**Basics of Java Practice Exercise – Day 4**

Submitted by – Aniket Singh (Emp no. - 2605511)

**1. Create a class Bank Account having the members as given below:**

**Field Names:**

**• accNo (int)**

**• custName (String)**

**• accType (String)**

**• balance (float)**

**Method Description:**

**• public void deposit(float amt) : This method allows you to credit an amount into the current balance. If amount is negative, throw an exception Negative Amount to block the operation from being performed.**

**• public float getBalance() : This method returns the current balance. If the current balance is below the minimum required balance, then throw an exception LowBalanceException accordingly.**

**Have a constructor to which you will pass, accNo, custName, acctype and initial balance. And check whether the balance is less than 1000 or not in case of savings account and less than 5000 in case of a current account. If so, then raise a LowBalanceException. In either case if the balance is negative then raise the NegativeAmount exception accordingly.**

**Input 1:**

**123 John Current 4000**

**Output 1:**

**LowBalance**

**Input 2:**

**123 John Saving -900**

**Output 2:**

**NegativeAmount**

**Code –**

**package** practicsday4;

**import** java.util.\*;

**class** NegativeAmount **extends** Exception{

**public** NegativeAmount(String msg) {

**super**(msg);

}

}

**class** LowBalanceException **extends** Exception{

**public** LowBalanceException(String msg) {

**super**(msg);

}

}

**public** **class** BankAccount {

**int** accno;

String custName;

String accType;

**float** balance;

**public** BankAccount(**int** accno, String custName, String accType, **float** balance) **throws** NegativeAmount, LowBalanceException {

**super**();

**this**.accno = accno;

**this**.custName = custName;

**this**.accType = accType;

**if**(balance<0) {

**throw** **new** NegativeAmount("Negative Balance");

}

**if**(accType.equalsIgnoreCase("saving") && balance<1000) {

**throw** **new** LowBalanceException("Low Balance for saving account");

}

**else** **if**(accType.equalsIgnoreCase("current") && balance<5000) {

**throw** **new** LowBalanceException("Low Balance for current account");

}

**this**.balance=balance;

}

**public** **void** deposit(**float** amt) **throws** NegativeAmount{

**if**(amt<0) {

**throw** **new** NegativeAmount("Negative amount entered");

}

balance=balance+amt;

}

**public** **float** getBalance() **throws** LowBalanceException{

**if**(accType.equalsIgnoreCase("saving") && balance<1000) {

**throw** **new** LowBalanceException("Low Balance for saving account");

}

**else** **if**(accType.equalsIgnoreCase("current") && balance<5000) {

**throw** **new** LowBalanceException("Low Balance for current account");

}

**return** balance;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** accno;

String custName;

String accType;

**float** balance;

System.***out***.println("Enter Account Number: ");

accno=sc.nextInt();

System.***out***.println("Enter Customer Name: ");

custName=sc.next();

System.***out***.println("Enter Account Type: ");

accType=sc.next();

System.***out***.println("Enter Account Initial Balance: ");

balance=sc.nextFloat();

**try** {

BankAccount acc1= **new** BankAccount(accno,custName,accType,balance);

}

**catch**(NegativeAmount | LowBalanceException e) {

System.***out***.println(e.getMessage());

}

System.***out***.println();

System.***out***.println("Enter 1 to Deposit");

System.***out***.println("Enter 2 to Check Balance");

System.***out***.println();

**int** num,dep,balenew;

num=sc.nextInt();

**if**(num==1) {

System.***out***.println("Enter Amount to Deposit: ");

**try** {

dep=sc.nextInt();

BankAccount acc2=**new** BankAccount(accno,custName,accType,balance);

acc2.deposit(dep);

System.***out***.println("Updated Balance: "+acc2.balance);

}

**catch**(NegativeAmount | LowBalanceException e) {

System.***out***.println(e.getMessage());

}

}

**else** **if**(num==2) {

**try** {

BankAccount acc2=**new** BankAccount(accno,custName,accType,balance);

System.***out***.println("Balance is: ");

**float** temp=acc2.getBalance();

System.***out***.println(temp);

}

**catch**(NegativeAmount | LowBalanceException e) {

System.***out***.println(e.getMessage());

}

}

**else** {

System.***out***.println("Wrong Choice:");

}

}

}

**West City Union is a cricket club that maintains an average rating of the players and provides them with coins based on the rating obtained by the critics. The club can only have three critics. Create a class called Cricket Rating with members as given below:**

**Field Names:**

**• playerName (String)**

**• critic1 (float)**

**• critic2 (float)**

**• critic3 (float**

**• avgRating (float)**

**• Coins (String)**

**Method Description:**

**• void calculateAverageRating(critic1,critic2) : This method Calculates Rating based on two critics.**

**• void calculateAverageRating(critic1,critic2,critic3) : This method Calculates Rating based on three critics.**

**• String calculateCoins() : This method returns the type of coin achieved by the player based on the rating.**

**• void display() : This method displays all the information.**

**The type of coin achieved by the player based on the rating is given below:**

**➢ If the avgRating is greater than or equal to 7 then the player gains gold coin.**

**➢ If the avgRating is greater than or equal to 5 and less than 7 then the player gains silver coin.**

**➢ If the avgRating is greater than or equal to 2 and less than 5 then the player gains copper Coin.**

**➢ If the avgRating is less than 2 then throw a NotEligibleException.**

**Provide appropriate constructor(s) that accept values to be passed to the attributes.**

**Implement the Tester class.**

**Input 1:**

**John 9.3 9.67 8.7**

**Output 1:**

**John 9.22 Gold**

**Input 2:**

**Henry 1.5**

**Output 2:**

**NotEligible**

**Code –**

**package** practicsday4;

**import** java.util.\*;

**class** NotEligible **extends** Exception{

**public** NotEligible(String message) {

**super**(message);

}

}

**public** **class** CricketRating {

String playerName;

**float** critic1,critic2,critic3,avgRating;

String coins;

**public** CricketRating(String playerName, **float** critic1, **float** critic2, **float** critic3) {

**super**();

**this**.playerName = playerName;

**this**.critic1 = critic1;

**this**.critic2 = critic2;

**this**.critic3 = critic3;

}

**public** CricketRating(String playerName, **float** critic1, **float** critic2) {

**super**();

**this**.playerName = playerName;

**this**.critic1 = critic1;

**this**.critic2 = critic2;

}

**public** **void** calculateAverageRating(**float** critic1,**float** critic2) {

**this**.avgRating=(critic1+critic2)/2;

}

**public** **void** calculateAverageRating(**float** critic1,**float** critic2,**float** critic3) {

**this**.avgRating=(critic1+critic2+critic3)/3;

}

**public** String calculateCoins() **throws** NotEligible{

**if**(avgRating>=7) {

**return** "Gold";

}

**else** **if**(avgRating>=5){

**return** "Silver";

}

**else** **if**(avgRating>=2){

**return** "Copper";

}

**else** {

**throw** **new** NotEligible("Not Eligible");

}

}

**public** **void** display() {

**try** {

**this**.coins=calculateCoins();

System.***out***.println(playerName+ " " + avgRating+ " "+coins);

}

**catch**(NotEligible e) {

System.***out***.println(e.getMessage());

}

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

String playerName;

**float** critic1,critic2,critic3;

**int** num;

System.***out***.println("Enter Player Name: ");

playerName=sc.next();

System.***out***.println();

System.***out***.println("Enter Number of Critics: ");

num=sc.nextInt();

System.***out***.println("Enter Critic1 Rating: ");

critic1=sc.nextFloat();

System.***out***.println("Enter Critic1 Rating: ");

critic2=sc.nextFloat();

**if**(num==2) {

CricketRating cr=**new** CricketRating(playerName,critic1,critic2);

cr.calculateAverageRating(critic1, critic2);

System.***out***.println();

cr.display();

}

**else** **if**(num==3) {

System.***out***.println("Enter Critic1 Rating: ");

critic3=sc.nextFloat();

CricketRating cr=**new** CricketRating(playerName,critic1,critic2,critic3);

cr.calculateAverageRating(critic1, critic2, critic3);

System.***out***.println();

cr.display();

}

**else** {

System.***out***.println();

System.***out***.println("Wrong Input");

}

}

}

**3. Cathey Bank wants to conduct examinations for the post of Probationary Officers, Assistants, and Special Cadre Officers. It has rolled out an online application which is available on the Bank’s website. The applicants can fill in the application form and submit it with accurate details. Assuming that each Applicant is represented by the following class members:**

**Field Names:**

**• applicantName (String)**

**• postApplied (String)**

**• applicantAge (int)**

**Define a user-defined Exception CatheyBankException.**

**Design a Validator class which has following methods for validating applicant details.**

**Method Description:**

**• validate(Applicant applicant): This method receives the Applicant and calls the respective methods to validate the values. If validation fails, it throws user-defined exceptions CatheyBankException with the exception message as given below:**

**➢ If the violation is for applicant name, then throw InvalidNameException with the message “Invalid Applicant Name”.**

**➢ If the violation is for post, then throw InvalidPostException with the message “Invalid Post”.**

**➢ If the violation is for age, then throw InvalidAgeException with the message “Invalid Age”.**

**If all the values are valid print the message “All details are valid”.**

**• isValidApplicantName(String name): This method validates applicantName. It cannot be null or empty. If the rule is violated, then it should return false else it should return true.**

**• isValidPost(String post): This method validates the post the applicant applied for. It should be one among one of the following posts: “Probationary Officer”, “Assistant”, or “Special Cadre Officer”. If the rule is violated, then it should return false else it should return true.**

**• isValidAge(Integer age): This method validates the age of the applicant. It should be greater than 18 years and less than 30 years. If the rule is violated, then it should return false else it should return true.**

**Create a Tester class and implement it.**

**Input 1:**

**Mary Assistant 34**

**Output 1:**

**Invalid Age**

**Input 2:**

**Mary Clerk 27**

**Output 2:**

**Invalid Post**

**Input 3:**

**Probationary Officer 30**

**Output 3:**

**Invalid Applicant Name**

**Input 4:**

**Joseph Probationary Officer 30**

**Output 4:**

**All details are valid**

**Code –**

**package** practicsday4;

**import** java.util.\*;

**class** CatheyBankException **extends** Exception {

**public** CatheyBankException(String message) {

**super**(message);

}

}

**class** InvalidNameException **extends** CatheyBankException {

**public** InvalidNameException(String message) {

**super**(message);

}

}

**class** InvalidPostException **extends** CatheyBankException {

**public** InvalidPostException(String message) {

**super**(message);

}

}

**class** InvalidAgeException **extends** CatheyBankException {

**public** InvalidAgeException(String message) {

**super**(message);

}

}

**class** Applicant {

String applicantName;

String postApplied;

**int** applicantAge;

**public** Applicant(String applicantName, String postApplied, **int** applicantAge) {

**this**.applicantName = applicantName;

**this**.postApplied = postApplied;

**this**.applicantAge = applicantAge;

}

**public** String getApplicantName() {

**return** applicantName;

}

**public** String getPostApplied() {

**return** postApplied;

}

**public** **int** getApplicantAge() {

**return** applicantAge;

}

}

**class** Validator {

**public** **void** validate(Applicant applicant) **throws** CatheyBankException {

**if** (!isValidApplicantName(applicant.getApplicantName())) {

**throw** **new** InvalidNameException("Invalid Applicant Name");

}

**if** (!isValidPost(applicant.getPostApplied())) {

**throw** **new** InvalidPostException("Invalid Post");

}

**if** (!isValidAge(applicant.getApplicantAge())) {

**throw** **new** InvalidAgeException("Invalid Age");

}

System.***out***.println("All details are valid");

}

**public** **boolean** isValidApplicantName(String name) {

**return** name != **null** && !name.isEmpty();

}

**public** **boolean** isValidPost(String post) {

**return** post != **null** && (post.equalsIgnoreCase("Probationary Officer") || post.equalsIgnoreCase("Assistant") || post.equalsIgnoreCase("Special Cadre Officer"));

}

**public** **boolean** isValidAge(**int** age) {

**return** age > 18 && age < 30;

}

}

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

String applicantName;

String postApplied;

**int** applicantAge;

System.***out***.println("Enter Applicant Name: ");

applicantName=sc.nextLine();

System.***out***.println("Enter Post Applied for: ");

postApplied=sc.nextLine();

System.***out***.println("Enter Applicant Age: ");

applicantAge=sc.nextInt();

Applicant applicant1 = **new** Applicant(applicantName,postApplied,applicantAge);

Validator validator = **new** Validator();

System.***out***.println();

**try** {

validator.validate(applicant1);

} **catch** (CatheyBankException e) {

System.***out***.println("Validation Error: " + e.getMessage());

}

}

}