Types of Variables

Based on the position of declaration and behavior we can categorize variables in three types:

1. instance variable
2. static variable
3. local variable

Instance Variables:

1. Instance variables are those variables declared inside a class but outside of any method, constructor or block and without static modifier.
2. These variables are created when an object of the class is created and destroyed when the object is destroyed.
3. For each and every object a separate copy of instance variables is created.
4. instance variables value may or may not vary from object to object.
5. All objects are stored in heap memory and as a part of object all instance variables are also stored on heap memory.
6. Accessing of instance variables
   In same class - I . All instance variables are directly accessible in instance area(constructor, instance method and instance block)
   II. instance variables are not directly accessible in static area in same class
   In Other class - we can access instance variables of one class in other class instance area as well as static area with the help of object reference only.

```
class Employee
{
 int empNo;
String name;
String city ="Indore";
double salary;

Employee() { }

Employee(int empNo, String name, double salary)
{
  this.empNo = empNo;
  this.name = name;
  this.salary = salary;
}

void setEmployeeData (int empNo, String name, double salary)
{
  this.empNo = empNo;
  this.name = name;
  this.salary = salary;
```

all are known as instance variables

instance area

1. All instance variables are directly accessible in instance area(constructor, instance method and instance block) in same class.
2. instance variables are not directly accessible in static area in same class

```
}

void displayEmployeeData()
{
 System.out.println("Employee No :" + empNo);
System.out.println("Employee Name :" +name);
System.out.println("Employee City :" + city);
System.out.println("Employee Salary :" + salary);
}
}

class TestEmployee
{
public static void main(String[] str)
{
Employee e = new Employee();
e. displayEmployeeData();
e. setEmployeeData (100,"Neeraj",35000.0);
e. displayEmployeeData();
Employee e1 = new Employee(101,"Suresh",26000);
e1.displayEmployeeData();
System.out.println(e.empNo);
System.out.println(e.name);
System.out.println(e.city);
System.out.println(e.salary);

}
}
```

Employee class zero argument constructor is called at the time of object creation. And here we are not initializing instance variables explicitly, So JVM will initialize instance variables by their default values.

empNo = 0
name=null
city=Indore
salary=0.0

Initialization of instance variables through method invocation

Output:

Employee No : 0
Employee Name : null
Employee City : Indore
Employee Salary : 0.0

Output:

Employee No : 100
Employee Name : Neeraj
Employee City : Indore
Employee Salary : 35000.0

we can access instance variables of one class in other class instance area as well as in static area with the help of object reference only.
In **TestEmpoyee** class static main method , if we want to access instance members(data and method) of **Employee class**, then first we have to create object of Employee class and with this object reference we can access instance variables and instance method of Employee class .

Employee class argument constructor (**Employee(int EmpNo, String name, double salary)**)is called at the time of object creation. And all instance variables will be intilized implicitly.

Output:

Employee No : 101
Employee Name : Suresh
Employee City : Indore
Employee Salary : 26000.0

Static Variables:

1.static variables are those variables declared inside a class but outside of any method, constructor or block and with static modifier.
2.These variables are created when a class is loaded into the memory and destroyed when the class is unloaded from the memory.
3. only one copy of static variables is created in memory at the time of class loading and this copy is shared to all the objects.
4. All static variables are stored in method area.
5. If we are not initializing static variables explicitly, then it is the responsibility of JVM to initialize static variables by their default values .
6. Accessing of static variables

In same class - All static variables are directly accessible in instance area(constructor, instance method and instance block) as well as in static area

In other class - we can access static variables of one class in other class instance area as well as in static area with the help of object reference only.

We can also access static variables with the class name also

So static variables can be accessible in three ways –

1. Direct
2. With object reference
3. With class Name

```
class Employee1
{
 int empNo;
String name;
static String city =”Indore”;        // static variable
double salary;

Employee1()  {  }

Employee1(int empNo, String name, double salary)
{
  this.empNo = empNo;
  this.name = name;
  this.salary = salary;
 }

void setEmployeeData (int empNo, String name, double salary)
{
  this.empNo = empNo;
  this.name = name;
  this.salary = salary;
}
```

```
void displayEmployeeData()
{
 System.out.println("Employee No :" + empNo);
System.out.println("Employee Name :" +name);
System.out.println("Employee City :" + city);
System.out.println("Employee Salary :" + salary);
}

static void displayEmployeeData1()
{
 System.out.println("Employee City :" + city);
}

}
```

Instance variable and static variables are directly accessible in instance area

Only static varaibles are directly accessible in **static area**. We can not access instance varaibles directly in static area.

```
class TestEmployee1
{
public static void main(String[] str)
{
Employee1 e = new Employee1();
e. displayEmployeeData();
e. setEmployeeData (100,"Neeraj",35000.0);
e. displayEmployeeData();
Employee1 e1 = new Employee1(101,"Suresh",26000);

e1.displayEmployeeData();
e1.displayEmployeeData1();

System.out.println(e.city);
System.out.println(Employee1.city);

}
}
```

empNo = 0
name=null
city=Indore
salary=0.0

Initialization of instance variables through method invocation

Output:

Employee No : 0
Employee Name : null
Employee City : Indore
Employee Salary : 0.0

Output:

Employee No : 100
Employee Name : Neeraj
Employee City : Indore
Employee Salary : 35000.0

Static varaibles are also accessible through object reference and class name

Local variable

1.local variables are those variables declared inside a method, constructor or block.
2.These variables are created when a method is invoked or constructor/ block is executed and destroyed at the completetion of execution of block, constructor and method.
3. All local variables are stored in stack area.

4. If we are not initializing local variables explicitly, then at the time of accessing we will get complilation error. It is the responsibility of programmer to initialize all local variables.

Example

```
int area()
  {
     int length=10;        //local variable
     int breadth = 5;      //local variable
     int rectarea = length*breadth;          //local variable
     return rectarea;
     }
```

Important Not Regarding instance, static and local varaibles:

1. **For instance variables, access modifiers (no-midifier/default,public, private and protected) can be given. Also we can use non-access modifier final ,volatile and transiant keyword with instance varible.**
2. **For static variables, access modifiers (no-midifier/default,public, private and protected) can be given. Also we can use non-access modifier final ,volatile with static variable, but we can not use transiant keyword with static variable**
3. **For local variables, access modifiers (no-midifier/default,public, private and protected) can not be given. We can use only use non-access modifier final with local varible. Other non-access modifiers are not allowed with local variables**
4. **synchronized, abstract,native, strictfp non-access modifier are not allowed with instance,static and local variables.**