Variables

* A variable is a container which holds the value while the program execution.
* variable is a name of memory location
* The value stored in a variable can be changed during program execution.
* variables must be declared before use.

⇒ steps to be followed for creating and using variables

1. variable declaration
2. variable initialization
3. variable use

(1) variable declaration

Syntax: [M] <datatype> <variablename>;

Ex: int age;

(2) variable initialization

Syntax [M] <datatype> <variablename> = <value>;

Ex: int age = 28;
Public int pincode = 412111;

(3) Variable use

~~Syntax~~:

Ex: System.out.println(age);

Important Points

(1) declaring a single variable in one variable declaration statement

[modifiers] <datatype> <valname> ;

Ex: byte b;
int a;
char ch;

(2) initializing a single variable

[valname] = <value> ;

Ex: b = 24;
d = 124.78;

(3) declaring and initializing a single variable in one variable declaration statement.

[modifiers] <datatype> <valname> = <value>;

Ex: int a=99;
boolean b=true;

(4) declaring multiple variable of single type in one variable declaration statement.

Syntax: [modifiers] <datatype> <val1>, <val2>, <val3>... ;

Ex: byte b1, b2, b3;
char ch1, ch2, ch3;

(5) Initializing multiple variables with same value:

Ex: a=b=c=9899;
str1 = str2 = str3 = "ati";
i=j=k=7;

(6) declaring and initializing multiple variable of single type in one variable declaration statement.

Ex: int a=120, b=76, c=56;
boolean b1=true, b2=false;

Types of Variables

1. Instance variable
2. Local variable
3. static variable

(1) Instance variable:

- The variable that is declared within the class directly but outside of any method or Block or constructor, without using static keyword.
- for every object a seprate copy of instance variable will be created.
- instance variable is part of the object so called as object level variables.
- instance variable can't be accessed directly, but we can access by using object reference.

- Instance area we can access instance members directly.
- Jvm assigns default value to instance variables

Ex:

```
class Hello {
    int a;
    float f;
    double d;
        void display ( )
        {
          sop (a);
          sop (f);
          sop (d);
          }
Public static void main (String args[])
    {
      Hello h= new Hello ( );
      h.display ( );
      }
}
```

Instance variables without static keyword

- Memory will be allocated for instance variable at the time of creating the object.

Ex:-
```
class Test {
    int x=10;
    P.S.V.m (String args[])
    {
        SOP(x); //Error => instance variable
    }                  can't be accessed directly inside
}                      static => create object.
```

(2) Static variable

- variable defined in the class using static modifier/keyword are called as static variable.
- also called class variable. or class level variables.
- memory will be allocated for static variable at the time of loading the class.
- Static variable is shared location for all the objects of the class [[one of object changes the value gets reflected for other objects also]]
- local variable can't be declared as static
- Static variable can be accessed either by using class name or by using object reference.
- default value will be assigned to static variable by ~~compiler~~ (JVM).

Ex:

```
class StaticExample {
    int x=10;
static int y=20;
    P.S.V.m (String args[])
        {
    StaticExample se = new StaticExample();
  SOP (se.x);
  SOP (se.y);
   SOP (StaticExample.y);
     StaticExample.y=70;
     se.x=50;
   SOP (se.x);
   SOP (StaticExample.y);
StaticExample se2=new StaticExample();
    se2.x=100;
    StaticExample.y=500;
  SOP (se2.x);
  SOP (se1.x);
  SOP (StaticExample.y);
   SOP (se.y);
   SOP (se2.y);
}
```

(3) Local variables :

- The variables declared inside the methods, constructors and blocks are called local variables.
- Local variables are temporary variables and stored inside stack.
- The local variables will be created while executing the block in which we declared it and destoryed once the block completed.
- Local variables can't be static.
- JVM will not assign default value to local variable, every time we have to initialize it while declaring the variable.
- Local variables can't be referenced(accessed), with class name or reference variable.
- only applicable modifier for local variable is "final"

```
{
    local variable
}
```

```
class Localvariable {
    int x=20;
    int y=50;

    void addition()
    { int total=0;
      total=x+y;
      SOP(total);
    }
P.S.V.m(String args[])
    {
Localvariable obj1 = new Localvariable();
      obj1.addition();

    }
}
```

Difference Between Instance variable & static variable

Instance variable	static variable
(1) Instance variables are declared in a class, but outside a method, constructor or any block	Static variables are declared with static keyword in a class but outside a method, constructor or a Block.
(2) Instance variables are created when an object is created with the use of keyword 'new'	static variables are created when the program starts (i.e class loaded on to memory)

(3) instance variables are accessed using object reference
`objectReference.VariableName;`
	accessed by calling with class name or object reference.
	`className.VariableName;`
	`objectReference.variableName;`

(4) object level variable	class level variable

|| Method ||

main method :->

- The execution of the program starts from main method in java programming language.
- If the JVM unable to find required main method then we will get runtime Exception (Error)

NoSuchMethodError : main

- Jvm always searches for the main with following

Syntax:

```
Public static void main (String[] args)
```

- Public : for calling from anywhere
- static : No object created for calling main
- void : will not return anything
- main : configured as "main" in JVM
- (String[] args) : command line arguments.

features of main method

(1) we can change the order of the modifiers

(2) we can declare String[] in any format as follows.

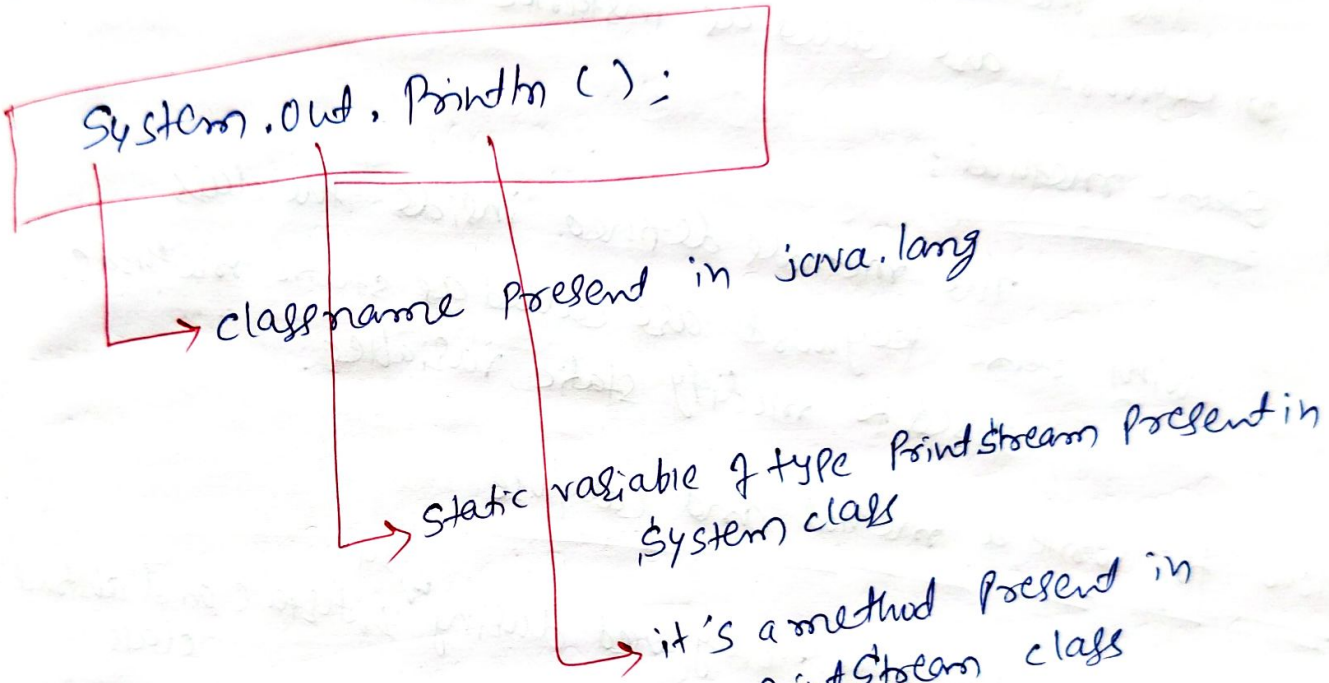
```
(String[] args)
(String [] args)
(String args[])
```

(3) instead of args we can change any valid Java Identifiers

(4) allowed modifiers with main method are final, synchronized, strictfp

(5) method hiding concept is applicable for main method.

(6) overloading of main method is possible.



Method: set of instructions to perform a certain task or operation.

- Method provides the functionality or operation or behaviour of an object.

- Syntax

```
[Modifiers] <return type> <methodname> (<Parameters>)
        {
          //method implementation
        }
```

Types of method :=>

1. Instance method
2. Static method

* Instance method :

The methods defined inside the class without static keyword are called as instance method.

* Static method :

The methods defined inside the class with static keyword are called as static methods.
used to modify static variables.

How to write a method and call/use

step (1) method declaration : only used during with Interface and abstract class
step (2) method defination.
step (3) calling method

Ex:
```
Public int sum (int a , int b)
{
  //method implementation
}
```
Formal Parameter

defination of method

```
objectName. sum (10, 20) ;
```
calling

Actual Parameter

Example : Types of method

```
public class MethodTypes {

    int a = 10;
static int b = 20;
    void instenceMethod ()
        {
          sop (a);
          sop (b);
        }
static void staticMethod ()
        {
          sop (a) : // Error
          //sop (b) ; // static var
        }
Public static void main (string[] args)
    {
      MethodTypes mt = new methodTypes();
      mt.instenceMethod();
      mt.staticMethod();
      methodTypes.staticMethod();
      //methodTypes.instence Method(); //Error
    }

}
```

difference between instance and static methods

Instance Method	Static Method
① without static keyword method is defined inside class.	with static keyword method is defined inside class.
② Instance method can access both instance variable and static variable directly	static method allow only static variables. (directly)
③ which requires an object of its class to be created before it can be called	without creating object we can call static methods
④ calling instance method objname.methodname();	calling static method ① objname.methodname(); ② classname.methodname(); ③ reference variable with null value
⑤ Belongs to object of class	Belongs to class level

Methods Return Type and method with Parameters

Method Return Type

- You can define a method for performing an operation After finishing operation it may produce some result or may not
- Result produced from method can be used in two ways
  1. use the result inside method
  2. returning the result to the called method
- when you want to return result to the caller of the method then you must specify return type depending on the type of result produced.
- if you don't want to return the result to the caller of the method then you must specify method with the return type as "void".

```
class MethodReturnType {
    int show (int a)
    {
      return a+1;
    }
    P.S.V. main (String [] args)
        {
    MethodReturnType mrt = new MethodReturnType();
        int x = mrt.show(10);
          SOP(x);
        }
                                    }
```

Method Parameters / Method Signature

Method parameters are the addition information we are passing to a method to perform certain task

Syntax:

```
[modifier] <returntype> methodname (<Parameters>)
{
    // method implementation.
}
```

Ex:

```
Public void addition (int a , int b)
{
  Sop (a+b);
}
```

method Parameters (arguments)

- There are two types of method Parameters (arguments) formal Parameters and actual Parameters
- Parameters specified with method definition or declaration are called formal Parameters
- arguments/Parameters specified with method call are called as actual Parameters.

```
class Methods {
    int addition (int a, int b)
    {
        return (a+b);
    }
    Public static void main (String[] args)
    {
        Methods m = new Methods();
        int total = m.addition (10, 20);
        SOP (total);
    }
}
```

Formal Parameters

Actual Parameters

Note: after addition method performing addition of 10 and 20 it will return 30 to calling method. and the value will be stored in total variable for further use.