Dt : 18/10/2022

Assignment:(Solution)

Construct BankTransaction application.

Balance.java

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

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;
      };
     }
}
```

Transaction.java

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

WithDraw.java

```
package test;
public class WithDraw implements Transaction{
```

```
      //WithDraw Logic
     public void process(int amt) {
      if(amt<b.bal)
      {
            System.out.println("Amt WithDrawn:"+amt);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 if
      else
      {
            System.out.println("InSufficient fund...");
      }
     }
}
```

Deposit.java

```
package test;
public class Deposit implements Transaction{
      //Deposit Logic
   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...");
   }
}
```

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 wd = new WithDraw();
                            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 dp = new Deposit();
                        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
        }
}
```

o/p:

```
Enter the pinNo:
2222
====Choice====
1.WithDraw
2.Deposit
Enter the Choice:
1
Enter the amt:
1200
Amt WithDrawn:1200
Balance amt:800.0
Transaction Completed...
```

==========

*imp

Generalization process:

=>The process in which one object is created holding all the mebers of Parent and only Overriding members from the Child is known as "Generalization process".

syntax for Generalization using classes:

```
PClass ob = (PClass)new CClass();
```

syntax for Generalization using Interfaces:

```
ITest ob = (ITest)new IClass();
```

syntax for Generalization using AbstractClasses:

```
AClass ob = (AClass)new EClass();
```

Ex:

PClass.java

```
package test;
public class PClass {
    public void m1(int x) {
     System.out.println("====PClass m1(x)===");
     System.out.println("The value x:"+x);
    }
    public void m2(int y) {
     System.out.println("====PClass m2(y)===");
     System.out.println("The value y:"+y);
    }
}
```

CClass.java

```
package test;
public class CClass extends PClass{
      public void m1(int x)//Overriding method
      {
          System.out.println("====CClass m1(x)===");
          System.out.println("The value x:"+x);
         }
         public void m3(int z) //Non-Overriding method
         {
          System.out.println("====CClass m3(z)===");
          System.out.println("The value z:"+z);
         }
}
```

ITest.java

```
package test;
public interface ITest {
   public abstract void dis1(int x);
}
```

IClass.java

```
package test;
public class IClass implements ITest{
     public void dis1(int x)//Implemented and Overriding method
     {
          System.out.println("====dis1(x)===");
          System.out.println("The value x:"+x);
     }
     public void dis2(int y)//Non-Implemented and Non-Overriding
method
     {
          System.out.println("====dis2(y)===");
          System.out.println("The value y:"+y);
     }
}
```

AClass.java

```
package test;
public abstract class AClass {
   public abstract void show1(int x);
}
```

EClass.java

```
package test;
public class EClass extends AClass{
	public void show1(int x)//Implemented and Overriding method
	{
		System.out.println("===show1(x)====");
		System.out.println("The value x:"+x);
	}
	public void show2(int y)//Non-Implemented and Non-
Overriding method
	{
		System.out.println("===show2(y)====");
		System.out.println("The value y:"+y);
	}
}
```

DemoGeneralization.java(MainClass)

```
package maccess;
import test.*;
public class DemoGeneralization {
	public static void main(String[] args) {
		System.out.println("****Generalization using
Classes****");
		PClass ob1 = (PClass)new CClass();
		ob1.m1(11);
		ob1.m2(12);
		// ob1.m3(13);//Error
		System.out.println("****Generalization using
Interfaces****");
		ITest ob2 = (ITest)new IClass();
		ob2.dis1(111);
		//ob2.dis2(222);//Error
		System.out.println("****Generalization using
AbstractClasses****");
		AClass ob3 = (AClass)new EClass();
		ob3.show1(234);
		//ob3.show2(345);//Error
	}
}
```

o/p:

*******Generalization using Classes*******

====CClass m1(x)===

The value x:11

====PClass m2(y)===

The value y:12

*******Generalization using Interfaces*******

====dis1(x)===

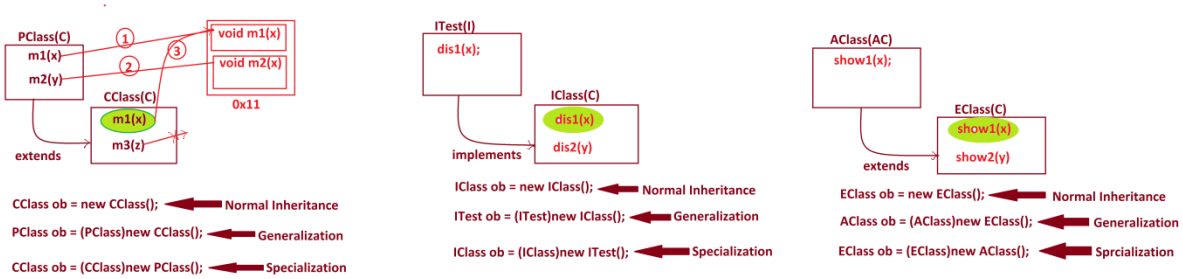
The value x:111

*******Generalization using AbstractClasses*******

===show1(x)====

The value x:234

Diagram:



==========