1. Create a package named myPack.CurrConvertor to implement the currency converter

(Dollar to INR, EURO to INR, Yen to INR and vice versa). Write a java program which uses this

package to convert the given currency into other currency type.

Use the pre-defined exception java.lang.NumberFormatException (checked) to throw

the error whenever the given input currency is not in the required format.

package myPack;

public class CurrConvertor{

double INR;

double USD;

double Euro;

double Yen;

public CurrConvertor(){

INR=0.0;Euro=0.0;

USD=0.0;Yen=0.0;

}

public void fromINR(double INR){

this.INR=INR;

USD=0.014\*INR;

Euro=0.013\*INR;

Yen=1.49\*INR;

}

public void fromUSD(double USD){

this.USD=USD;

INR=71.18\*USD;

Euro=0.9\*USD;

Yen=106.16\*USD;

}

public void fromEuro(double Euro){

this.Euro=Euro;

USD=1.11\*Euro;

INR=78.92\*Euro;

Yen=117.69\*Euro;

}

public void fromYen(double Yen){

this.Yen=Yen;

USD=0.0094\*Yen;

Euro=0.0085\*Yen;

INR=0.67\*Yen;

}

public void Display(){

System.out.println("\nINR: "+INR);

System.out.println("USD: "+USD);

System.out.println("Euro: "+Euro);

System.out.println("Yen: "+Yen);

}

}

import myPack.CurrConvertor;

import java.util.Scanner;

public class TestPackage{

public static void main(String args[]){

myPack.CurrConvertor cc=new myPack.CurrConvertor();

Scanner in=new Scanner(System.in);

double x;

int opt=0;

do{

try{

System.out.println("\nChoose input currency format\n 1-INR\n 2-USD");

System.out.println(" 3-Euro\n 4-Yen\n 0-Exit");

System.out.print(" Your choice: ");

opt=in.nextInt();

if(opt==1){

System.out.print(" Enter INR value ");x=in.nextDouble();

cc.fromINR(x);

cc.Display();

}

else if(opt==2){

System.out.print(" Enter USD value ");x=in.nextDouble();

cc.fromUSD(x);

cc.Display();

}

else if(opt==3){

System.out.print(" Enter Euro value ");x=in.nextDouble();

cc.fromEuro(x);

cc.Display();

}

else if(opt==4){

System.out.print(" Enter Yen value ");x=in.nextDouble();

cc.fromYen(x);

cc.Display();

}

else if(opt!=0)

throw new NumberFormatException();

else;

}

catch(NumberFormatException e){

System.out.println("Invalid Input Format ");

}

}while(opt!=0);

}

}

/\*

Output:

Choose input currency format

1-INR

2-USD

3-Euro

4-Yen

0-Exit

Your choice: 1

Enter INR value 1000

INR: 1000.0

USD: 14.0

Euro: 13.0

Yen: 1490.0

Choose input currency format

1-INR

2-USD

3-Euro

4-Yen

0-Exit

Your choice: 2

Enter USD value 1000

INR: 71180.0

USD: 1000.0

Euro: 900.0

Yen: 106160.0

Choose input currency format

1-INR

2-USD

3-Euro

4-Yen

0-Exit

Your choice: 3

Enter Euro value 1000

INR: 78920.0

USD: 1110.0

Euro: 1000.0

Yen: 117690.0

Choose input currency format

1-INR

2-USD

3-Euro

4-Yen

0-Exit

Your choice: 4

Enter Yen value 1000

INR: 670.0

USD: 9.4

Euro: 8.5

Yen: 1000.0

Choose input currency format

1-INR

2-USD

3-Euro

4-Yen

0-Exit

Your choice: 5

Invalid Input Format

Choose input currency format

1-INR

2-USD

3-Euro

4-Yen

0-Exit

Your choice: 0

\*/

2. Create a class named “Account” which contains customer name, acct\_num, branch, balance,

PAN. Write a method for deposit and withdrawal.

a. In deposit, if the customer deposits amount more than 50000, then throw the user defined

exception “PANRequiredException”.

b. In withdrawal, check for the minimum balance before the withdrawal i.e., the minimum

balance > balance - withdrawal. Else throw “MinBalRequiredException” exception.

If the withdrawal is more than the balance then throw

“NotEnougMoneyInAccountException”.

c. Search for a particular acct\_num. If not present then throw

“AccountNotFoundException”.

Custome-define Exception:

When defining a custom-defined exception use the String toString() method to display the custom

defined error messages (description) instead of regular error format – exception object:error

description.Design a class called Account as described below:

Account

­cname:String

­pan:int

­accno:int

­branch:String

­balance:float

+Account(cname,pan,accno,branch,bala

nce)

+getMethod(): all ­ fields

+deposit(accno, amt):void

+withdraw(accno, amt):void

+search(accno)throws AccountNotFound

+toString():String

import java.util.Scanner;

class PanRequiredException extends Exception{

public String toString(){

return "PanRequiredException: PAN Number Required";

}

}

class MinBalRequiredException extends Exception{

public String toString(){

return "MinBalRequiredException: Balance Tending Below Minimum Limit";

}

}

class NotEnoughMoneyInAccountException extends Exception{

public String toString(){

return "NotEnoughMoneyInAccountException: Insufficient Funds To Process Request";

}

}

class AccountNotFoundException extends Exception{

public String toString(){

return "AccountNotFoundException: Requested Account Non-Existent";

}

}

class Account{

private String cname;

private int pan;

private int accno;

private String branch;

private double balance;

public Account(String cname,int pan,int accno,String branch,double balance){

this.cname=cname;

this.pan=pan;

this.accno=accno;

this.branch=branch;

this.balance=balance;

}

public String getCName(){return cname;}

public int getPAN(){return pan;}

public int getAccNo(){return accno;}

public String getBranch(){return branch;}

public double getBalance(){return balance;}

public void deposit(int accno,double amt) throws PanRequiredException{

if(amt>50000)

throw new PanRequiredException();

else

balance+=amt;

}

public void withdraw(int accno,double amt) throws MinBalRequiredException,NotEnoughMoneyInAccountException{

if(balance<amt)

throw new NotEnoughMoneyInAccountException();

else if((balance-amt)<1000.0)

throw new MinBalRequiredException();

else

balance-=amt;

}

public void search(int accno)throws AccountNotFoundException{

if(this.accno==accno){

System.out.println("\nName: "+cname);

System.out.println("PAN: "+pan);

System.out.println("Account Number: "+accno);

System.out.println("Branch: "+branch);

System.out.println("Current Balance: "+balance);

}

else

throw new AccountNotFoundException();

}

public String toString(){

return "Account Number "+accno+" has produced ";

}

}

public class AccountHandling{

public static void main(String args[]){

String name,branch;

int pin,accno;

double bal;

int num;

double amt;

int ano;

int opt;

Scanner in=new Scanner(System.in);

System.out.print("\nEnter limit ");num=in.nextInt();

Account acc[]=new Account[100];

for(int i=0;i<num;i++){

System.out.println("Enter Account Details ");

in.nextLine();

System.out.print("Enter Name: ");name=in.nextLine();

System.out.print("Enter PIN number: ");pin=in.nextInt();

System.out.print("Enter Account number: ");accno=in.nextInt();

in.nextLine();

System.out.print("Enter Branch: ");branch=in.nextLine();

System.out.print("Enter Balance: ");bal=in.nextDouble();

acc[i]=new Account(name,pin,accno,branch,bal);

}

do{

System.out.println("\nWhat do? \n 1-Deposit\n 2-Withdraw");

System.out.println(" 0-Exit");

System.out.print(" Your choice: "); opt=in.nextInt();

if(opt==1){

System.out.print("Enter amount to deposit: ");

amt=in.nextDouble();

System.out.print("Enter account number: ");

ano=in.nextInt();

for(int i=0;i<num;i++){

try{

acc[i].search(ano);

try{

acc[i].deposit(ano,amt);

System.out.println("\nAfter deposit ");

acc[i].search(ano);

break;

}

catch(PanRequiredException pre){

System.out.println(acc[i].toString()+pre.toString());break;

}

}

catch(AccountNotFoundException anfe){

System.out.println(acc[i].toString()+anfe.toString());

}

}

}

else if(opt==2){

System.out.print("Enter amount to withdraw: ");

amt=in.nextDouble();

System.out.print("Enter account number: ");

ano=in.nextInt();

for(int i=0;i<num;i++){

try{

acc[i].search(ano);

try{

acc[i].withdraw(ano,amt);

System.out.println("After deposit ");

acc[i].search(ano);

break;

}

catch(MinBalRequiredException pre){

System.out.println(acc[i].toString()+pre.toString());break;

}

catch(NotEnoughMoneyInAccountException nemiae){

System.out.println(acc[i].toString()+nemiae.toString());break;

}

}

catch(AccountNotFoundException anfe){

System.out.println(acc[i].toString()+anfe.toString());

}

}

}

else if(opt!=0)

System.out.println("Invalid option ");

}while(opt!=0);

}

}

/\*

Output:

Enter limit 3

Enter Account Details

Enter Name: 1

Enter PIN number: 1

Enter Account number: 1

Enter Branch: 1

Enter Balance: 50000

Enter Account Details

Enter Name: 2

Enter PIN number: 2

Enter Account number: 2

Enter Branch: 2

Enter Balance: 200000

Enter Account Details

Enter Name: 3

Enter PIN number: 3

Enter Account number: 3

Enter Branch: 3

Enter Balance: 150000

What do?

1-Deposit

2-Withdraw

0-Exit

Your choice: 1

Enter amount to deposit: 10000

Enter account number: 1

Name: 1

PAN: 1

Account Number: 1

Branch: 1

Current Balance: 50000.0

After deposit

Name: 1

PAN: 1

Account Number: 1

Branch: 1

Current Balance: 60000.0

What do?

1-Deposit

2-Withdraw

0-Exit

Your choice: 2

Enter amount to withdraw: 70000

Enter account number: 1

Name: 1

PAN: 1

Account Number: 1

Branch: 1

Current Balance: 60000.0

Account Number 1 has produced NotEnoughMoneyInAccountException: Insufficient Funds To Process Request

What do?

1-Deposit

2-Withdraw

0-Exit

Your choice: 2

Enter amount to withdraw: 59500

Enter account number: 1

Name: 1

PAN: 1

Account Number: 1

Branch: 1

Current Balance: 60000.0

Account Number 1 has produced MinBalRequiredException: Balance Tending Below Minimum Limit

What do?

1-Deposit

2-Withdraw

0-Exit

Your choice: 0

\*/