**Exception Handling in Java**

**1. Create a class BankAccount 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 NegativeAmount toblock the operation frombeing performed.**

**• public float getBalance() : This method returnsthe current balance. If the current balance isbelow the minimum required balance, then throw an exception LowBalanceExceptionaccordingly.**

**Have a constructorto 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 acurrent account.**

**If so, then raise a LowBalanceException. In either case if the balance is negative then raise the NegativeAmount exception accordingly.**

**package** day4assignment;

**import** java.util.Scanner;

**class** lowbalanceexception **extends** Exception {

**public** lowbalanceexception(String message) {

**super**(message);

}

}

**class** negativebalance **extends** Exception {

**public** negativebalance(String message) {

**super**(message);

}

}

**class** Bankaccount{

**int** accno;

String custname;

String acctype;

**float** balance;

**public** Bankaccount(**int** accno,String custname,String acctype,**float** balance)**throws** lowbalanceexception,negativebalance {

**this**.accno=accno;

**this**.custname=custname;

**this**.acctype=acctype;

**this**.balance=balance;

**if**(balance<0) {

**throw** **new** negativebalance("negative balance");

}

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

**throw** **new** lowbalanceexception("low balance");

}

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

**throw** **new** lowbalanceexception("low balance");

}

}

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

**if**(amt<0) {

**throw** **new** negativebalance("negative balance unable to deposite");

}

**this**.balance+=amt;

}

**public** **float** getbalance()**throws** lowbalanceexception{

**if**(acctype.equalsIgnoreCase("saving")&& balance<1000 ||acctype.equalsIgnoreCase("current")&& balance<5000) {

**throw** **new** lowbalanceexception("low balance");

}

**return** **this**.balance;

}

}

**public** **class** Bank{

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

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

System.***out***.println("enter accno:");

**int** accno = sc.nextInt();

sc.nextLine();

System.***out***.println("enter custname:");

String custname = sc.nextLine();

System.***out***.println("enter accotype(current/saving):");

String acctype = sc.nextLine();

System.***out***.println("enter intial amount:");

**float** balance = sc.nextFloat();

**try** {

Bankaccount account = **new** Bankaccount(accno,custname,acctype,balance);

System.***out***.println("account created");

System.***out***.println("the intial balance:" +account.getbalance());

System.***out***.println("enter amount to deposit:");

**float** deposit = sc.nextFloat();

account.deposit(deposit);

System.***out***.println("updated balance:" +account.getbalance());

}

**catch** (lowbalanceexception | negativebalance e) {

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

}

}

}

**Output:**

enter accno:

123

enter custname:

john

enter accotype(current/saving):

current

enter intial amount:

4000

low balance

**output:**

enter accno:

123

enter custname:

john

enter accotype(current/saving):

saving

enter intial amount:

-900

negative balance

**2. WestCity 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 CricketRating 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 twocritics.**

**• 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 basedonthe 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.**

**Implementthe Tester class**.

**package** day4assignment;

**import** java.util.Scanner;

**class** NotEligibleException **extends** Exception {

**public** NotEligibleException(String message) {

**super**(message);

}

}

**class** Cricket {

String playername,coins;

**float** critic1,critic2,critic3,avgrating;

**public** Cricket(String playername,**float** critic1,**float** critic2,**float** critic3) {

**this**.playername=playername;

**this**.critic1=critic1;

**this**.critic2=critic2;

**this**.critic3=critic3;

calculateAverageRating(critic1, critic2, critic3);

}

**public** Cricket(String playername, **float** critic1, **float** critic2) {

**this**.playername = playername;

**this**.critic1 = critic1;

**this**.critic2 = critic2;

**this**.critic3 = 0;

calculateAverageRating(critic1, critic2);

}

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

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

calculateCoins();

}

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

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

calculateCoins();

}

**public** **void** calculateCoins() {

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

coins = "Gold";

} **else** **if** (avgrating >= 5) {

coins = "Silver";

} **else** **if** (avgrating >= 2) {

coins = "Copper";

} **else** {

**try** {

**throw** **new** NotEligibleException("NotEligible");

} **catch** (NotEligibleException e) {

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

}

}

}

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

**if** (avgrating >= 2) {

System.***out***.println(playername + " " + String.*format*("%.2f", avgrating) + " " + coins);

}

}

}

**public** **class** Cricketrating {

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

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

System.***out***.print("Enter Player Name,Critic 1,Critic 2,Critic 3: ");

String playername = sc.nextLine();

**float** critic1 = sc.nextFloat();

**float** critic2 = sc.nextFloat();

**float** critic3 = sc.nextFloat();

**if** (critic3 == 0) {

Cricket player = **new** Cricket(playername,critic1,critic2);

player.display();

} **else** {

Cricket player = **new** Cricket(playername,critic1,critic2,critic3);

player.display();

}

}

}

**Output:**

Enter Player Name,Critic 1,Critic 2,Critic 3:

john

9.3

9.67

8.7

john 9.22 Gold

**output:**

Enter Player Name,Critic 1,Critic 2,Critic 3:

henry

1.5

0

0

NotEligible

**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.**

**package** day4assignment;

**import** java.util.Scanner;

**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, postApplied;

**int** applicantAge;

**public** Applicant(String name, String post, **int** age) {

**this**.applicantName = name;

**this**.postApplied = post;

**this**.applicantAge = age;

}

}

**class** Validator {

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

**if** (!isValidApplicantName(applicant.applicantName)) **throw** **new** InvalidNameException("Invalid Applicant Name");

**if** (!isValidPost(applicant.postApplied)) **throw** **new** InvalidPostException("Invalid Post");

**if** (!isValidAge(applicant.applicantAge)) **throw** **new** InvalidAgeException("Invalid Age");

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

}

**private** **boolean** isValidApplicantName(String name) { **return** name != **null** && !name.isEmpty(); }

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

**return** post.equals("Probationary Officer")

|| post.equals("Assistant")

|| post.equals("Special Cadre Officer");

}

**private** **boolean** isValidAge(**int** age) { **return** age > 18 && age < 30; }

}

**public** **class** Tester {

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

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

System.***out***.println("enter name,post,age:");

Applicant applicant = **new** Applicant(sc.next(), sc.next(), sc.nextInt());

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

**try** {

validator.validate(applicant);

} **catch** (CatheyBankException e) {

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

}

}

}

**Output**:

enter name,post,age:

mary

Assistant

34

Invalid Age