Dt : 1/9/2023

faq:

define Factory method?

=>The method which creates object by hiding the Object creation process is known as Factory method.

==========

Ex:

Convert IComparable-Application into Anonymous InnerClasses model.

ProjectName : App_Anonymous4_IcComparable

packages,

p1 : IComparable.java

```
package p1;
public interface IComparable {
    public abstract int compareTo(int x,int y);
}
```

p2 : DemoAnonymous4.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

public class DemoAnonymous4 {

        public static void main(String[] args) {

    Scanner s = new Scanner(System.in);
```

```
System.out.println("Enter the value-1:");
int v1 = s.nextInt();
System.out.println("Enter the value-2:");
int v2 = s.nextInt();
if(v1>0 && v2>0)
{
    System.out.println("****Choice****");
    System.out.println("\t1.GreaterValue"
            + "\n\t2.SmallerValue");
    System.out.println("Enter the Choice:");
    int choice = s.nextInt();
    switch(choice)
    {
    case 1:
        //GreaterValue as Anonymous
        IComparable gv = new IComparable()
        {
            public int compareTo(int x,int y)
            {
                if(x>y) return x;
                else return y;
            }
```

```
            };
            int res1 = gv.compareTo(v1, v2);
            System.out.println("GreaterValue:"+res1);
            break;
        case 2:
            //SmallerValue as Anonymous
            IComparable sv = new IComparable()
            {
                public int compareTo(int x,int y)
                {
                    if(x<y) return x;
                    else return y;
                }
            };
            int res2 = sv.compareTo(v1, v2);
            System.out.println("SmallerValue:"+res2);
            break;
        default:
            System.out.println("Invalid Input...");
        }//end of switch
    }//end of if
    else
```

```
    {
        System.out.println("Invalid input..");
    }
    s.close();
      }
}
```

ClassFiles:

IComparbale.class

DemoAnonymous4.class(MainClass)

DemoAnonymous4$1.class

DemoAnonymous4$2.class

==========

Assignment:

Convert IArithmetic-Application into Anonymous InnerClasses model.

==========

Note:

=>"Anonymous InnerClasses as Implementation classes" model is modified as "LambdaExpressions" in Java8 version.

==========

**imp*

LambdaExpressions in Java:(Java8 - new feature)

=>The process of declaring method without method name is known as LambdaExpression or Anonymous method.

structure of LambdaExpression:

```
(para_list)->
{
 //method_body
}
```

Note:

=>The LambdaExpression is attached with the abstract method of Interface and,the LambdaExpression is called for execution using Interface abstract method_name.

syntax:

```
interface ITest
{
  public abstract void m1(int x);
}

ITest ob = (int x)->
```

```
{
 //method_body
};
```

Ex:

ProjectName : App_LambdaExpression1

packages,

p1 : ITest.java

```
package p1;
public interface ITest {
    public abstract void m1(int x);
    public default void m2(int y) {
        System.out.println("****default m2(y)****");
        System.out.println("The value y:"+y);
    }
}
```

p2 : DemoLambdaExpression1.java

```
package p2;
import p1.*;
public class DemoLambdaExpression1 {
     public static void main(String[] args) {
          //LambdaExpression
         ITest ob = (int x)->
         {
       System.out.println("***Implemented m1(x)****");
       System.out.println("The value x:"+x);
         };

         ob.m1(11);
         ob.m2(12);
     }
```

}

o/p:

******Implemented m1(x)*******

The value x:11

*******default m2(y)*******

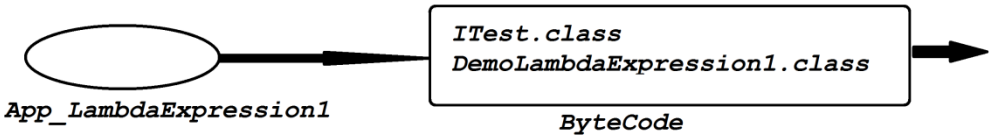
The value y:12

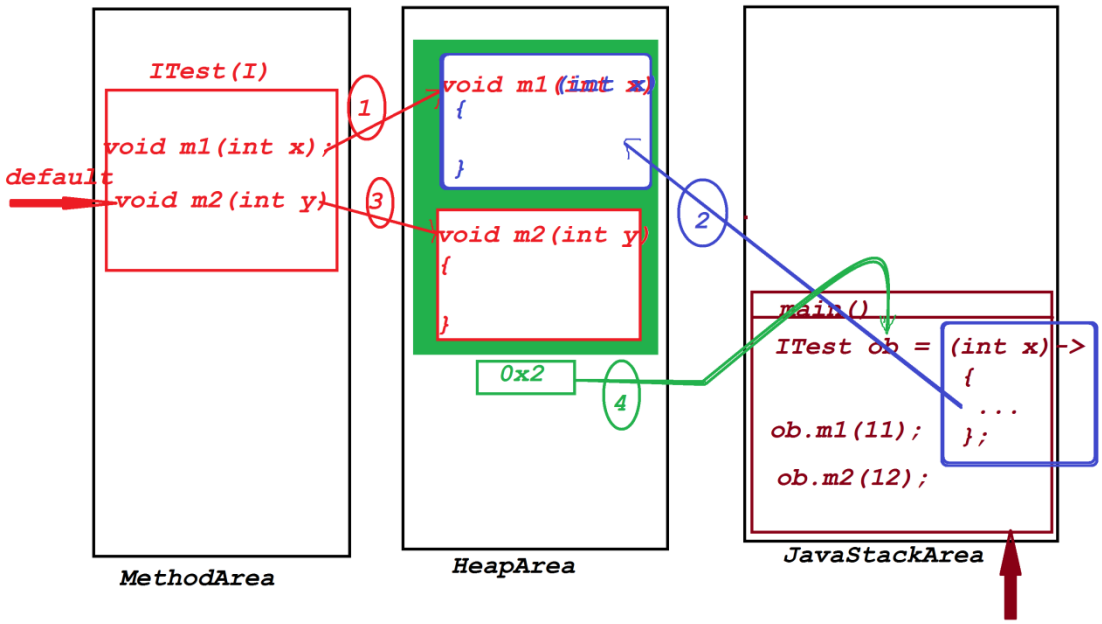
Diagram:(Demonstrating LambdaExpression)

ClassFiles:

ITest.class

DemoLambdaExpression1.class(MainClass)





==========

Advantage of LambdaExpression:

=>when we use LambdaExpressions,the separate class files are not generated and,the 'Loading and Linking time' of Execution process is saved and generate high performance of an application.

==========