Dt : 2/9/2023

****imp***

Execution Scope of LambdaExpression:

=>LambdaExpression is executed within the method-scope where the LambdaExpression is declared.

Ex:

If LambdaExpression declared in main() method then it is executed under the scope of main() method.

==========

Coding Rules of LambdaExpression:

Rule-1 : The interface which is providing abstract method to hold LambdaExpression must be declared with only one abstract method and the interface is known as "Functional Interface".

Rule-2 : The parameter-names which are used in LambdExpressions,the same names must not be used for Local variables under the same method Scope.

Rule-3 : The LambdaExpressions cannot access variables from Functional Interface directly,but we can access with Interface name.

==========

ProjectName : App_LambdaExpression2

packages,

p1 : IDisplay.java

```
package p1;
public interface IDisplay
{
     public static final int a=10;
     public abstract void dis(int k);
}
```

p2 : DemoLambdaExpression2.java(MainClass)

```
package p2;
import p1.*;
public class DemoLambdaExpression2
{
     public static int b=20;
     public int c=30;
     public static void main(String[] args)
     {
        int d=40;
        //LambdaExpression
        IDisplay ob = (int k)->
        {
        System.out.println("The value k:"+k);
        //System.out.println("The value a:"+a);
        System.out.println("The value b:"+b);
        //System.out.println("The value c:"+c);
        System.out.println("The value d:"+d);
        };
        ob.dis(12);
     }

}
```

o/p:

The value k:12

The value b:20

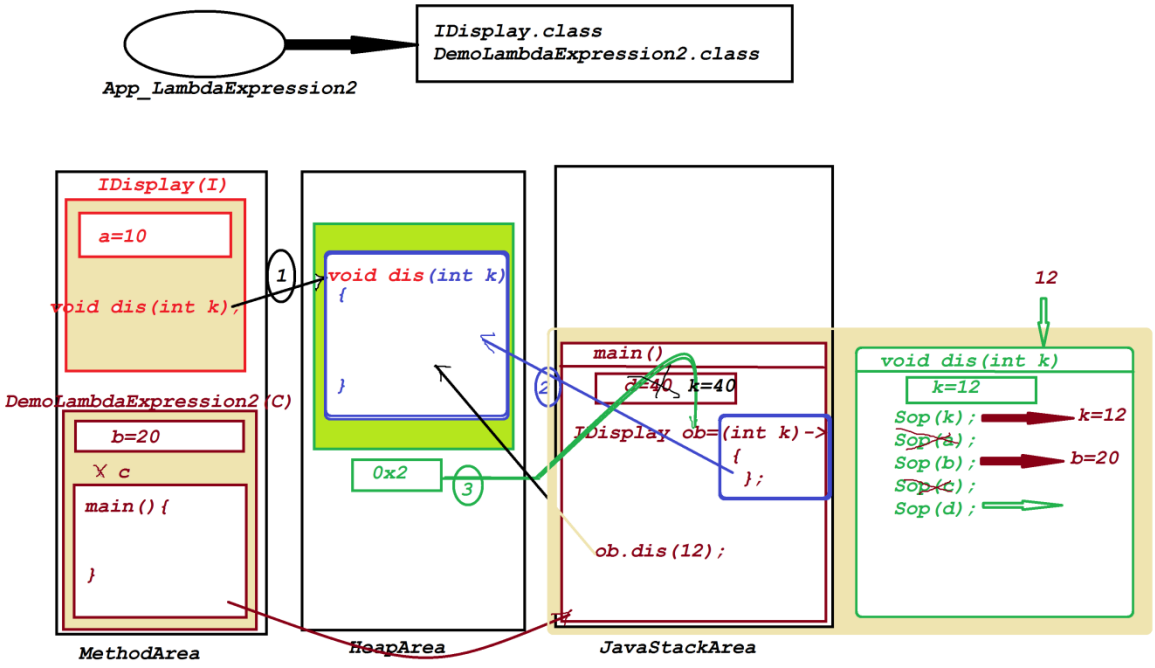
The value d:40

Execution Flow of above application:

ClassFiles:

IDisplay.class

DemoLambdaExpression2.class(MainClass)



==========

Ex:

ProjectName : App_LambdaExpression3

packages,

p1 : IComparable.java

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

p1 : Access.java

```
package p1;
public class Access {
     public static IComparable getRef(int choice)
     {
     return switch(choice)
     {
     case 1 : yield (int x,int y)->
                      {
                          if(x>y) return x;
                          else return y;
                      };
     case 2 : yield (int x,int y)->
                     {
                     if(x<y) return x;
                     else return y;
                     };
      default : yield null;
     };
     }
}
```

p2 : DemoLambdaExpression3.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;
```

```
public class DemoLambdaExpression3 {
	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();
        IComparable ob = Access.getRef(choice);
        if(ob==null) {
            System.out.println("Invalid choice...");
        }else {
                System.out.println("Result:"+ob.compareTo(v1, v2));
        }
    }
    else
```

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

==========