Role of non static field (instance variable) while creating the Object :

* As we know non static fields are the part of the object.

* Whenever we create an object in java, a separate copy of all the non static fields will be created with each & every object.

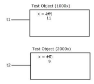
Test.java

```
public class Test
{
    int x = 10;

    public static void main(String[] args)
    {
        Test t1 = new Test();
        Test t2 = new Test();

        ++t1.x;   --t2.x;

        IO.println(t1.x); //11
        IO.println(t2.x); //9
    }
}
```



//Program

```
package com.ravi.oop;

public class Test
{
    int x = 10;

    public static void main(String[] args)
    {
        Test t1 = new Test();
        Test t2 = new Test();

        ++t1.x; --t2.x;

        IO.println(t1.x);
        IO.println(t2.x);

    }
}
```

What is a static field :

* If we declare a field inside a class with static modifier then It is called static field.
  Example :
  public class Bank
  {
      public **static** String IFSC_CODE = "SBI01001";   //static field
  }

* Whenever we load a class into JVM memory then **static fields** are getting the **memory** and initialized with **default value**.

* We can use these static fields with the help of class name till class is not un-loaded from JVM memory.

Role of static field while object creation :

* Actually, for static field object is not required. We can directly call the static field OR static method with the help of class name.

* Whenever we create an object in java then a **single copy of static field will be created a the same single copy will be sharable by all the objects.**

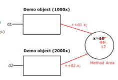
```
package com.ravi.oop;

public class Demo
{
    static int x = 10;    //Static field

    public static void main(String[] args)
    {
        Demo d1 = new Demo();
        Demo d2 = new Demo();

        ++d1.x; ++d2.x;

        IO.println(d1.x); //12
        IO.println(d2.x); //12
    }
}
```



So the conclusion is,
Non static field = **Multiple Copies** with each & every object
static field = **Single copy** for All the objects
So we can say static fields are used to SAVE THE MEMORY.

When we should declare a field as a non static field and when we should declare as static :

Non static field :

* If the **value of the field** is **different** with respect to object (every object field is having different value) then we should use non static field so, multiple copies will be created for holding different values.

Static field :

* If the **value of the field** is **common** for all the objects then we should use static field so single copy will be created and sharable by all the objects.

Example 1 :

```
class Student
{
    int roll;
    String name;
    String address;
    static String collegeName = "NIT";
    static String courseName = "Java";
}
```

Example 2 :

```
class Customer
{
    int accountNumber;
    String name;
    double balance;
    static String IFSCCODE = "RBI0100";
    static String location = "Ameerpet";
}
```

//Program on static & non static field :

```
package com.ravi.blc;

public class Student
{
  int rollNumber;
  String studentName;
  String studentAddress;
  static String collegeName = "NIT";
  static String courseName = "Java";

  public void setStudentData(int roll, String name, String address)
  {
      rollNumber = roll;
      studentName = name;
      studentAddress = address;
  }

  public String getStudentDetails()
  {
      return "[Roll Number is :"+rollNumber+", Student Name is :"+studentName+", Student
Address is :"+studentAddress+", College Name is :"+collegeName+ ", Course Name is
:"+courseName+"]";
  }

}
```

```
package com.ravi.elc;

import com.ravi.blc.Student;

public class StudentDemo
{
    public static void main(String[] args)
    {
        Student raj = new Student();
        raj.setStudentData(101, "Raj", "Ameerpet");
        String studentDetails = raj.getStudentDetails();
        IO.println(studentDetails);

        IO.println("...................");

        Student priya = new Student();
        priya.setStudentData(201, "Priya", "S R Nagar");
        studentDetails = priya.getStudentDetails();
        IO.println(studentDetails);

    }
}
```

**What is Data Hiding ?

Data hiding is nothing but declaring our non static fields with private access modifier so our data will not be accessible from outer world (Outside of BLC class) that means no one can access our data directly from outside of the class.

*We should provide the accessibility of our data through methods so we can perform VALIDATION ON DATA which are coming from outer world.

```
package com.ravi.blc;

public class Customer
{
  private double balance = 10000;  //Data hiding

   public void deposit(double amount)
   {
       if(amount<=0)
       {
           System.err.println("Sorry!! Amount cannot be deposited");
           System.exit(0);
       }
       balance = balance + amount;
   }

   public void withdraw(double amount)
   {
       if(amount > balance)
       {
           System.err.println("Sorry, Insufficient Balance");
           System.exit(0);
       }
       balance = balance - amount;


   }

   public double getBalance()
   {
       return balance;
   }



}
```

```
package com.ravi.data_hiding;

public class BankApplication
{
    public static void main(String[] args)
    {
        Customer raj = new Customer();
        raj.deposit(5000);
        raj.withdraw(2000);
    }
}
```