What is Instance Variable OR Non static field :

* It is a variable which we declare at **class level so, we have default value.**

* If we declare a variable **inside a class but outside of a method without static modifier then It is called non static field.**

* The life of any non static field starts with **Object creation** that means we cannot think about non static field without **object creation**.
  Example :

```
public class Demo
{
   int x = 100; //non static field

    public static void main(String[] args)
    {
        System.out.println("x value is :"+x); //error
    }
}
```

* As far as It's accessibility is concerned, non static fields are accessible **within the same** class as we all as depends upon the **access modifier** applied on non static field.

* Whenever we want to represent **object properties** we should use non static field.

* The life (not scope) of non staic field starts with object and If the object is destroyed then automatically non static field will also be deleted from the memory.

How to initiaize the object properties through methods :

* If we create our BLC and ELC classes into two different packages then from another package we cannot access our non static field directly, We need to declare the non static field with public access modifier (**Strongly not recommended**)
  So, the alternate way is to initialize the object properties (NSF) is public methods as shown in the program.

```
//Program :
-------------
package com.ravi.blc;

import java.util.Scanner;

//BLC
public class Employee
{
    int employeeNumber;
    String employeeName;
    double employeeSalary;

    public void setEmployeeData()
    {
         Scanner sc = new Scanner(System.in);
         System.out.print("Enter employee Number :");
         employeeNumber = Integer.parseInt(sc.nextLine());
         System.out.print("Enter employee Name :");
         employeeName = sc.nextLine();
         System.out.print("Enter employee Salary :");
         employeeSalary = Double.parseDouble(sc.nextLine());
         sc.close();
    }

    public void getEmployeeData()
    {
       System.out.println("Employee Number is :"+employeeNumber);
       System.out.println("Employee Name is :"+employeeName);
       System.out.println("Employee Salary is :"+employeeSalary);
    }


}
```

```
package com.ravi.elc;

import com.ravi.blc.Employee;

//ELC
public class EmployeeDemo
{
   public static void main(String[] args)
   {
      Employee scott = new Employee();
      scott.setEmployeeData();
      scott.getEmployeeData();
   }

}
```

How to initialize my object properties via methods using parameter variable :

* Instead of initializing the non static field with method without parameterm, It is recommended to initialize the non static field **method with parameter as shown in the program.**

```
package com.ravi.blc;

public class Customer
{
   int customerId;
   String customerName;

   public void setCustomerData(int cid, String cname)
   {
      customerId = cid;
      customerName = cname;
   }

   public void getCustomerData()
   {
      System.out.println("Customer id is :"+customerId);
      System.out.println("Customer Name is :"+customerName);
   }

}
package com.ravi.elc;

import com.ravi.blc.Customer;

public class CustomerDemo
{
   public static void main(String[] args)
   {
      Customer raj = new Customer();
      raj.setCustomerData(101, "Raj");
      raj.getCustomerData();

      System.out.println(".............");

      Customer priya = new Customer();
      priya.setCustomerData(102, "Priya");
      priya.getCustomerData();

   }

}
```

Note : Upto here, We know total 3 ways to initialize the object properties which are as follows :

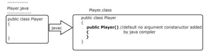
1) Using Object reference(raj.rollNumber = 122)
2) Using Method without parameter (Scanner class)
3) Using Method with Parameter (Park Story)

Introduction of Constructor : [Introduction only]

* If the name of the class and name of the method, both are exactly same and It does not contain any return type then It is called Constructor.

```
public class Student
{
   public Student() //Constructor
   {
   }
}
```

* IN JAVA, WHENEVER WE WRITE A CLASS AND IF WE DON'T WRITE ANY TYPE OF CONSTRUCTOR IN THE CLASS THEN AUTOMATICALLY javac (java compiler) will add one default no argument constructor in that particular class.
  Example :



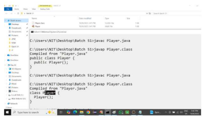
```
public class Player
{

}
```

```
public class Player
{
   public Player() //default no argument constaructor added
   {                      by java compiler
   }
}
```

* Every java class must contain **at-least one constructor** either implicitly added by java compiler OR explicitly written by developer. [We cannot think about java class without constructor]

* The access modifier of default no argument constructor, added by java compiler depends upon class access modifier that means If the class is public then the default no argument constructor added by java compiler is also public, If class is not public then constructor is also not public.



```
C:\Users\NIT\Desktop\Batch 51>javac Player.java

C:\Users\NIT\Desktop\Batch 51>javap Player.class
Compiled from "Player.java"
public class Player {
  public Player();
}

C:\Users\NIT\Desktop\Batch 51>javac Player.java

C:\Users\NIT\Desktop\Batch 51>javap Player.class
Compiled from "Player.java"
class Player {
  Player();
}
```