Dt : 12/9/2023

Assignment:(Solution with Exception handling process)

Construct BanTransaction-Application

ProjectName : App_bankTransaction1

packages,

p1 : Balance.java

```
package p1;
public class Balance {
  public float bal=2000.0F;
  public float getBalance() {
       return bal;
   }
}
```

p1 : CheckPinNo.java

```
package p1;
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;
     };
     }
}
```

p1 : Transaction.java

```
package p1;
@FunctionalInterface
```

```
public interface Transaction {
   public Balance b = new Balance();
   public abstract void process(int amt)throws
WithDraw;
 }
```

p1 : WithDraw.java

```
package p1;
@SuppressWarnings("serial")
public class WithDraw extends Exception implements
Transaction
{
   public WithDraw(String msg)
   {
        super(msg);
   }
   @Override
   public void process(int amt)throws WithDraw
   {
        try
        {
            if(amt>b.bal)//Exception
            {
                WithDraw ob = new
WithDraw("InSufficient Fund");
                throw ob;
            }
            System.out.println("Amt WithDrawn:"+amt);
            b.bal = b.bal-amt;
            System.out.println("Balance
Amt:"+b.getBalance());
            System.out.println("Transaction
Completed...");
        }//end of try
        catch(WithDraw ob)
        {
            throw ob;
        }
   }
}
```

p1 : Deposit.java

```
package p1;
public class Deposit implements Transaction
{
	@Override
	public void process(int amt)
	{
	System.out.println("Amt deposited:"+amt);
	b.bal=b.bal+amt;
	System.out.println("Balance Amt:"+b.getBalance());
	System.out.println("Transaction Completed...");
	}
}
```

p2 : BankMainClass.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

@SuppressWarnings("serial")

public class BankMainClass extends Exception

{

	public BankMainClass(String msg)

	{

		super(msg);

	}

	public static void main(String[] args)
```

```
  {
Scanner s = new Scanner(System.in);
try
{
  int count=0;
  xyz:while(true)
  {
    try
    {
            System.out.println("Enter the PinNo:");
            int pinNo = s.nextInt();
            if(!(pinNo>=1111 && pinNo<=9999))//Exception
            {
                    BankMainClass bmc = new BankMainClass
                                    ("Invalid pinNo..");
                    throw bmc;

            }
            boolean k = new CheckPinNo().verify(pinNo);
            if(!k)//Exception
            {
                    BankMainClass bmc = new BankMainClass
                    ("PinNo donot exist...");
        throw bmc;
}
System.out.println("*****Choice******");
System.out.println("\t1.WithDraw"
              + "\n\t2.Deposit");
System.out.println("Enter the Choice:");
int choice = s.nextInt();
switch(choice)
{
case 1:
        try
        {
               System.out.println("Enter the amt:");
               int a1 = s.nextInt();
               if(!(a1>0 && a1%100==0))//Exception
               {
                       BankMainClass bmc=new BankMainClass
                                       ("Invalid amt...");
                       throw bmc;
               }
               WithDraw wd = new WithDraw("");
```

```
          wd.process(a1);
      }//end of try
      catch(BankMainClass bmc)
      {
           System.out.println(bmc.getMessage());
      }
      catch(WithDraw ob)
      {
          System.out.println(ob.getMessage());
      }
      break xyz;//Stop the loop
case 2:
      try
      {
          System.out.println("Enter the amt:");
          int a2 = s.nextInt();
          if(!(a2>0 && a2%100==0))
          {
                BankMainClass bmc=new BankMainClass
                            ("Invalid Amt..");
                throw bmc;
          }
```

```
                    Deposit dp = new Deposit();
                    dp.process(a2);
                }//end of try
                catch(BankMainClass bmc)
                {
                     System.out.println(bmc.getMessage());
                }
                break xyz;//stop the loop
            default:
                System.out.println("Invalid Choice...");
                break xyz;//stop the loop
        }//end of switch
    }//end of try
    catch(BankMainClass bmc)
    {
        System.out.println(bmc.getMessage());
        count++;
        if(count==3)
        {
            System.out.println("Transaction Blocked..");
            break;//loop is stopped
        }
```

```
            }
          }//end of loop
        }//end of try
        finally
        {
          s.close();
        }
          }
}
```

o/p:

```
Enter the PinNo:
1111
*****Choice******
          1.WithDraw
          2.Deposit
Enter the Choice:
2
Enter the amt:
1200
Amt deposited:1200
Balance Amt:3200.0
Transaction Completed...
```

===

faq:

define Annotation?

=>The tag based information which is added to programming components is known as Annotation.

=>we use "@" symbol to represent annotations.

=>Annotations can be added to variables,methods,Classes and Interfaces

=>These Annotations will give information to compiler at Compilation stage.

=>The following are some important annotations in CoreJava:

(a)@SuppressWarnings

(b)@Override

(c)@FunctionalInterface

(a)@SuppressWarnings:

=>@SuppressWarnings annotation will give information to compiler to close the raised warnings.

(b)@Override:

=>@Override annotation will give information to compiler to check the method is Overriding method or not.

(c)@FunctionalInterface:

=>@FunctionalInterface will give information to compiler to check the interface is Functional Interface or not.

==