Friday, March 25, 2016

Programming Language

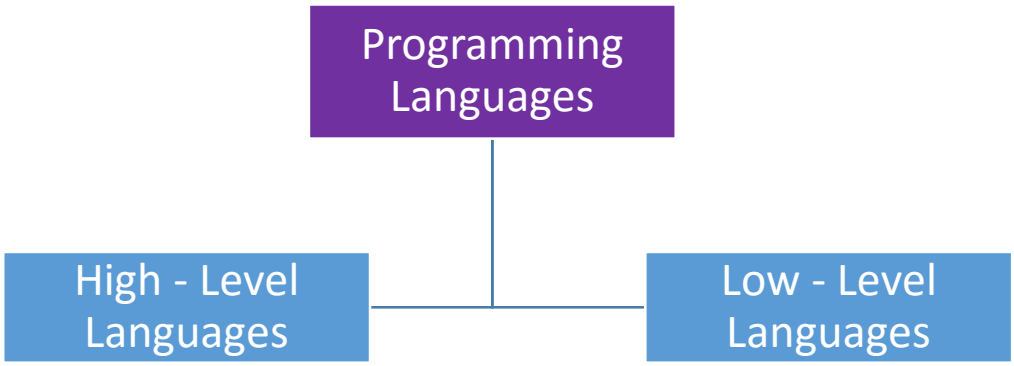
What is a programming Language…?

A programming language is a standardized communication technique for expressing instructions to a computer.

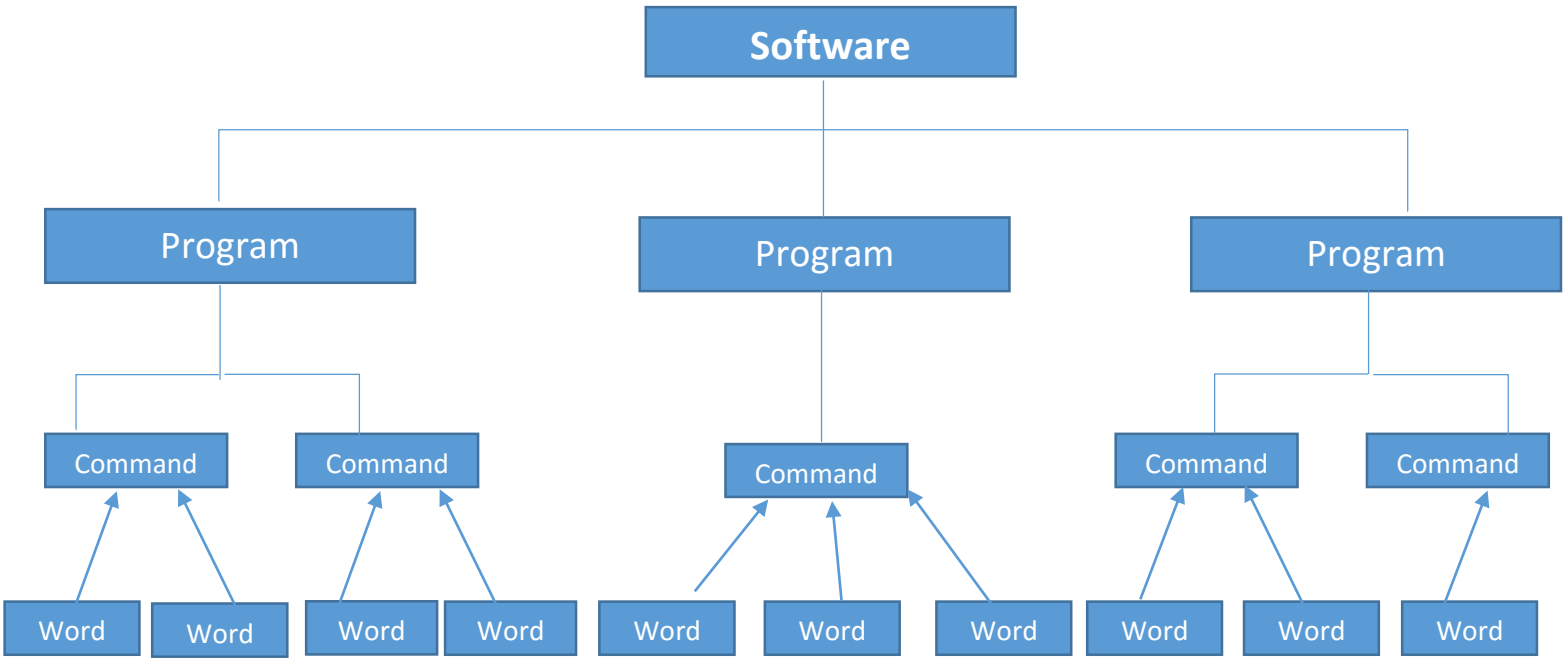
What are the programming languages you know…?

01. Java	02.C++	03.Pascal	04.VB 6	05.VB.Net
06. ASP.Net	07. C#	08.Android	09.AlGol	10.PHP
11. Go				

Types of programming Languages



How a software is made…?



Your 1st Java Program

```
class MyFirstProgram{
        public static void main(String [] args){
                System.out.println("Welcome to Java...!");
        }
}
```

Open and close the brackets same times.

Rules for Set a Class Name

- Must start with a letter, a currency character (**$**), or a connecting character such as the underscore (_).
- First letter should be capitalized.
- Words are linked together to form the name, the first letter of the inner words should be uppercase. **("CamelCase" Format)**
- Cannot start with a number.
- Can't use a Java keyword

For example:

Dog
Account
PrintWriter
$arath
Test_Marks
_school

Main Method/ PSVM

public static void main(String args[]) {}

public static void main(String [] args) {}

public static void main(String [] abcd) {}

public static void main(String ravINdu[]) {}

public static void main(String sss123) {}

Print And Println

```
class A{
        public static void main(String args[]){
                System.out.println("A");
                System.out.print("B");
}       }
```

Output:

```
C:\Users\Ravi\Desktop\OOPC-I>java A
A
B_
```

```
class A{
        public static void main(String args[]){
                System.out.print ("A");
                System.out.println("B");
}       }
```

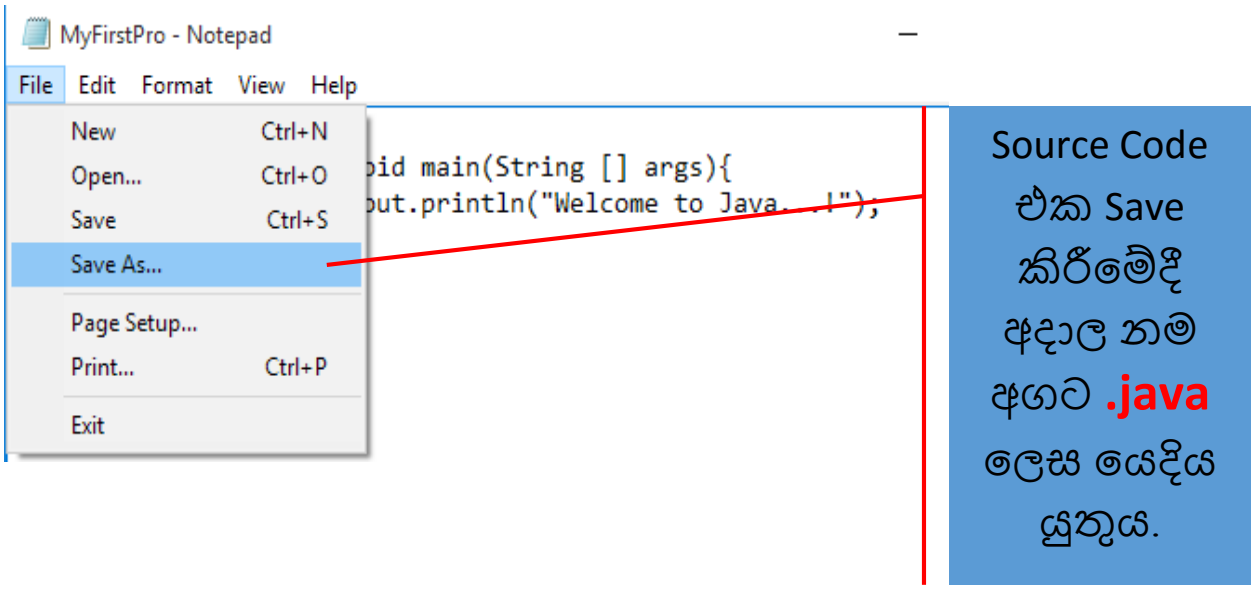
Output:

```
C:\Users\Ravi\Desktop\OOPC-I>java A
AB
_
```

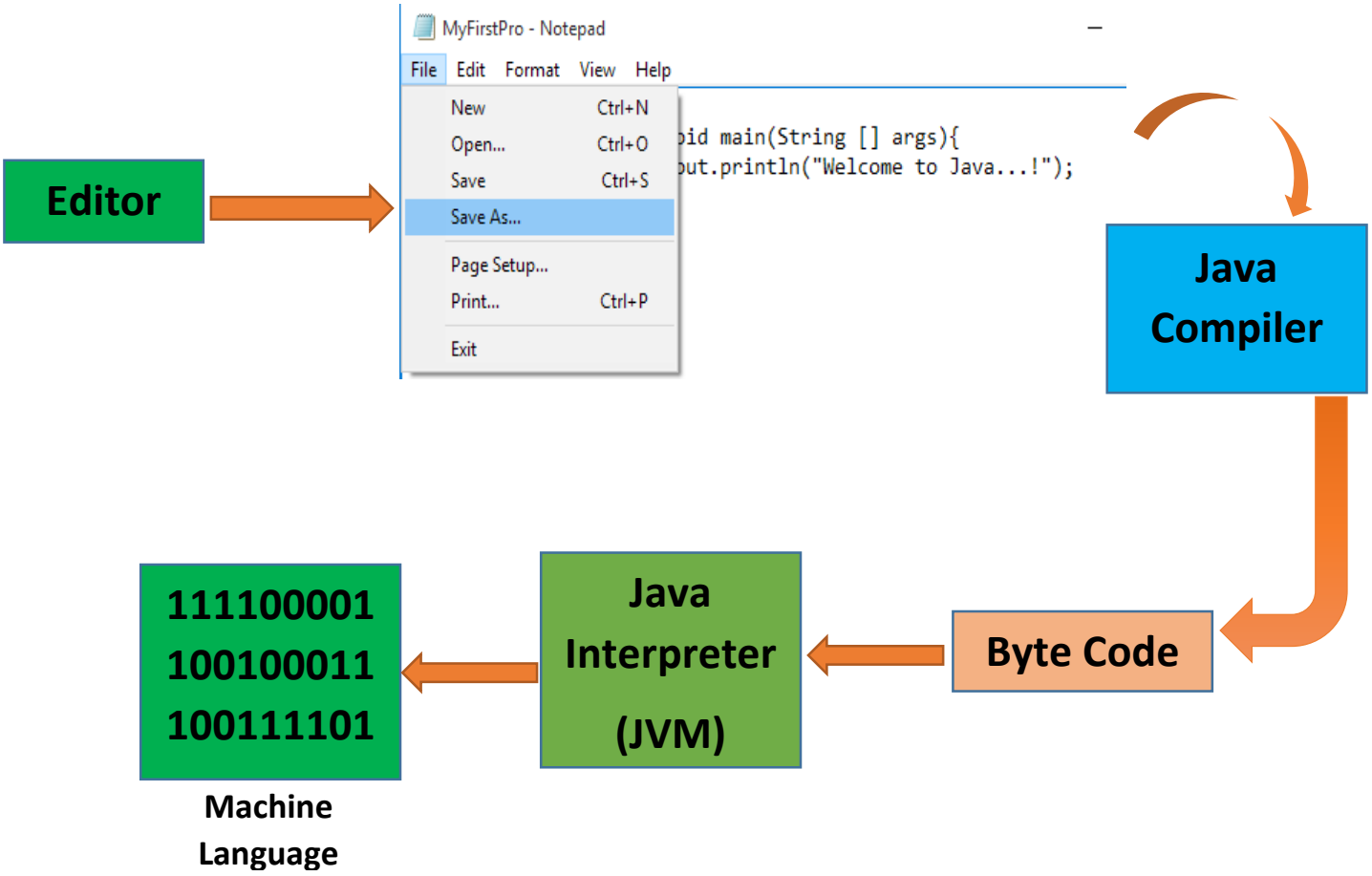
Compiler Cursor Position

- print හි කාර්යය වන්නේ Print කිර කසරය වම් දකුණු පසට ගෙනයයි.
- println 1දී " " ඇතුලත දී ඇති දෙය Print කර කසරය පහලට ගෙනයයි.

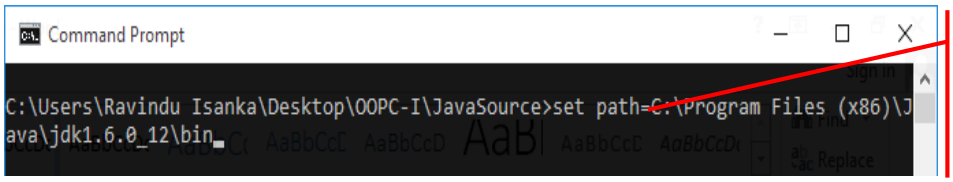
How to save Java File



How to a Java program works



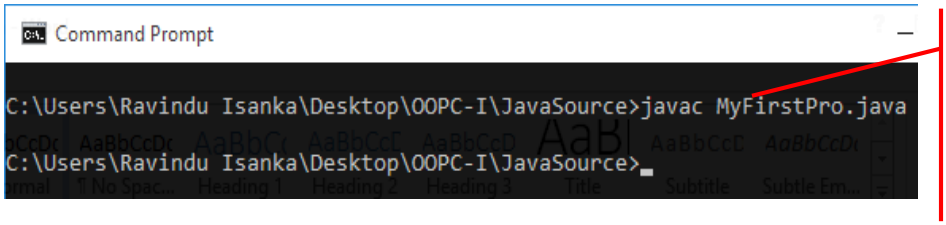
How to set a Path…?



This is jdk's bin file location.

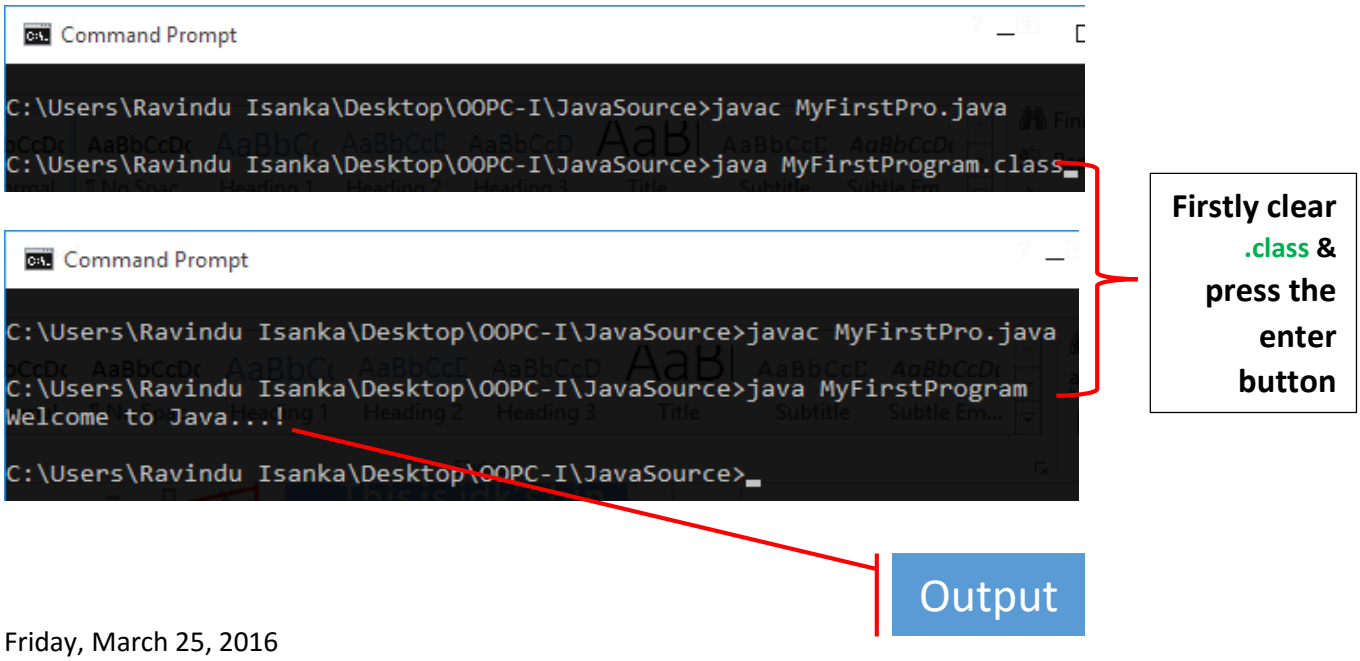
How to compile & run Java File….

1. Compile Java Source File…



Command to compile the program

1. Run class file



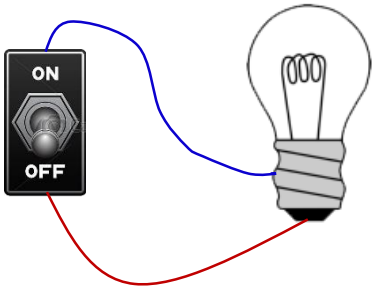
Friday, March 25, 2016

Data Types

What is a data...?

- A number
- A letter
- A name
- A place
- It can be anything
  - Raw facts and figures

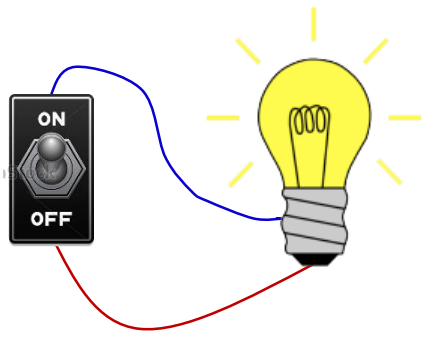
How does the computer understand data...?



Switch set to 0

Bulb off

No electrical pulse



Switch set to 1

Bulb On

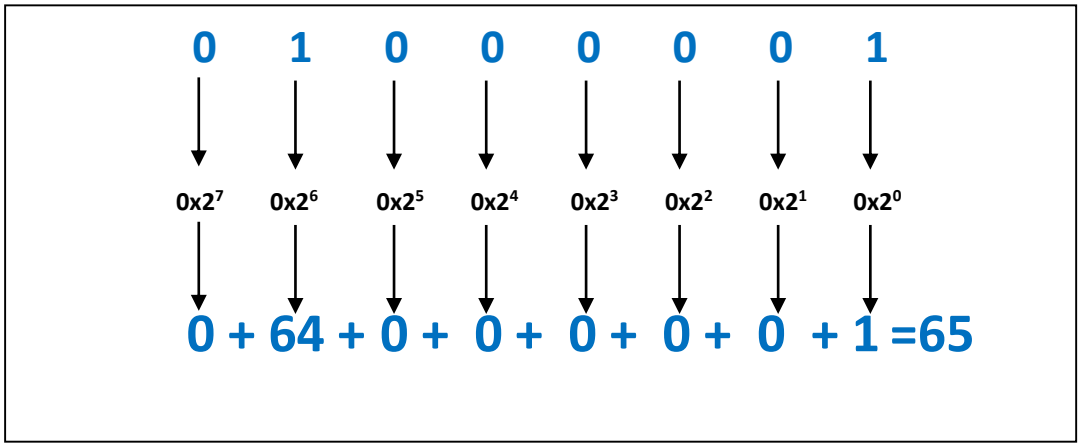
Electrical pulse

How do we convert data 1s & 0s?

```
2|65------1
2|32------0
2|16------0
2|8-------0
2|4-------0
2|2-------0
  1
```

$65_{10} = 1000001_2$

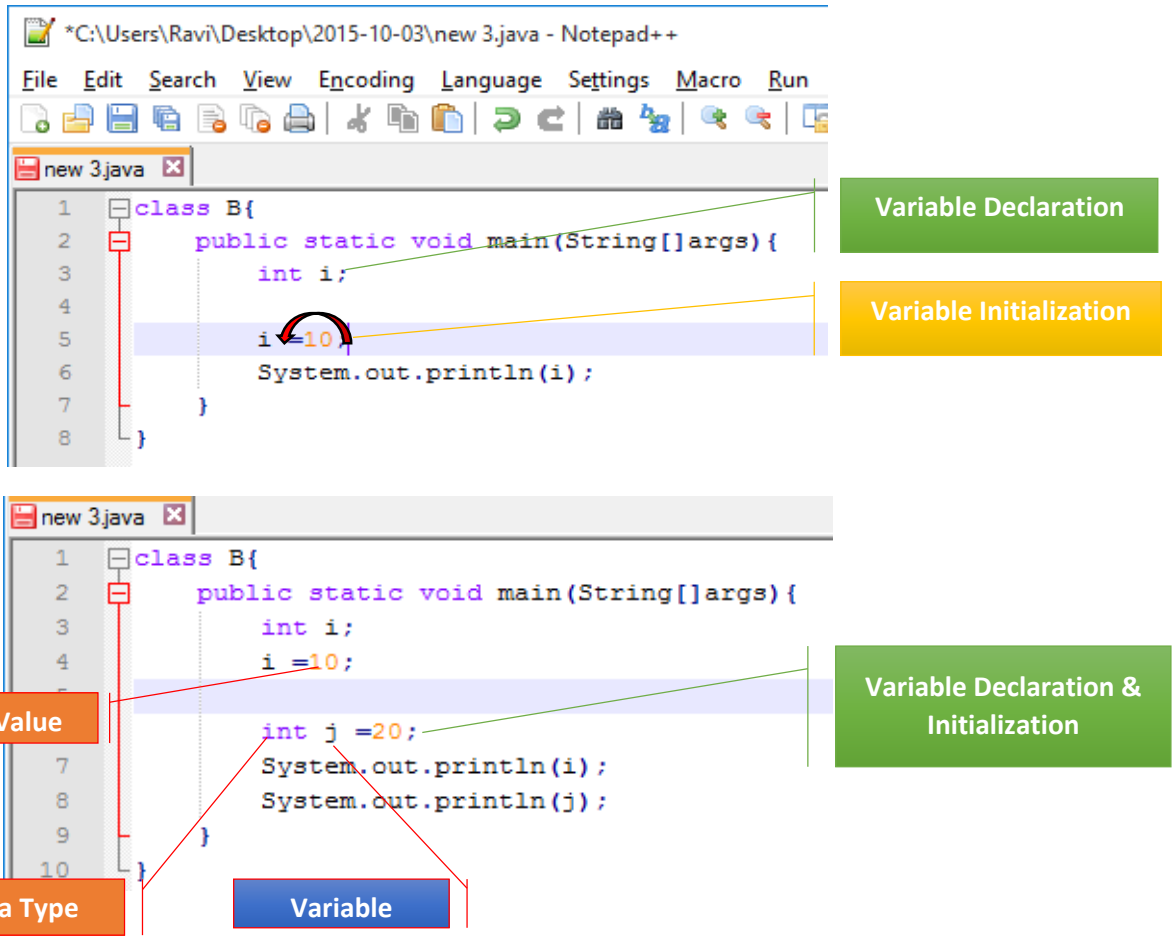
How do we convert bits back to its number?



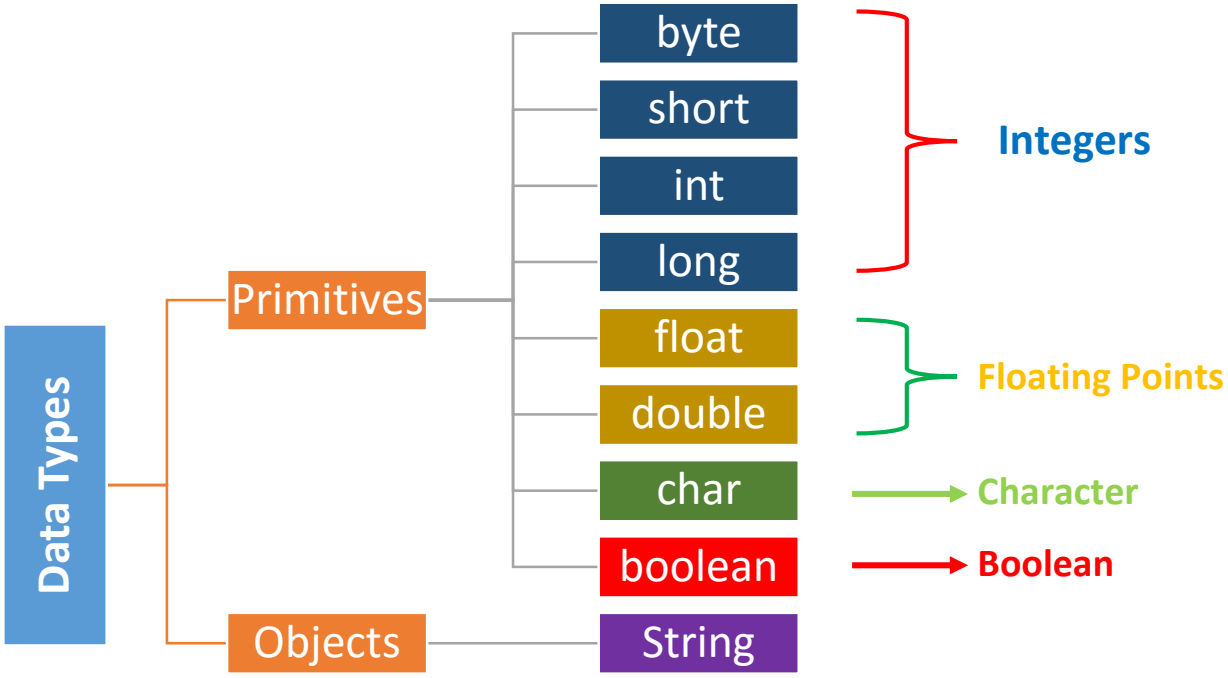
What is a variable…?

X=55 **X=5** **X=100001** **X=1.256**

- **X is a variable which can keep any value between 0 and infinity.**
- **Store data in run time.**



Data Type



Integer

1. long වර්ගයේ විචල්‍යයකට (variable) පූර්ණ සංඛ්‍යා යෙදීමේ දී (initialization) අදාල අගයේ අගට L හෝ l යෙදීමේ කිසිදු වරදක් නොමැත.

Floating Points

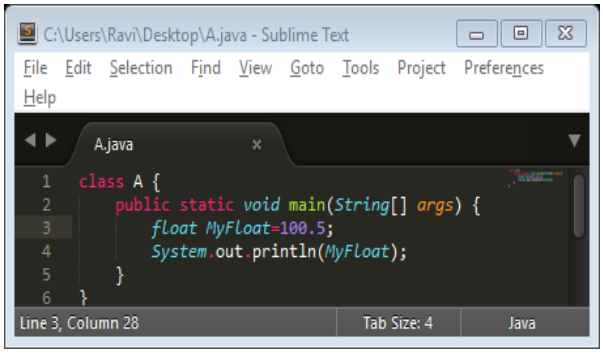
Float

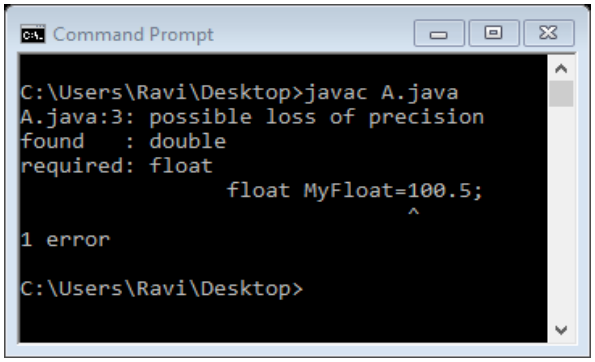
1. float වර්ගයේ විචල්‍යයකට (variable) පූර්ණ සංඛ්‍යා යොදා (initialization) compile කරීමේදී කිසිදු error එකක් නොලැබෙන අතර එහි output ලැබෙනුයේ දශම සංඛ්‍යා ස්භාවයෙනි.

```
class A {
    public static void main(String[] args) {
        float MyFloat=100;
        System.out.println(MyFloat);
    }
}
```

```
C:\Users\Ravi\Desktop>javac A.java

C:\Users\Ravi\Desktop>java A
100.0

C:\Users\Ravi\Desktop>
```

2. float වර්ගයේ විචල්‍යයකට (variable) දශම (double) සංඛ්‍යා යොදා (initialization) compile කරීමේදී error ලැබෙනු ඇත.





මෙහිදී දශම සංඛ්‍යා ලබාදිය යුත්තේ Floating Value ස්වභාවයෙනි.එනම්,

float myf=100.51F හෝ **float myf=100.51f** ලෙසය.

```
class A {
    public static void main(String[] args) {
        float MyFloat1=100.5F;
        float MyFloat2=12.56f;

        System.out.println(MyFloat1);
        System.out.println(MyFloat2);
    }
```

```
C:\Users\Ravi\Desktop>javac A.java

C:\Users\Ravi\Desktop>java A
100.5
12.56

C:\Users\Ravi\Desktop>
```

Double

2. double වර්ගයේ විචල්‍යයකට (variable) පූර්ණ සංඛ්‍යා යොදා (initialization) compile කරීමේදී කිසිදු error එකක් නොලැබෙන අතර එහි output ලැබෙනුයේ දශම සංඛ්‍යා ස්භාවයෙනි.

```
class A {
    public static void main(String[] args) {
        double mydouble=77;
        System.out.println(mydouble);
    }
}
```

```
C:\Users\Ravi\Desktop>javac A.java

C:\Users\Ravi\Desktop>java A
77.0

C:\Users\Ravi\Desktop>
```

3. double වර්ගයේ විචල්‍යයකට (variable) පූර්ණ සංඛ්‍යා හෝ දශම සංඛ්‍යා යෙදීමේ දී (initialization) අදාල අගය අගට D හෝ d යෙදීමේ කිසිදු වරදක් නොමැත.

```
class A {
    public static void main(String[] args) {
        double mydouble1=77d;
        double mydouble2=12.7d;
        System.out.println(mydouble1);
        System.out.println(mydouble2);
```

```
C:\Users\Ravi\Desktop>javac A.java

C:\Users\Ravi\Desktop>java A
77.0
12.7

C:\Users\Ravi\Desktop>
```

Char

char c = ‘a‘;
char c = 0→65535;
char c = ‘\b‘ , ‘\f‘ , ‘\n‘ , ‘\r‘ , ‘\t‘ , ‘\‘ , ‘\\‘ , ‘\”‘;
char c = ‘\u0000’ - ’\uffffff‘;
char c = ‘0x0000 - 0xffff‘;
char c = -98; //CE
char c = (char) -98; //OK

1. Variable සඳහා Character ලබාදීමේදී සෑම විටම තනි උද්ධෘත [(‘ ’) single quotations] යටතේ ලබාදිය යුතුය.
2. char Variable සඳහා සෑම විටම එක් අවස්ථාවකදී තබා ගත හැක්කේ එක් character පමණි.
3. char Variable එකට int අගයක් ලබාදී compile & run කිරීමෙ මගින් එම අගයට අදාල character එක නිරූපණය කරයි.

```
    public static void main(String[] args) {
        char mychar1='A';
        char mychar2=65;
        char mychar3=97;
        System.out.println("mychar1="+mychar1);
        System.out.println("mychar2_ int 65="+mychar2);
        System.out.println("mychar3_ int 97="+mychar3);
    }
}
```

```
C:\Users\Ravi\Desktop>java A
mychar1=A
mychar2_ int 65=A
mychar3_ int 97=a

C:\Users\Ravi\Desktop>
```

Boolean

1. Boolean variable එකට තබාගත හැක්කේ true සහ false යන keyword 2 පමණි.

```
    public static void main(String[] args) {
        boolean mybool1=true; boolean mybool2=false;
        System.out.println("mybool1="+mybool1);
        System.out.println("mybool2="+mybool2);
    }
}
```

```
C:\Users\Ravi\Desktop>javac A.java

C:\Users\Ravi\Desktop>java A
mybool1=true
mybool2=false

C:\Users\Ravi\Desktop>
```

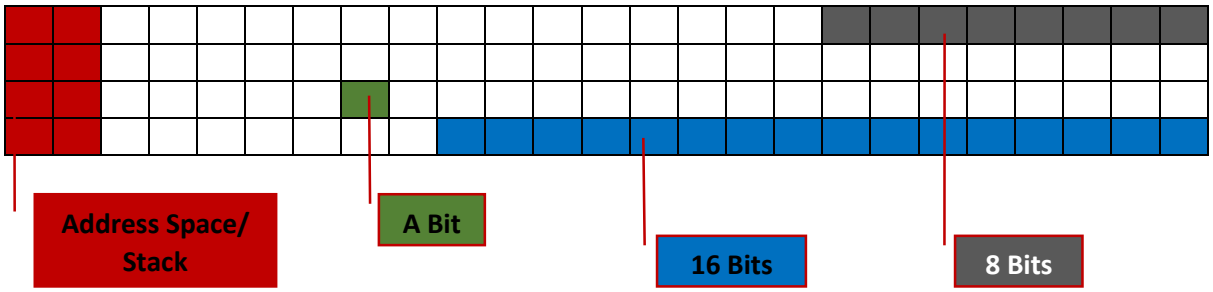
String

1. සෑම විටම ද්විත්ව උද්ධෘතපාඨ (Double Quotations) තුල ලිවිය යුතුය.

```
    public static void main(String[] args) {
        String myst1="Ravindu Isanka";
        System.out.println(myst1);
    }
}
```

```
C:\Users\Ravi\Desktop>javac A.java

C:\Users\Ravi\Desktop>java A
Ravindu Isanka

C:\Users\Ravi\Desktop>
```

What Dose The Memory Of Computer Look Like



Type	Bits	Bytes
Integers		
byte	**8**	**1**
short	**16**	**2**
int	**32**	**4**
long	**64**	**8**
Floating Point		
double	32	4
Float	64	8
Character		
char	**16**	**2**
Boolean		
boolean	**1**	

ඉහතින් දක්වා ඇති ඇත්තේ, ආදාල Data Type එකේ variable එකක් නිර්මාණය කර එයට යම් අගයක් ඇතුලත් කරන විට Ram එක තුල ලබාගන්නා ඉඩ ප්‍රමාණයන්ය.

පහත පරිදි විවිධ variables Ram එක තුල ඉඩ වෙන් කරගන්නා අයුරු Stack & Heap ඇදීම මගින් නිරූපණය කළ හැක.

Ex:-1

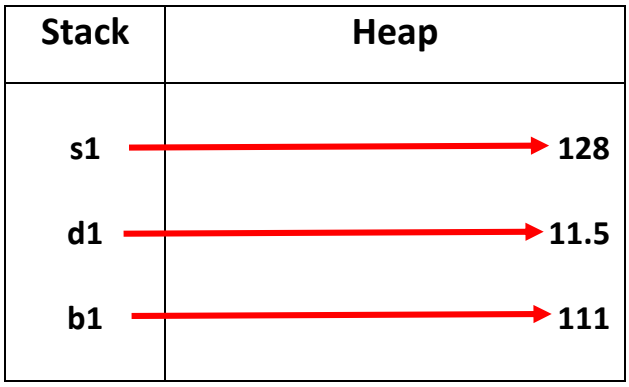
```
class Daya4{
    public static void main(String []args){
        short myshort1=128;
        System.out.println(myshort1);
    }
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac Day4_2.java

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java Daya4
128

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

EX:-2

```
class Daya4{
    public static void main(String []args){
        short s1=128;
        double d1=11.5;
        byte b1=111;
        System.out.println(s1);
```

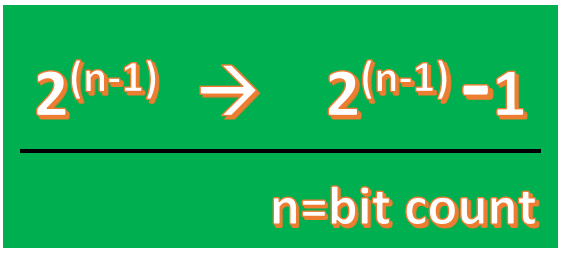
```
:\Users\Ravi\Desktop\OOPC-I _ Day4>javac Day4_2.java

:\Users\Ravi\Desktop\OOPC-I _ Day4>java Daya4
28
1.5
11

:\Users\Ravi\Desktop\OOPC-I _ Day4>
```



Variable With Whole Numbers (Without Decimal Points)

Data Type	Value Range	Bits	Byte
byte x	-128→127 -2^{8-1} to+ $2^{8-1}-1$	8	1
short x	-2^{16-1} to+ $2^{16-1}-1$	16	2
int x	-2^{32-1} to+ $2^{32-1}-1$	32	4
long x	-2^{64-1} to+ $2^{64-1}-1$	64	8
double x	-2^{32-1} to+ $2^{32-1}-1$	32	4
float x	-2^{64-1} to+ $2^{64-1}-1$	64	8
char x	-2^{16-1} to+ $2^{16-1}-1$	16	2
boolean x	true & false	1	

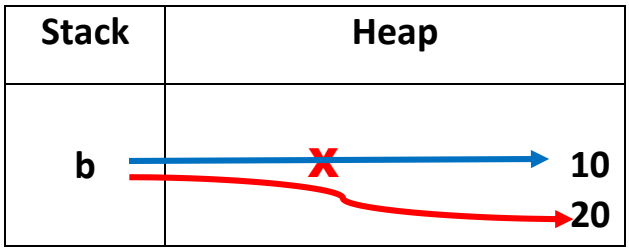
- විවිධ Data Type වලට අදාල අගය පරාසයන් පහත සූත්‍රයභාවිතයෙන් සොයාගත හැක.



Ex:-01

```
class Day4_4A{
    public static void main(String[]args){
        byte b=10;
        b=20;
        System.out.println(b);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac Day4_4.java

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java Day4_4A
20

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```



Casting

මෙහිදී සිදුකරනුයේ යම් Data Type එකකින් නිර්මාණය කර ඇති විචල්‍යයක අගය වෙනත් Data Type එකට ගැළපෙන පරිදි පරිවර්තනය කිරීමයි. මෙය casting ලෙස හඳුන්වන අතර Auto Casting සහ Manual Casting වශයෙන් ආකාර දෙකක් පවති.

Ex:-01

```
class Day4_4{
    public static void main(String[]args){
        int b1=10;
        byte i1=(byte)b1;
        System.out.println(i1);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac Day4_5.java

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java Day4_4
10

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

Ex:-02

```
class Day4{
    public static void main(String[]args){
        double z=10.44;
        float d =z;
        System.out.println();
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new (7).java"
new (7).java:4: possible loss of precision
found   : double
required: float
                float d =z;
                          ^
1 error

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

- මෙහිදී සිදුකර ඇත්තේ double වර්ගයේ විචල්‍යමය අගයක් float වර්ගයේ විචල්‍යයකට සමාන කර තිබීමයි. නමුත් එහිදී ලැබෙනුයේ double වර්ගයේ අගයක් සොයාගත් අතර නමුත් අවශ්‍යව පවතින්නේ float වර්ගයේ එකක් බවත්ය.

  මෙම ගානම පහත අයුරින් cast කර විචල්‍ය සඳහා අගයන් ලබාදීමේ දී එසේ එය සිදු නොවේ.....

```
class Day4{
    public static void main(String[]args){
        double z=10.44;
        float d =(float)z;
        System.out.println(d);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new (7).java"

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java Day4
10.44

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

Ex:-03

```
class Day4{
    public static void main(String[]args){
        byte b =130;
            System.out.println(b);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new (8).java"
new (8).java:3: possible loss of precision
found   : int
required: byte
                byte b =130;
                       ^
1 error

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

- මෙහිදී byte වර්ගයේ variable ඒකක් සඳහා 130 යන අගය ලබා දී ඇත. නමුත byte වල අගය පරාසය පිහිටා ඇත්තේ -129 සිට 128 දක්වා බැවින් එහිදී error එකක් ලැබෙනුයේ එම අගය int වර්ගයට අයත් බැවිනි.

A/C _Automatic Conventions

boolean byte short int long float double
char

Casting

```
class Day4{
    public static void main(String[]args){
        float f =(float)1212.34;
            System.out.println(f);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new 9.java"

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java Day4
1212.34

C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new 9.java"
```

Comments

```
class Day4{
    public static void main(String[]args){
        //casting sample - This is a single line comment
        float f =(float)1212.34;
        /*Using sop
            12255ss54c5s45c4s54c
                        This is a multiline comment
        */
            System.out.println(f);
    }
}
```

Literals Value

Literal values for all primitive type

A primitive literal is merely a source code representation of the primitive data types. The following are examples for primitive literals:

'b'	**//char literal**
42	**//int literal**
false	**//boolean literal**
25555555.222	**//double literal**

Integer literals

There are three ways to represent integer numbers in the java language: **decimal (base 10), Octal (base 8) and hexadecimal (base 16)**.most exam questions with integer literals use decimal representations.

1. Decimals Literals

Decimal integer need no explanation: you've been using then since grade one or earlier, in the java language, they are represented as is, with no prefix of any kind as follows.

Int length=10;

```
class DAya4{
    public static void main(String[]args){
        int length=343;
        System.out.println(length);
    }
}
```

2. Octal Literals

Octal integers use only the digits 0 to 7. In java, you represent a literal in octal form by placing zero in front of the numbers, as follow.

```
class DAya4{
    public static void main(String[]args){
        int six=06;          //Equal to decimal 6
        int seven=07;        //Equal to decimal 7
        int eight=010;       //Equal to decimal
        int nine=011;        //Equal to decimal 9
        System.out.println(six);
        System.out.println(seven);
        System.out.println(eight);
        System.out.println(nine);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new 11.java"

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java DAya4
6
7
8
9

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

- පහතින් දක්වා ඇති අවස්ථාවේ දී variable එක සඳහා අගය ඇතුලත් කිරීමේදී ඔක්ටල් සංඛ්‍යා පද්ධතියේන් ඔබ්බට ගිය අගයක් බැවින් එය integer values සඳහා වලංගු නොවේ.

```
class DAya4{
    public static void main(String[]args){
        short s =0184;
        System.out.println(s);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new 12.java"
new 12.java:3: integer number too large: 0184
                short s =0184;
                         ^
1 error

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

3. Hexadecimal Literals

Hexadecimal (hex for short) numbers are constructed using 16 distinct symbols. Because we never invented single digit symbols for the numbers 10 through 15, we use alphabetic characters to represent these digits. Java will accept uppercase or lowercase letters those extra digits counting form 0 through 15 in hex looks like this.

```
class HexTest{
    public static void main(String[]args){
        int x=0X0001;
        int y=0x7fffffff;
        int z=0xDeadCafe;
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new 13.java"

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java HexTest
1
134217727
-559035650

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

```
class HexTest{
    public static void main(String[]args){
        int x=0X11A;
        System.out.println(x);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new 14.java"

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java HexTest
282

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

```
class HexTest{
    public static void main(String[]args){
        long l=0X11AL;
        System.out.println(l);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC-I _ Day4>javac "new 15.java"

C:\Users\Ravi\Desktop\OOPC-I _ Day4>java HexTest
282

C:\Users\Ravi\Desktop\OOPC-I _ Day4>
```

Don't be misled by changes in case for hexadecimal digits or the 'x' preceding it. **OXCAFE and oxcafe** are both legal and have same value. All three integer literals **(octal, decimal and hexadecimal)** are defined as int by default, but they may also be specified as long by placing a suffix of L or l after the number:

Long lo =11055**L**;

Long so=0XFFFF**L**; // note the Lowercase l

Floating point Literals

Floating point numbers are defined as a number a, decimal, symbol are more numbers representing the fraction.

double =11301874.9881024

In the preceding example the number 11201974.9881024 is the literal value. Floating point literals are defined as double (64 bits) by default, so if want to assign a floating –point literal to a variable of type float (32 bit) you must attach the suffix F or f to the number.

float f = 23.2564; //compile error

float f = 498.56245F;

You may also optionally a **D** or **d** to double literals, but it is not necessary because this is the default behavior

double d = 111206.564646D; // optional not required

```
class BS{
    public static void main(String[]args){
        double d=0111.11;
        System.out.println(d);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac B.java

C:\Users\Ravi\Desktop\OOPC_1_17>java BS
111.11

C:\Users\Ravi\Desktop\OOPC_1_17>
```

පහත උදාහරණයේ Double Varibe එක සඳහා ලබා දී ඇත්තේ Hexadecimal අගයකි. මෙහිදී නොගැලැපෙන අගය වර්ගයක නිසා පහත error එක ලැබේ.

```
class BS{
    public static void main(String[]args){
        double d=0x111.11;
        System.out.println(d);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac C.java
C.java:3: malformed floating point literal
                double d=0x111.11;
                         ^
1 error

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Boolean Literals

Boolean literals are the source code representation for Boolean values. A Boolean value can only be define as a true or false. Although in C (and some other language) it is common to use to numbers to represent true or false this will not work in java. Again repeat after me “Java is not C++”

boolean b =true;

boolean b2=false;

```
class Bool{
    public static void main(String[]args){
        boolean b1=true;
        boolean b2=false;
        System.out.println(b1);
        System.out.println(b2);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac D.java

C:\Users\Ravi\Desktop\OOPC_1_17>java Bool
true
false

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Boolean Literals කිසිම අවස්ථාවක java භාෂාව තුලදී අංකික වටිනාකම් දරන්නේ නොමැත. පහත දක්වා ඇත්තේ ඒ සඳහා උදාහරණයකි.

```
class Bool{
    public static void main(String[]args){
        boolean b1=true;
        boolean b2=false;

        boolean b3 =0; //This line will prodeuce a compile error
        System.out.println(b1);
        System.out.println(b2);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac E.java
E.java:6: incompatible types
found   : int
required: boolean
                boolean b3 =0; //This line will prodeuce a compile error
                           ^
1 error

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Character Literals

A char literal is represented by a single character in single quotes.

char ='a';

char='@'

You can also type in the Unicode value of character, using the Unicode notation of prefixing the value with **\u** as follows:

Cha letterN ='\u004E' //The Letter N

- We can't use **uppercase U** before Unicode value.

```
class Character_lit{
    public static void main(String[]args){
        char a ='A';
        char  Letter='\u004E';

        System.out.println(a);
        System.out.println(Letter);

    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac F.java

C:\Users\Ravi\Desktop\OOPC_1_17>java Character_lit
A
N

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Remember character are just 16 bit unsigned integers under the hood. That means you can assign a number literal assuming it will fit into the unsigned 16 bit range (65536 or less). For example, following are all legal:

char a =0x892; //hexadecimal literals

char b =982; //int literals

char c =(char)70000; //The cast is required, out of char range

char e =0101;

char f=0x41;

char g=(char)//Ridiculous ,but Legal

And the following are not legal and produce compile error

char h =-29; // possible loss of precision ; need a cast

char I = 70000; // possible loss of precision (Out of range); need a cast

You can also use an escape code if you want to represent a character that can't be typed in as a literals, including the characters for linefeed, new line, horizontal tab backspace, and single quotes.

char c = '\"' //a double quote

char d ='\n' // A newline

```
class A_2_Z{
    public static void main(String[]args){
        char c ='\"';
        char d='\n';

        System.out.print("Hi My Name is ");
        System.out.print(c);
        System.out.print("Amal");
        System.out.print(c);
        System.out.print(d);
        System.out.println("How are you?");
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac I.java

C:\Users\Ravi\Desktop\OOPC_1_17>java A_2_Z
Hi My Name is "Amal"
How are you?

C:\Users\Ravi\Desktop\OOPC_1_17>
```

String Literals

A literal is a source code representation of a value of a string objects. For example the following is an example of two ways to represent a string literal:

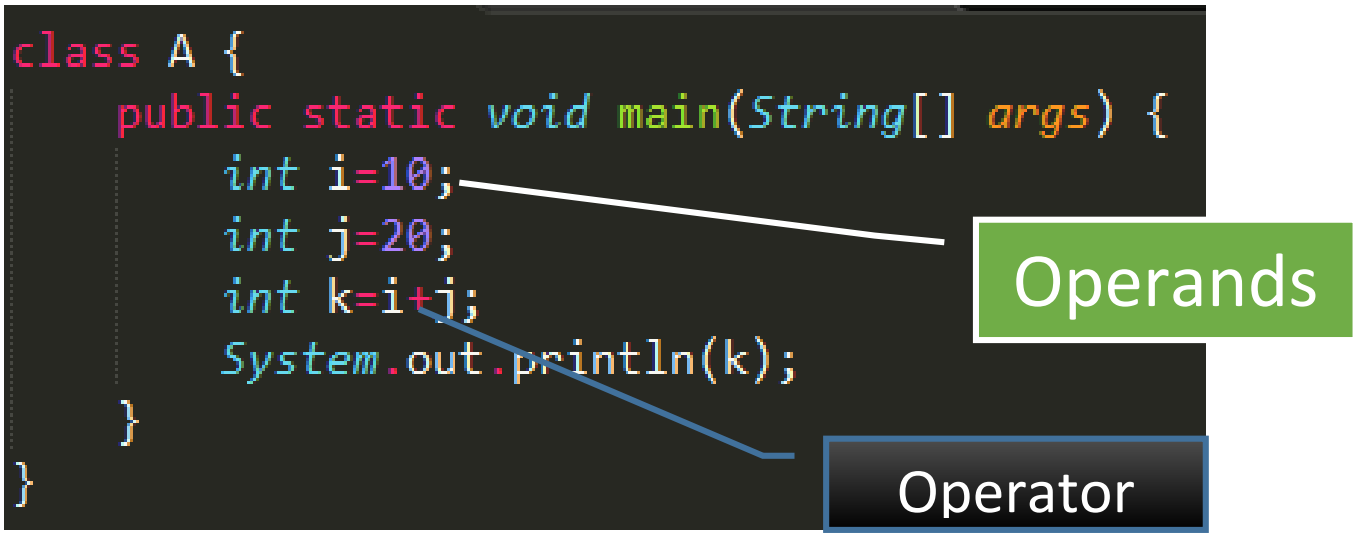
String s ="Bill Joy"

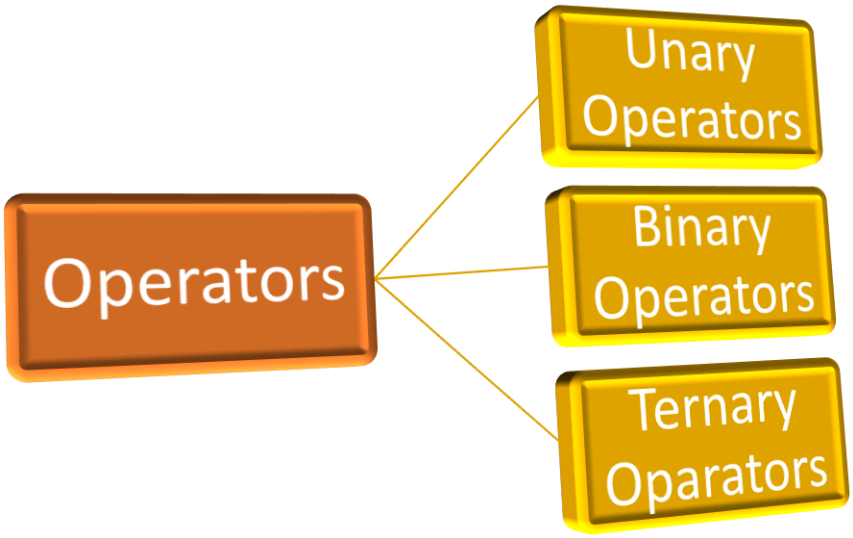
Although string are not primitives, there're include in this section because they can be represented as literals

```
class B{
    public static void main(String[]args){
        String s1="AB";
        String s2="CD";
        String s3=s1+s2;
        String s4="Hi-"+s3;

            System.out.println(s1);
            System.out.println(s2);
            System.out.println(s3);
            System.out.println(s4);

    }
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac J.java

C:\Users\Ravi\Desktop\OOPC_1_17>java B
AB
CD
ABCD
Hi-ABCD

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Operators





Basic Arithmetic Operators

- **+ → Addition**
- **- → Subtraction**
- *** → Multiplication**
- **/ → Division**

```
class C {
    public static void main(String[] args) {

        int i =-10;
        int j = 20;

        int z1 = j-i;
        int z2 = j+i;
        int z3 = j*i;
        int z4 = j/i;

        System.out.println("20-10="+z1);
        System.out.println("20+10="+z2);
        System.out.println("20x10="+z3);
        System.out.println("20/10="+z4);
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac Oparators_3.java

C:\Users\Ravi\Desktop\OOPC_1_17>java C
20-10=30
20+10=10
20x10=-200
20/10=-2

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Modulus Operator

- මෙමගින් වම් පස අගය දකුණු පස අගයෙන් බෙදා ඉතිරිය පිළිතුර ලෙස ලබා දෙයි.

```
class C {
    public static void main(String[] args) {

        int i =7;
        int j =4;

        int k = i%j;

        System.out.println(k);
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>java C
3

C:\Users\Ravi\Desktop\OOPC_1_17>
```

- Byte , Short වර්ගයේ අගයන් දෙකක යම් ගණිත කර්මයකට භාජනය වීමෙන් පසු එමගින ලැබෙන අගය int අගයක් බවට පත්වේ. එබැවින් උක්තව දැකවූ පරිදි අවසන් අගය int variabe එකක් යටතේ පැවතිය යුතුය.

Ex:-01

```
class D{
    public static void main(String[] args) {
        byte b1=10;
        byte b2=20;

        byte b3=b1+b2;

        System.out.println(b3);
    }

}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac Oparators_5.java
Oparators_5.java:6: possible loss of precision
found   : int
required: byte
                byte b3=b1+b2;
                         ^
1 error

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Ex:-02

```
class D{
    public static void main(String[] args) {
        byte b1=10;
        byte b2=20;

        int b3=b1+b2;

        System.out.println(b3);
    }
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac Oparators_5.java

C:\Users\Ravi\Desktop\OOPC_1_17>java C
The Value Of k is  =4
The Value Of k1 is =3

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Ex:-03

```
class D{
    public static void main(String[] args) {
        byte b1=10;
        byte b2=20;

        short b3=b1+b2;

        System.out.println(b3);
    }

}
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac Oparators_5.java
Oparators_5.java:6: possible loss of precision
found   : int
required: short
                short b3=b1+b2;
                          ^
1 error

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Ex:-04

```
class D{
    public static void main(String[] args) {
        short s1=25;
        short s2=2;

        short s3=s1*s2;

        System.out.println(s3);
    }
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac Oparators_5.java
Oparators_5.java:6: possible loss of precision
found   : int
required: short
                short s3=s1*s2;
                          ^
1 error

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Ex:-05

```
class D{
    public static void main(String[] args) {
        short s1=25;
        short s2=2;

        int s3=s1*s2;

        System.out.println(s3);
```

```
C:\Users\Ravi\Desktop\OOPC_1_17>javac Oparators_5.java

C:\Users\Ravi\Desktop\OOPC_1_17>java D
50

C:\Users\Ravi\Desktop\OOPC_1_17>
```

Arithmetic Assignment

- **+= → Addison Assignment**
- **-= → Subtraction Assignment**
- ***= → Multiplication Assignment**
- **/= → division Assignment**
- **%= → Modulus Assignment**

a += 5 → a= a+ 5

a -= 5 → a= a- 5

a *= 5 → a= a* 5

a /= 5 → a= a/ 5

a %= 5 → a= a% 5

Increment & Decrement

++ → Increment

-- → Decrement

මෙය අවස්ථා දෙක්ක යටතේ භවිතා කරයි. එය,

Prefix සහ Postfix වශයෙන් හඳුන්වයි.

```
class AB{
    public static void main(String[] args) {
        int i =10;

        i++;
        System.out.println(i);
        ++i;
        System.out.println(i);
        i++;
        ++i;
        System.out.println(i);
        i--;
        System.out.println(i);
        --i;
        System.out.println(i);
        --i;
        i--;
        System.out.println(i);
    }
}
```

Postfix Increment

Prefix Increment

Prefix Decrement

Postfix Decrement

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_a.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
11
12
14
13
12
10

C:\Users\Ravi\Desktop\OOPC_I>
```

- මෙහි **Prefix Increment** අවස්ථාවේදී **(++x)** සිදුවනුයේ පළමුව විචල්‍ය සඳහා එකක් එකතු වී ප්‍රතිඵලය ලබාදීමය. **Prefix Decrement** අවස්ථාවේදී **(--x)** ද පළමුව විචලයෙන් එකක් අඩු වී ප්‍රතිඵලය ලබාදීමය.

නමුත් **Postfix Increment** අවස්ථාවේදී **(x++)** සිදුවනුයේ ප්‍රතිඵලය ලබාදීමෙන් අනතුරුව විචල්‍ය සඳහා එකක් එකතු වීමය. **Postfix Decrement** අවස්ථාවේදී **(x++)** ද ප්‍රතිඵලය ලබාදීමෙන් අනතුරුව විචලයෙන් එකක් අඩු වීම සිදුවෙයි.

EX:-01

```
class AB{
    public static void main(String[] args) {
        int i =10;

        System.out.println(++i);
        System.out.println(i);
        System.out.println(i++);
        System.out.println(i);
        System.out.println(--i);
        System.out.println(i);
        System.out.println(i--);
        System.out.println(i);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
11
11
11
12
11
11
11
10

C:\Users\Ravi\Desktop\OOPC_I>
```

- Prefix Increment සහ Prefix Decrement පහත අයුරින්ද භාවිතා කළ හැක.

y=++x → y=x++

y=--x → y=x--

Ex:-02

```
class AB{
    public static void main(String[] args) {
        int p =45;
        int q=p++; //int q=++p

        System.out.println("Value of p is "+p);
        System.out.println("Value of q is "+q);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
Value of p is 46
Value of q is 45

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-03

```
class AB{
    public static void main(String[] args) {
        int p =45;
        int q=p--; //int q=--p

        System.out.println("Value of p is "+p);
        System.out.println("Value of q is "+q);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_d.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
Value of p is 44
Value of q is 45

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-04

```
class AB{
    public static void main(String[] args) {
        double d =3.55;

        System.out.println(++d);
        System.out.println(d);
        System.out.println(d++);
        System.out.println(d);

        System.out.println(--d);
        System.out.println(d);
        System.out.println(d--);
        System.out.println(d);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_e.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
4.55
4.55
4.55
5.55
4.55
4.55
4.55
3.55

C:\Users\Ravi\Desktop\OOPC_I>
```

Relational Operators

- == Equal to
- != not equal
- > Greater Than
- < less Than
- >= Greater than or equal to
- <= less than or equal to

මෙහිදී යොදගනු ලබන සියලුම operands, සංඛ්‍යා විය යුතුය. ලැබෙන ප්‍රතිඵල Boolean අගයන් වේ.

Ex:-01

```
class AB {
    public static void main(String[] args) {
        int i=10;
        int j=20;
        boolean b=i<j;
        System.out.println(b);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
true

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-02

```
class AB {
    public static void main(String[] args) {
        int i=10;
        int j=20;
        System.out.println("\n<--If  i=10 & 20-->\n");
        System.out.println("i==j <--> "+(i==j));
        System.out.println("i!=j <--> "+(i!=j));
        System.out.println("i>j  <--> "+(i>j));
        System.out.println("i<j  <--> "+(i<j));
        System.out.println("i>=j <--> "+(i>=j));
        System.out.println("i<=j <--> "+(i<=j));
    }
}
```

```
<--If  i=10 & 20-->

i==j <--> false
i!=j <--> true
i>j  <--> false
i<j  <--> true
i>=j <--> false
i<=j <--> true

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-03

```
class AB {
    public static void main(String[] args) {
        int i=10;
        int j=10;

        boolean b=i<=100;

        System.out.println(b);
    }
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
true

C:\Users\Ravi\Desktop\OOPC_I>
```

Boolean Logical

1. **&** **(Logical AND)**
2. **|** **(Logical OR)**
3. **^** **(Logical XOR)**
4. **||** **(Short- circuit OR)**
5. **&&** **(Short- circuit AND)**
6. **!** **(Logical unary Not)**
7. **&=** **(AND Assignment)**
8. **|=** **(OR Assignment)**
9. **^=** **(XOR Assignment)**
10. **==** **(Equal to)**
11. **!=** **(Not Equal to)**

A	B	A OR B A | B	A And B A & B	A XOR B A ^ B	Not A !A
True	True	True	True	False	False
False	True	True	False	True	True
True	False	True	False	True	False
False	False	False	False	False	True

AND

මෙම තාර්කික ද්වාරය හරහා ගමන් කරන් අගයන් දෙකම සත්‍යනම් පමණක් අවසන් පළිතුර සත්‍ය වේ. එක් අගයක් හෝ අගයන් දෙකම අසත්‍ය විට පළිතුර අසත්‍ය වේ.

Ex:-01

```
1  class AB {
2      public static void main(String[] args) {
3
4          boolean b1=true;
5          boolean b2=false;
6          boolean b3= b1 & b2;
7          System.out.println(b3);
8      }
9  }
```

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_boolean_B.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
false

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-02

```
class AB {
    public static void main(String[] args) {

        boolean b1=true;
        boolean b2=false;
        boolean b3=true;
        boolean b4=b1&b2&b3;

        System.out.println(b4);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
false

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-03

```
class AB {
    public static void main(String[] args) {

        boolean b1=true;
        boolean b2=false;
        boolean b3=true;
        boolean b4=b1&b2&b3;

        System.out.println(b4);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
false

C:\Users\Ravi\Desktop\OOPC_I>
```

OR

මේහිදී අගයන් දෙකම හෝ එක් අගයක් සත්‍ය වේ නම් එහි අවසන් පළිතුර සත්‍ය වේ. අගයන් දෙකම අසත්‍ය විටදී පමණක් අවසන් පිළිතුර අසත්‍ය වේ.

Ex:-01

```
class AB {
    public static void main(String[] args) {

        boolean b1=false;
        boolean b2=true;
        boolean b3=true;
        boolean b4=false;

        System.out.println(b1|b2);
        System.out.println(b2|b3);
        System.out.println(b1|b4);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
true
true
false

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-02

```
class AB {
    public static void main(String[] args) {

        boolean b1=false;
        boolean b2=true;
        boolean b3=b1|b2;
        boolean b4=b3&true|false&b2;

        System.out.println(b4);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
true

C:\Users\Ravi\Desktop\OOPC_I>
```

XOR_^

මෙම අවස්ථාවෙදී අගයන් දෙකම අසත්‍ය හෝ සත්‍ය නම් අවසන් පිලිතුර අසත්‍ය වේ.

Ex:-01

```
class AB {
    public static void main(String[] args) {

        boolean b1=true;
        boolean b2=false;
        boolean b3=true;
        boolean b4=false;

        System.out.println("true^true= "+(b1^b3));
        System.out.println("true^false= "+(b1^b2));
        System.out.println("false^true= "+(b2^b3));
        System.out.println("false^false= "+(b2^b4));
    }
}
```

```
C:\Users\Ravi\Desktop>javac ss.java

C:\Users\Ravi\Desktop>java AB
true^true= false
true^false= true
false^true= true
false^false= false
```

Ex:-02

! මෙම සලකුණ මගින් ලැබෙන පිළිතුරේ විරුද්ධ අගය ලබාගත හැකිය.

```
    public static void main(String[] args) {

        boolean b1=true;
        boolean b2=!b1;
        System.out.println(b2);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_boolean_h.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
false

C:\Users\Ravi\Desktop\OOPC_I>
```

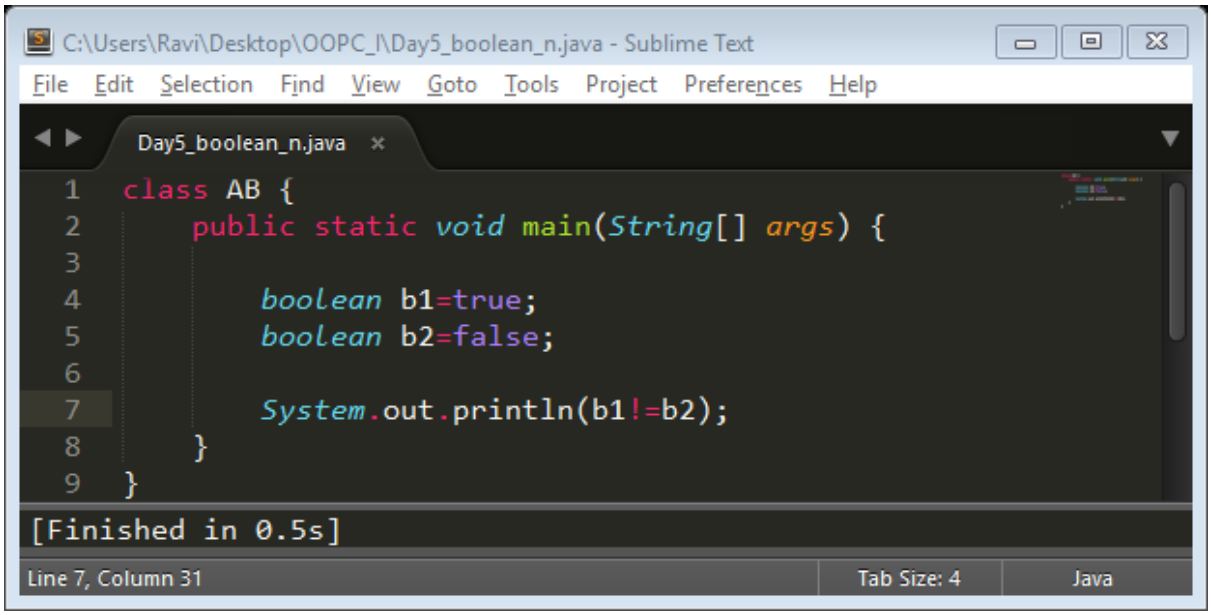
Assignment

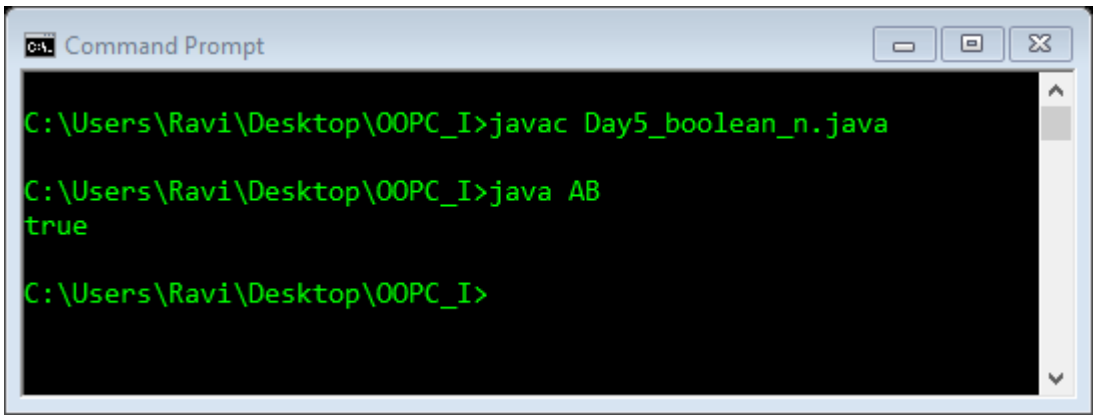
Ex:-01

```
class AB {
    public static void main(String[] args) {

        boolean b1=true;
        boolean b2=false;

        System.out.println(b1==b2);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_boolean_n.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
false

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-02





Short Circuit And / OR

- සොර්ට් සර්කිට් **&&** භවිථයෙදී සිදුවන්නේ පළමු අගය නිවවරදී නම් දෙවැනි පිළිතුර වෙත ගමන් කරයි. නමුත් පළමු පිළිතුර වැරදි නම් දෙවන අගය වෙත ගමන් නොකරයි.
- සොර්ට් සර්කිට් || භවිථයෙදී සිදුවන්නේ පළමු අගය නිවවරදී නම් දෙවැනි පිළිතුර වෙත ගමන් නොකර ප්ලිතුර සත්‍ය ලෙස ලබාදෙයි. නමුත් පළමු පිළිතුර වැරදි නම් දෙවන අගය වෙත ගමන් කර එය පරීක්ෂා කර බලා අවසන් පිළිතුර ලබාදෙයි.

Ex:-01

```
class AB {
    public static void main(String[] args) {

        boolean b1=true;
        boolean b2=true;

        System.out.println(b1&&b2);
        System.out.println(b1||b2);
    }
```

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_boolean_o.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
true
true
```

Ex:-02

C:\Users\Ravi\Desktop\OOPC_I\Day5_boolean_o.java - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

Day5_boolean_o.java

```
1  class AB {
2      public static void main(String[] args) {
3
4          boolean b1=false;
5          boolean b2=true;
6
7          System.out.println(b1&&b2);
8          System.out.println(b1||b2);
9      }
```

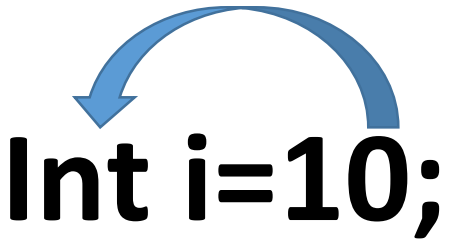
[Finished in 0.5s]

Line 7, Column 35 Tab Size: 4 Java

Command Prompt

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_boolean_o.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
false
true

C:\Users\Ravi\Desktop\OOPC_I>
```

Assignment Operator



Ternary Operator / Coordinal Operator

?

Expression1 ? expression2 : expression3

Ex:-01

Day5_boolean_condinal.java

```
class AB {
    public static void main(String[] args) {

        int i =100;
        int j=i<105?500:600;

        System.out.println(j);
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I>javac Day5_boolean_condinal.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
500

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-02

Day5_boolean_condinal.java

```
class AB {
    public static void main(String[] args) {

        int i =100;
        int j=i>105?500:600;

        System.out.println(j);
    }
}
```

Command Prompt

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
500

C:\Users\Ravi\Desktop\OOPC_I>javac Day5_boolean_condinal.java

C:\Users\Ravi\Desktop\OOPC_I>java AB
600

C:\Users\Ravi\Desktop\OOPC_I>
```

Ex:-02

Day5_boolean_condinal_2.java

```
1  class AB {
2      public static void main(String[] args) {
3
4          int i =160;
5          double d =(i<=10)&!(true&false)?30:20;
6
7          System.out.println(d);
8      }
9  }
```

```
C:\Users\Ravi\Desktop\OOPC_I>java AB
20.0

C:\Users\Ravi\Desktop\OOPC_I>
```

Instance operator

Without using a compound operator

Y=y-6;

X=x+2*5;

With compound operator

y -=6;

x +=2*5

Flow Control

මෙය කොටස් 2ක් යටතේ අධ්‍යනය කරයි. එනම්,

1. Selection
2. Iteration

Selection

මෙමගින් සිදුකනුයේ යම් දෙයක් පදනම් කරගනිමින් තීරන ලබාදෙමින් වැඩසටහනේ ගලයෑම වෙනස් කිරීමය.

if

```
if (condition)
    Statement;
```

පහතින් දක්වා ඇත්තේ if භාවිතයෙන් නිර්මාණය කර ඇති වැඩසටහන් කිහිපයකි.

Ex:-01

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(i<100)
        System.out.println("i is less than 100");
                    System.out.println("End");

    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 1.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is less than 100
End

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(true)
        System.out.println("i is less than 100");
                    System.out.println("End");

    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 1.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is less than 100
End

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-02

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(i>100)
        System.out.println("i is less than 100");
                    System.out.println("End");

    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
End

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex: 03

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=150;
        if(i>100)
        System.out.println("i is greatrt than 100");
                    System.out.println("ABCD");

    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 2.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is greatrt than 100
ABCD

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

සෑම flow control එකක්. මගින්ම පාලනය කළ හැක්කේ condition එයට ආසන්නවම පවතින, Statement එක පමණි.

නමුත් condition එක සඳහා {} යොදමින් statements ඇතුලත් කළ විට ඒ සියල්ලම පාලනය කළ හැක.

```
if (condition) {
        Statements;
}
Sop();
```

ඉහත නීතිය සෑම flow control එකක්සඳහම වලංගුය.

Ex:-04

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=150;
        if(i>100){
            //Statments
        System.out.println("i is greatrt than 100");
                    System.out.println("abcd");
                    System.out.println("ABCD");
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 3.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is greatrt than 100
abcd
ABCD

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-05

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(i>100)
            //Statments
        System.out.println("i is greatrt than 100");
                    System.out.println("abcd");
                    System.out.println("ABCD");
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
abcd
ABCD

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

If else

```
if (condition)
        Statement 1;
else
        statement 2;
```

මෙහිදී condition එක true අවස්ථාෙවිදී if යටතේ පවතින දෑ සිදු කරන අතර ,

Condition එක False අවස්ථාෙවිදී else යටතේ පවතින දෑ සිදු කරයි.

පහත නිදර්ශන මගින් එය දක්වා ඇත.

Ex:-01

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=150;
        if(i>100)
            //Statments
        System.out.println("i is greater than 100");
        else
            System.out.println("i is not greater than 100");

        System.out.println("End");
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is greater than 100
End

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-02

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(i>100)
            //Statments
        System.out.println("i is greater than 100");
        else
            System.out.println("i is not greater than 100");

        System.out.println("End");
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is not greater than 100
End

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-03

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(i>100)
            //Statments
            System.out.println("i is greater than 100");
            System.out.println("ABC");
        else
            System.out.println("i is not greater than 100");
            System.out.println("abc");
        System.out.println("End");
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 5.java"
new 5.java:9: 'else' without 'if'
                else
                ^
1 error

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-04

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(i>100){
            //Statments
            System.out.println("i is greater than 100");
            System.out.println("ABC");
        }
        else
            System.out.println("i is not greater than 100");
            System.out.println("abc");
        System.out.println("End");
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is not greater than 100
abc
End

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

smskslkslklas

EX:-5

Sdskdslkdlsdlsklkds

```
class ABC{
    public static void main(String[]args){
        System.out.println("Start");
        int i=10;
        if(i>100){
            //Statments
            System.out.println("i is greater than 100");
            System.out.println("ABC");
        }
        else{
            System.out.println("i is not greater than 100");
            System.out.println("abc");

        System.out.println("End");
    }
}}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Start
i is not greater than 100
abc
End

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

If එකට අදාලව else එක අත්‍යවශ්‍ය නොවේ. අපට අවශ්‍යනම් පමණයි if සමට else භාවිතා කරන්ෙන්.

නමුත් if නොමැතිව else භාවිතා කිරීමට නොමැතිය. එනම් else ට if අත්‍යවශය ෙවි.

If එකක් සඳහා else කිහිපයක් භාවිතා කළ ෙනාහැක. පහතින් දක්වා අැත්ෙත් ඒහි පිළිවලය.

```
If(){
}else{
}
```

```
If(){}
If(){
}
else {
}
```

```
If(){}
If(){
}else{}
if(){
}else{}
```

If else if

```
if (condition 1)
        Statement 1;
else if (condition 2)
        statement 2;
else if (condition 3)
        statement 3;
else
        statement 4;
```

```
if (condition 1){
        Statements;
}else if (condition 2){
        statements;
}else if (condition 3){
        statements;
}else{
        statements ; }
```

Ex:-01

```
class ABC{
    public static void main(String[]args){
        int i=30;
        if (i==100) {
            System.out.println("i equals to 10");
        }
        else if (i==20) {
            System.out.println("i equals to 20");
        }else if (i==30) {
            System.out.println("i equals to 30");

        }else if (i==40) {
            System.out.println("i equals to 40");
        }else if (i==50) {
            System.out.println("i equals to 50");
        }else{
            System.out.println("Not Above");
        }
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 8.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
i equals to 30

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

මෙයහිදී i සඳහා ලැබාෙදන අගය යම් if condition එකක් තුළ පවතීනම් ඊට අදාල වූ sout ඒක ලබාෙදයි.

I හි අගය ලැබෙන තෙක් සියලුම if conditions පිළිවලින් පරීක්ෂා කරනු ලබයි. ගැලෙපන කිසිම අගයක් ෙනාමැතිනම් else වෙත පැමිණ එහි පවතින කාර්ය සිදුකරයි. පහතින් එය දක්වා ඇත.

Ex:-02

```
lass ABC{
   public static void main(String[]args){
        int i=12;
        if (i==100) {
            System.out.println("i equals to 10");
        }
        else if (i==20) {
            System.out.println("i equals to 20");
        }else if (i==30) {
            System.out.println("i equals to 30");

        }else if (i==40) {
            System.out.println("i equals to 40");
        }else if (i==50) {
            System.out.println("i equals to 50");
        }else{
            System.out.println("Not Above");
        }
   }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 8.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Not Above

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-03

```
class ABC{
    public static void main(String[]args){
        int i=12;
        if (i==100) {
            System.out.println("i equals to 10");
        }
        else if (i==20) {
            System.out.println("i equals to 20");
        }else if (i==30) {
            System.out.println("i equals to 30");

        }else if (i==40) {
            System.out.println("i equals to 40");
        }else if (i==50) {
            System.out.println("i equals to 50");
        }else{
            System.out.println("Not Above");
        }
        System.out.println("Programe End...!");
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Not Above
Programe End...!

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

- කිසිම අවස්ථාවක **else condition** එක වැඩසටහන මැදදී තැබිය නොහැක. සෑම විටම එය තැබිය යුත්තේ වැඩසටහන අවසානයේය. අවශ්‍ය නම් එය නොදා සිටිය හැකිය

Nested Flow Controls

Ex:-01

```
class ABC{
    public static void main(String[]args){
        int i,j;
        i=10;   j=20;

        if (i==10) {
            if (j==0) {
                System.out.println("i==10,j==0");

            }else if (j==20) {
                System.out.println("i==10,j==20");
            }else {
                System.out.println("i==10,j ???");
            }
        }else {
            if (i==20 && j==20) {
                System.out.println("i==20 && j==20");
            }
        }
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 10.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
i==10,j==20

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-02

```
class ABC{
    public static void main(String[]args){
        int i,j;
        i=20;    j=20;

        if (i==10) {
            if (j==0) {
                System.out.println("i==10,j==0");

            }else if (j==20) {
                System.out.println("i==10,j==20");
            }else {
                System.out.println("i==10,j ???");
            }
        }else {
            if (i==20 && j==20) {
                System.out.println("i==20 && j==20");
            }
        }
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 10.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
i==20 && j==20

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Switch Case

```
switch (expression){
        case  1 : statement 1;
        case  2 : statement 2;
        case  3 : statement 3;
        case  4 : statement 4;
        case  n : statement n;
        default : statement default;
}
```

EX:-01

```
class ABC{
    public static void main(String[]args){
        int x=1;
        switch(x){
            case 1: System.out.println("i equals to 1");
            case 3: System.out.println("i equals to 3");
            case 5: System.out.println("i equals to 5");
            default: System.out.println("default");
            case 2: System.out.println("i equals to 2");
            case 4: System.out.println("i equals to 4");
        }
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
i equals to 1
i equals to 3
i equals to 5
default
i equals to 2
i equals to 4

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-02

```
class ABC{
    public static void main(String[]args){
        int x=4;
        switch(x){
            case 1: System.out.println("i equals to 1");
            case 3: System.out.println("i equals to 3");
            case 5: System.out.println("i equals to 5");
            default: System.out.println("default");
            case 2: System.out.println("i equals to 2");
            case 4: System.out.println("i equals to 4");
        }
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
i equals to 4

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-03

```
class ABC{
    public static void main(String[]args){
        int x=10;
        switch(x){
            case 1: System.out.println("i equals to 1");
            case 3: System.out.println("i equals to 3");
            case 5: System.out.println("i equals to 5");
            default: System.out.println("default");
            case 2: System.out.println("i equals to 2");
            case 4: System.out.println("i equals to 4");
        }
    }}
```

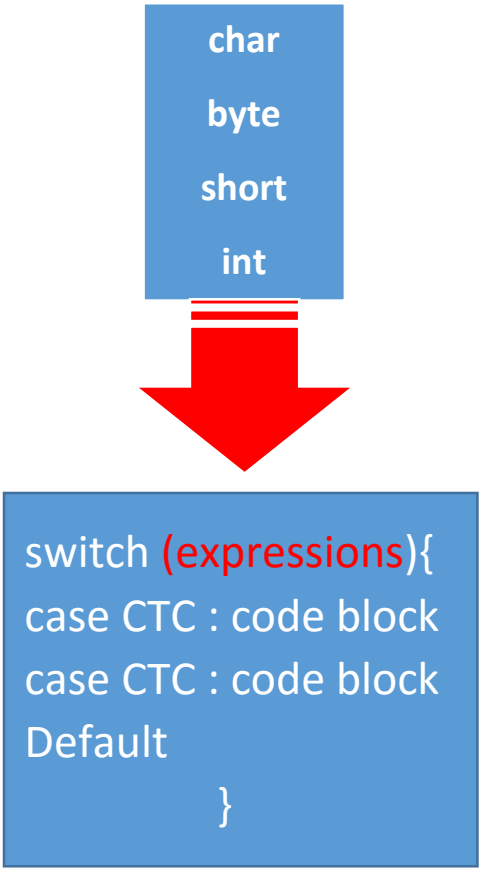
```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 11.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
default
i equals to 2
i equals to 4

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-04

```
class ABC{
    public static void main(String[]args){
        int x=3;
        switch(x){
            case 1: System.out.println("i equals to 1");
            break;
            case 3: System.out.println("i equals to 3");
            break;
            case 5: System.out.println("i equals to 5");
            break;
            case 2: System.out.println("i equals to 2");
            break;
            case 4: System.out.println("i equals to 4");
            break;
            default: System.out.println("default");
        }
    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
i equals to 3

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Legal Expressions For Switch And Case



Ex:-05

```
class ABC{
    public static void main(String[]args){
        int x =10776;
        switch(x%4){
            case 0 :System.out.println("Olu");
            break;
            case 1 :System.out.println("Nelum");
            break;
            case 2 :System.out.println("Manel");
            break;
            case 3 :System.out.println("Kumudu");
            break;
        }

    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Olu

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-06

```
class ABC{
    public static void main(String[]args){
        char ch ='A';
        switch(ch){
            case 'A' :System.out.println("Olu");
            break;
            case 1 :System.out.println("Nelum");
            break;
            case 2 :System.out.println("Manel");
            break;
            case 3 :System.out.println("Kumudu");
            break;
        }

    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Olu

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-07

```
class ABC{
    public static void main(String[]args){
        char ch ='A';
        switch(ch){
            case 65 :System.out.println("Olu");
            break;
            case 1 :System.out.println("Nelum");
            break;
            case 2 :System.out.println("Manel");
            break;
            case 3 :System.out.println("Kumudu");
            break;
        }

    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 13.java"

C:\Users\Ravi\Desktop\OOPC_I - __B>java ABC
Olu

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Ex:-08

```
class ABC{
    public static void main(String[]args){
        char ch ='A';
        switch(ch){
            case 65 :System.out.println("Olu");
            break;
            case 'A':System.out.println("Nelum");
            break;
            case 2 :System.out.println("Manel");
            break;
            case 3 :System.out.println("Kumudu");
            break;
        }

    }}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __B>javac "new 13.java"
new 13.java:7: duplicate case label
                        case 'A':System.out.println("Nelum");
                        ^
1 error

C:\Users\Ravi\Desktop\OOPC_I - __B>
```

Switch case වලදී string ඇල්ලිය හැකකේ jdk 1.7 ගේ සිට ඉහල වර්ෂන් වලදීය.

Iteration

While

```
while (condition){

      statements;

}
```

මෙම while loop එකකදී සිදු වන්නේ එකම දේ නැවත නැවත සඳහු කිරීමය.

මෙහිදී condition එක වෙන පැමින එය සත්‍ය නම් {} අැති statements සියල්ල සුදුකර නැවති condition ඒක වෙන ගමන් කර නැවත condition true නම් නැවත while loop ඒක ක්‍රියාත්මක ෙවී.

ෙමම ක්‍රියාව condition එක false වන තෙක්ම සිදු වේ.

Ex:-01

```
class FlowC{
    public static void main(String []args){
        int x=0;
            while(x<=100){
            System.out.println("x="+x);
            x++;
        }

    }
}
```

ඉහත නිදසුනෙන් සුදුවන්නේ, 0 සිට 100 දක්වා සියලුම පූර්ණ සංඛ්‍යාවන් print කිරීමය.

Ex:-02

```
class FlowC{
    public static void main(String []args){
        int x=0;
            while(x<=50){
            System.out.println("x="+x);
            x=x+2;// x+=2 or
        }

    }
}
```

```
class FlowC{
    public static void main(String []args){
        int x=0;
            while(x<=50){
            if(x%2==0){
                System.out.println("x="+x);
            }
            x++;
        }}}
```

ඉහත නිදසුනෙන් සුදුවන්නේ, 0 සිට 50 දක්වා වූ 2න් ඉතිරි නැතිව බෙදෙන සියලුම පූර්ණ සංඛ්‍යාවන් print කිරීමය.

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac new4.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java FlowC
x=0
x=2
x=4
x=6
x=8
x=10
x=12
x=14
x=16
x=18
x=20
x=22
x=24
x=26
x=28
x=30
x=32
x=34
x=36
x=38
x=40
x=42
x=44
x=46
x=48
x=50

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-03

```
class FlowC{
    public static void main(String []args){
        int x=10;
            while(x<20)
            System.out.println("Hi");
            System.out.println("Hi");
        }}
```

මෙහිදී hi යන්න පමණක් නොනැවතී print වේ. ඉහත අවස්ථානකදී while loop ඒකට වුවද {} නොමැති විට condition එකට ආසන්නම executable Line එක පමණක් while Loop එකට අනුකූලව ක්‍රියාත්මක වේ.

```
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
Hi
```

Ex:-04

```
class FlowC{
    public static void main(String []args){
        int x=10;
            while(x<20){
            System.out.println("Hi");
            System.out.println("Bye");
}}}
```

මෙහිදී hi සහ bye යන 2ම නොනැවතී print වේ.

```
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
```

Ex:-05

```
class FlowC{
    public static void main(String []args){
        int x=10;
            while(true){
            System.out.println("Hi");
            System.out.println("Bye");
}}}
```

```
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
```

Ex:-06

```
1  class FlowC{
2      public static void main(String []args){
3          int x=10;
4              while(false){
5              System.out.println("Hi");
6              System.out.println("Bye");
7  }}}
```

```
Command Prompt

C:\Users\Ravi\Desktop\OOPC_I - __C>javac new7.java
new7.java:4: unreachable statement
                            while(false){
                                         ^
1 error

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-07

```
class FlowC{
    public static void main(String []args){
        int x=10;
            while(true){
            System.out.println("Hi");
            System.out.println("Bye");
}
        System.out.println("End");
}}
```

මෙහිදී ස්ථීර ලෙසම while loop එක true බැවින්, කිසීම විටෙක end සඳහන් sop එක වෙත ළඟා විය නොහැක.

පහත පරිදි එවිට error එකක් ලෙ ස එය පෙන්වාදෙයි.

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac new8.java
new8.java:8: unreachable statement
                        System.out.println("End");
                        ^
1 error

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-08

```
class FlowC{
    public static void main(String []args){
        int x=10;
            while(false){
            System.out.println("Hi");
            System.out.println("Bye");
}
        System.out.println("End");
}}
```

මෙහිදී ස්ථීර ලෙසම while loop එක false බැවින්, කිසීම විටෙක while loop එක ඇතුලට ගමන් කළ නොහැක. ඒ නිසා while loop එකේ {} (condition) erroe ෙපන්වයි.

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac new8.java
new8.java:4: unreachable statement
                                while(false){
                                ^
1 error

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-09

```
class FlowC{
    public static void main(String []args){
        boolean b=true;
                while(b){
                System.out.println("Hi");
                System.out.println("Bye");
}
            System.out.println("End");
}}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac new8.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java FlowC
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
Bye
Hi
```

Ex:-10

```
class FlowC{
    public static void main(String []args){
        boolean b=false;
                while(b){
                System.out.println("Hi");
                System.out.println("Bye");
}
            System.out.println("End");
}}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac new7.java
new7.java:4: unreachable statement
                                while(false){
                                             ^
1 error

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

do while

```
do{
statements;
    }
while(condition);
```

මෙහිදී පළමුව do යටතේ පවතින statements සිදු කරීම සඳහා condition එකේ ප්‍රථිපළය බල නොපායි.

එනම් condition පරීක්ෂා කරනුයේ පළමු වතාවට do යටතේ පවතින කොටස සිදු කිරීමෙන් අනතුරුවය.

එවිට condition ඒක false නම් නැවත do වෙත ගමන් කරනු නොලබයි.

True නම් නැවත do වෙත ගමන් කර ඒහි අඩංගු කාර්යයන් සිදුකරයි.

Ex:-01

```
class DoWhile{
    public static void main(String []args){
        int x=300;
        do{
            System.out.println(x);
            x-=1;
        }while(x>=100);
    }
}
```

```
258
257
256
255
254
253
252
251
250
249
248
247
246
245
```

මෙහිදී සිදු වන්නේ 300 සිට 100 දක්වා වූ සියලුම සංඛ්‍යා 1 බැගින් අඩු වෙමින් පසු පසට print වීමය.

Ex:-02

```
class DoWhile{
    public static void main(String []args){
        int x=500;
        do{
            if(x%3==0 && x%5==0){
                System.out.println(x);
            }
            x-=1;
        }while(x>=200);

    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac Check.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java DoWhile
495
480
465
450
435
420
405
390
375
360
345
330
315
300
285
270
255
240
225
210

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

මෙහිදී සිදු වන්නේ 500 සිට 200 දක්වා වූ සියලුම 3න් සහ 5න් බෙදෙන පසු පසට print වීමය.

& හෝ && යන දෙකම භාවිතා කළ හැක. && යෙදීම මගින් program ඒක ෙවිගවත් ෙව්.

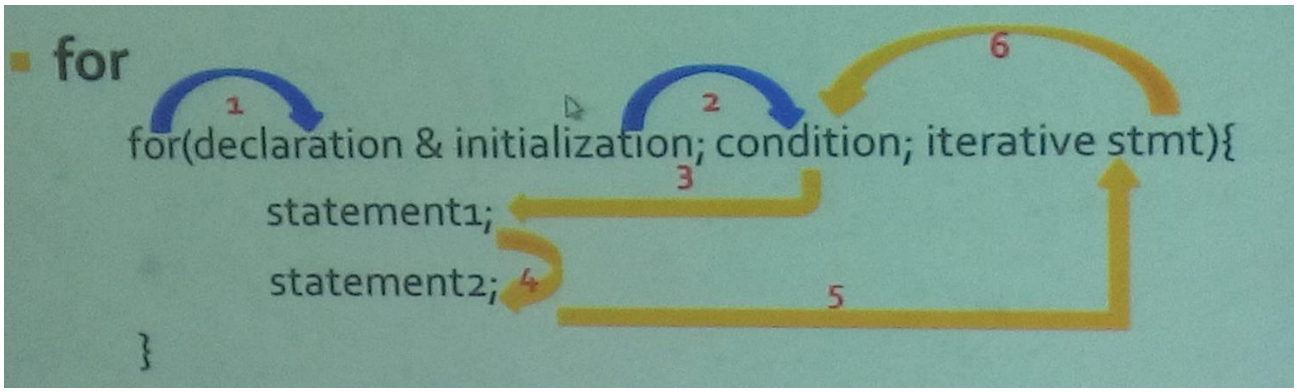
Ex:-03

```
class DoWhile{
    public static void main(String []args){
        do{
            System.out.println("A");
        }while(true);
        System.out.println("B");
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac Check.java
Check.java:6: unreachable statement
                    System.out.println("B");
                    ^
1 error

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

for

```
for (declaration & initialization ; condition; iterative stmt) {
        statement1;
        statement1;
}
```



Ex:-01

```
class ForLoop{
    public static void main(String []args){
        for(int x=0;x <=100;x++){
            if(x%2==1){
            System.out.print("Value of x is:=");
            System.out.println(x);
            }
        }
    }
}
class ForLoop{
    public static void main(String []args){
        for(int x=1;x <=100;x+=2){
            System.out.print("Value of x is:=");
            System.out.println(x);
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac For_2.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java ForLoop
Value of x is:=1
Value of x is:=3
Value of x is:=5
Value of x is:=7
Value of x is:=9
Value of x is:=11
Value of x is:=13
Value of x is:=15
Value of x is:=17
Value of x is:=19
Value of x is:=21
Value of x is:=23
Value of x is:=25
Value of x is:=27
Value of x is:=29
Value of x is:=31
Value of x is:=33
Value of x is:=35
Value of x is:=37
Value of x is:=39
Value of x is:=41
Value of x is:=43
Value of x is:=45
Value of x is:=47
Value of x is:=49
Value of x is:=51
```

Ex:-02

```
class ForLoop{
    public static void main(String []args){
        for(int x=10;x>0;x--){
            for(int y=1;y<x;y++){
            System.out.print("*");
            }
            System.out.println("*");
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac For_6.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java ForLoop
**********
*********
********
*******
******
*****
****
***
**
*
```

Ex:-03_Infinite For Loop

```
class ForLoop{
    public static void main(String []args){
        for(;;){

            System.out.print("X");
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>java ForLoop
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXX
C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-04

```
class ForLoop{
    public static void main(String []args){
        int i;
        int j;
        for(i=10;i<20;i++){

            System.out.println(i);
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>java ForLoop
10
11
12
13
14
15
16
17
18
19

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-05

```
class ForLoop{
    public static void main(String []args){

        for(int i=5,j=10;i<20;i++){
            System.out.println(i);
            System.out.println(j);
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>java ForLoop
5
10
6
10
7
10
8
10
9
10
10
10
11
10
12
10
13
```

break & Continue

break

Used to stop the entire loop

- ***Inside either a loop or*** switch ***statement.***

Ex:-01

```
class ForLoop{
    public static void main(String []args){
        for(int x=0;x<10;x++){
            if(x==5) break;
            System.out.print(x+" ");
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac Check.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java ForLoop
0 1 2 3 4
C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-02

අවශ්‍යතාවය මත flow controls සඳහා label name ලබාදිය හැක. පහතින් දක්වා ඇත්තේ එයයි.

මෙහිදී while Condition එක true නමුත් , for loop එක නම් කර ඇති label1 break කිරීම නිසා සම්පූර්ණම for loop එක නවති.

```
class Flow_Label{
    public static void main(String[]args){
        lable1: for(int i=0;i<5;i++){
            int y=0;
            while(y<9){
                if(y==6){
                    break lable1;
                }
                System.out.println("Inside While" +y);
                y++;
            }
        }
    }
}
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac For_7.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java Flow_Label
Inside While0
Inside While1
Inside While2
Inside While3
Inside While4
Inside While5

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Ex:-03

මෙහිදී නවතිනුයේ while Loop එක පමණි. ඒ නිසා පහත out put ලැබේ.

```
class Flow_Label{
    public static void main(String[]args){
        lable1: for(int i=0;i<5;i++){
            int y=0;
            while(y<9){
                if(y==6){
                    break ;
                }
                System.out.println("Inside While" +y);
                y++;
            }
        }
    }
}
```

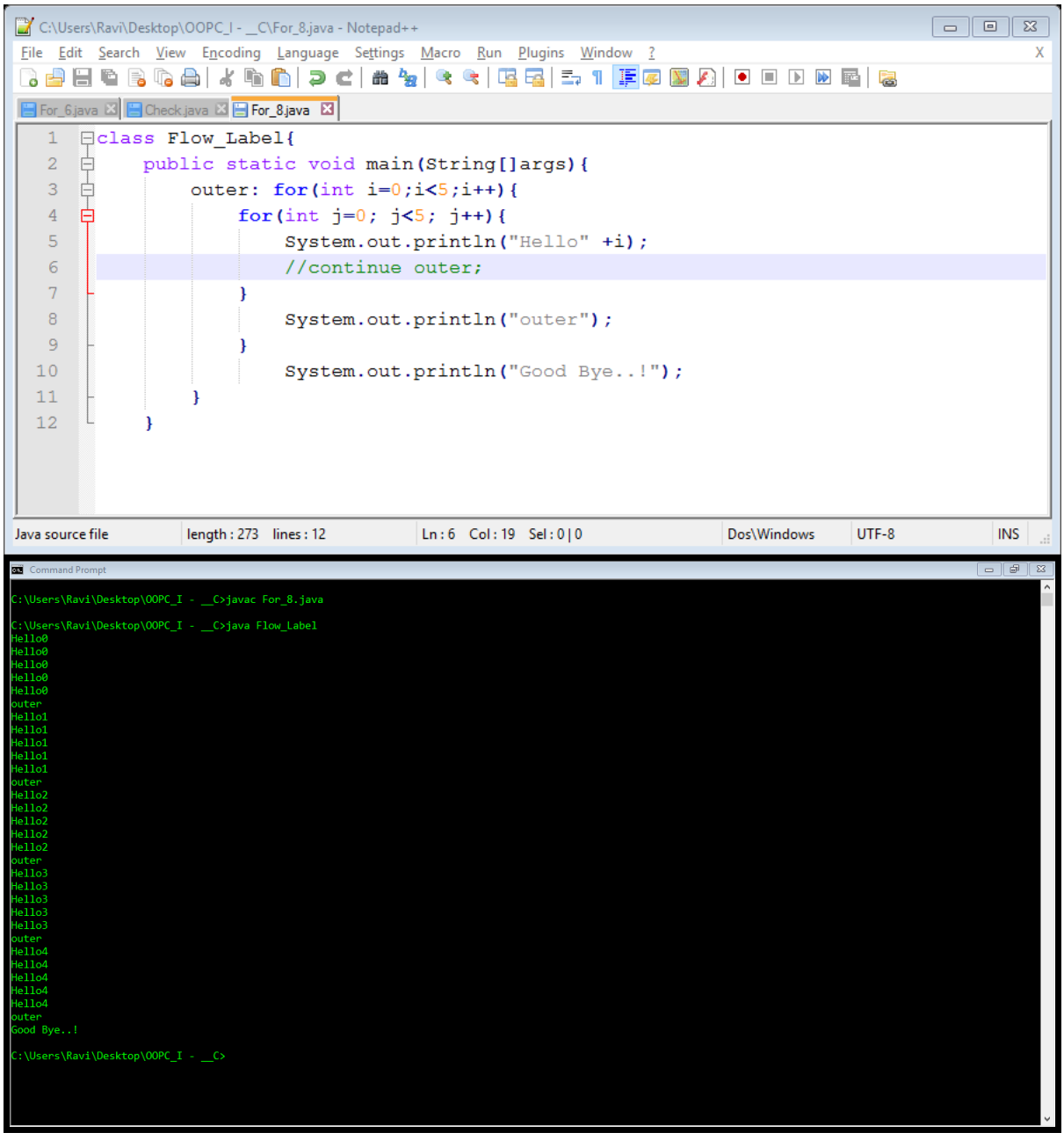
```
C:\Users\Ravi\Desktop\OOPC_I - __C>java Flow_Label
Inside While0
Inside While1
Inside While2
Inside While3
Inside While4
Inside While5
Inside While0
Inside While1
Inside While2
Inside While3
Inside While4
Inside While5
Inside While0
Inside While1
Inside While2
Inside While3
Inside While4
Inside While5
Inside While0
Inside While1
Inside While2
Inside While3
Inside While4
Inside While5
Inside While0
Inside While1
Inside While2
Inside While3
Inside While4
Inside While5

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

continue

used to stop just the current iteration

- must be inside a loop

Ex:-01



නමුත් ඉහත code සඳහාම continue යන keyword එක භාවිතා කරමින්, ක්‍රියාත්මක වෙමින් පවතින කොටස කනවතා අනිත් කොටසට ගමන් කළ හැක. පහතින් දක්වා ඇත්තේ ඒ සඳහා නිදසුනකි.

Ex:-02

```
class Flow_Label{
    public static void main(String[]args){
        outer: for(int i=0;i<5;i++){
            for(int j=0; j<5; j++){
                System.out.println("Hello" +i);
                continue outer;
            }
                System.out.println("outer");
            }
                System.out.println("Good Bye..!");
        }
    }
```

```
C:\Users\Ravi\Desktop\OOPC_I - __C>javac For_8.java

C:\Users\Ravi\Desktop\OOPC_I - __C>java Flow_Label
Hello0
Hello1
Hello2
Hello3
Hello4
Good Bye..!

C:\Users\Ravi\Desktop\OOPC_I - __C>
```

Object Orientation

What is an object..?

Object එකක් වශයෙන් ඇසට පෙනෙන ඉස්ස්ලුම දේ හැදින්විය හැක.

මේවා feature සහ behaviors වශයෙන් කොටස් දෙකක් යටතේ සකච්ඡා කරයි.

Example For OOP

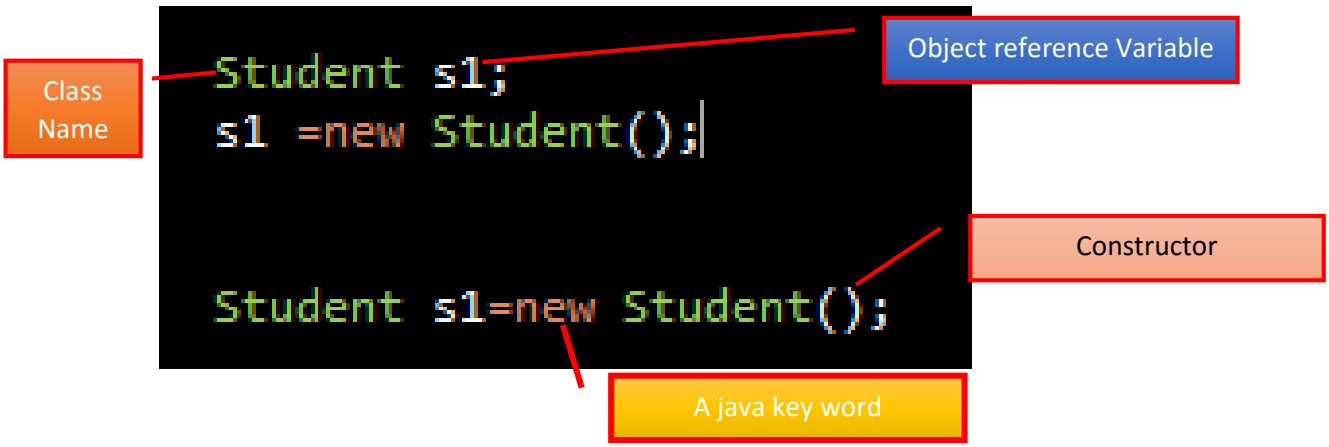
Ex:-1

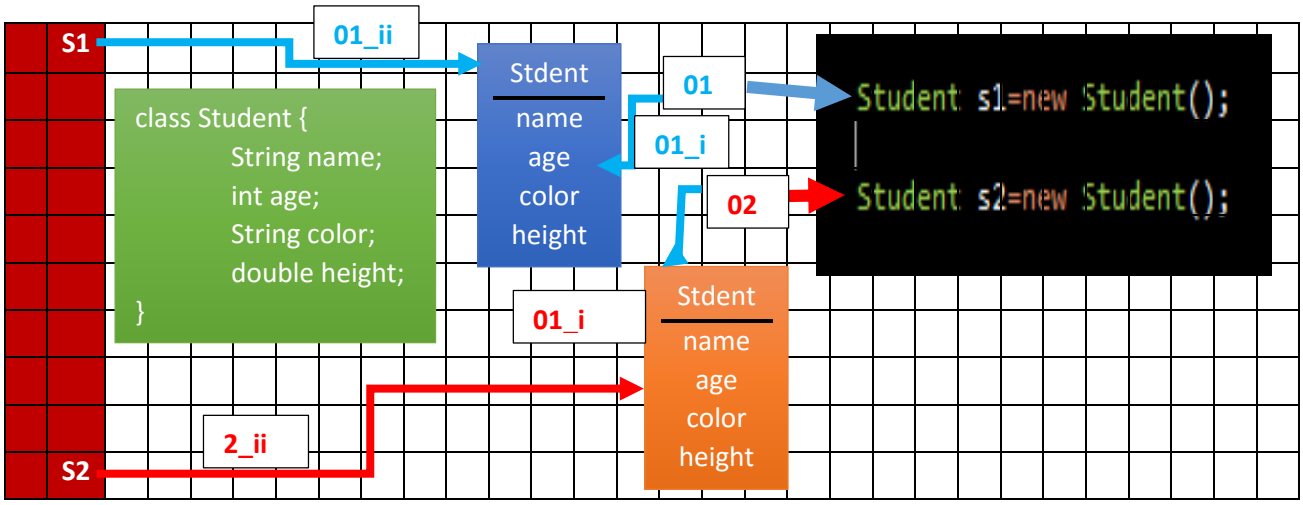
Stack	Heap
S1 →	name age color height

```
class Student {
    String name;
    int age;
    String color;
    double height;
}
class CreateStudent{
    public static void main(String[] args) {
        Student s1=new Student();

    }
}
```

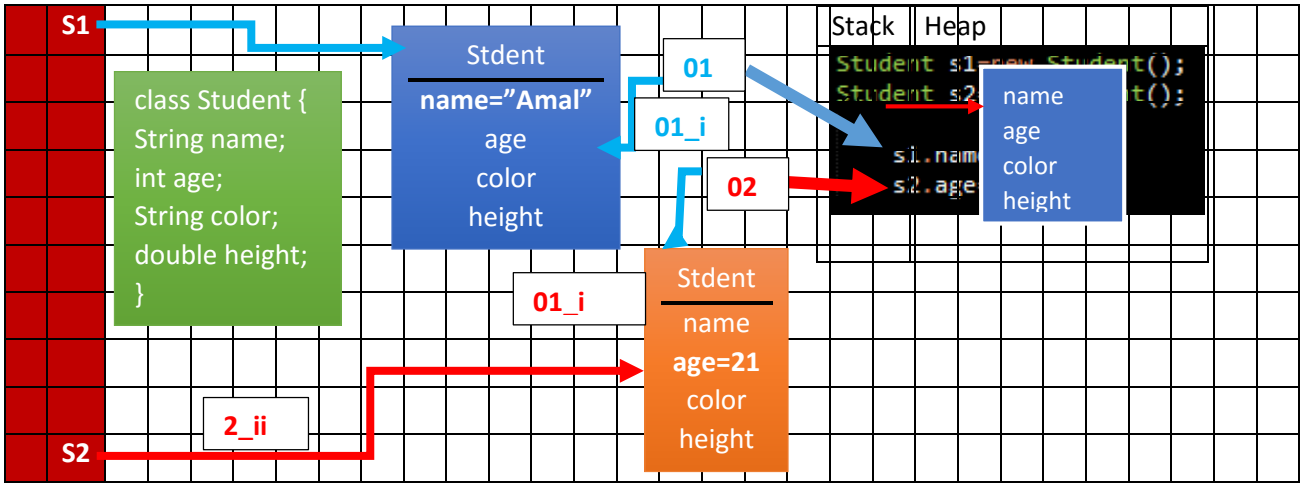
Ram තුල Object එකක් නර්මාණය වන විට නිශ්චිත ස්ථානයක් නොගනී.

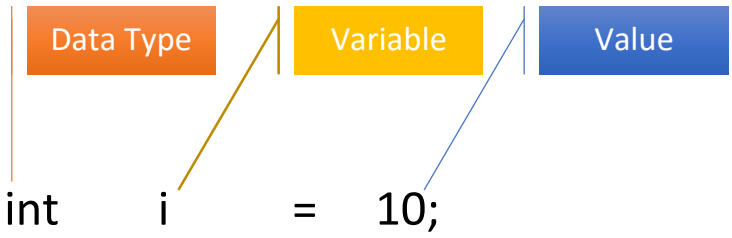
- new යන keyword එක ම.ගින් සිදුකරනුයේ ram එකේ අලුතින් ඉඩක් වෙන්කර ගැනීමයි.

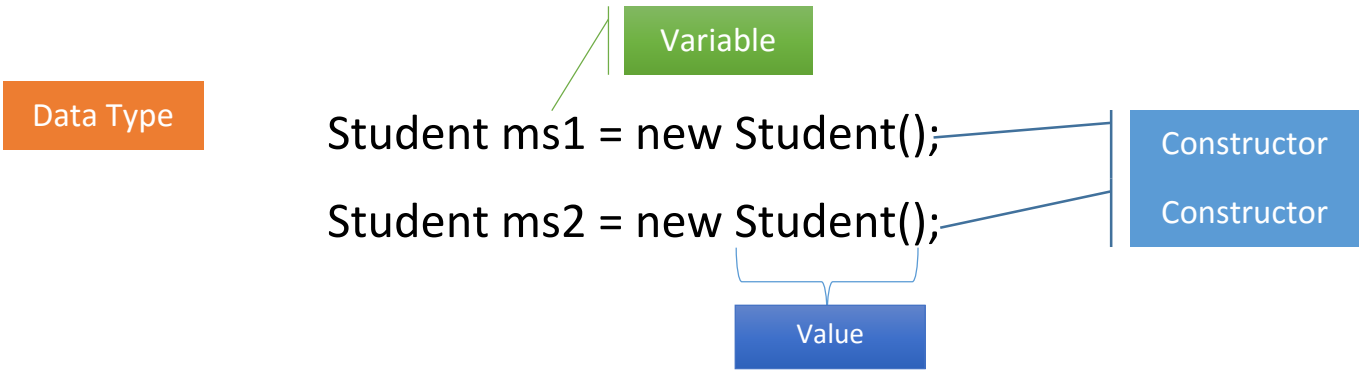




Ex:-02







Variable Type

Variable වර්ග 3ක් පවතින අතර එය පහත පරිදි දැක්විය හැකිය.

1. Instance Variable
2. Static Variable/Global Variable
3. Local Variable
    a. Method Local Variable
    b. Block Variable

Instance Variable

මෙම විචල්‍ය වර්ගය හඳුනා ගැනීමට පහසුම ආකාරය වනුයේ ඒවා පිහ්ටන්නේ class {} තුල පමනක් බැවින්ය.

```
class A {
    double d;
    String s;
    int i;
    char c;
}
```

class එකක පවතින instance variable PSVM එකේ සිට call කරන විට නොලේබෙන අතර , ඒ අදාල class එකේ නමින් object එකක් නර්මාණය කර call කළ හැක.

Ex:-01

```
class A {
    double d;
    String s;
    int i;
    char c;

    public static void main(String[] args) {
        A a1=new A();
    System.out.println("hi");
    System.out.println(a1.d);
    System.out.println(a1.s);
    System.out.println(a1.i);
    System.out.println(a1.c);
}
}
```

Ex:-02

```
class A {
    double d;
    String s;
    int i;
    char c;
}
class B{
    public static void main(String[] args) {
    System.out.println(i);
}}
```

```
C:\Users\Ravi\Desktop\Day10>javac MyOopPro_3.java
MyOopPro_3.java:10: cannot find symbol
symbol  : variable i
location: class B
        System.out.println(i);
                           ^
1 error
```

මෙම අවස්ථාවෙදී i සොයා ගැනිමට නොහැකි බව error 1ක් මගින් දැනුම් දෙයි.

Ex:-03

```
class A {
    double d;
    String s;
    int i=10;
    char c;
}
class B{
    public static void main(String[] args) {
        A a1 = new A();
    System.out.println(a1.i);
}}
```

```
C:\Users\Ravi\Desktop\Day10>java B
10
```

මෙම ක්‍රමය අනුගමනය කරමින් අවශ්‍ය ප්‍රතිපලය ලබාගත හැක.

EX;-04

```
class A {

    byte b1;    short s1;   int i1;     long l1;
    float f1;   double d1;  char c1;    boolean b2;

    String s2;  C c2;
}
class B{
    public static void main(String[] args) {
        A a1 = new A();
    System.out.println(a1.b1);
    System.out.println(a1.s1);
    System.out.println(a1.i1);
    System.out.println(a1.l1);
    System.out.println(a1.f1);
    System.out.println(a1.d1);
    System.out.println(a1.c1);
    System.out.println(a1.b2);
    System.out.println(a1.s2);
    System.out.println(a1.c2);

}}

class C{}
```

```
C:\Users\Ravi\Desktop\Day10>javac MyOopPro_4.java

C:\Users\Ravi\Desktop\Day10>java B
0
0
0
0
0.0
0.0

false
null
null
```

යම් variable එකක් සඳහා අගයක් ආදේශ කිරිඉම්කින් තොර වූ විට එහි default Value එක එයට ආදේශ කරගනී.

Object type එකක වුව ද default value එක null වේ. එනම් කිසිවක් නොමැත යන්නයි.

Ex:-05

```
class X {

    byte b=10; int i; boolean b1; Z z1;
}
class Y{
    public static void main(String[] args) {
        X a1 = new X();
    System.out.println(a1.b);
    System.out.println(a1.i);
    System.out.println(a1.b1);
    System.out.println(a1.z1);
}}
```

```
C:\Users\Ravi\Desktop\Day10>javac MyOopPro_5.java
MyOopPro_5.java:3: cannot find symbol
symbol  : class Z
location: class X
        byte b=10; int i; boolean b1; Z z1;
                                      ^
1 error
```

Ex:-06

```
class X {

    byte b=10; int i; boolean b1; Z z1=new Z();
}
class Y{
    public static void main(String[] args) {
        X a1 = new X();
        System.out.println(a1);
    System.out.println(a1.b);
    System.out.println(a1.i);
    System.out.println(a1.b1);
    System.out.println(a1.z1);
}}
class Z{}
```

```
C:\Users\Ravi\Desktop\Day10>java Y
X@697eb767
10
0
false
Z@7e3b014c
```

Stack	Heap
m1 →	M I → 10 20 x → 0 50

Ex:-07

```
class M{
    int i=10; byte x;
}
class N{
    public static void main(String[] args) {
        M m1=new M();

        System.out.println(m1.i);
        System.out.println(m1.x);

        m1.i=20;    m1.x=50;

        System.out.println(m1.i);
        System.out.println(m1.x);
    }
}
```

```
C:\Users\Ravi\Desktop\Day10>javac MyOopPro_6.java

C:\Users\Ravi\Desktop\Day10>java N
10
0
20
50
```

Ex:-08

```
class M{
    int i=10; byte x;
}
class N{
    public static void main(String[] args) {
        M m1=new M();

        System.out.println(m1.i);
        System.out.println(m1.x);

        m1.i=20;    m1.x=50;

        System.out.println(m1.i);
        System.out.println(m1.x);

        M m2=new M();

        System.out.println(m2.i);
        System.out.println(m2.x);
    }}
```

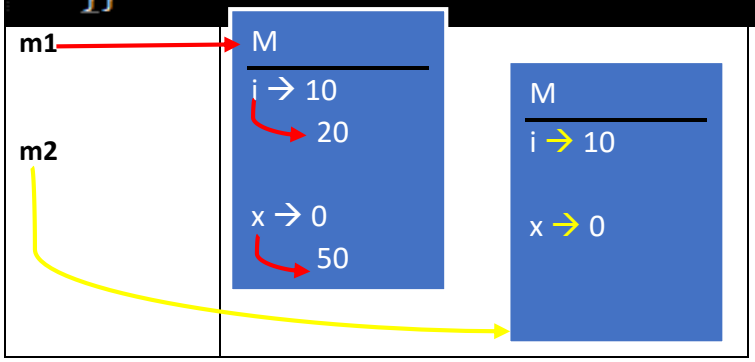
```
C:\Users\Ravi\Desktop\Day10>java N
10
0
20
50
10
0
```

Stack	Heap
a1 →	b→10 i→0 b1→ false z1→ Z

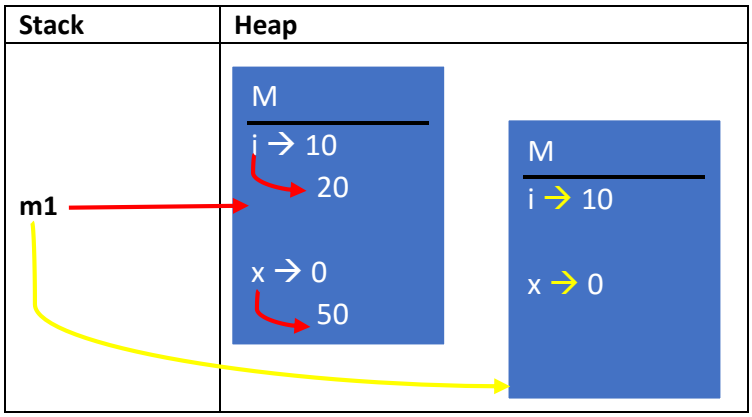
Ex:-09

```
class M{
    int i=10; byte x;
}
class N{
    public static void main(String[] args) {
        M m1=new M();
            System.out.println(m1);
            System.out.println(m1.i);
            System.out.println(m1.x);
        m1.i=20;     m1.x=50;
            System.out.println(m1.i);
            System.out.println(m1.x);
        m1=new M();
            System.out.println(m1);
            System.out.println(m1.i);
            System.out.println(m1.x);
    }}
```

```
M@44d79c75
10
0
20
50
M@2760e8a2
10
0
```



මෙහිදී object 2කේ memory location print කළ විට ලැබෙනුයේ address 2කි. එමගින් තහවුරු වන්නේ අලුතින් ඔබ්ජෙක්ට් එකක් නිර්මානය වූ බවය.



Ex:-10

මෙහිදී object 1ක සඳහා අගක් ලැබී නොමැති විට එය null ලෙස පෙන්වන අතර අගයක් ලැබුන විට memory location address 1ක ලැබෙ.

```
class M{
    int i=10; byte x;
    String s="ABCD";

    G g1=new G();
}
class N{
    public static void main(String[] args) {
        M m1=new M();
            System.out.println(m1.s);
            System.out.println(m1.g1);
    }}
class G{}
```

```
C:\Users\Ravi\Desktop\Day10>javac MyOopPro_8.java

C:\Users\Ravi\Desktop\Day10>java N
ABCD
G@41675ec4
```

Ex:-11

```
class M{
    int i=10; byte x;
    String s="ABCD";

    G g1=new G();
}
class N{
    public static void main(String[] args) {
        M m1=new M();
            System.out.println(m1.g1.myInt);
            System.out.println(m1.g1.myString);
    }}
class G{
    int myInt;
    String myString;
}
```

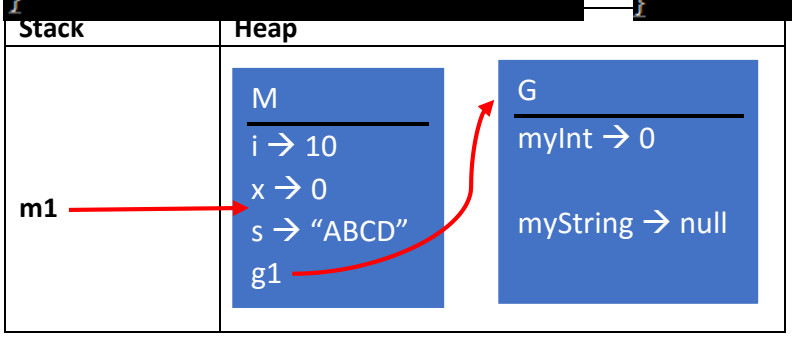
```
C:\Users\Ravi\Desktop\Day10>java N
0
null
```

g1 යන ඔබ්ජෙට් විචල්‍ය මගින් G වර්ගයේ ඔබ්ජෙක්ට් එකක් නර්මාණය කරයි.

➢ **Instance Variable** සඳහා අගයන්ලබාදීමේ දී **Declaration & Initiation** එකම පෙලියක්දී සිදු කළ යුතුය නැතහොත් අගය ලබාදීමෙන් වැලකිය යුතුය. පේලි දෙකකදී සිදුකළ නොහැක.

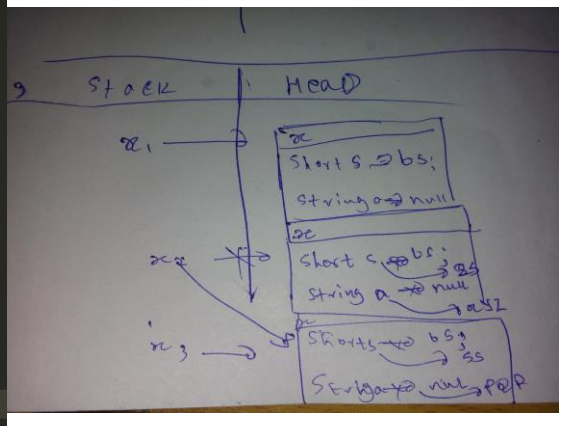
```
class A{
        int a ; // Correct
        int a1=10; //correct
    public static void main(String[] args) {
        long mylong;
        mylong=226562;
    }
    char c1;

}
```

```
class A{
        int a ; // Correct
        a=10; //incorrect
    public static void main(String[] args) {
        long mylong;
        mylong=226562;
    }
    char c1;

}
```



Ex:-12

```
class X{short s=65; String a;}
class Y{
    public static void main(String []args){
        X x1=new X();
        X x2=new X();
        X x3=new X();
            x2.s=25;
            x2.a="XYZ";
        x3.s=55;
        x3.a="PQR";
            x2=x3;
        System.out.println(x2);
        System.out.println(x2.s);
        System.out.println(x2.a);
            System.out.println(x3);
            System.out.println(x3.s);
            System.out.println(x3.a);
            }   }
```



Static Variable / Global Variable

- මෙම විචල්‍ය වර්ගයද පවතින්නේ class {} තුලය. මෙය පහසුවෙන් හඳුනා හැක්කේ එම විචල්‍ය ඉදිරියේ **static** යන වචනය යොදා ඇත.

```
class A{
    static int i;
    static String s;
}
```

- **Static / Global Variable** සඳහා අගයන් ලබාදීමේදී **Declaration & Initiation** එකම පෙලියක්දී සිදු කළ යුතුය. නැතහොත් අගය ලබාදීමෙන් වැලකිය යුතුය.පේලි දෙකකදී සිදුකළ නොහැක.

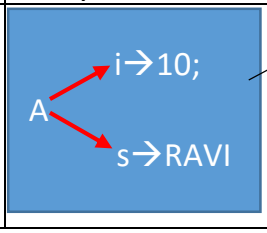
```
class A{
    static int i=10;
    static String s="RAVI";
}
```

- වෙනත් class එකක පවතින විචල්‍ය වැඩසටහනේ ඕනෑම ස්ථානයක සිට call කළ හැක.

```
class A{
    static int i=10;
    static String s="RAVI";
}
class B{
    public static void main(String[] args) {
        System.out.println(A.i);
        System.out.println(A.s);
}
}
```

```
C:\Users\Ravi\Desktop\Day12>java B
10
RAVI

C:\Users\Ravi\Desktop\Day12>
```

මෙහිදී ram එක තුල මෙම විචල්‍ය ගබඩා වනුයේ class එකේ නමින්ය.

Stack	Heap
	

Static Area

- class {} තුල එකම විචල්‍ය නැවත නර්මාණය කළ නොහැක. එකිනෙකට වෙනස් විචල්‍ය අවශ්‍ය ප්‍රමාණයක්නර්මාණය කරගත හැක.

```
class A{
    static int i=10;
    static String s="RAVI";
    int i;
}
```

```
C:\Users\Ravi\Desktop\Day12>javac Day12_1.ja
Day12_1.java:4: i is already defined in A
        int i;
            ^
1 error
```

```
class A{
    static byte b;      static short s;
    static int i;       static long l;
    static float f;     static double d;
    static char c;      static boolean b1;
    static C c1;
}
class C{}
class B{
    public static void main(String[] args) {
        System.out.println(A.b);
        System.out.println(A.s);
        System.out.println(A.i);
        System.out.println(A.l);
        System.out.println(A.f);
        System.out.println(A.d);
        System.out.println(A.c);
        System.out.println(A.b1);
        System.out.println(A.c1);
    }   }
```

```
C:\Users\Ravi\Desktop\Day12>javac Day12_1.java

C:\Users\Ravi\Desktop\Day12>java B
0
0
0
0
0.0
0.0

false
null

C:\Users\Ravi\Desktop\Day12>
```

Stack	Heap
	A.b → 0 A.s → 0 A.i- → 0 A.l → 0 A.f → 0.0 A.d → 0.0 A.c → A.b1 → false A.c1 → null

- විචල්‍ය පවතින class එක තුලදී class name සමග හෝ class name එක නොමැතිව එය call කළ හැක.

```
class A{
    static byte b;      static short s;     static int i;
    static long l;      static float f;     static double d;
    static char c;      static boolean b1;  static C c1;

    public static void main(String[] args) {
        System.out.println(b);
        System.out.println(s);
        System.out.println(i);
        System.out.println(l);
        System.out.println(f);
        System.out.println(d);
        System.out.println(c);
        System.out.println(b1);
        System.out.println(c1);
    }
}
class C{}
```

- Class name එක නොමැතිව වෙනත් class එකක සිට call කළ විට පහත අයුරින් errors ලැබේ.

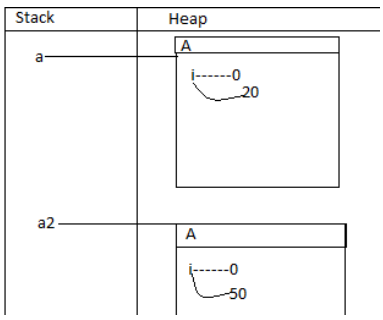
```
class A{
    static byte b;      static short s;     static int i;
    static long l;      static float f;     static double d;
    static char c;      static boolean b1;  static C c1;
}class B{

    public static void main(String[] args) {
        System.out.println(b);
        System.out.println(s);
        System.out.println(i);
        System.out.println(l);
        System.out.println(f);
        System.out.println(d);
        System.out.println(c);
        System.out.println(b1);
        System.out.println(c1);
    }
}
class C{}
```

```
C:\Users\Ravi\Desktop\Day12>javac Day12_1.java
Day12_1.java:8: cannot find symbol
symbol  : variable b
location: class B
                System.out.println(b);
                                   ^
Day12_1.java:9: cannot find symbol
symbol  : variable s
location: class B
                System.out.println(s);
                                   ^
Day12_1.java:10: cannot find symbol
symbol  : variable i
location: class B
                System.out.println(i);
                                   ^
Day12_1.java:11: cannot find symbol
symbol  : variable l
location: class B
                System.out.println(l);
                                   ^
Day12_1.java:12: cannot find symbol
symbol  : variable f
location: class B
                System.out.println(f);
                                   ^
```

Ex:-01

```
class A{
    static int i;
}
class B{
    public static void main(String[]args){
        A a=new A();
                System.out.println(A.i);
                System.out.println(a.i);
            A.i=20;
                System.out.println(a.i);
                System.out.println(A.i);
        A a2=new A();
            a2.i=50;
                System.out.println(a.i);
                System.out.println(a2.i);
                System.out.println(A.i);

    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 13>java B
0
0
20
20
50
50
50

C:\Users\PlayBoy\Desktop\Day 13>
```



Ex:-02

```
class AA{
    static int i=200;
    static float f=234.3f;
    public static void main(String[]args){
        System.out.println(i);
        System.out.println(f);
        System.out.println(BB.i);
        System.out.println(BB.f);
    }
}
class BB{
    static int i=11;
    static float f=43.4f;
}
```

Class {} තුල පවතින variable වලට direct call කල හැක.

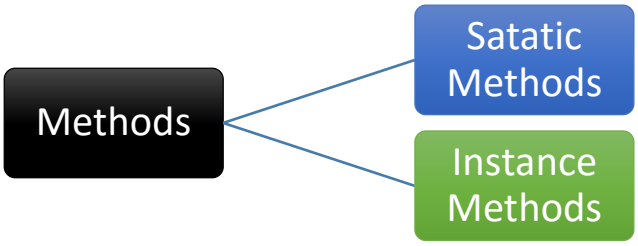
වෙනත් class එක්ක පවතින variable වලට call කිරීමේදී class එකේ නමින් කතාකල යුතුය.

Method Local Variable

Method Local Variable අධ්‍යනය කිරීමට පෙර Method පිලිබඳ අවබෝධයක් ලබාගමු..

Methods

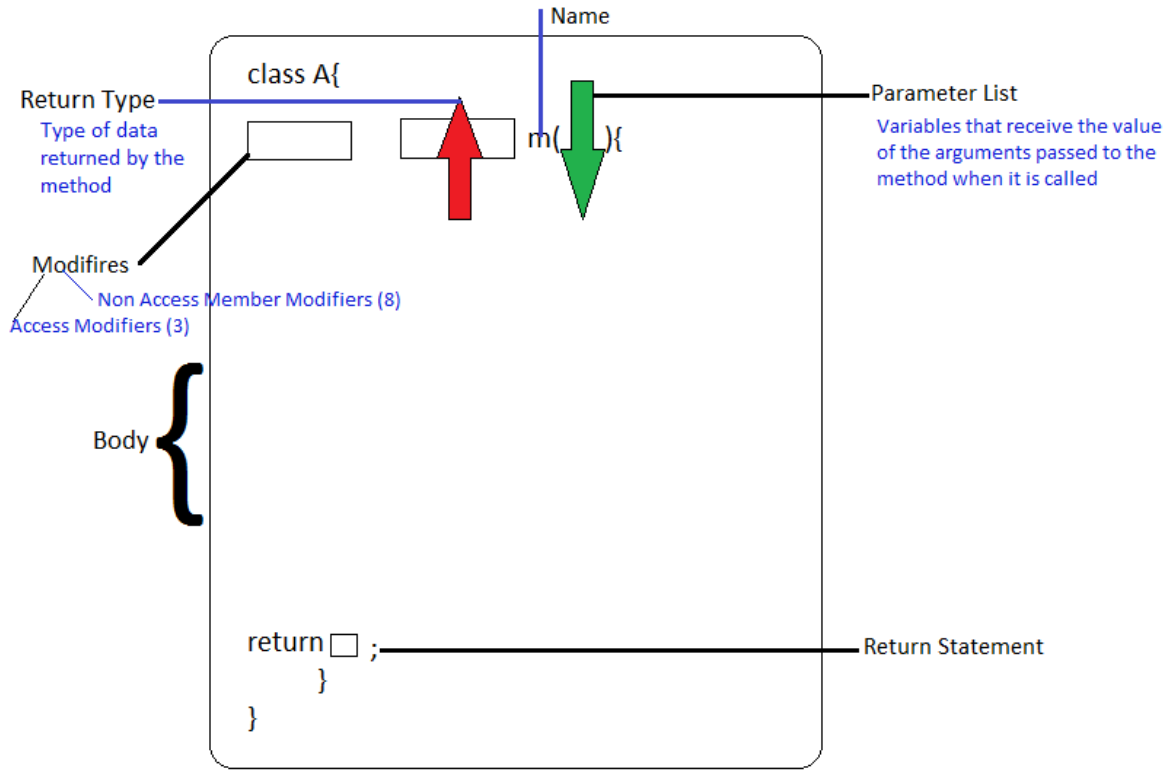
ප්‍රධාන වශයෙන් Methods වර්ග දෙකකි.



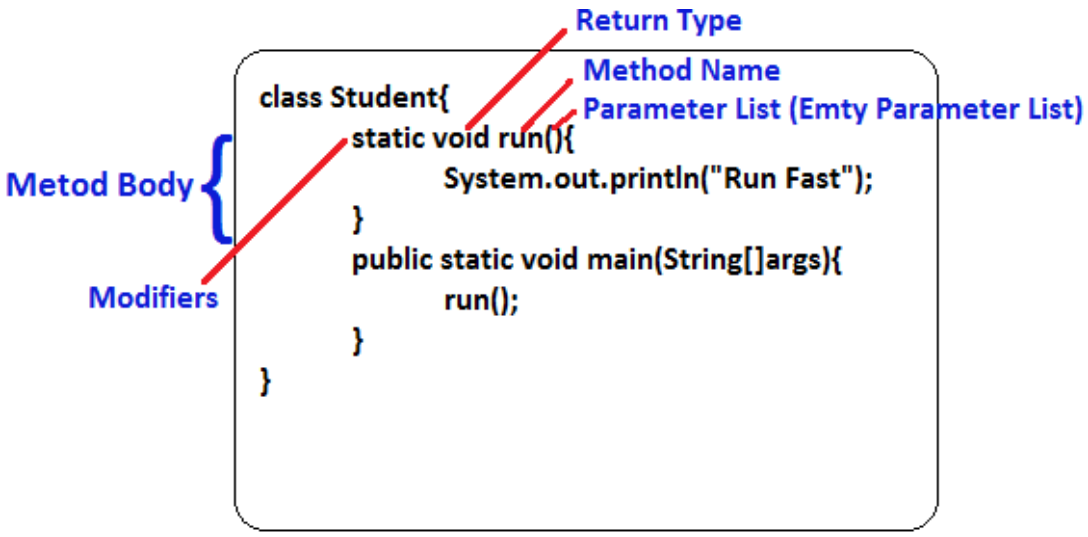

```
class B{
    static void m(){
        System.out.println("Im method");
    }
    public static void main(String[]args){
        System.out.println("Main Method")
        m()
    }

}
```

මෙහි දක්වා ඇත්තේ Method සඳහා නිදසුනකි.

Structure of a method



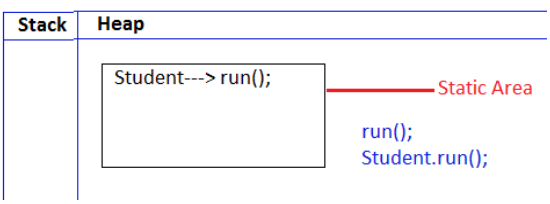
Ex:-01



මෙහි Return Statement අන්තර්ගත නොවේ.

Method Call කිරීමේදී පහත ආකාරකිපයෙන්ම call කළ හැක.

```
class Student{
    static void run(){
        System.out.println("Run Fast");
    }
    public static void main(String[]args){
        run();
        Student.run();
    }
}
```



- Class එකක සිට වෙනත් class එක්ක පවතින method call කළ යුත්තේ class name එක භාවිතයෙනි.

Ex:-02

```
class x{
    static void myMethod(){
        System.out.println("My Method in class x");
    }
}
class Y{
    public static void main(String[]args){
        System.out.println("Main method in class Y");
        x.myMethod();
    }
}
```

Ex:-03_(Void Return Type)

```
class x{
    static void myMethod(){
        System.out.println("My Method in class x");
        System.out.println("1");
        System.out.println("End of myMethod");
    }
}
class Y{
    public static void main(String[]args){
        System.out.println("Main method in class Y");
        x.myMethod();
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 13>java Y
Main method in class Y
My Method in class x
1
End of myMethod

C:\Users\PlayBoy\Desktop\Day 13>
```

void return type එක්ක වුවද එමගින් කිසිවක් එලියට **return** කිරීමේ හැකියාවක් නොමැත. එම නිසා **method** එක සඳහා **return statement** යෙදීම අවශය නොවේ. එවැන්නක් දීම හේතුවෙන් **error** ලැබේ.

void මගින් සිදුකරනුයේ අදාල **method** එකේ පවතින **code execute** කිරීම පමණක් සිදුකරයි.

Ex:-04

```
class x{
    static int myMethod(){
        System.out.println("My Method in class x");
        System.out.println("1");
        System.out.println("End of myMethod");

    }
}
class Y{
    public static void main(String[]args){
        System.out.println("Main method in class Y");
        x.myMethod();
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 13>javac "new 7.java"
new 7.java:7: missing return statement
        }
        ^
1 error

C:\Users\PlayBoy\Desktop\Day 13>
```

➢ **void** හැර වෙනත් **return type** භාවිතයේදී ඊට අදාල වූ **return value** එකක් **return** කළ යුතුය.

Ex:-05_ (Primitive Return Type _int)

```
class x{
    static int myMethod(){
        System.out.println("My Method in class x");
        System.out.println("1");
        System.out.println("End of myMethod");
        return 5;
    }
}
class Y{
    public static void main(String[]args){
        System.out.println("Main method in class Y");
        x.myMethod();
    }
}
```

මෙහිදී අදාල වූ return value එක ලබා දෙන නිසා return statement missing යන error message එක ලබා නොදෙයි.

```
C:\Users\PlayBoy\Desktop\Day 13>java Y
Main method in class Y
My Method in class x
1
End of myMethod

C:\Users\PlayBoy\Desktop\Day 13>
```

➢ **Return statement** එක යෙදීමේදී **return type** එකට ගැලපෙන අගයක්ම විය යුතුය. පහත නිදසුනෙන් එය පෙන්නුම් කරයි.

Ex:-06

```
class x{
    static int myMethod(){
        System.out.println("My Method in class x");
        System.out.println("1");
        System.out.println("End of myMethod");
        return 5.5;
    }
}
class Y{
    public static void main(String[]args){
        System.out.println("Main method in class Y");
        x.myMethod();
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 13>javac "new 9.java"
new 9.java:6: possible loss of precision
found   : double
required: int
                return 5.5;
                       ^
1 error

C:\Users\PlayBoy\Desktop\Day 13>
```

Ex:-07

Return type එක void අවස්ථාවෙදී return statement යෙධිම සිදු නොකළ යුතුය.

```
class x{
    static void myMethod(){
        System.out.println("My Method in class x");
        System.out.println("1");
        System.out.println("End of myMethod");
        return 5.5;
    }
}
class Y{
    public static void main(String[]args){
        System.out.println("Main method in class Y");
        x.myMethod();
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 13>javac "new 9.java"
new 9.java:6: cannot return a value from method whose result type is void
                return 5.5;
                       ^
1 error

C:\Users\PlayBoy\Desktop\Day 13>
```

Ex:-08

Return statement යෙදීමෙන් අනතුරුව ඉන් පසුව පවතින code execute කරීමට නොහැක. Return Statement එකක් යෙදිය යුත්තේ method {} එක අවසානයේදීය.

Return statement එකට පහලින් වෙනත් code පවතින විට errors 2ක් ලැබේ.

```
class x{
    static int myMethod(){
        System.out.println("My Method in class x");
        System.out.println("1");
        return 55;
        System.out.println("End of myMethod");
    }
```

```
C:\Users\PlayBoy\Desktop\Day 13>javac "new 10.java"
new 10.java:6: unreachable statement
                System.out.println("End of myMethod");
                ^
new 10.java:7: missing return statement
        }
        ^
2 errors

C:\Users\PlayBoy\Desktop\Day 13>
```

Ex:-09

Return statement එක මගින් return කරන අගය පහත අයුරින් අල්ලගත හැක.

```
class x{
    static int myMethod(){
        System.out.println("My Method in class x");
        System.out.println("1");
        System.out.println("End of myMethod");
        return 55;
    }
}
class Y{
    public static void main(String[]args){
        System.out.println("Main method in class Y");
        int k=x.myMethod();
        System.out.println("k :"+k);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 13>javac "new 11.java"

C:\Users\PlayBoy\Desktop\Day 13>java Y
Main method in class Y
My Method in class x
1
End of myMethod
k :55

C:\Users\PlayBoy\Desktop\Day 13>
```

Ex:-10(Object Data Type_(String)-Return Type)

```
class x{
    static String abc(){
        return "hellow";
    }
}
class Y{
    public static void main(String[]args){
        String myString =x.abc();
        System.out.println(myString);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 13>javac "new 12.java"

C:\Users\PlayBoy\Desktop\Day 13>java Y
hellow

C:\Users\PlayBoy\Desktop\Day 13>
```

Ex:-11

```
class A{
        static void abc(){
                System.out.println("hi,I am method abc()");
        }
}
```

```
class B{
        psvm{
                A.abc();
        }
}
```

Stack	Heap
	A —— abc()

Ex:-12

```
class A{
        static void testvoid( ){
                System.out.println("hi,I am method abc()");
                }
        static int testint( ){
                System.out.println("hi,I am method tetint");
                return 20; }
        static boolean testboolean( ){
                System.out.println("hi, I am testboolean");
                return true;        }
        static String teststring( ){
                System.out.println("hi,I am teststring");
                return "This is a String"; }
        }
```

```
class B{
        psvm(){
        A.testvoid();
         int i =A.testint();
        System.out.println("value of i:"+i);
         boolean b1=A.testboolean();
        System.out.println("value of b1:"+b1);
         String s1=A.teststring();
        System.out.println("value of s1:"+s1);
        }
}
```

Ex:-13

```
class A1{
    static void testvoid(){
        System.out.println("hi,I am method abc()");
    }
    static int testint(){
        System.out.println("hi,I am method tetint");
        return 20;
    }
    static boolean testboolean(){
        System.out.println("hi, I am testboolean");
        return true;
    }
    static String teststring(){
        System.out.println("hi,I am teststring");
        return "This is a String";
    }
}
class B1{
    public static void main(String []args){
        System.out.println(A.testvoid());
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 14>javac "new 3.java"
new 3.java:20: 'void' type not allowed here
                System.out.println(A.testvoid());
                                               ^
1 error
```

මෙහිදී void modifier මගින් කිසිවක් return නොකරන බැවින් මෙම error එක ලැබේ.

method එක execute කරීමට ද ඉඩ ලබා නොදේ.

Parameter List

Ex:-1

```
class X{
    static void m(int i){
        System.out.println("Value of i :"+i);
    }
}
class Y{
    public static void main(String []args){
        X.m(10);
    }
}
```

මෙහිදී return type එකට ගැලපෙන value එකක් parameter list එකට ලබාදිය යුතුය.

Ex:-2

```
class X{
    static void m(int k){
        System.out.println("Value of i :"+k);
    }
}
class Y{
    public static void main(String []args){
        X.m(10,10.23);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 14>javac "new 6.java"
new 6.java:8: m(int) in X cannot be applied to (int,double)
                X.m(10,10.23);
                  ^
1 error
```

Parameter list එක තුල ඉල්ලා තිබෙන්නේ return type 1 ක් නම් ; return කළ යුත්තේධ එක් return value එකකි.

- **Primitive data types, string සහ objective type value වුවද මෙහිදී return කළහැක.**

Ex:-3

```
class X{
    static void m(int k,double d){
        System.out.println("Value of k :"+k);
        System.out.println("Value of d :"+d);
        double tot=k+d;
        System.out.println("Total"+tot);
    }
}
class Y{
    public static void main(String []args){
        X.m(10,10.23);
    }
}
```

මෙහිදී return value පිළිවලින් ලබාදී ඇති නිසා පහත පරිධි output ලැබේ.

```
C:\Users\PlayBoy\Desktop\Day 14>java Y
Value of k :10
Value of d :10.23
Total20.23
```

Ex:-04

```
class X{
    static String hellowcreator(String name){
        String s =name;
        s="Hellow "+s;
        return s;
    }
}
class Y{
    public static void main(String []args){
        String p=X.hellowcreator("Ravi");
        System.out.println(p);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 14>javac "new 8.java"

C:\Users\PlayBoy\Desktop\Day 14>java Y
Hellow Ravi

C:\Users\PlayBoy\Desktop\Day 14>
```

Ex:-05

```
class X{
    static void helloPrinter(String name,int count){
        String s =name;
        int c=count;
        for(int i=1;i<c;i++){
            System.out.println("Hellow "+s+" - "+i);
        }

    }
}
class Y{
    public static void main(String []args){
        String t="Ravi";
        X.helloPrinter(t,10);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 14>java Y
Hellow Ravi - 1
Hellow Ravi - 2
Hellow Ravi - 3
Hellow Ravi - 4
Hellow Ravi - 5
Hellow Ravi - 6
Hellow Ravi - 7
Hellow Ravi - 8
Hellow Ravi - 9
```

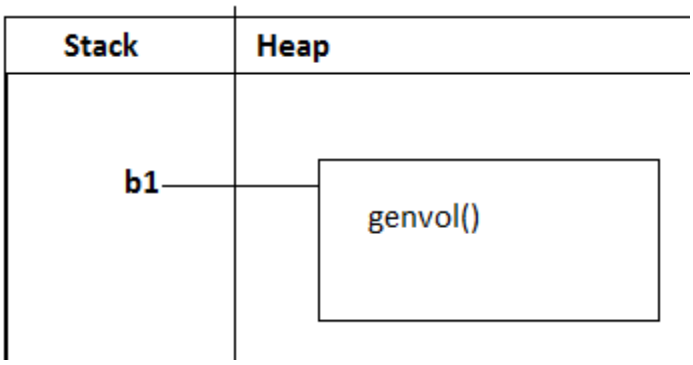
- ✓ **Parameter list** එක සැලකීමේදී එය තුල
  - **Return type value** කීපයක් **“,”** වෙන්වී පැවතිය හැක.
  - මෙහි **primitive type , objective type** හෝ **array** වුවද පැවතිය හැක.
  - සමහර අවස්ථා වලදී කිසිවක් නොමැතිව පැවතිය හැක . එය **empty parameter list** ලෙස හඳුන්වයි.

Instance Method

Ex:-1

```
class Box{
    double genvol (double w,double h,double l){
            double vol =w*h*l;
            return vol;
    }
}
class CreateBox{
    public static void main(String []args){
            Box b1 =new Box();
                double volume=b1.genvol(10.0,10.0,10.0);
            System.out.println("Volume is:"+volume);

            //System.out.println("Volume is:"+b1.genvol(10.0,10.0,10.0));
            }
}
```

```
C:\Users\PlayBoy\Desktop\Day 14>javac "new 12_box.java"

C:\Users\PlayBoy\Desktop\Day 14>java CreateBox
Volume is:1000.0

C:\Users\PlayBoy\Desktop\Day 14>
```



✓ Static variable, Static area තුල සේව් වන අතර method local variables, objects එක තුල සේව් වේ.

Ex:-2

```
class MySelf{
    void details(int age,String name,double heiht){
        System.out.println("Name :"+name);
        System.out.println("Age :"+age);
        System.out.println("Height :"+heiht);
    }

    public static void main (String[]args){

        MySelf ms =new MySelf();
         ms.details(20,"Ravindu Bandara",190.123);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 14>jav

C:\Users\PlayBoy\Desktop\Day 14>jav
Name :Ravindu Bandara
Age :20
Height :190.123

C:\Users\PlayBoy\Desktop\Day 14>
```

	Inside static method	Inside instance method (context)
Static variable / method	Ok	Ok
Instance Variable / method	no	Ok

Method Local Variable

Method එකක් ඇතුලත පවතින variable, method local variable ලෙස හඳුන්වයි.

```
class K{
    public static void main(String[]args){
        int i;
        int j=10;
        int k;
            k=20;
        double d,d1=55.67,d2;
        String s,s2="ABC",s3="AA",s4;
    }
}
```

Explain:-1

මේවා method එක ඇතුලත declare කිරීම සහ භාවිත කිරීම සිදු කරයි. මෙම variables , Method එකට පමණක් සීමා වේ.

```
class K{
    public static void main(String[]args){
        int i;
            i=12;
        int j=10;
        int k;
            k=20;
        double d,d1=55.67,d2;
        String s,s2="ABC",s3="AA",s4;

        System.out.println(j);
        System.out.println(k);
        System.out.println(d1);
        System.out.println("s2="+s2);
```

```
C:\Users\PlayBoy\Desktop\Day 15>java K
10
20
55.67
s2=ABC

C:\Users\PlayBoy\Desktop\Day 15>
```

මෙහි දක්වෙන ඕනෑම ආකාරයකින් variable declaration සිදුකර initialization සිදු කළ හැක.

Explain:-2

යම් අවස්ථාවක method local variables declaration එකෙන් පසුව භාවිතයට ගන්නේ නම් අනිවාර්යෙන්ම එය initialize කළ යුතුය. මෙම

variables default values තබන්නේ නැත. ඒ නිසා initialize part එක අනිවාර්ය වේ. නැතිනම් පහත අයුරින් error ලැබේ.compile නොවෙන නිසා class file නිර්මාණය නොවේ.

```
class K{
    public static void main(String[]args){
        int i;
        System.out.println(i);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 15>javac MethodLocalV_3.java
MethodLocalV_3.java:4: variable i might not have been initialized
                System.out.println(i);
                                   ^
1 error
```

නමුත් භාවිතයට නොගනිනම් එවැනි error එකක් නොලැබේ.

```
class K{
    public static void main(String[]args){
        int i;
        //System.out.println(i);
    }
}
```

Explain:-3

Method කීපයක එකම නමින් variables පැවතිය හැක. Instance, static බේදයක ද එයට බල නොපායි. Class (instance variables) එකක් තුල මෙම ක්‍රියාව සිදුකළ නොහැක.

```
class K{
    public static void main(String[]args){
        int i;
        String s;
    }
    void ab(){
        int i;
        String s;
    }
    static void cd(){
        int i;
        String s;
    }
}
```

Explain:-4

එමෙන්ම Method එක්ක වුවද එකම නාමින් variables කීපයක් පැවතිය නොහැක.

```
class K{
    public static void main(String[]args){
        int i;
        String s;
    }
    void ab(){
        int i;
        String s;
        double i;
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 15>javac MethodLocalV_6.java
MethodLocalV_6.java:9: i is already defined in ab()
                double i;
                       ^
1 error
```

Explain:-5

```
class K{
    String abc(int a , String n){
        int age=a;
        String name=n;
        return "Name:"+name+"-Age:"+age;
    }
}
class CreateStudent{
    public static void main(String[]args){
        K k1=new K();
        String rv=k1.abc(20,"Amal");
            System.out.println(rv);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 15>java CreateStudent
Name:Amal-Age:20
```

Explain:-6

```
class P{
    int m=50;
void ab(int i,int j){
    int k=20;
    System.out.println(k);
    System.out.println(this.m);
    System.out.println(m);
}}
class Q{
    public static void main(String[]args){
            int x=10;
            int y=12;
        P p1=new P();
            p1.ab(x,y);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 15>javac MethodLocalV_9.java

C:\Users\PlayBoy\Desktop\Day 15>java Q
20
50
50
```

මෙවැනි අවස්ථාවක

This magin curntly excute wana object eka labagani,

Explain:-7

```
class V{
    int i;
    String s;
    void ab(){
        int i=10;
        String s="A";
        System.out.println(i);
        System.out.println(s);
        System.out.println(this.i);
        System.out.println(this.s);
    }
    void bc(){
        int i=20;
        String s="B";
        System.out.println(i);
        System.out.println(s);
        System.out.println(this.i);
        System.out.println(this.s);
    }
}
class X{
    public static void main(String[]args){
        V v1=new V();
        V v2=new V();
        v1.i=200;
        v1.s="V1";
        v2.i=400;
        v2.s="V2";
        v1.bc();
        v2.bc();

    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 15>java X
20
B
200
V1
20
B
400
V2

C:\Users\PlayBoy\Desktop\Day 15>
```

Block Variables

Block එකක් ඇතුලත පවතින variable, block variable ලෙස හඳුන්වයි.

```
class A{
    public static void main(String[]args){
        {
            int i;
                i=20;
            String s="A",s1;
        }
    }
}
```

මේවා block එක ඇතුලත declare කිරීම සහ භාවිත කිරීම සිදු කරයි. මෙම variables , bblock එකට පමණක් සිමා වේ.

මෙහි දක්වෙන ඕනෑම ආකාරයකින් variable declaration සිදුකර initialization සිදු කළ හැක.

Explain:-1

```
class A{
    public static void main(String[]args){
        {
            int i;  String s;
                System.out.println(s);
                System.out.println(i);
                    }
            }   }
```

යම් අවස්ථාවක block variables declaration එකෙන් පසුව භාවිතයට ගන්නේ නම්

```
C:\Users\PlayBoy\Desktop\Day 15>javac Block_2.java
Block_2.java:5: variable s might not have been initialized
                                System.out.println(s);
                                                   ^
Block_2.java:6: variable i might not have been initialized
                                System.out.println(i);
                                                   ^
2 errors
```

අනිවාර්යෙන්ම එය initialize කළ යුතුය. මෙම variables default values තබන්නේ නැත. ඒ නිසා initialize part එක අනිවාර්ය වේ.නැතිනම් පහත අයුරින් error ලැබේ.compile නොවෙන නිසා class file නිර්මාණය නොවේ.

Explain:-2

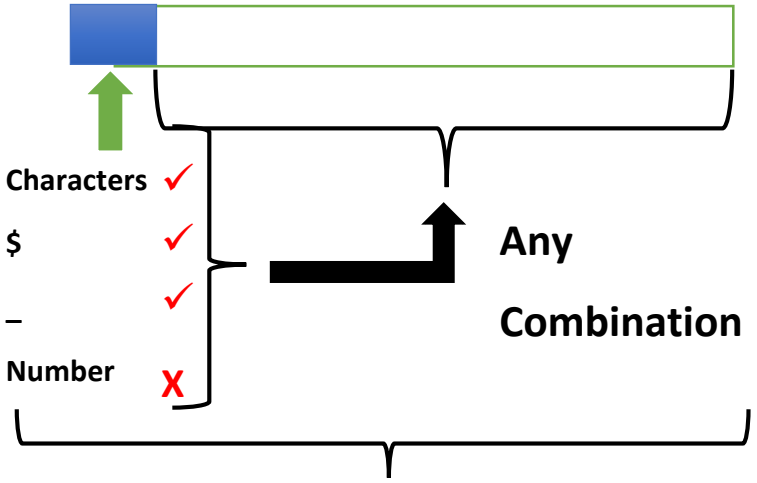
```
class A{
    public static void main(String[]args){
        for(int k=0;k<10;k++){
            System.out.println(k);
        }
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 15>javac Block_5.java

C:\Users\PlayBoy\Desktop\Day 15>java A
0
1
2
3
4
5
6
7
8
9
```

Non – static සහ static වශයෙන් කොටස් 2ක් පවතී ඒ පිලිබඳව පසුවට සාකච්චා කරමු. after constructor lesson.

Identifier

Java භාෂාව තුල බවිතාවන සියලුම components සඳහා නම් භාවිතා කරන අතර ඒවා ***Identifiers*** වශයෙන් හඳුන්වයි.



No Keywords

No Limit

No Space

Case Sensitive

Sun's Java code conventions...

classes	Nouns	Dog Accounts PrintWeiter
methods	verb-noun	getBalance doCalculation setCustomNames
variables	meaningful names	buttonWidth accountBalance
constants	Final	MIN_Value
Interface	Adjectives	Runnable Serializable

Arrays

Arrays මගින් සිදු කරනුයේ එක් variable එකක් බාවිතයෙන් අගයන් රාශියක් එයට ඇතුලත් කර එම අගයන් බාවිතයට ගැනීමයි. Arrays වර්ග ප්‍රධාන වශයෙන් 4ක් පවතී;

1. 1D Array
2. 2D Array
3. 3D Array
4. Anonymous Array

1D Arrays

Explain:-01

මෙහි දක්වා ඇත්තේ 1D arrays Declaration part එකයි.

```
class A{
    public static void main(String[]args){
        int i[];
        int []j;

        String s[];
        String []l;
    }
}
```

Explain*:-02*

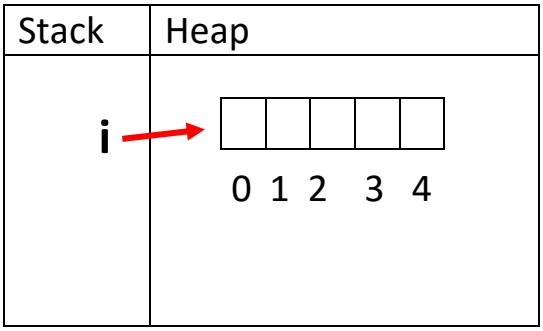
Array එකක් ඇතුලත තබාගත යුතු elements ගන්න එයට ඇතුලත් කිරීම **constructing** ලෙස හඳුන්වයි. එය පහත පරිදී වේ.

```
class A{
    public static void main(String[]args){
        int i[]; //Declaration
        i=new int[5];//Constructing
    }
}
```

Explain:-03

Arrays පහත අයුරින් Ram එකේ ඉඩ වෙන් කරගැනී. විශේෂත්වය වනුයේ එය ඉඩ වෙන් කරගැනීමේදී 0 සිට ගණනය අර්ම්බයි. Elements 5 ක අවස්ථාවකදී එය 0, 1,2,3,4 අයුරින් ඉඩ වෙන් කරගැනී.

```
class A{
    public static void main(String[]args){
        int i[]; //Declaration
        i=new int[5];//Constructing
    }
}
```

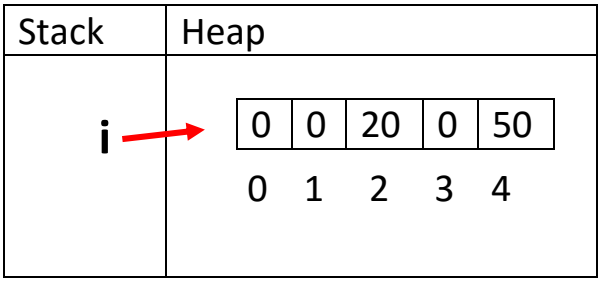


Explain:-04

```
class A{
    public static void main(String[]args){
        int i[]; //Declaration
        i=new int[5];//Constructing

        i[2]=20;
        i[0]=10;
        i[4]=50;
            System.out.println(i[0]);
            System.out.println(i[4]);
            System.out.println(i[2]);
            System.out.println(i[1]);
            System.out.println(i[3]);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 16>java A
10
50
20
0
0
```

මෙහිදී array එකේ elements වලට අදාල අගයන් initialize කිරිඉමේදී heap එකට fill වන අතර initialize නොකරන elements සඳහා default value initialize වේ.



Explain:-05

```
class A{
    public static void main(String[]args){
        String s[]=new String[10];

                s[1]="kamal";
                s[2]="amal";
                s[4]="sunil";
                s[3]="wimal";
                s[6]="nimal";
                s[5]="sunimal";
                s[7]="sapumal";
                s[9]="kumari";
                s[0]="somawathi";
                s[8]="chinchimanavika";
            System.out.println(s[0]);
            System.out.println(s[1]);
            System.out.println(s[2]);
            System.out.println(s[3]);
            System.out.println(s[4]);
            System.out.println(s[5]);
            System.out.println(s[6]);
            System.out.println(s[7]);
            System.out.println(s[8]);
            System.out.println(s[9]);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 16>java
somawathi
kamal
amal
wimal
sunil
sunimal
nimal
sapumal
chinchimanavika
kumari
```

Explain:-06

```
class A{
    public static void main(String[]args){
        String s[]=new String[4];

                s[0]="kamal";
                s[1]="amal";
                s[2]="sunil";
                s[3]="wimal";
                s[4]="wimali";

            System.out.println(s[0]);
    }
}
```

```
class A{
    public static void main(String[]args){
        String s[]=new String[4];

                s[0]="kamal";
                s[1]="amal";
                s[2]="sunil";
                s[3]="wimal";
                s[4]="wimali";

            System.out.println(s[6]);
    }
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
        at A.main(Arrays_5.java:9)

C:\Users\PlayBoy\Desktop\Day 16>javac Arrays_5.java

C:\Users\PlayBoy\Desktop\Day 16>Java A
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
        at A.main(Arrays_5.java:9)
```

Explain:-07

```
class A{
    public static void main(String[]args){
        String s[]=new String[10];

                s[1]="kamal";
                s[2]="amal";
                s[4]="sunil";
                s[3]="wimal";
                s[6]="nimal";
                s[5]="sunimal";
                s[7]="sapumal";
                s[9]="kumari";
                s[0]="somawathi";
                s[8]="chinchimanavika";
            for(int i=0;i<10;i++){
                    System.out.println(s[i]);

                }
    }
```

```
C:\Users\PlayBoy\Des
somawathi
kamal
amal
wimal
sunil
sunimal
nimal
sapumal
chinchimanavika
kumari
```

Explain:-08

යම් අවස්ථාවක array එකේ **variable name.length** ලෙස call කිරීමෙන් array එකේ element ගන්න ලබාගත හැක. පහත නිදසුනෙන් එය අවබෝධ කරගත හැක.

```
class A{
    public static void main(String[]args){
        String s[]=new String[10];

            s[1]="kamal";
            s[2]="amal";
            s[4]="sunil";
            s[3]="wimal";
            s[6]="nimal";
            s[5]="sunimal";
            s[7]="sapumal";
            s[9]="kumari";
            s[0]="somawathi";
            s[8]="chinchimanavika";
          for(int i=0;i<s.length;i++){
                System.out.println(s[i]);
                } } }
```

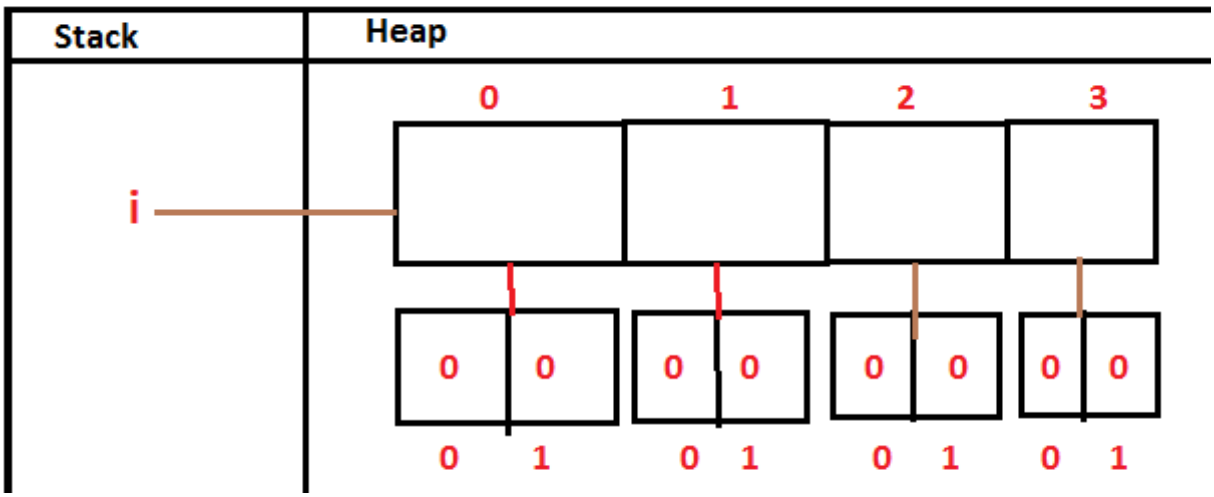
2D Arrays

2D arrays වලදී පහත අයුරින් declarations part එක සිදු කල හැක. එහිදී [ ] [ ] 2ක් භාවිතා කරයි.

```
class A{
    public static void main(String[]args){
        int []i[];
        String s[][];
        byte []b[];
        short [][]s1;
            } }
```

Explain:-01

```
class A{
    public static void main(String[]args){
        int []i[];
           i=new int[4][2];
              //int i[][]=new int[4][2];
       }   }
```



Explain:-02

```
class A{
    public static void main(String[]args){
        int []i[];
           i=new int[4][2];
           i[0][0]=10;
           i[0][1]=20;
           i[1][0]=30;
           i[1][1]=40;
           i[2][0]=50;
           i[2][1]=60;
           i[3][0]=70;
           i[3][1]=80;

           System.out.println(i[3][0]);
           System.out.println(i[2][0]);
           System.out.println(i[1][0]);
           System.out.println(i[0][0]);
           System.out.println(i[0][1]);
           System.out.println(i[1][1]);
           System.out.println(i[2][1]);
           System.out.println(i[3][1]);
       }   }
```

```
C:\Users\PlayBoy\Desktop\Day 16>java A
70
50
30
10
20
40
60
80
```

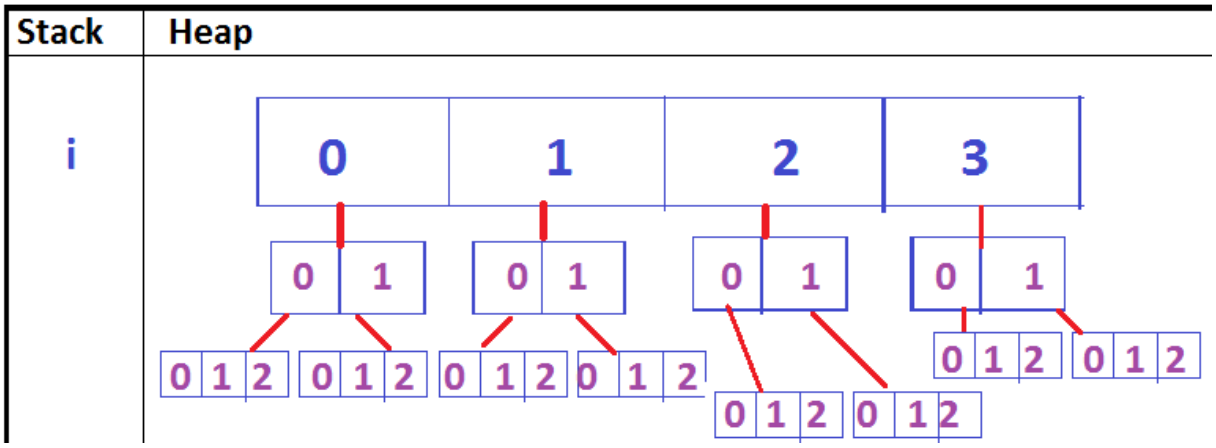
Explain:-03

```
class A{
    public static void main(String[]args){
        int []i[];
           i=new int[4][2];
           i[0][0]=10;
           i[0][1]=20;
           i[1][0]=30;
           i[1][1]=40;
           i[2][0]=50;
           i[2][1]=60;
           i[3][0]=70;
           i[3][1]=80;

           for(int x=0;x<4;x++){
                   for(int y=0;y<2;y++){
                       System.out.println(i[x][y]);
                   }
           }

       }   }
class A{
    public static void main(String[]args){
        int []i[];
           i=new int[4][2];
           i[0][0]=10;
           i[0][1]=20;
           i[1][0]=30;
           i[1][1]=40;
           i[2][0]=50;
           i[2][1]=60;
           i[3][0]=70;
           i[3][1]=80;

           for(int x=0;x<i.length;x++){
                   for(int y=0;y<i[y].length;y++){
                       System.out.println(i[x][y]);
                   }
           }

       }   }
```

3D Arrays

3D arrays වලදී පහත අයුරින් declarations part එක සිදු කල හැක. එහිදී [ ] [ ] [ ] 3ක් භාවිතා කරයි.

```
class A{
    public static void main(String[]args){
        int i[][][]=new int[4][2][3];
    }
}
```



Anonymous array

මෙම arrays වර්ගය dimensions ප්‍රමාණය තීරනය කිරීමට අවශ්‍ය නොවන අතර එය elements ගණන අනුව තීරනය වේ.

1D Array

```
class A {
    public static void main(String[] args) {
        int i[]={1,2,4,56,34,123,34};

            System.out.println(i.length);
    }
}
```

2D Arrays

3D Arrays

```
class A{
    public static void main(String[] args) {
        int i [][][]={{{1,2},{3,4},{5,6}} ,{{7,8},{9,10},{11,12}},{{13,14},{15,16},{17,18}},{{19,20},{21,22},{23,24}}};

            for (int a=0;a<i.length;a++) {
                for (int b=0;b<i[a].length;b++) {
                    for (int c=0;c<i[a][b].length;c++) {
                        System.out.println(i[a][b][c]);
                    }
                }

            }

    }
}
```

Enhance For Loop

Ex:-01

```
class A{
    public static void main(String[] args) {
        String s[]=new String[5];
        s[0]="RAvindu";
        s[1]="Shehan";
        s[2]="Shamith";
        s[3]="kumara";
        s[4]="imesh";

        for (String s1:s) {
            System.out.println(s1);
        }   }   }
```

Ex:-02

```
class A{
    public static void main(String[] args) {
        int i[][]={{1,2,3,4} , {5,6,7,8} };
            for (int a[]:i) {
                for (int b:a) {
                    System.out.println(b);
                }
            }   }
        }
```

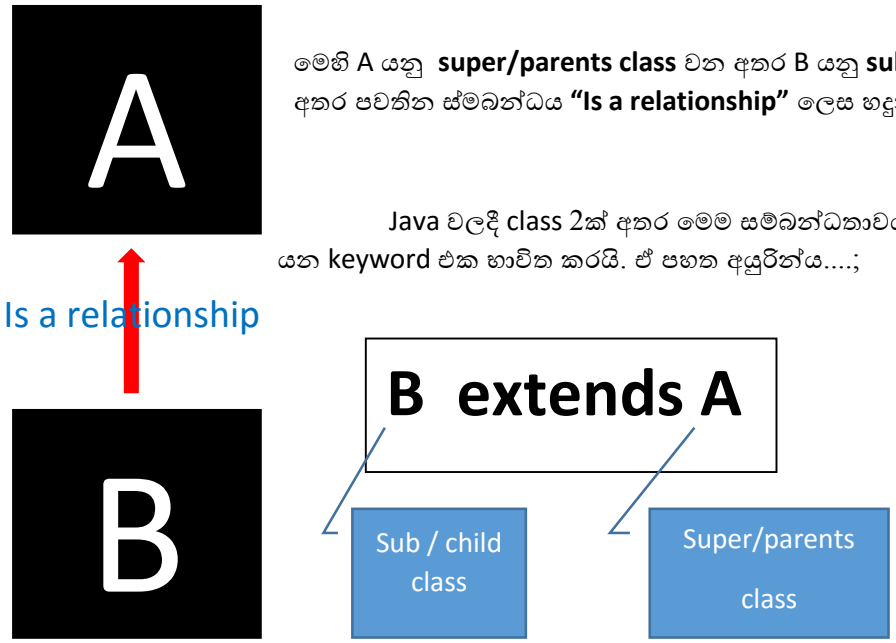
Object Orientation Concept

1).Inheritance

Java language එකේදී inheritance ලෙස හඳුන්වනුයේ .....;

යම් java class එකක් සතු futures & behaviors වෙනත් java class එකක් සතු විෂමය . එනම් යම් super class එක්ක පවතින features & behaviors එම super/parents class එකට අයත් sub/child class එකකට අයත් විෂමයි.

ප්‍රායෝගික ජීවිතයේදී දෙමව්පියන් සතු හැඩ හුරුකම් සහ ගති පැවතුම් දරුවනට ලැබෙන අයුරින්ම , java programming language එකෙදීත් ඉහත පරිදි එලසම පවතී.



මෙහි A යනු **super/parents class** වන අතර B යනු **sub/child class** වේ. A හා B අතර පවතින සම්බන්ධය **“Is a relationship”** ලෙස හඳුන්වයි.

Java වලදී class 2ක් අතර මෙම සම්බන්ධතාවය ගොඩනැගීමට **“Extends”** යන keyword එක භාවිත කරයි. ඒ පහත අයුරින්ය....;

Ex:-01

```
class X{
    int i=200;
    void ab(){
            System.out.println("Method ab()");
        }
}
class Y extends X{

}
class Z{
    public static void main(String args[]){
        Y y1=new Y();
            System.out.println(y1.i);
            y1.ab();

    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 17>java Z
200
Method ab()
```

මෙහි Y sub class එක වන අතර X එහි super class එක වේ. එම නිසා y class එකට X හි feature & behaviour අයිති වේ.

- **class කිපයක features & Behaviors class එකකට ලබාගැනීමට අවශ්‍යනම් එවිට කළ යුත්තේ පවතිඅන super class එක අනෙක් class එකේ child class එක බවට පත් කිරීම වැනි ගැලපෙන ක්‍රියාවලියකි.**
- **කිසිම අවස්ථවකදී class එකට එකවර class 2ක් extends කළ නොහැක.**

Ex:-02

```
class X extends N{
    int i=200;
    void ab(){
            System.out.println("Method ab()");
        }
}
class Y extends X{}
class N{int k=89;}
class Z{
    public static void main(String args[]){
        Y y1=new Y();
            System.out.println(y1.i);
            y1.ab();
            System.out.println(y1.k);

    }
}
```

- Private යන access modifier එක භාවිතා කිරීමෙන් ඒ අදාල කොටස් එම සීමාවට පමණක් අයත් වේ.

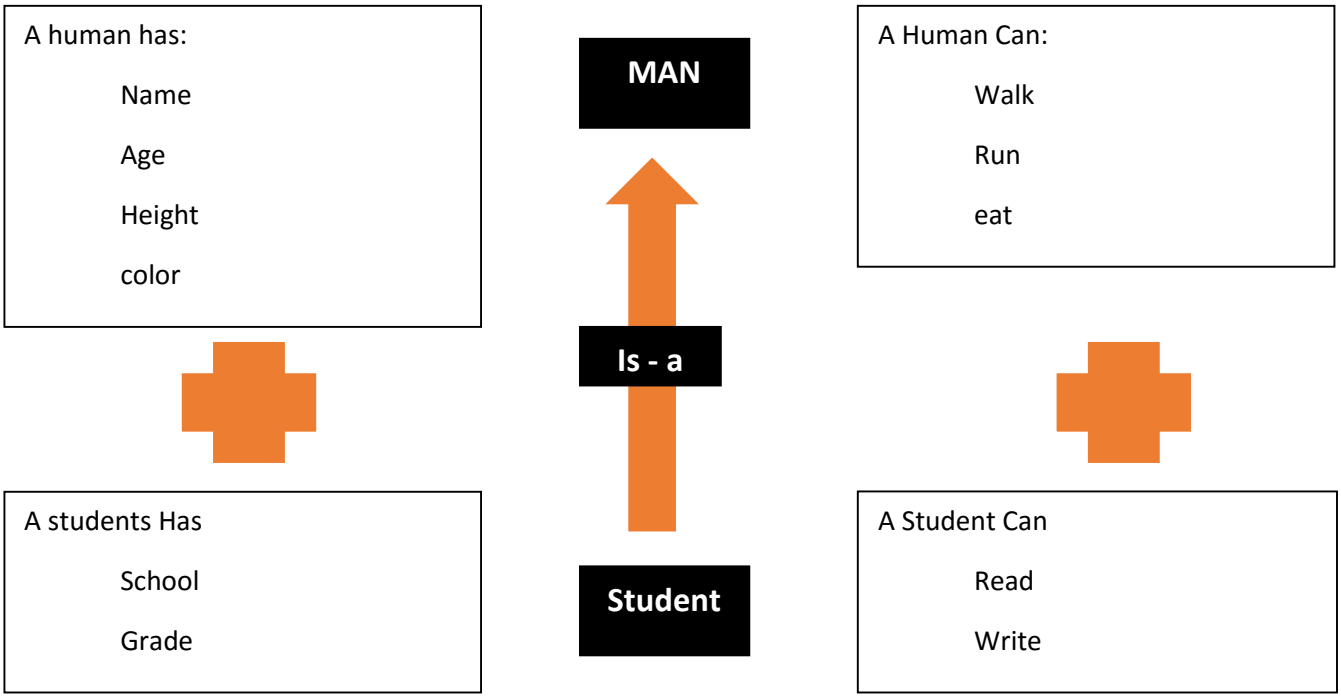
Ex:-03

```
class X extends N{
    int i=200;
    void ab(){
            System.out.println("Method ab()");
        }
}
class Y extends X{}
class N{
        private int k=89;}
class Z{
    public static void main(String args[]){
        Y y1=new Y();
            System.out.println(y1.i);
            y1.ab();
            System.out.println(y1.k);

    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 17>javac Inheritance_4.java
Inheritance_4.java:15: k has private access in N
                        System.out.println(y1.k);
                                           ^
1 error
```

Ex:-04

```
class X{
    int i=200;
    void ab(){
        System.out.println("Method ab()");
    }
}
class Y extends X{
    int l=900;
}
class Z extends Y{
    public static void main(String args[]){
        Z z1=new Z();
        System.out.println(z1.i);
        z1.ab();
        System.out.println(z1.l);

    }
}
```



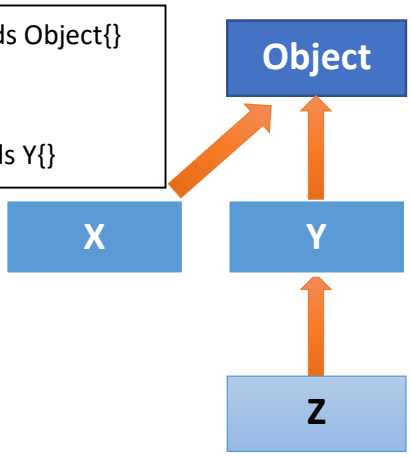
Object Grande Super Parents Class

Java programing language එකෙදී අප විසින් විවිධ classes' වර්ග විවිධ නම් වලින් නිර්මාණය කරන අතර ඒ සියල්ලම automatically extends ව පවතින class එක **"Object Class"** හෙවත් එය javaවල **"Grande Super Parents class"** ලෙස හඳුන්වයි.

යම් අවස්තාවක Manually යම් class එකක් වෙනත් class එකකට extends කර පවතීනම් එම අවස්ථාවේදී Objects class එක සමග ඇති සම්බන්ධයෙන් ඉවත් වී manually extends කළ class එක සමග සම්බන්ධතාවය පවත්වාගනී.

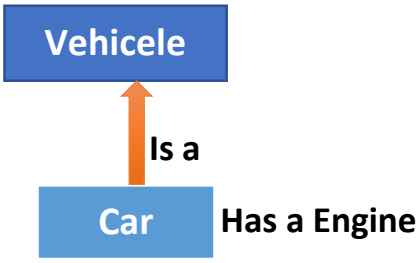
එම සම්බන්ධතාවයඇද දක්වීම class hierarchy අදීම වශයෙන් හඳුන්වයි.

```
class X extends Object{}
class Y{}
class Z extends Y{}
```



2).Has a Relationship

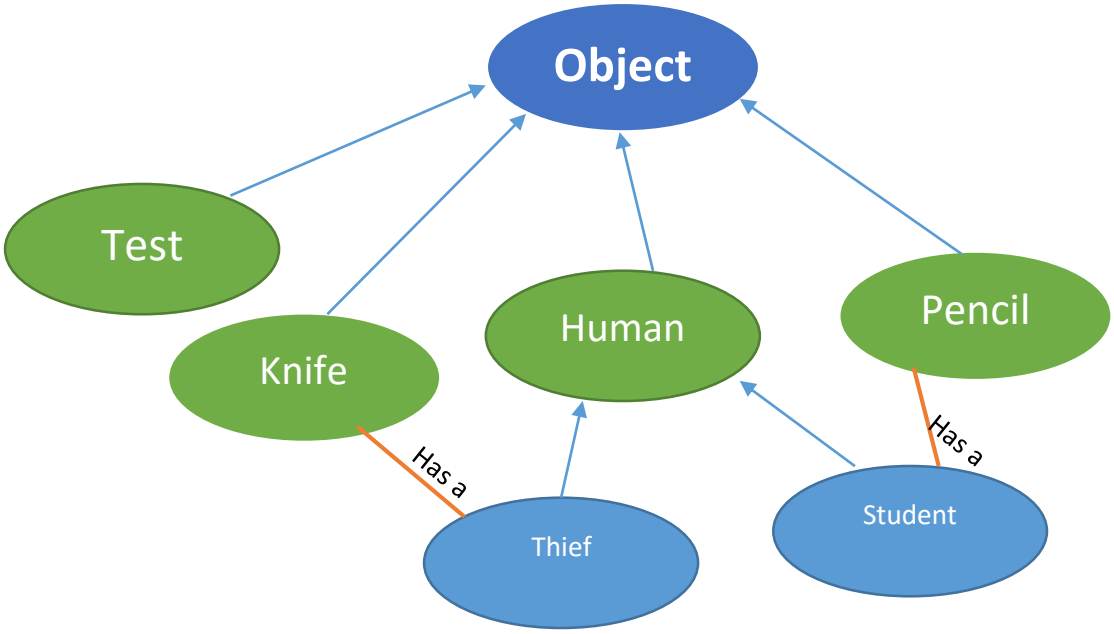
```
class Vehicel{
}
Class Car extends Vehicele{
}
class Engine{}
```



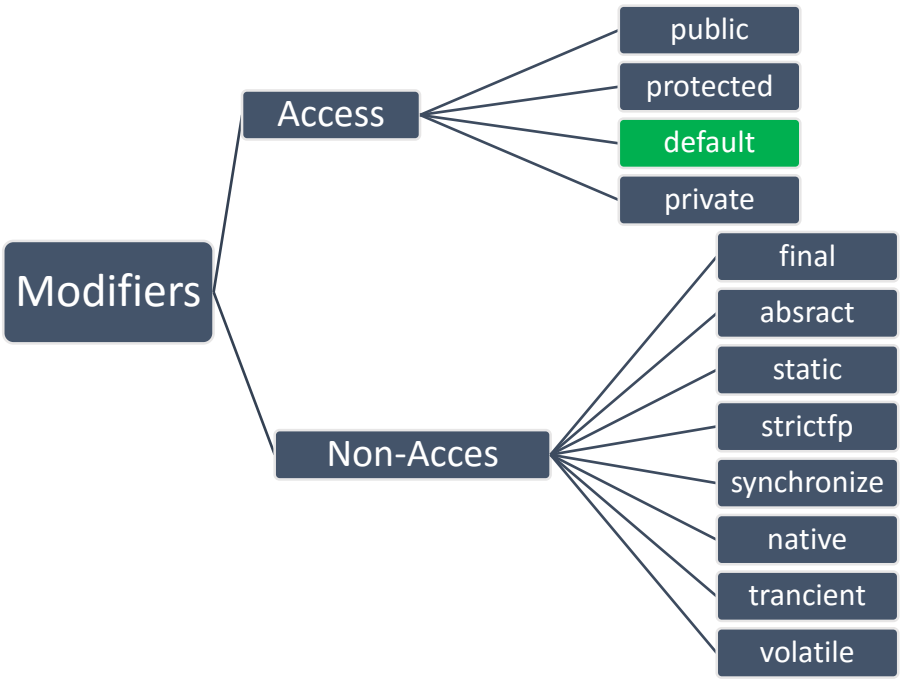
Ex:-01

```java
class Human{
    int age;
    String name;
    void run(){
        System.out.println("I can run");
    }
}
class Students extends Human{
    String grade ="5C";
    Pencil p;
    void read(){
        System.out.println("I Can read");
    }
}
class Thief extends Human{
    String type ="killer";
    Knife k = new Knife();
    void kill(){
        System.out.println("I Can kill");
    }
}
class Pencil{
    String brand="HB";
}
class Knife{
    String brand;
}
```

```java
class Test{
    public static void main(String[] args) {
        Students s =new Students();
        s.name="Rohan Perera";
        s.age=16;
        System.out.println(s.name);
        System.out.println(s.age);
        System.out.println(s.grade);

        s.p=new Pencil();
        System.out.println(s.p.brand);
        s.run();
        s.read();
        System.out.println();
        System.out.println("-- -- -- -- -- --");
        System.out.println();

        Thief t = new Thief();
        t.name="*** Upali";
        t.age=28;
        System.out.println(t.name);
        System.out.println(t.age);
        System.out.println(t.type);
        t.k.brand="Rambo";
        System.out.println(t.k.brand);
        t.run();
        t.kill();
    }
}
```



Modifiers



Explain:-01

Access Modifiers & Non-Access modifiers වැඩසටහනක පවතින විට ඒවා පවතින ස්ථානය බලනොපායි.

```
public class K {
    static public void main(String[]args){
        System.out.println("fsdfsdf");
    }
}
public class K {
    public static void main(String[]args){
        System.out.println("fsdfsdf");
    }
}
```

Explain:-02

Access Modifier 1ක් සමග Non-Access modifiers 1 ක් හෝ කීපයක් වැඩසටහනක පැවතිය හැක. **නමුත් Access Modifiers 2ක් හෝ ඊට වැඩි ප්‍රමාණයක් එක ලග පවතිය නොහැක.**

```
public class K {
    final static public void main(String[]args){
        System.out.println("fsdfsdf");
    }
}
```

- Java source file declaration Rules වලට අනුව එක් source file එකක් තුල පැවතිය හැක්කේ එක් public class එකක් පමණි. එමෙන්ම public class එකක් පවතින source file save කළ යුත්තේද එම class එකේ නමින්මය.

```
public class A{
    public static void main(String[]args){
        System.out.println("HO");
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 18>javac modifires_1.java
modifires_1.java:1: class A is public, should be declared in a file named A.java
public class A{
       ^
1 error

C:\Users\PlayBoy\Desktop\Day 18>
```

Ex:-03

```
public class X{
    public int i=10;
    public  void ab(){
        System.out.println("Method ab()");
    }
}
class Y{
    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();
    }
}
```

Public variables same package එකේ පවතින විට එකම source code එකේ වෙනත් class එක්ක සිට හෝ එකම package එකේ different class වල සිට වුව access කළ හැක.

```
class Y{
    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();
    }
}
```

```
public class X{
    public int i=10;
    public  void ab(){
        System.out.println("Method ab()");
    }
}
```

Explain:-04

```
class Y extends X{
    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();

        Y y1=new Y();
        System.out.println(y1.i);
        y1.ab();
    }
}
```

එකම source package එකේ පවතින වෙනත් class එකක් මෙහි දැක්වෙන පරිදි extends කිරීම පමනක් සිදු කර (Sub class එකක් භාවිතයෙන්) වුවද public variables & methods මගින් ප්‍රතිපල ලබාගත හැක.

Explain:-05

Different package එක්ක පවතින different class එක සිට public variables & method වැනි දේවල් access කළ හැක.

මෙහිදී import statement භාවිතා කර package එකේ පවතින class file එකට call කළ හැකි අතර එහිදී import කරන class එක public class එකක් බවට පත්කළ යුතුය.

```
package B;
import K.X;
public class Z{
    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();
    }
}
```

Explain:-06

Different package, different class වල පවතින sub class බවත කිරීමෙන් වුවද public variables and method access කළහැක.

```
package B;
import K.X;

public class K extends X{
    public static void main(String[] args) {
        K k1=new K();
        System.out.println(k1.i);
        k1.ab();
    }
}
```

Same Class	OK
Same Package Different Class	OK
Same Package Sub Class	OK
Different Package Different Class	OK
Different Package Sub Class	OK

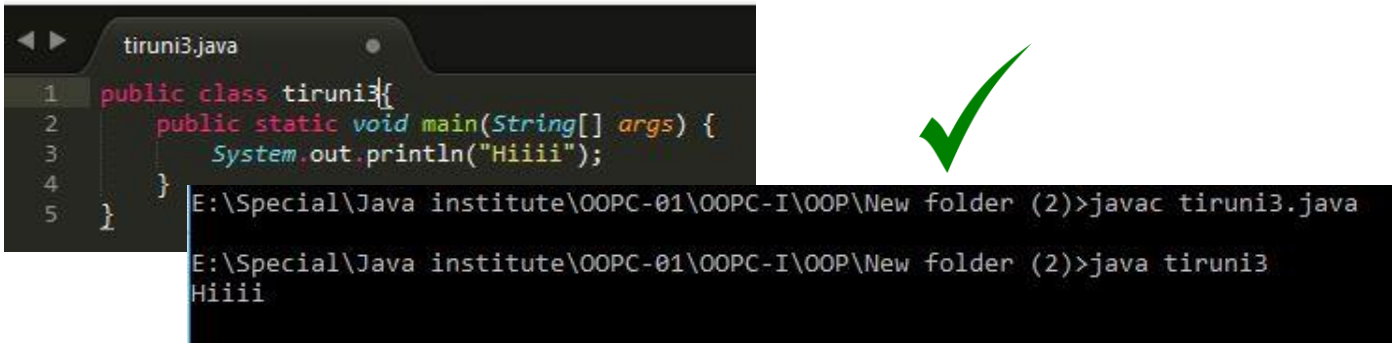
private variables And methods සැම විටම භාවතා කළ හැක්කේ එම class එකේ සිට පමණි.

Access Modifiers

➢ Public:

මෙහි විශේෂත්වය වන්නේ මෙය ඕනෑම class file එකක සිට Access කල හැකිවීමය.Java source file rules වලට අනුව එකම java file එකක් තුල public class දෙකක් තිබිය නොහැක. තවද මෙහි විශේෂත්වය වන්නේ public class එකක් ඇති java file එකක් save කලයුත්තේ එහි class name එකෙන්ම විමය.

Explain:-01



- Public class එකක් තුල ඇති variables different class එකක් මගින්ද access කළ හැක.
- මෙම class2ක වෙනම note pad දෙකක තිබුනද එය automatically අනිත් notepad එකද compile කරගනුලැබේ.

Explain: -02

```
package K;

/**
 *
 * @author Shehan Nawoda
 */
public class X {
    public int i=10;
    public void ab (){
        System.out.println("Method ab");
    }
}
class Y{

    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();
    }
}
```

```
run:
10
Method ab
BUILD SUCCESSFUL (total time: 1 second)
```

Explain: -03

```
package K;

/**
 *
 * @author Shehan Nawoda
 */
public class X {
    public int i=10;
    public void ab (){
        System.out.println("Method ab");
    }
}
```

```
package K;
class Y{

    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();
    }
}
```

```
run:
10
Method ab
BUILD SUCCESSFUL (total time: 0 seconds)
```

Explain: -04

```
package K;
class Y extends X{

    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();

        Y y1=new Y();
        System.out.println(y1.i);
        y1.ab();
    }
}
```

```
package K;

/**
 *
 * @author Shehan Nawoda
 */
public class X {
    public int i=10;
    public void ab (){
        System.out.println("Method ab");
    }
}
```

```
run:
10
Method ab
10
Method ab
BUILD SUCCESSFUL (total time: 0 seconds)
```

- **Import** යන **keyword** භාවිතා කර වෙනත් **package** එකක පවතින **class** එකක් වෙනත් **package** එකක සිට **call** කල හැකිවීමය.

Explain: -05

```
package K;
class Y extends X{

    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();

        Y y1=new Y();
        System.out.println(y1.i);
        y1.ab();
    }
}
```

```
package K;

/**
 *
 * @author Shehan Nawoda
 */
public class X {
    public int i=10;
    public void ab (){
        System.out.println("Method ab");
    }
}
```

```
package KK;

import K.X;
public class Z {
    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.ab();
    }
}
```

```
run:
10
Method ab
10
Method ab
BUILD SUCCESSFUL (total time: 0 seconds)
```

Same class	✓
Same package different class	✓
Same package sub class	✓
Different package different class	✓
Different package sub class	✓

Private

මෙමෙ “private” යන access modifier එක variables වලට සහ methods වලට භාවිතාවන අතර class සඳහා භාවිතා නොවේ.

Ex:-01

```
package pri;

public class X {
    private int i=20;
    private void abc(){
        System.out.println("Method abc()");
    }
    public static void main(String[] args) {
        X x1=new X();
        System.out.println(x1.i);
        x1.abc();
    }
}
```

```
run:
20
Method abc
BUILD SUCCESSFUL (total time: 0 seconds)
```

Same class	✓
Same package different class	✗
Same package sub class	✗
Different package different class	✗
Different package sub class	✗

Protected

මෙම protected යන keyword එක බාවිතා කරමින් සෑදියහැක්කේ variables සහ method පමණි. මෙය බාවිතා කරමින් protected class සෑදිය නොහැක.

Explain: -01

```
package pri;

public class X{
    protected int i=10;
    protected void xxx(){
        System.out.println(i);
        i+=1;
        System.out.println(i);
    }
    public static void main(String[] args) {
     X x1=new X();
        System.out.println(x1.i);
        x1.xxx();
    }

}
```

Same class	✓
Same package different class	✓
Same package sub class	✓
Different package different class	✗
Different package sub class	✗

Default/Package Access Level

මෙම default යන keyword එක බාවිතා කරමින් සෑදියහැක්කේ variables සහ method පමණි. මෙය බාවිතා කරමින් default class සෑදිය නොහැක. මෙයට package access level ලෙසද හඳුන්වනු ලැබේ. සාමාන්‍ය ලෙස නිර්මාණය කරන method & variables default වර්ගයට අයත් වේ.

Same class	✓
Same package different class	✓
Same package sub class	✓
Different package different class	✗
Different package sub class	✗

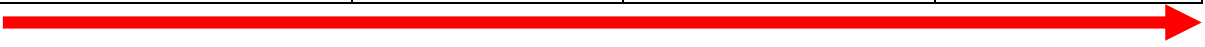
Summary of Access Modifiers

	Public	Private	Protected	Default
Same Class	Ok	OK	OK	OK
Same Package Different Class	OK	NO	OK	OK
Same Package Sub Class	OK	NO	OK	OK
Different Package Different Class	OK	NO	NO	No
Different Package Sub Class	OK	NO	OK	NO

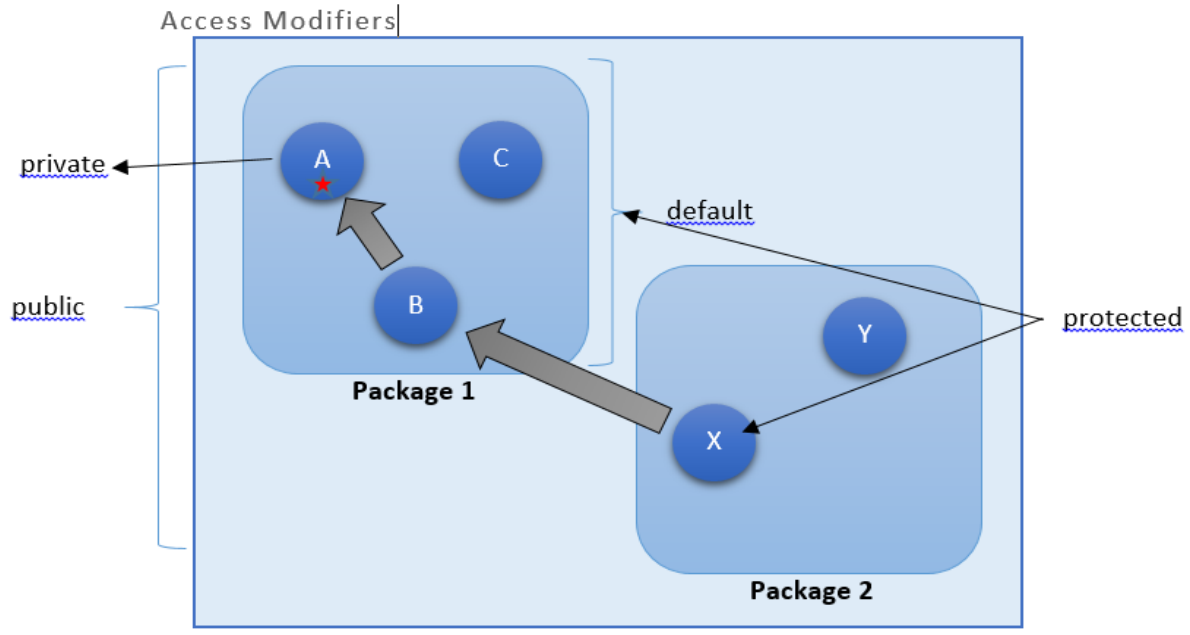
Situation	Public	Private	Protected	Default
Same Class	✓	✓	✓	✓
Same Package Different Class	✓	✗	✓	✓
Same Package Sub Class	✓	✗	✓	✓
Different Package Different Class	✓	✗	✗	✗
Different Package Sub Class	✓	✗	✓	✗

ඉහත වගුව සැලකීමෙන් ආරක්ෂාවක් වැඩිම **Access modifier** එක සිට ආරක්ෂාව අඩුම **Access modifier** එක දක්වා පිළිවලින් පිළිවලින් පෙළ ගැස්වූ විට පහත පරිදිය.

Private | **default** | **Protected** | **Public**



Where do we use Modifiers…?



Method local Variables	Instance variable	Methods
final	final public protected default private trancient volatile	final public protected default private Synchronize Static Strictfp native abstract

Encapsulation

Class එක්ක පවතින **variable** ආරක්ෂා කිරීම සඳහා මෙම **concept** එක භාවිතා කරයි.

Not Encapsulated Class

```
class Account{
    public int balance;
    public String name;
}

class A{
    public static void main(String[] args) {
        Account a =new Account();
            a.name="Kamal";
            a.balance=10000;
            System.out.println(a.balance);
    }
}
```

Encapsulated Class

Class තුල පවතින variables, **private variables** බවට පත් කිරීම encapsulate වශයෙන් හඳුන්වයි.

Explain: -01

```
class Account{
    private int balance; //Enscapsulated
    private String name; //Enscapsulated
}

class A{
    public static void main(String[] args) {
        Account a =new Account();
            a.name="Kamal";
            a.balance=10000;
            System.out.println(a.balance);
    }
}
```

Can't change values from every where

```
C:\Users\PlayBoy\Desktop\Day 19>javac encaps_02.java
encaps_02.java:9: name has private access in Account
                        a.name="Kamal";
                         ^
encaps_02.java:10: balance has private access in Account
                        a.balance=10000;
                         ^
encaps_02.java:11: balance has private access in Account
                        System.out.println(a.balance);
                                           ^
3 errors
```

- **Encapsulate** කිරීමෙන් පසු කිසිම තැනකදී **variable value** වෙනස් කළ නොහැක. ඒ නිසා එයට විසඳුමක් ලෙස **method()** භාවිතා කල හැක.

Explain: -02

```
class Account{
    private int balance; //Enscapsulated

    public void setBalance(int n){
        balance =n;
        System.out.println(n);
    }
}

class A{
    public static void main(String[] args) {
        Account a =new Account();
            a.setBalance(100050);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 19>java A
100050
```

- මෙහිදී අවශ්‍යතාවයට ගැලපෙන පරිදි **method** එකක් භාවිතා කරමින් **variable** එකට අගය ඇතුලත් කළ හැක.

Explain: -03

```
class Account{
    private int balance; //Enscapsulated

    public void setBalance(int balance){
        this.balance=balance;
        System.out.println(balance);
    }
}

class A{
    public static void main(String[] args) {
        Account a =new Account();
            a.setBalance(100050);
    }
}
```

- මෙහිදී එකම **variable** එක ස්ථාන කිපයක භාවිත කිරීමේදී **this** යන **keyword** එක භාවිතා කර එම අවස්ථාවේ **run** වෙන **variable** එක භාවිත කළ හැක.

How To Set Values

```
class Account{
        private int balance;

        public void setBalance(int balance){
                this.balance=balance;
        }
}
```

```
class A{
        public static void main(String[] args) {
                Account a =new Account();
                        a.setBalance(100050);
        }
}
```

How To Get Values

```
class Account{
        private int balance;
        public void setBalance(int balance){
                this.balance = balance;
        }
        public int getBalance(){
                return balance;
        }
}
```

```
class A{
        public static void main(String arg[]){
                Account a=new Account();
                a.setBalance(1000);

        System.out.println(a.getBalance());
        }
}
```

SetName & GetName

```
class Ac{
    private int balance;
    public void setBalance(int balance){
        this.balance=balance;
    }
    public int getBalance(){
        return balance;
    }
}

class A{
    public static void main(String[] args) {
        Ac a = new Ac();
            a.setBalance(12356);
            System.out.println(a.getBalance());
    }
}
```

Features

- All variables are private.
- Options
  - getters – Accesser
  - setters – Mutators

Advantageous

- Data hiding
- Immutability
- Data Validation

Disadvantageous

- More Codes
- More time

Advantageous/Benefits

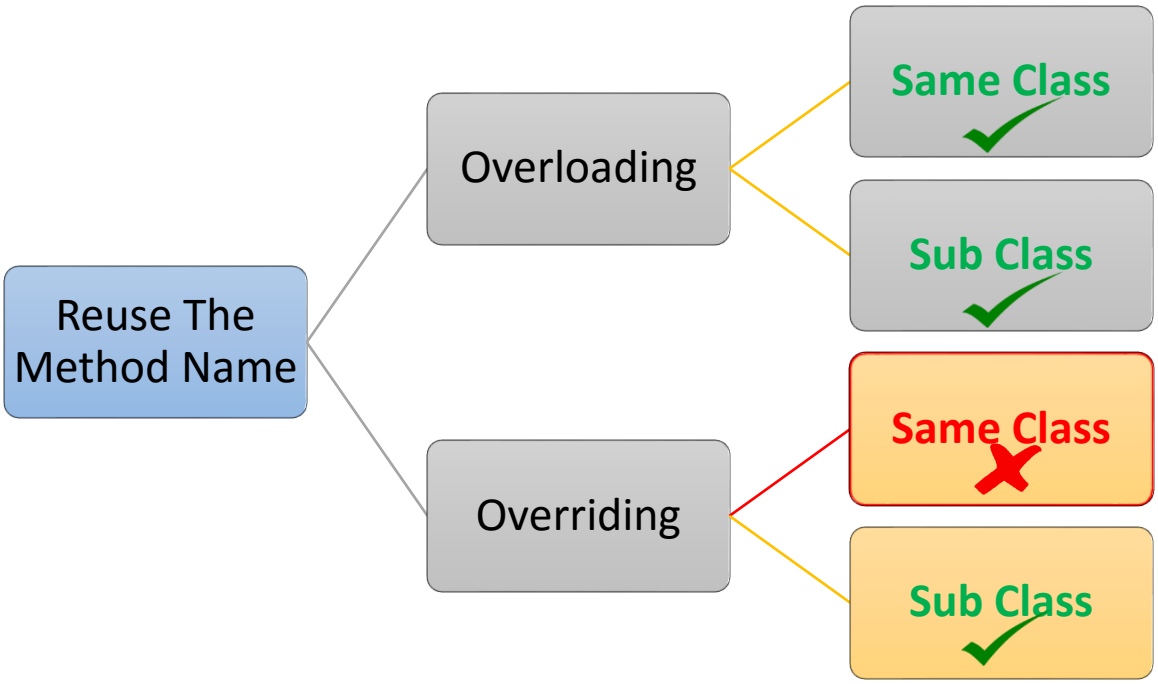
- Data hiding
- Immutability
- Data Validation

```
class A{
        private int balance;

        public int getBalance(){
            return  balance;            //Data hiding
        }
        public void setBalance(int balance){
            if(balance>0){              //Immutability
                this.balance=balance;
            }
        }
```

Ex: -01

```
class Foo{
    public double left=9.0;
    public double right=3.0;

    public void setLeft(double leftNum){
        left =leftNum;
        right=leftNum/3;
        System.out.println("left value="+left+"  left value="+right);
    }
    public static void main(String[] args) {
        Foo f =new Foo();
            f.setLeft(10);
    }
}
```

Reuse The Method Name



Overloading

Explain: -01

```
class A{
    void abc(){
        System.out.println("ABC1");
    }
    void abc(int i){
        System.out.println("ABC2");
    }
    void abc(int i, String s){
        System.out.println("ABC3");
    }
    void abc(String s , int i){
        System.out.println("ABC4");
    }
    void abc(double d){
        System.out.println("ABC5");
    }
}
```

Explain: -02

```
class A{
    void abc(){
        System.out.println("ABC1");
    }
    void abc(int i){
        System.out.println("ABC2");
    }
    void abc(int i, String s){
        System.out.println("ABC3");
    }
    void abc(String s , int i){
        System.out.println("ABC4");
    }
    void abc(double d){
        System.out.println("ABC5");
    }
}
class Test{
    public static void main(String[] args) {
        A a1=new A();
            a1.abc();
            a1.abc(10);
            a1.abc(10,"ravi");
            a1.abc("Rvi",10);
            a1.abc(3.5);
    }
}
```

Explain: -03

```
class A{
    void abc(){
        System.out.println("ABC 1");
    }
    void abc(int i){
        System.out.println("ABC 2");
    }
}
class B extends A{
    void abc(byte b){
        System.out.println("ABC 3");
    }
}
```

Same name Different parameters පවතින method භාවිතා කළ හැක්කේ same class වල සහ sub class තුල පමණි.

Explain: -04

```
class A{
    void m(String s){
        System.out.println("String");
    }
    void m(A a){
        System.out.println("A");
    }
    public static void main(String[] args) {
        A a =new A();
        a.m(null);
    }
}
```

- මෙම අවස්ථාවේ දී null යන parameter value එක m(A a) & m(String s) යන methods දෙකටම match වන නිසා පහත error එක ලැබේ.

```
C:\Users\PlayBoy\Desktop\Day 20>javac overloadin4.java
overloadin4.java:10: reference to m is ambiguous, both method m(java.lang.String) in A and method m(A) in A match
            a.m(null);
              ^
```

Explain: -05

```
class A{
    void m(String s){
        System.out.println("String");
    }
    void m(A a){
        System.out.println("A");
    }
    public static void main(String[] args) {
        A a =new A();
        //a.m(null);
        a.m(new A());
        a.m("sds ");
    }
}
```

Overriding

Rules for overriding a method

- Argument list exactly match
- Same or wider access level
- Same or narrower checked exceptions
- Instance method can be overridden only if they are inherited by the subclass
- The return type must be the same as, or a subtype of, the return type declared in the original overridden method in the superclass – covariant returns.
- You cannot override a method marked final.
- You cannot override a method marked static.
- The overriding method CAN throw any unchecked (runtime) exception, regardless of whether the overridden method declares the exceptions.

Ex: -01

Rule: -Argument list exactly match

```
class A{
    void abc(){
        System.out.println("ABC I");
    }
}
class B extends A{
    void abc(){
        System.out.println("ABC 3");
    }
}
class Test{
    public static void main(String[] args) {
        A a =new A();
            a.abc();
        B b1=new B();
            b1.abc();

    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 20>java Test
ABC I
ABC 3
```

Explain: -02

Rule: - Same or wider access level

```
1  class A{
2      public void abc(){
3          System.out.println("ABC I");
4      }
5  }
6  class B extends A{
7      void abc(){
8          System.out.println("ABC 3");
9      }
10 }
11 class Test{
12     public static void main(String[] args) {
13
14     }
15 }
```

```
C:\Users\PlayBoy\Desktop\Day 20\overriding2.java:7: abc() in B cannot override
abc() in A; attempting to assign weaker access privileges; was public
    void abc(){
         ^
1 error
```

```java
class A{
    public void abc(){
        System.out.println("ABC I");
    }
}
class B extends A{
    private void abc(){
        System.out.println("ABC 3");
    }
}
class Test{
    public static void main(String[] args) {

    }
}
```

```
:\Users\PlayBoy\Desktop\Day 20\overriding2.java:7: abc() in B cannot override
bc() in A; attempting to assign weaker access privileges; was public
   private void abc(){
                ^
```

```java
class A{
    public void abc(){
        System.out.println("ABC I");
    }
}
class B extends A{
    protected void abc(){
        System.out.println("ABC 3");
    }
}
class Test{
    public static void main(String[] args) {

    }
}
```

```
:\Users\PlayBoy\Desktop\Day 20\overriding2.java:7: abc() in B cannot override
bc() in A; attempting to assign weaker access privileges; was public
   protected void abc(){
                  ^
```

overriding2.java

```
1  class A{
2      private void abc(){
3          System.out.println("ABC I");
4      }
5  }
6  class B extends A{
7      protected void abc(){
8          System.out.println("ABC 3");
9      }
10 }
11 class Test{
12     public static void main(String[] args) {
13
14     }
15 }
```

```
[Finished in 0.3s]
```

```
1  class A{
2      private void abc(){
3          System.out.println("ABC I");
4      }
5  }
6  class B extends A{
7      void abc(){
8          System.out.println("ABC 3");
9      }
10 }
11 class Test{
12     public static void main(String[] args) {
13
14     }
15 }
```

```
Finished in 0.6s]
```

මෙහි දී **access level** එක ඊට සමාන හෝ වැඩිවිය යුතුය. එසේ නොවන සියල්ලම **access** කිරීමට නොහැක. මෙහි දක්වා ඇති **ex** වලින් එය මනාව පැහැදිලි වේ.

```java
class A{
    protected void abc(){
        System.out.println("ABC I");
    }
}
class B extends A{
    void abc(){
        System.out.println("ABC 3");
    }
}
class Test{
    public static void main(String[] args) {

    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 20\overriding2.java:7: abc() in B cannot override
abc() in A; attempting to assign weaker access privileges; was protected
    void abc(){
         ^
```

```java
class A{
    protected void abc(){
        System.out.println("ABC I");
    }
}
class B extends A{
    public void abc(){
        System.out.println("ABC 3");
    }
}
class Test{
    public static void main(String[] args) {

    }
}
```

```
Finished in 0.4s]
```

Explain: -03

Same or narrower checked exceptions---- Study After exception lesson

Explain: -04

Instance method can be overridden only if they are inherited by the subclass

Explain: -05

The return type must be the same as, or a subtype of, the return type declared in the original overridden method in the superclass – covariant returns.

Ex:-01

```
class K{
    int myMethod(){
        return 20;
    }
}
class J extends K{
    int myMethod(){
        return 20;
    }
}
```

```
1  class K{
2      int myMethod(){
3          return 20;
4      }
5  }
6  class J extends K{
7      byte myMethod(){
8          return 20;
9      }
```

```
C:\Users\PlayBoy\Desktop\Day
20\overriding5_1.java:7: myMethod() in J
cannot override myMethod() in K;
ttempting to use incompatible return type
ound    : byte
equired: int
    byte myMethod(){
         ^
 error
Finished in 0.4s]
```

Ex: -02

```
1  class X{
2
3  }
4  class Y extends X{
5
6  }
7  class A{
8      X my(){
9          X x1=new X();
10         return x1;
11     }
12 }
13
14 class B extends A{
15     X my(){
16         X x2=new X();
17         return x2;
18     }
19 }
```

```
[Finished in 0.4s]
```

Ex: -03

```
1   class X{
2
3   }
4   class Y extends X{
5
6   }
7   class A{
8       X my(){
9           X x1=new X();
10          return x1;
11      }
12  }
13
14  class B extends A{
15      Y my(){
16          Y y1=new Y();
17          return y1;
18      }
19  }
[Finished in 0.4s]
```

Ex: -04

```
class X{}
class Y extends X{}
class A{
    Object my(){
        X x1=new X();
        return x1;
    }
}

class B extends A{
    Y my(){
        Y y1=new Y();
        return y1;
    }
}
```

```
class X{}
class Y extends X{}

class A{
    X m(){
        return new X();
    }
}
class B extends A{
    X m(){
        return new X();
    }
    public static void main(String[] args) {
        System.out.println(new B().m());
        System.out.println(new A().m());
    }
}
```

Super class එකේ පවතින return type එකම හෝ එහි subtype එකක් පැවතිය යුතුය. sub class එක ඇතුලේ පැවතිය යුතුය.

Explain: -05

Rule: -You cannot override a method marked final.

final යන **non access modifiers** භාවිත කළ විට **methods or variables values** වෙනස් කළ නොහැක.

```
1  class A{
2      final void my(){}
3  }
4  class B extends A{
5      void my(){}
6  }
```

```
C:\Users\PlayBoy\Desktop\Day 20\overriding5_13.java:5: my() in B cannot override
my() in A; overridden method is final
    void my(){}
         ^
1 error
[Finished in 0.4s]
```

Explain:-06

Rule: - You cannot override a method marked static.

```
class A{
    static void my(){}
}
class B extends A{
    static void my(){}
}
```

static area එක තුල මෙය **load** වෙන නිසා මෙය **compile** වේ.

Polymorphism

Polymorphism යන වචනය ග්‍රීක වචනය වන අතර මෙයින් කියවෙන්නේ ‘‘බහුරූපීඵවයි’’ (Many Forms).

මෙම සංකල්පය සඳහා inheritance භාවිතා වේ.

Explain: -01

```
class A{}
class B extends A{}
class Test{
    public static void main(String[] args) {
        A a0=new A();
        A a1=new B();//Polymorphism Behaviour_

    }
}
```

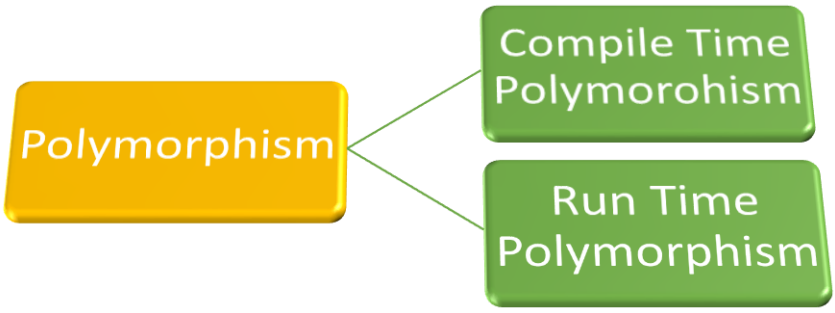
මෙහි object reference variable type එක අවස්ථා 2 දිම A වේ. නමුත් a0 සහ a1 හි values වෙනස් වේ. එනම් බහුරූපී අවස්ථාවකි. මෙය polymorphism වශයෙන් හඳුන්වයි.

මෙහි B class එකේ super class එක A වේ. නමුත් පහත අවස්ථාව සිදුවිය නොහැක.

```
1  class A{}
2  class B extends A{}
3  class Test{
4      public static void main(String[] args) {
5          B b1=new A();_
6
7      }
8  }
```

```
C:\Users\PlayBoy\Desktop\Day 21\TestJava3.java:5: incompatible types
found   : A
required: B
        B b1=new A();
             ^
```

Polymorphism වර්ග 2ට කොටස් කරනු ලබයි.



```
class A{
    void ab(){System.out.println("ab");}
    void ab(int i){System.out.println("ab(int)");}
}
class B extends A{
    void ab(String s){System.out.println("ab(String)");}
}
class Test{
    static public void main(String []args){
        A a0=new A();
        B b1=new B();
        a0.ab();
        a0.ab(10);
        b1.ab();
        b1.ab(10);
        b1.ab("Hello");
    }
}
```

Run Time Polymorphism

Explain: -01

```
package polymorphism;

class A{
    void ab(){
        System.out.println("ab()in A");
    }
}
class B extends A{

}
public class Test{
    public static void main(String[] args) {
        A a1=new B();
        a1.ab();

    }
}
```

```
B
ut - Polymorphism (run) ×
run:
ab()in A
```

Explain: -03

```
package polymorphism;

class A{
}
class B extends A{
    void ab(){
        System.out.println("ab()in B");
    }
}
public class Test{
    public static void main(String[] args) {
        A a1=new B();
        a1.ab();

    }
}
```

```
A
ut - Polymorphism (run) ×
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - Erroneous sym type: polymorphism.A.ab
```

ඉහත ex එකේ supper class A හි ab() method එක ඉවත් කළ විට compiler විසින් errors ලබාදේ. නමුත් super class එක තුල ab() method එක පවතින විට compile error නොලේබෙන නමුත් sub class එක වන B හි පවතින ab method එක ක්‍රියාත්මක වේ. මෙම කොටස් run time polymorphism වශයෙන් හඳුන්වයි.

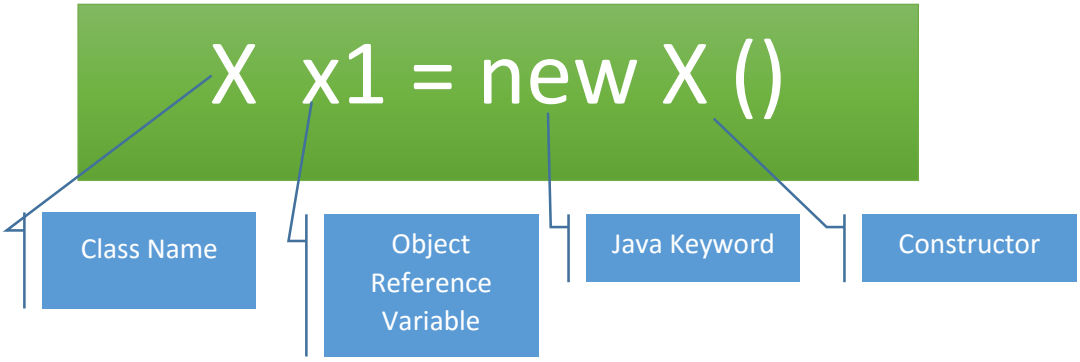
```
1
2   package polymorphism;
3
4   class A{
5       void ab(){
6           System.out.println("ab()in A");
7       }
8   }
9   class B extends A{
10      void ab(){
11          System.out.println("ab()in B");
12      }
13  }
14  public class Test{
15      public static void main(String[] args) {
16          A a1=new B();
17          a1.ab();
18
19      }
20  }
```

```
Test
Output - Polymorphism (run) ×
run:
ab()in B
```

Constructors

Explain: -01

```
class X{
    X(){System.out.println("X cons");
    }
    public static void main(String[] args) {
        X x1=new X();
        System.out.println("main method");
    }
}
```



Default Constructor

```
class X{
    X(){super();
    }
    public static void main(String[] args) {
        X x1=new X();
        System.out.println("main method");
    }
}
```

මෙහි highlight කර ඇති කොටස default constructor එක වේ. මෙය compiler විසින් automatically මෙය නිර්මාණය කරගැනී. එය තුල super() statement එකත් සමගම පවතී.

Default constructor එක සලකීමෙදි, එය සමානයන් පවත්වගන්යේ constructor එක පවතින class එකේ access level එකය. අවශ්‍ය නම් එහි access level එක වෙනස් කරගත හැක.

Explain: - 01

```
1  class X{
2      X(){
3          System.out.println("X cons...");
4          super();
5      }
6      public static void main(String[] args) {
7          X x1=new X();
8          System.out.println("main method");
9      }
10 }
```

```
C:\Users\PlayBoy\Desktop\Day 21\constructors_2.java:4: call to super must be
first statement in constructor
        super();
              ^
```

Default constructor එකේදී **super(); statement** එක පැවතිය යුත්තේ පලමු පේලිය තුලය. එසේ නොවේනම් ඉහත **compile error** එක ලැබේ. **this** යන **keyword** එකද භාවිතා කළ හැක.

Constructor chain

Sacasvasvasvasv

Explain: -01

```
class Y{
    private Y(){
        super();
        System.out.println("Y constr..");
    }
}
class X extends Y{
    public X(){
        super();
     }
    int i=10;
    public static void main(String[] args) {
        X x1=new X();
        System.out.println("main method");
        System.out.println(x1.i);
    }
}
```

```
sers\PlayBoy\Desktop\Day 21\constructors_5.java:9: Y() has private access in Y
    super();
    ^
```

Explain: -02

```
class Y{
    public Y(){
        super();
        System.out.println("Y constr..");
    }
}
class X extends Y{
    protected X(){
        super();
     }
    int i=10;
    public static void main(String[] args) {
        X x1=new X();
        System.out.println("main method");
        System.out.println(x1.i);
    }
}
```

```
ished in 0.4s]
```

Explain :-02

```java
class A{
    A(){System.out.println("A");}
}
class B extends A{
    B(){
        System.out.println("B");
    }
    public static void main(String[] args) {
        B b=new B();
    }
}
```

Command Prompt

```
C:\Users\PlayBoy\Desktop\Day 21>javac constructors_6.java

C:\Users\PlayBoy\Desktop\Day 21>java B
A
B

C:\Users\PlayBoy\Desktop\Day 21>
```

Explain: -04

```java
class X{
    X(){
        super();
        i=500;
    }
    int i;
    public static void main(String[] args) {
        X x1=new X();
        System.out.println("Main Method");
        System.out.println(x1.i);
    }
}
```

Command Prompt

```
C:\Users\PlayBoy\Desktop\Day 21>javac constructors_7.java

C:\Users\PlayBoy\Desktop\Day 21>java X
Main Method
500

C:\Users\PlayBoy\Desktop\Day 21>
```

Constructor මගින් **object** එකක් නිර්මාණය කිරීමෙදී එයට අවශ්‍ය වෙනස්කම් සිදු කළ හැක.

Explain: -05

```
1  class X{
2      X(String s){
3          super();
4          i=500;
5      }
6      int i;
7      public static void main(String[] args) {
8          X x1=new X();
9          System.out.println("Main Method");
10         System.out.println(x1.i);
11     }
12 }
```

```
C:\Users\PlayBoy\Desktop\Day 21\constructors_8.java:8: cannot find symbol
symbol  : constructor X()
location: class X
        X x1=new X();
             ^
```

Constructor parameter එකට ගැලපෙන අගයන් ලබා දිය යුතුය.

```
class X{
    X(String s){
        super();
        i=500;
    }
    int i;
    public static void main(String[] args) {
        X x1=new X("Hello");
        System.out.println("Main Method");
        System.out.println(x1.i);
    }
}
```

```
ished in 0.4s]
```

Constructing overloading

```
class X{
    X(String s){
        super();
        i=500;
        System.out.println(s);
    }
    X(){
        super();
    }
    int i;
    public static void main(String[] args) {
        X x1=new X();
        System.out.println("Main Method");
        System.out.println(x1.i);
    }
}
```

```
shed in 0.3s]
```

constructing භාවිත කරමින object නිර්මාණය කළ හැක. මෙලෙස සිදුකරන ක්‍රියාව constructing overloading නම් වේ.

Explain: -06

```
1  class M{
2      M(int i){
3          System.out.println(i);
4          System.out.println("cons M(int)");
5      }
6  }
7  class N extends M{
8      N(){
9          super();
10         System.out.println("cons N()");
11     }
12 }
13 class Test{
14     public static void main(String[] args) {
15         N n1=new N();
16     }
17 }
```

```
C:\Users\PlayBoy\Desktop\Day 21\constructors_9_test.java:9: cannot find symbol
symbol  : constructor M()
location: class M
        super();
        ^
```

මෙහි N() constructer එක එහි super සොයාගෙන යන අතර M class එකේ default constructer එකක් නොමැති නිසා error ලැබේ.

Explain:-07

```
class M{
    M(int i){
        System.out.println(i);
        System.out.println("cons M(int)");}
    M(String s1){
        System.out.println(s1);
        System.out.println("cons M(String)");}
}
class N extends M{
    N(){
        super("Hellow");
        System.out.println("cons N()");}
    N(int xx){
        super(xx);
        System.out.println("N(int)cons");}
}
class Test{
    public static void main(String[] args) {
        N n1=new N();
        N  n2=new N(150);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 21>javac cons

C:\Users\PlayBoy\Desktop\Day 21>java Test
Hellow
cons M(String)
cons N()
150
cons M(int)
N(int)cons

C:\Users\PlayBoy\Desktop\Day 21>
```

Explain:08

This

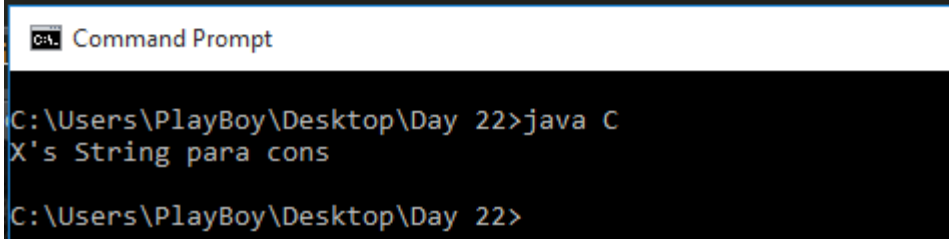
```
class N{
    N(){
        this(10);
        System.out.println("Cons N()");
    }
    N(int x){
        super();
        System.out.println("Cons N(int)");
    }
}
class Test{
    public static void main(String[] args) {
        N n1=new N();
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 21>javac thsiscons.java

C:\Users\PlayBoy\Desktop\Day 21>java Test
Cons N(int)
Cons N()
```

Explain: -08

```
class X{
    X(){
        System.out.println("X's default cons");
    }
    X(int i){
        System.out.println("X's int para cons");
    }
    X(String s){
        System.out.println("X's String para cons");
    }
}
class A{
    static X x1=new X();
}
class B{
    static X x2=new X(10);
}
class C{
    static X x3=new X("ABC");
    public static void main(String[] args) {

    }
}
```

Run වීමට අවශ්‍ය classes වල static දේ පමනක ලෝඩ් වේ.



Explain: -09

```
class X{
    X(){
        System.out.println("X's default cons");
    }
    X(int i){
        System.out.println("X's int para cons");
    }
    X(String s){
        System.out.println("X's String para cons");
    }
}
class A{
    static X x1=new X();
}
class B{
    static X x2=new X(10);
}
class C{
    static X x3=new X("ABC");
    public static void main(String[] args) {
        System.out.println("Main Method");
        A a1=new A();
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 22>javac Test_02.java

C:\Users\PlayBoy\Desktop\Day 22>java C
X's String para cons
Main Method
X's default cons

C:\Users\PlayBoy\Desktop\Day 22>
```

```
class X{
    X(){
        System.out.println("X's default cons");
    }
    X(int i){
        System.out.println("X's int para cons");
    }
    X(String s){
        System.out.println("X's String para cons");
    }
}
class A{
    static X x1=new X();
}
class B extends A{
    static X x2=new X(10);
    B(){
        System.out.println("B's conster");
    }
}
class C extends B{
    static X x3=new X("ABC");
    public static void main(String[] args) {
        System.out.println("Main Method");
        C c1=new C();
```

```
Select C:\Windows\system32\cmd.exe

C:\Users\PlayBoy\Desktop\Day 22>java C
X's default cons
X's int para cons
X's String para cons
Main Method
B's conster
```

```java
class X{
    X(){System.out.println("X's default cons");}
    X(int i){System.out.println("X's int para cons");}
    X(String s){System.out.println("X's String para cons");}
}
class A{
    static X x1=new X();
}
class B extends A{
    static X x2=new X(10);
    B(){System.out.println("B's conster");}
}
class C extends B{
    C(){System.out.println("C's conster");}
    static X x3=new X("ABC");
    public static void main(String[] args) {
        System.out.println("Main Method");
        C c1=new C();
    }
}
```

```
C:\Windows\system32\cmd.exe

C:\Users\PlayBoy\Desktop\Day 22>javac Test_04.java

C:\Users\PlayBoy\Desktop\Day 22>java C
X's default cons
X's int para cons
X's String para cons
Main Method
B's conster
C's conster
```

```
class A{
        A(){
                Super();
        }
}
```

With private Constructors

Abstraction

Explain: -01

මෙම උදාහරණයට අනුව තමන්ගේ class එකේ object නිර්මාණය කළ නොහැකිය.

```
abstract class A{
    int i=10;
    public static void main(String[] args) {
        System.out.println("Hi...!");
        A a1=new A();
        System.out.println(a1.i);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 22\Abastraction.java:5: A is abstract; cannot be instantiated
        A a1=new A();
          ^
  1 error
```

Constructors පවතී.

```
abstract class A{
    A(){
        super();
        System.out.println("A");
    }
    int i=10;
}
class B{
    public static void main(String[] args) {
        System.out.println("Hi !!!!");
    }
}
```

```
[Finished in 0.4s]
```

Explain: -02

```
abstract class A{
    A(){
        super();
        System.out.println("A");
    }
    int i=10;
}
class B extends A{
    public static void main(String[] args) {
        System.out.println("Hi !!!!");
            B b1=new B();
        System.out.println(b1.i);
    }
}
```

```
C:\Windows\system32\cmd.exe

C:\Users\PlayBoy\Desktop\Day 22>javac Abastraction_04.java

C:\Users\PlayBoy\Desktop\Day 22>java B
Hi !!!!
A
10
```

මෙහිදී වෙනත් class වල පවතින components accesses කළ හැක්කේ extends වොන්වෙප්ට් එක භාවිතා කළ යුතුය. Inheritance භාවිතයෙන් access කළ හැක.

Explain: -03

```
abstract class A{
    A(){
        super();
        System.out.println("A");
    }
    int i=10;
    static int x=20;
    static void ab(){
        System.out.println("a");
    }
    void xy(){
        System.out.println("b");
    }
}
class B extends A{
    public static void main(String[] args) {
        B b1=new B();
        System.out.println(b1.i);
        System.out.println(A.x);
        B.ab();
        b1.xy();
    }
}
```

```
Select C:\Windows\system32\cmd.exe

C:\Users\PlayBoy\Desktop\Day 22>javac Abastraction_04

C:\Users\PlayBoy\Desktop\Day 22>java B
A
10
20
a
b
```

Explain:-04

```
abstract class A{
    void xy(){
        System.out.println("b");
    }
    abstract void myMethod();
    abstract int myMethod1(int x,int y);
    abstract String myMethod2(String s,int i);
}
```

abstract class ඇතුලත abstract methods පැවතිය හැක.

එනම් අස්ම්පුර්න class ඇතුලේ අසම්පුර්ණ methods පැවතිය හැක.

Explain:-05

Abstract method පවතින විට class extends කළ නොහැක.

```
abstract class A{
    void xy(){System.out.println("b");}

    abstract void ab();
}
class B extends A{

}
```

```
C:\Users\PlayBoy\Desktop\Day 22\Abastraction_06.java:6: B is not abstract and does not
override abstract method ab() in A
class B extends A{
^
1 error
[Finished in 0.4s]
```

Explain:-06

```
abstract class A{
    void xy(){System.out.println("b");}

    abstract void ab();
}
abstract class B extends A{

}
```

```
nished in 0.4s]
```

B class එක abstract කර ගනිම මගින් compile error ඉවත් කර ගත්තද.., නමුත් එය තුල object නිර්මාණය කළ නොහැක.

```
1  abstract class A{
2      void xy(){System.out.println("b");}
3
4      abstract void ab();
5  }
6  abstract class B extends A{
7      B b1=new B();
8
9  }
```

```
C:\Users\PlayBoy\Desktop\Day 22\Abastraction_06.java:7: B is abstract; cannot be instantiated
    B b1=new B();
         ^
```

ඉහත ගැටලුව විසඳීමට පහත අයුරින් extends කරගත් class එක ඇතුලත abstract method එක method body එක සමග නිර්මාණය කරගන අවශ්‍ය වෙනස්කම් සිදු කළ හැක.

```java
abstract class A{
    void xy(){System.out.println("b");}

    abstract void ab();
}
class B extends A{
    void ab(){
        System.out.println("Hi Bye");
    }
    public static void main(String[] args) {
        B b1=new B();
          b1.ab();
    }

}
```

Interface

Interface බොහෝ සෙයින් fully abstract class එකක් මෙනි. මෙය තුල main method ,method හෝ constructions නොපවතී. ඒ නිසා object නිර්මාණය සිදු කළ නොහැක. නමුත් වෙනත් class එකක් ඇතුලත main methods පැවතිය හැක.

Ex:-01

```java
interface ABC{
    RV(){}
}
```

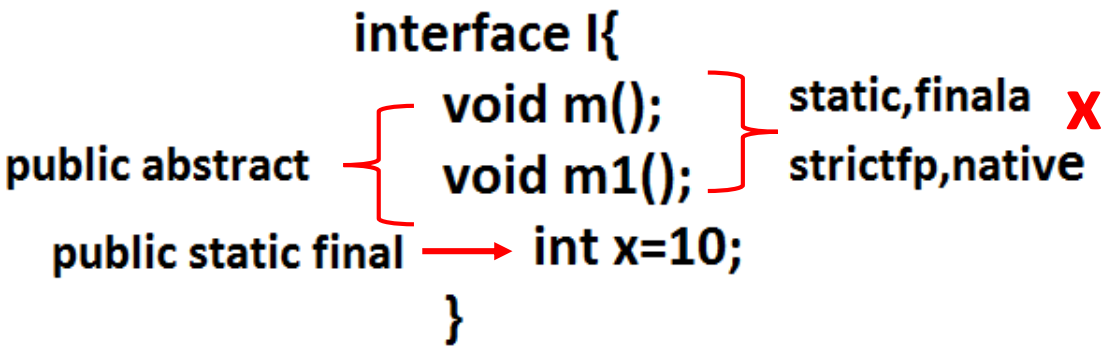
```
C:\Users\PlayBoy\Desktop\Day 23\Interface2.java:2: <identifier> expected
    RV(){}
      ^
```

Ex:-02

```java
interface ABC{}
class B{
    public static void main(String[] args) {
        ABC abc=new ABC();
    }
}
```

```
:\Users\PlayBoy\Desktop\Day 23\Interface2.java:4: ABC is abstract; cannot be instantiated
        ABC abc=new ABC();
                ^
```

මෙහිදී abstract යන keyword එක interface මුලට යොදා compile කලද අවුලක් නොමැත.



Explain:-01

```
interface ABC{
    void a();
    int b(int x,int y);
        String s="AAA";
        double d=10.56;
}
class B{
    public static void main(String[] args) {

    }
}
```

```
[Finished in 0.4s]
```

- Interface තුල පවතින සියලුම method, abstract methods වේ.
- පවතින සියලුම variables සියල්ලම "public static final " මට්ටමේ පවතී.

මෙහිදී compiler මහින් මෙම keywords සියල්ල compiler විසින් default add කරගනී.

Ex:-03

```
interface ABC{
    void a();
    int b(int x,int y);
        String s="AAA";
        double d=10.56;
}
class B{
    public static void main(String[] args) {
        System.out.println(ABC.s);
        System.out.println(ABC.d);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 23>java B
AAA
10.56
```

```
interface ABC{
    void bounce();
    public void bounce1();
    abstract void bounce2();
    public abstract void bounce3();
    abstract public void bounce4();
}
```

```
[Finished in 0.3s]
```

- **Interface** 2ක් අතර **is a relationship** එකක් ගොඩ නැගීමේදී භාවිතා කරනුයේ **extends** යන **keyword** එකය.
- **Class** එකකට **interface** එකක් **super type** එකක් වශයෙන් භාවිත කළ හැක. එහිදී භාවිතා කරන්නේ **implements** යන **keyword** එකය.

```
interface ABC{
    void m();
    String s="interface ABC";
    final public static double d=10.56;
}
interface DEF extends ABC {
        void m1();
        String s1="interface DEF";
}
class Test{
    public static void main(String[] args) {
        System.out.println(ABC.s+" & "+ ABC.d);
        System.out.println(DEF.s1);

    }
}
```

- නමුත් **interface** එකකට **super type** ලෙස **class** එකක් පැවතිය නොහැක.

Explain:- 02

```
interface A{
    void m1();

}
interface B{
    void m2();
}
interface C{
    void m3();
}
class D extends Z implements A,B,C{

}
```

Class එකකට **extends** කළ හැක්කේ එක **class** එකක් වුවද **interface** රාශියක් **implements** කළ හැක.

තවද **extends** කිරීමෙන් අනතුරුවයි **interface implement** කළ යුත්තේ.

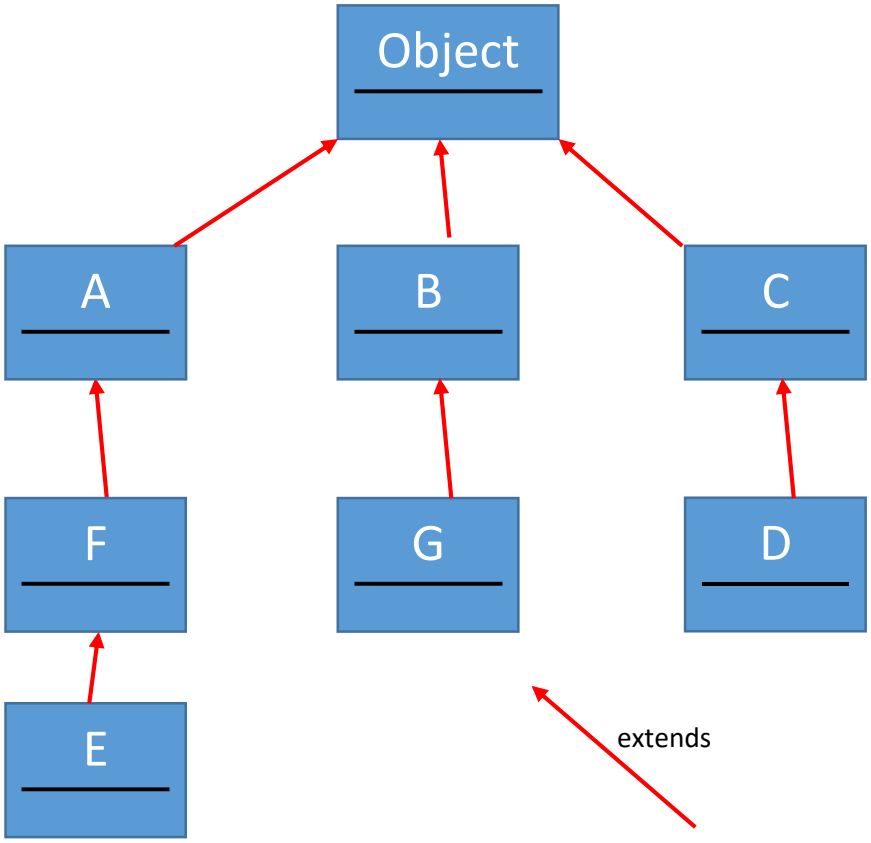
Interface එකකට තව **interface** කීපයක් **extends** කරගැනීමටද හැකියාව ඇත.

```
interface A{
    void m1();

}
interface B{
    void m2();
}
interface C extends A,B{
    void m3();
}

[Finished in 0.4s]
```

Class Diagrams

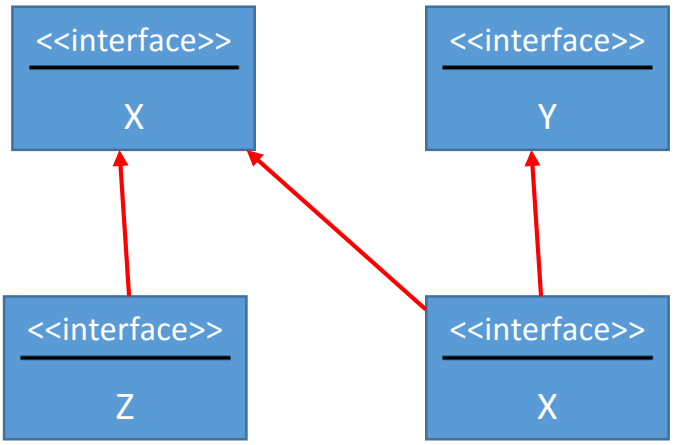
Ex:-01

```
class A{}
class B{}
class C{}
class D extends C{}
class E extends F{}
class F extends A{}
class G extends B{}
```



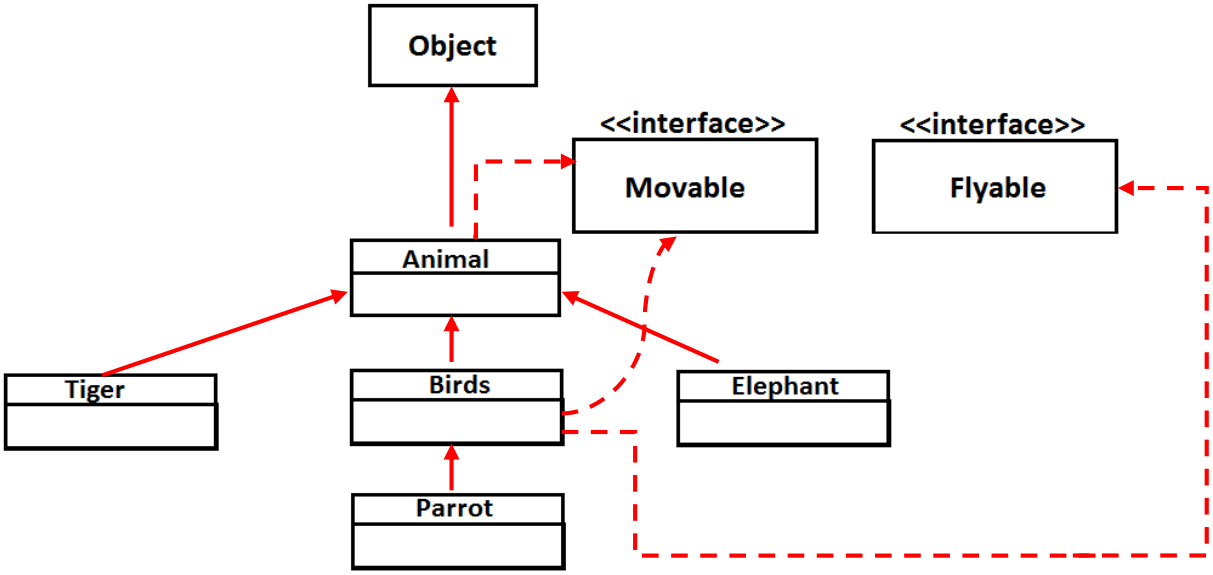
Ex:-02

```
interface X{}
interface Y{}
interface Z extends X{}
interface U extends X,Y{}
```



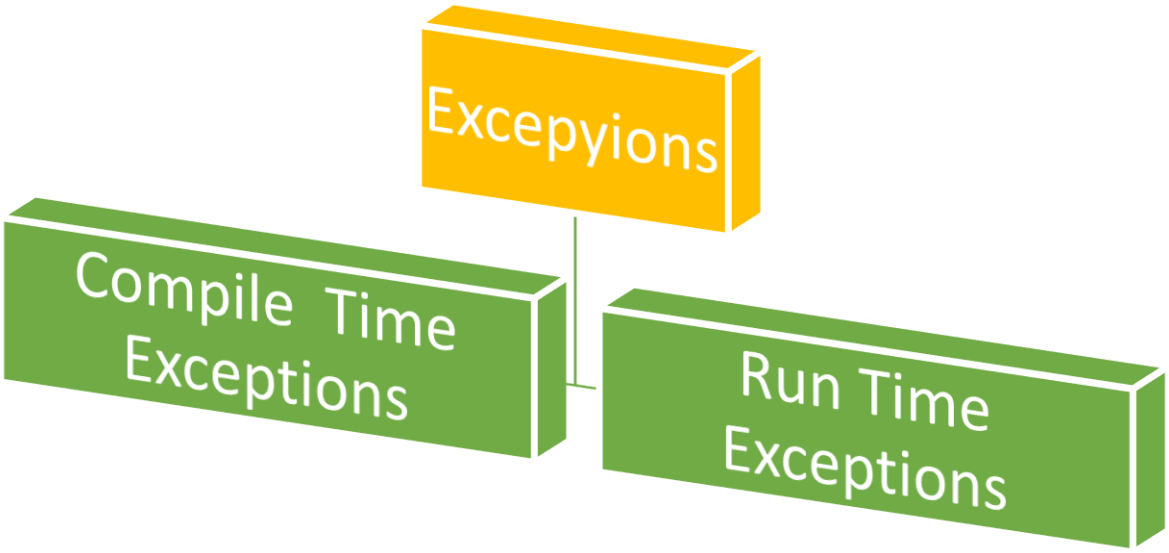
Ex:-03

```
class Elephant extends Animal{}
class Tiger extends Animal{}
class Parrot extends Birds{}
class Animal implements Movable{
    public void move(){
        System.out.println("Moving....");
    }
}
class Birds extends Animal implements Flyable,Movable{
    public void move(){
        System.out.println("Moving....");
    }
    public void fly(){
        System.out.println("flying..");
    }
}

interface Movable{
    void move();
}
interface Flyable{
    void fly();
}
```



Exceptions Handling

පරිගණක ක්‍රම ලේඛන භාවිතයෙන් විවිධ වැඩසටහන් නිර්මනය කල හැකි අතර එහිදී අඑතිවන ගැටළු exceptions වශයෙන හඳුන්වන අතර. මේවා run time & compile time වශයෙන් ආකාර 2ක් වේ.



මෙම exceptions handling මගින් ඇතිවන ගැටලු විසදීමට හැකි අතර එහිදී try-catch සහ throws යන keywords භාවිතා කරයි.

Explain:-01

```
class A{
    public static void main(String[] args) {
        System.out.println("Starting....");
            int i=10;
            int j=2;
        try{
            int z=i/j;
                System.out.println(z);
        }catch (Exception e) {
                System.out.println("Error...!!");
            }
        System.out.println("End");

    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 24>java A
Starting....
5
End
```

මෙහි දී j=2 බැවින් කිසිම වරදක නොමැතිව output ලැබේ.

නමුත් j=0 අවස්ථාව සලකීමෙදි z=i/j යන සමීකරණය මගින් ලැබෙන ප්‍රතිපලය පිළිබඳව පැහැදිළි ගැටලුවක් ඇත.

Try{}catch(){} block එක භාවිත කිරීමෙන් එය මගහරවාගත හැකි බව මෙම උදාහරණය මගින් පෙන්නුම් කරයි.

```
class A{
    public static void main(String[] args) {
        System.out.println("Starting....");
            int i=10;
            int j=0;
        try{
            int z=i/j;
                System.out.println(z);
        }catch (Exception e) {
                System.out.println("Error...!!");
            }
        System.out.println("End");

    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 24>java A
Starting....
Error...!!
End
```

Explain:-02

```
class A{
    public static void main(String[] args) {
        System.out.println("Starting....");
            int i=10;
            int j=0;
        try{
            int z=i/j;
                System.out.println(z);
        }catch (Exception e) {
                System.out.println("Error...!!");
                System.out.println(e);
            }
        System.out.println("End");
    }
}
```

```
Select Command Prompt

C:\Users\PlayBoy\Desktop\Day 24>java A
Starting....
Error...!!
java.lang.ArithmeticException: / by zero
End
```

පවතින error එක ලබාගැනීමට අවශ්‍යනම් කළ යුත්තේ exceptions name එකේ sop එකක් තුල යෙදීමය.

Explain:-03

```
class A{
    public static void main(String[] args) {
        System.out.println("Starting....");
            int i=10;
            int j=0;
        try{
            int z=i/j;
                System.out.println(z);
        }catch (ArithmeticException e) {
                System.out.println("Arith EX.!!");
                System.out.println(e);
        }catch (Exception ex) {
                System.out.println("Ex");
                System.out.println(ex);
            }
        System.out.println("End");
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 24>java A
Starting....
Arith EX.!!
java.lang.ArithmeticException: / by zero
End
```

Explain:-04

```
class A{
    public static void main(String[] args) {
        System.out.println("Starting....");
            int i=10;
            int j=0;
        try{
            int z=i/j;
                System.out.println(z);
        }catch (Exception ex) {
                System.out.println("Ex");
                System.out.println(ex);
        }catch (ArithmeticException e) {
                System.out.println("Arith EX.!!");
                System.out.println(e);
        }
        System.out.println("End");

    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 24\Exceptions_multicatch.java:12: exception java.lang.
ArithmeticException has already been caught
    }catch (ArithmeticException e) {
    ^
1 error
[Finished in 0.4s]
```

Catch අවශ්‍ය තරම් යෙදිය හැකිය. නමුත එය යෙධිය යුත්තේ කුඩා එකේ සිට විශාල එක දක්වාය.

Explain:-05

```
class A{
    public static void main(String[] args) {
        System.out.println("Starting....");
            int i=10;
            int j=0;
        try{
            int z=i/j;
                System.out.println(z);
        }catch (ArithmeticException e) {
                System.out.println("Arith EX.!!");
                System.out.println(e);
        }catch (Exception ex) {
                System.out.println("Ex");
                System.out.println(ex);
        }finally{
            System.out.println("P");
        }
        System.out.println("End");

    }
}
```

```
Select Command Prompt

C:\Users\PlayBoy\Desktop\Day 24>java A
Starting....
Arith EX.!!
java.lang.ArithmeticException: / by zero
P
End
```

finally යන block එක භාවිත කර විටදී එය try , catch දෙකෙන් කුමක් ක්‍රියාත්මක වුව ද finally block එක ක්‍රියාත්මක වේ.

```
class A{
    public static void main(String[] args) {
        System.out.println("Starting....");
            int i=10;
            int j=0;
        try{
            int z=i/j;
                System.out.println(z);
        }finally{
            System.out.println("P");
        }
        System.out.println("End");

    }
}
```

try , catch දෙකම ක්‍රියාත්මක නොවුව ද finally block එක වැඩකරයි.

Try එක නොමැතිව catch or finally යන block පැවතිය නොහැක.

Throws

```
class A{
    public static void main(String[] args) {
        B b1=new B();
        b1.ab();
    }
}
class B{
    void ab throws Exception(){
        System.out.println("Star");
        int x=10/0;
        System.out.println(x);
        System.out.println("Error");
    }
}
```

Exception එකක් සහිත code එකකට throws යන keyword එක බවිතාකළ විටදී එම code එකේ Error එකක් ඇති බවත් මේ සදහා try-catch එකක් බාවිතා කල යුතුබවත් error එකකින් දැනුම් දෙනු ලබය්).මෙමෙ keyword එක මගින් risky code එකක් බව අගවනු ලබය් මෙම .(keyword එක method එක තුල ඇතුලත් නොකරය් .Method එකේම name එකට පසුව මෙය යෙදේ.

```
class A{
    public static void main(String[] args) {
        A a1=new A();
        String s=a1.toString();
        System.out.println(s);
        System.out.println(a1.toString());
        System.out.println(a1);
    }
}
```

```
C:\Users\Shehan Nawoda\Desktop\New folder (3)\Private\Garbage Collection>java A
A@75e4f66a
A@75e4f66a
A@75e4f66a
```

මෙම toString name එක මෙය .වන එකකි inherite එකකින් Object class අපට අවශ්‍ය පරිදි override කල හැකපහත . දක්වා ඇත්තේකරනලද උදාහරණ ovarride යකි.

```
class A{
    public static void main(String[] args) {
        A a1=new A();
        String s=a1.toString();
        System.out.println(s);
        System.out.println(a1.toString());
        System.out.println(a1);
    }
    public String toString(){
        return "i am toStringMethod";
    }
    void a(){
        System.out.println("asaaadfsuhovsegioshgoierhisxhbuiryghso8rguu");
    }
}
```

```
C:\Users\Shehan Nawoda\Desktop\New folder (3)\Private\Garbage Collection>java A
i am toStringMethod
i am toStringMethod
i am toStringMethod
```

Garbage Collection

Java garbage collector විසින් ස්වයංක්‍රීයව memory management එක සිදුකරනු ලැබේ එනම් . coderට සිතිමට කිසිවක් ඉතුරු නොකරයි.

When Dose the Garbage Collector Run

මෙම garbage collector ක්‍රියාකරනුයේ JVM එක හරහාය .JVM විසින් garbage collector run විය යුත්තේ කොතැනදිද යන්න තීරණය කරනු ලබයි. අපටද එය run කරන ලෙස call කල හැක නමුත් එය .නිශ්චිතවම සිදුවේ යැයි කිවනොහැක.

When Dose an Object become eligible for garbage collection

Garbage collector පැමිණ අනවශ්‍ය දෑ ඉවත් කිරීමට කාරණා න්4 මේ .ක් ඇත4 එකක් හෝ ඉටු වූ විට garbage collector පැමිණ memory යෙන් අනවශ්‍ය දෑ ඉවත් කරනු ලැබේ. එම නීති 4 පහත දැක්වේ.

1. Nulling Reference
2. Reassigning a Reference

3. Local Reference
4. Isolating a Reference

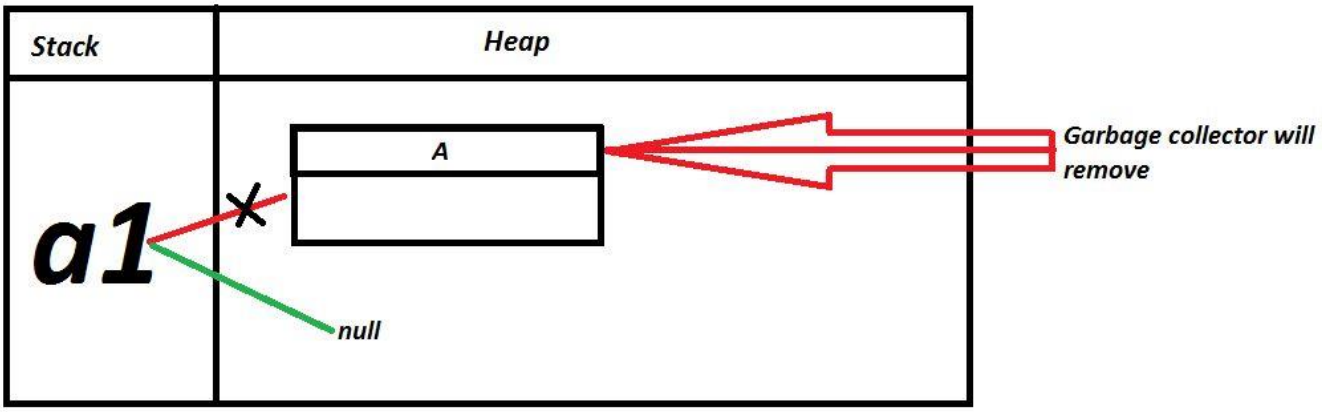
Nulling Reference

EX:-01

```
class A{
    public static void main(String[] args) {
        A a1=new A();
        a1=null;
        System.out.println(a1);
    }
}
```

```
C:\Users\Shehan Nawoda\D
null
```

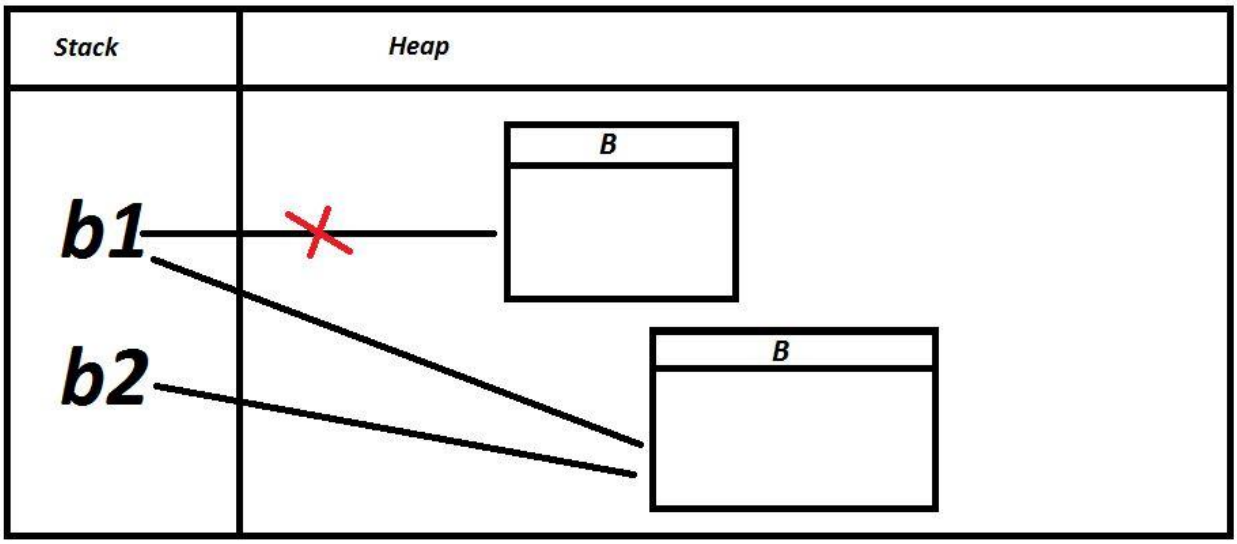
මෙම උදාහරණය අනුව පළමුව a1ගේ object එක ඉවත්කර a1ට null යන්න ආදේශ කරගනු ලැබේඑකේ ram එක object මෙහිදී . ඇති ස්ථානය පෙන්විය යුතු වුවද එය නොපෙන්වන්නේ garbage collector විසින් එය ඉවත් කරනු ලැබීම නිසාවෙනි ..a1ට null ආදේශ කර පෙන්වනු ලැබේ Ram .එක පහත දැක්වේ.



Reassigning a Reference

```
class A{
    public static void main(String[] args) {
    B b1=new B();
    B b2=new B();
    b1=b2;
    }
}
class B{}
```

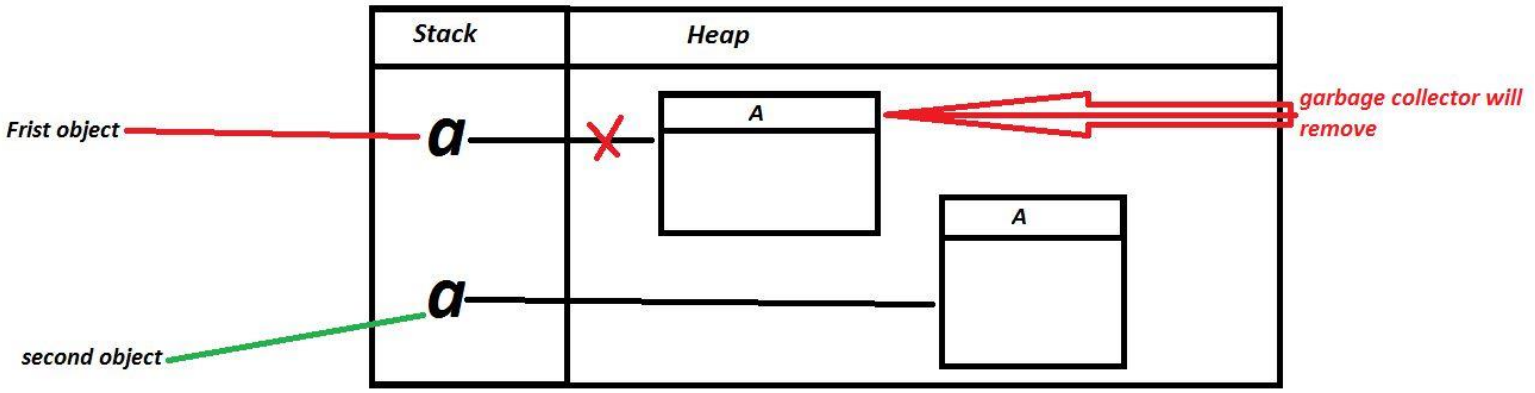
මෙහිදී object ලෙස B එකක් සැදේ .b1 සහ b2 ලෙස සෑදෙනB ගේ.දෙකක් වේ object B1=b2 ලෙස ආදේශ කර ඇති නිසා එක් එකක් ඉවත් object වේ .Garbage collector පැමිණි විගස එම ඉවත්එක object ඉවත් කරනුලැබේ.එක පහත දැක්වේ ram මෙහි .



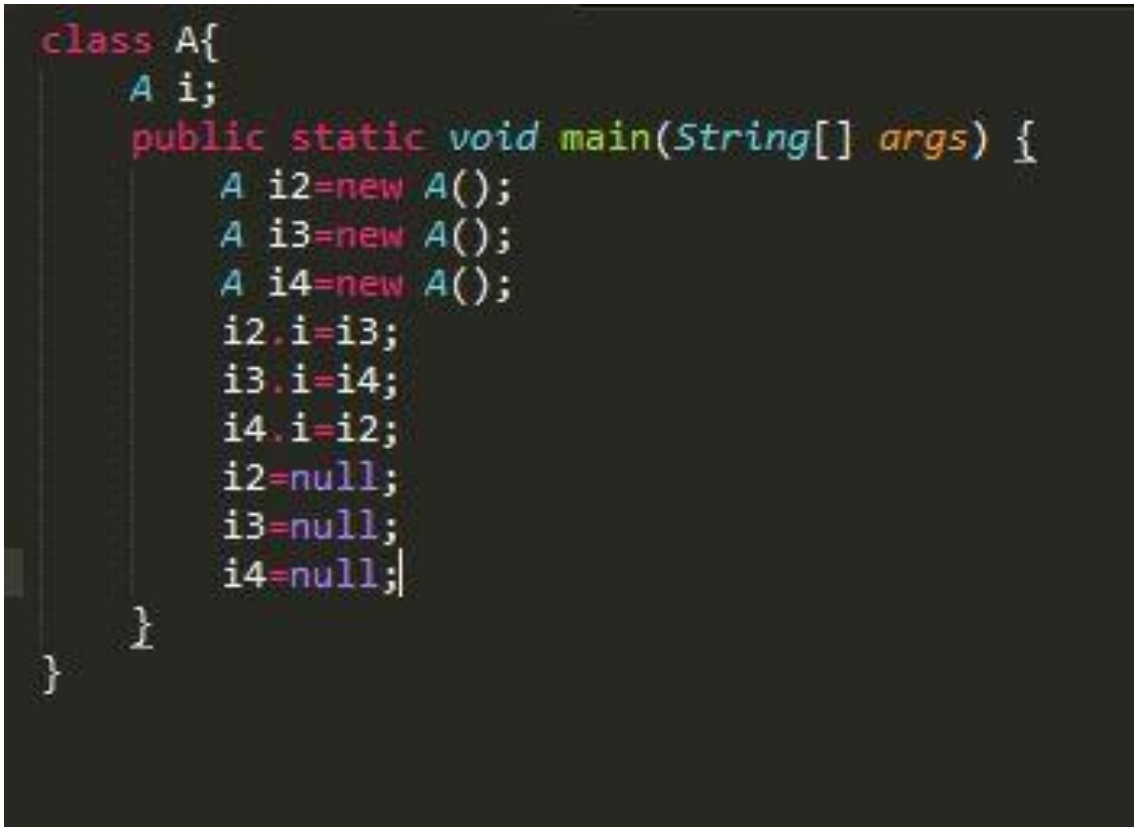
Local Reference

```
class A{
    static A m(){
        A a=new A();
        return a;
    }
    public static void main(String[] args) {
        System.out.println(m());
    }
}
```

මෙහිදී A ලෙස object එකක් සෑදේ නමුත් මෙහිදී call කර ඇත්තේ m() .එකයි method ලෙස a එබැවින් සෑදෙනඑක කථා කරන method අප විසින් .එක තනි වේ object වාරයක් පාසා අලුතින්objects හැදෙන බැවින් පරණ එම .බාවිතා නොවේ objects නිසා එම පරණ objectsgarbage collector විසින් ඉවත්කර දැමීමට ඉඩ ඇත. මෙහි Ram එක පහත දැක්වේ.



Isolating a Reference



Call for Garbage collector

Garbage collector JVM සෑම විටම එවනු නොලැබේ ලැබේඅපට . එය අවශ්‍ය වූ විට call කල හැක මේසදහා .System.gc(); ලෙස call කල යුතුවේ.

System.gc.();

EX:-01

```
class A{
    int i;
    public static void main(String[] args) {
        A a1=new A();
        a1.i=45;
        a1=new A();
        a1.i=30;
        System.out.println(a1.i);
        System.gc();
    }
}
```

Finalize()Method

මෙම method එක යොදා එකක් සැකසූ විට codegarbage collector ඒමට මොහොතකට කලින් මෙය Run වේ.

EX:-01

```
class A{
    int i;
    public static void main(String[] args) {
        A a1=new A();
        a1.i=45;
        a1=new A();
        a1.i=30;
        System.out.println(a1.i);
        System.gc();
    }
    public void finalize(){
        System.out.println("Garbage collector came:o!!!!!!!!");
    }
}
```

```
C:\Users\Shehan Nawoda\Desktop\New folder (3)\Private\Garbage Collection>java A
30
Garbage collector came:o!!!!!!!!
```

ඉහත කී පරිදි garbage collector එනවද නැද්ද යන්න බලාගත හැක්කේ මෙම finalize method එක මගිනි .Garbage collector පමිනුනහොත් ඊට පසුව finalize method එක run වන නිසා එම method එකේ ඇති දේවල් සිදුකරනු ලැබේකෙසේ . හෝ garbage collector නොපැමිණියහොත් එම finalize method එක run නොවේ.

```
class A{
    int i;
    public static void main(String[] args) {
        A a1=new A();
        a1.i=45;
        a1=new A();
        a1.i=30;
        System.out.println(a1.i);
    }
    public void finalize(){
        System.out.println("Garbage collector came:o!!!!!!!!");
    }
}
```

```
C:\Users\Shehan Nawoda\Desktop\New folder (3)\Private\Garbage Collection>java A
30
```

මෙහි දැක්වෙන්නේ garbage collector පැමිණියහොත් output එක 30 සහ Garbage collector ලෙස ලැබේ නොපැමිණියහොත් ..30 පමණක් ලැබේ.

Instance Of comparison

```
class A6{
    public static void main(String[] args) {
        String s = new String("SSS");
        System.out.println(s instanceof String);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java A6
true
```

```
class A6{
    public static void main(String[] args) {
        String s = new String("SSS");
        System.out.println(s instanceof String);
        String s1="AAAA";
        System.out.println(s1 instanceof String);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java A6
true
true
```

```
class A6{
    public static void main(String[] args) {
        String s = new String("SSS");
        System.out.println(s instanceof String);
        String s1="AAAA";
        System.out.println(s1 instanceof Object);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java A6
true
true
```

Instanceof යනු operator එකක් වන අතර එමගින් ලබාගැනීමට හැකිවන්නේ Boolean outputs පමණි.

Checks whether an object is of a particular type.

For object reference variable only.

Explain:-01

```
1  class A6{
2      public static void main(String[] args) {
3          String s = new String("SSS");
4          System.out.println(s instanceof String);
5          int i=10;
6          System.out.println(i instanceof Object);
7      }
8  }
```

```
C:\Users\PlayBoy\Desktop\Day 26\mmmInstanceOfComperisons.java:6: unexpected type
found   : int
required: reference
        System.out.println(i instanceof Object);
                           ^
```

```
1  class A6{
2      public static void main(String[] args) {
3          String s = new String("SSS");
4          System.out.println(s instanceof String);
5          int i=10;
6          System.out.println(i instanceof int);
7      }
8  }
```

```
C:\Users\PlayBoy\Desktop\Day 26\mmmInstanceOfComperisons_2.java:6: unexpected type
found   : int
required: reference
        System.out.println(i instanceof int);
                           ^
C:\Users\PlayBoy\Desktop\Day 26\mmmInstanceOfComperisons_2.java:6: unexpected type
found   : int
required: class or array
        System.out.println(i instanceof int);
                                        ^
```

කිසිම වෙලාවක primitive type variables මෙහිදී ගැලපීමක් සිදු නොවේ .Object type variables පමනක් මේ සඳහා ගැලපේ.

Explain:-02

```
class A{}
class B{
    public static void main(String[] args) {
        B x= new B();
        System.out.println(x instanceof A);
    }
}
```

```
C:\Users\PlayBoy\Desktop\Day 26\mmmInstanceOfComperisons_2.java:5: inconvertible types
found   : B
required: A
        System.out.println(x instanceof A);
                           ^
1 error
[Finished in 0.6s]
```

මෙහිදී මෙම error ලබිමට හේතුව ,

Class hierarchy එක සැලකීමේදී A හා B යනු කිසිම සම්බන්ධතාවයක් නැති වෙනත්ම කොටස් දෙකකි. අ ඒ නිසා x සහ A අතර සම්බන්ධයක් නොපවතී.

```
class A7{
    public static void main(String[] args) {
        int x[]=new int[5];
        System.out.println(x instanceof Object);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java A7
true
```

Explain:-02

```
interface I{}
class A{}
class B extends A{
    public static void main(String[] args) {
        B a=new B();
        System.out.println(a instanceof A);
        System.out.println(a instanceof I);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java B
true
false
```

මෙහි පළමු පර්තිඵලය true ලෙස ලබෙනුය් .Class hierarchy එක සැලකීමේදී B ගෙ super class එක A නිසාය.

දෙවනුව false ලැබෙනුයේ , interface I class hierarchy එකට සමබන්ධ නොවිම නිසාය.

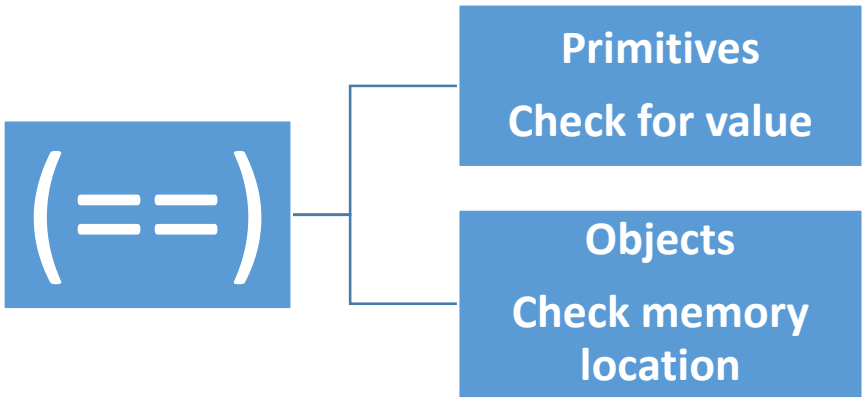
Explain:-03

```
class B{}
class A{
    public static void main(String[] args) {
        Object b=new B();
        System.out.println(b instanceof A);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java A
false
```

මෙහි A & B යන class දෙකෙම super class එක objects වේ. නමුත් b o.r.variable එක අල්ලන් ඉන්නේ B ගෙ සම්බන්ධයකි.

Equal to Operator



Explain:-01

```
class A4{
    public static void main(String[] args) {
        int i=10;
        byte b=10;
        float f=10.0f;
        char c='a';
        byte b1=97;
        System.out.println(i==b);
        System.out.println(i==b);
        System.out.println(c==b);
        System.out.println(c==b1);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java A4
true
true
false
true
```

primitive data types වලදී equal to (==) operator මගින් values සමානද යන්න පරීක්ෂ කරයි .

Explain:-02

```
class A5{
    int x=10;
    public static void main(String[] args) {
        A5 a=new A5();
        A5 a1=new A5();
        System.out.println(a==a1);
    }
}
```

```
Command Prompt

C:\Users\PlayBoy\Desktop\Day 26>java A5
false
```

Objective data types වලදී equal to (==) operator මගින් memory location සමානද යන්න පරීක්ෂ කරයි .

Source file Declamation Rules

1. Public class එකක් නිර්මාණය කළ විට එය save කළ යුත්තේ එම class එකේ නමින්මය.

ABCXYZ.java

```
public class ABCXYZ{
    public static void main(String[] args) {
        System.out.println("Hi !!!");
    }
}
```

- එසේ නොවේනම් පහත පරිදි error ලැබේ .Public class එකකින් බලාපොරොත්තු වෙන්නේ එය පොදුවේ භාවිතා කිරිඉමය.ඒ නිසා එය class name එකින්ම source file එක save කළ යුතුය.

SFD.java | ABCXYZ.java

```
1  public class ABCXYZ{
2      public static void main(String[] args) {
3          System.out.println("Hi !!!");
4      }
5  }
```

```
C:\Users\PlayBoy\Desktop\Day 26\SFD.java:1: class ABCXYZ is public, should be declared in a
file named ABCXYZ.java
public class ABCXYZ{
       ^
```

- Public class එකක් පවතින source file එකක් තුල වෙනත් class සහ interface පැවතීමට ඕනෑතරම් ඉඩ කඩ ඇත.

```
ABCXYZ.java
public class ABCXYZ{
    public static void main(String[] args) {
        System.out.println("Hi !!!");
    }
}
class A{}
class B{}
interface G{}
interface H{}
```

- Source file එකක් තුල පැවතිය හැක්කේ සැම විටම public class/interface එකක් පමණි. අනෙකුත් class හෝ interface ඕනෑතරම් පැවතිය හැක.

```
ABCXYZ.java
1  public class ABCXYZ{
2      public static void main(String[] args) {
3          System.out.println("Hi !!!");
4      }
5  }
6  class A{}
7  public class B{}
8  interface G{}
9  interface H{}

C:\Users\PlayBoy\Desktop\Day 26\ABCXYZ.java:7: class B is public, should be declared in a fil
named B.java
public class B{}
       ^
```

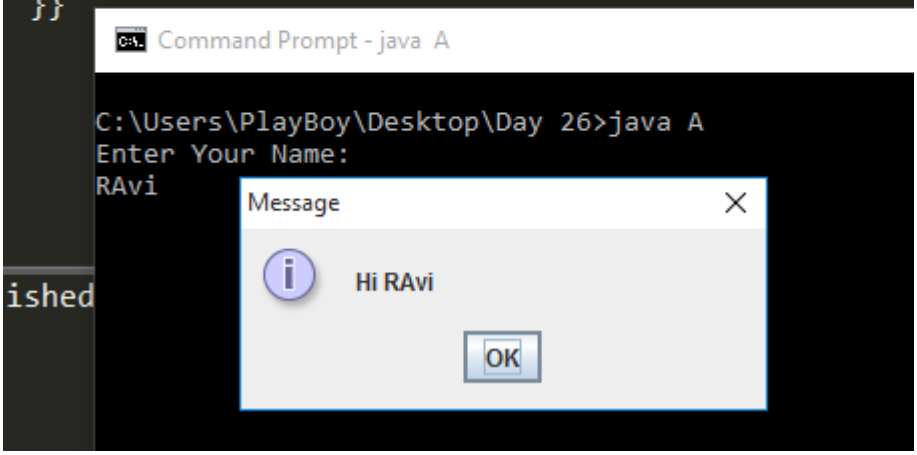
2. එක් source file declaration එකකට පැවතිය හැක්කේ එක් package එකක් පමණි . Package කීපයක් පැවතිය නොහැක.

```
1  package Day26;
2  package f;
3  class A{}
4  class B{}
5  interface G{}
6  interface H{}

C:\Users\PlayBoy\Desktop\Day 26\ljava.java:2: class, interface, or enum expected
package f;
^
```

```
1  package Day26;
2  class A{}
3  class B{}
4  interface G{}
5  interface H{}

[Finished in 0.3s]
```

3. Source file එකක් තුල ඕන තරම් import statement පැවතිය හැක.

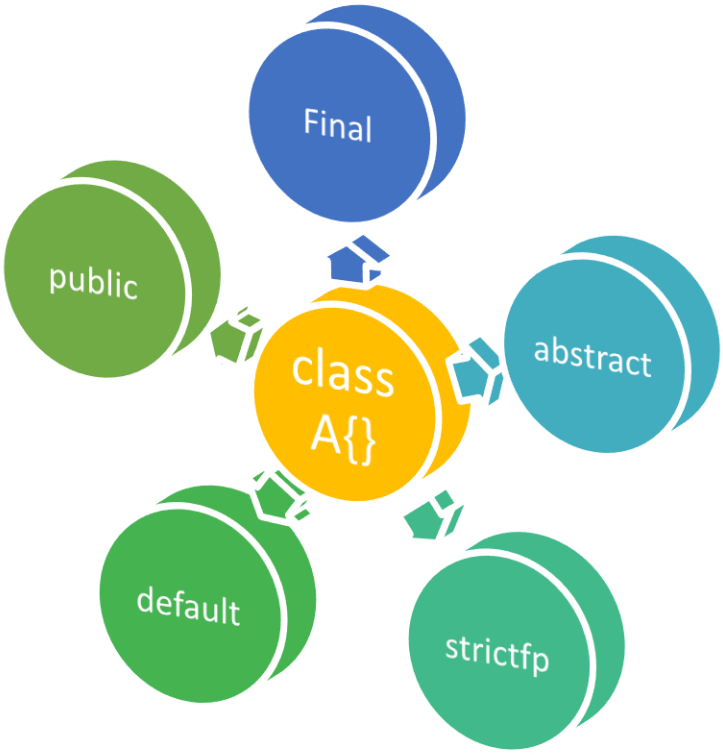
```
import javax.swing.JOptionPane;
import java.util.Scanner;

class A{
    public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter Your Name: ");
    String s=sc.nextLine();
    JOptionPane.showMessageDialog(null,"Hi "+s);
}}
```



පහත අයුරින්ද සිදු කළ හැක.

```
class A{
    public static void main(String[] args) {
    java.util.Scanner sc=new java.util.Scanner(System.in);
    System.out.println("Enter Your Name: ");
    String s=sc.nextLine();
    javax.swing.JOptionPane.showMessageDialog(null,"Hi "+s);
}}
```

Class එකක් ඉදිරියෙන් පහත සඳහන් නම් භාවිතා කළ හැක.


Final
public
class
A{}
abstract
default
strictfp