Dt : 25/10/2022

Advantage of LambdaExpressions:

=>when we use LambdaExpressions then separate class files are not generated ,and in this process the loading time of execution process is saved and generate HighPerformance of an application.

Coding Rules of LambdaExpressions:

Rule-1 : The target interface which is providing abstract method signature to hold LambdaExpression must be declared with only one abstract method,but can be declared with any number of Concrete methods. (This target interface is also known as Functional Interface)

Rule-2 : LambdaExpressions will access variables of target interface using Interface_name.

Rule-3 : The parameters which are used in LambdaExpressions,the Same parameter names must not be used as Local variables of same method_Scope.

--

faq:

wt is the diff b/w

(i)Normal Interface

(ii)Functional Interface

(iii)Marker Interface

(i)Normal Interface:

=>The Interface which can be declared with any number of abstract methods is known as Normal Interface.

Ex:

java.util.Collection<E>

java.util.Map<K,V>

(ii)Functional Interface:

=>The interface which is declared with only one abstract method is known as Functional Interface.

Ex:

java.lang.Runnable

java.lang.Comparable

(iii)Marker Interface:

=>The Empty-Interfaces are known as Marker Interfaces or Tagging Interfaces.

(The interface with 0-members is known as Empty Interface.

Ex:

java.io.Serializable

java.lang.Cloneable

==

Assignment-1:(Solution)

Convert IArithmetic application into LambdaExpressions.

IArithmetic.java

```
package test;
public interface IArithmetic {
  public abstract double calculate(int x,int y);
}
```

LambdaExpression3.java(MainClass)

```
package maccess;

import java.util.*;

import test.*;

public class LambdaExpression3 {

          public static void main(String[] args) {

      Scanner s = new Scanner(System.in);

      System.out.println("Enter the value1:");

      int v1 = s.nextInt();

      System.out.println("Enter the value2:");

      int v2 = s.nextInt();

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

      System.out.println("1.add\n2.sub\n3.mul\n4.div\n5.modDiv");

      System.out.println("Enter the Choice:");

      switch(s.nextInt())

      {

      case 1:

         //Addition class as LambdaExpression

         IArithmetic ad = (int x,int y)->
```

```
    {
            return x+y;
    };
    System.out.println("Sum="+ad.calculate(v1, v2));
    break;
case 2:
    //Subtraction class as LambdaExpression
    IArithmetic sb = (int x,int y)->
            {
                    return x-y;
            };
    System.out.println("Sub="+sb.calculate(v1, v2));
    break;
case 3:
    //Multiplication class as LambdaExpression
    IArithmetic ml = (int x,int y)->
            {
                    return x*y;
            };
    System.out.println("Mul="+ml.calculate(v1, v2));
    break;
case 4:
    //Division class as LambdaExpression
    IArithmetic dv = (int x,int y)->
```

```
                {
                        return (float)x/y;
                };
            System.out.println("Div="+dv.calculate(v1, v2));
            break;
        case 5:
            //ModDivision class as LambdaExpression
            IArithmetic md = (int x,int y)->
                {
                        return x%y;
                };
            System.out.println("ModDiv="+md.calculate(v1, v2));
            break;
        default:
            System.out.println("Invalid Choice...");
        }//end of switch
        s.close();
            }
}
```

===

Assignment-2:

Convert BankTransaction application into LambdaExpression.

Balance.java

```
package test;
public class Balance {
   public double bal=2000;
   public double getBalance() {
        return bal;
   }
}
```

Transaction.java

```
package test;
public interface Transaction {
    public static final Balance b = new Balance();
    public abstract void process(int amt);
}
```

CheckPinNo.java

```
package test;
public class CheckPinNo {
     public boolean verify(int pinNo) {
      return switch(pinNo) {
      case 1111:yield true;
      case 2222:yield true;
      case 3333:yield true;
      default:yield false;
      };
     }
}
```

BankMainClass.java(MainClass)

```
package maccess;

import test.*;

import java.util.*;

public class BankMainClass {

     public static void main(String[] args) {

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

```
int count=0;

pqr:

while(true) {

    System.out.println("Enter the pinNo:");

    int pinNo = s.nextInt();

    CheckPinNo cpn = new CheckPinNo();

    boolean k = cpn.verify(pinNo);

    if(k)

    {

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

            System.out.println("1.WithDraw\n2.Deposit");

            System.out.println("Enter the Choice:");

            switch(s.nextInt())

            {

            case 1:

                    System.out.println("Enter the amt:");

                    int a1 = s.nextInt();

                    if(a1>0 && a1%100==0)

                    {

                            //WithDraw class as LambdaExpression

                            Transaction wd = (int amt)->

                            {

                                    if(amt<Transaction.b.bal)

                                    {
                                        System.out.println("Amt WithDrawn:"+amt);

                                        Transaction.b.bal=Transaction.b.bal-amt;

                                        System.out.println("Balance
amt:"+Transaction.b.getBalance());

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

                              }//end of if

                              else

                              {

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

                              }

                        };

                    wd.process(a1);

          }//end of if

          else

          {

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

          }

          break pqr;//stop the loop

     case 2:

          System.out.println("Enter the amt:");

          int a2 = s.nextInt();

          if(a2>0 && a2%100==0)

          {

                    //Deposit as LambdaExpression
```

```
                                        Transaction dp = (int amt)->
                                        {
                                                System.out.println("Amt deposited:"+amt);
                                                Transaction.b.bal=Transaction.b.bal+amt;
                                                System.out.println("Balance 
amt:"+Transaction.b.getBalance());
                                                System.out.println("Transaction Completed...");
                                          };
                                        dp.process(a2);
                                }//end of if
                                else
                                {
                                        System.out.println("Invlid amt...");
                                }
                                break pqr;//stop the loop
                        default:
                                System.out.println("Invalid Choice...");
                        break pqr;//stop the loop
                        }//end of switch
                }//end of if
                else
                {
                        System.out.println("Invalid pinNo....");
                        count++;
```

```
          }


          if(count==3)
          {
                    System.out.println("Transaction blocked...");
                    break;//stop the loop
          }
      }//end of loop
          }
}
```

==

faq:

wt are the situations in realtime to use "Annonymous InnerClasses " and "LambdaExpressions"?

(i)Anonymous InnerClasses model is used for Normal Interfaces.

(ii)LambdaExpressions are used for Functional Interfaces.

===