| **Name:** | Mahadev Balla |
| --- | --- |
| **UID:** | 2023300010 |
| **Experiment No.** | 9 |

| **AIM:** | Implement programs to demonstrate exception handling. |
| --- | --- |
| **Program 1** | |
| **PROBLEM STATEMENT :** | There is an abstract class Account  Attribute:-  ● Name  ● Balance  ● Acc\_No  Method:-  ● Deposit - abstract method  ● withdraw - abstract method  ● display - abstract method  Saving Account inherits the Account class and provides the implementation for the methods accordingly  Saving Account class Attribute:-  ● interestRate  ● minBalance  Method  ● addInterest: handle Arithmetic Exception  ● transfer():  Note:  ● Balance cannot be less than 0.  ● In a Saving account if minBalance is set then for that the balance cannot go less than that amount. If it goes, an error must be shown.  ● let the user deposit to or withdraw from the account. For each transaction, a message is displayed to indicate the status of the transaction: successful or failed. In case of failure, the failure reason is reported.  ● The possible Exceptions are negative-amount-exception and insufficient-amount-exception.  For the above scenario write an interactive program in Java. Also, show output for different use cases. |
| **PROGRAM:** | import java.util.\*;  abstract class Account{  String depositorName;  long accountNumber;  double balance;  Account(String name, long accNumber, double bal) {  depositorName = name;  accountNumber = accNumber;  balance = bal;  }    abstract void deposit(double amount);  abstract void withdraw(double amount);  abstract void display();  }  class SavingsAccount extends Account{  double rate;  SavingsAccount(String name, long accNumber, double bal, double rate){  super(name, accNumber, bal); this.rate = rate;  }    void deposit(double amount){  balance += amount;  System.out.println("Amount deposited successfully.\nBalance : " + balance);  }    void withdraw(double amount){  balance -= amount;  System.out.println("Amount withdrawn successfully.\nBalance : " + balance);  }    void display(){  System.out.println("Account holder's name : " + depositorName);  System.out.println("Account number : " + accountNumber);  System.out.println("Balance : " + balance);  }  double getSimpleInterest(double principal, double rate, double time){  balance = principal \* (Math.pow((1 + (rate/100)), (time/365)));  return balance;  }  }  class UserException extends Exception{  public UserException(String str){  super(str);  }  }  class trycatchbank{  public static void main(String args[]){  Scanner sc = new Scanner(System.in);  int time; double balance, minbalance, rate; long accno;  System.out.println("Enter your details -");  System.out.print("Name : ");  String name = sc.nextLine();  do{  System.out.print("Account number : ");  accno= sc.nextInt();  }  while(accno<0);  do{  System.out.print("Set Min. Balance : ");  minbalance = sc.nextDouble();  if(minbalance<0){  try{  throw new UserException("User Defined Exception");  }  catch(UserException e){  System.out.println("Minimum balance should be greater than 0");  }  }  }  while(minbalance<0);  do{  System.out.print("Balance : ");  balance = sc.nextDouble();  if(balance<minbalance){  try{  throw new UserException("User Defined Exception");  }  catch(UserException e){  System.out.println("Balance should be greater than min. balance(" + minbalance + ")");  }  }  }  while(balance<minbalance);  do{  System.out.print("Rate of Interest : ");  rate = sc.nextDouble();  if(rate<0){  try {  throw new UserException("User Defined Exception");  } catch (UserException e) {  System.out.println("Rate of interest can't be negative.");  }  }  }  while(rate<0);  do{  System.out.print("Time Period(in days) : ");  time = sc.nextInt();  if(time<0){  try {  throw new UserException("User Defined Exception");  } catch (UserException e) {  System.out.println("Number of days can't be negative!!");  }  }  }  while(time<0);  SavingsAccount account = new SavingsAccount(name, accno, balance, rate);  System.out.println("New balance : " + account.getSimpleInterest(balance, rate, time));  int x;  do{  System.out.print("\n1. Deposit\n2. Withdraw\n3. Display\n4. Exit\nEnter your choice : ");  x = sc.nextInt();  switch(x){  case 1: System.out.print("Enter amount to be deposited : ");  double depamt = sc.nextDouble();  if(depamt<0){  try {  throw new UserException("User Defined Exception");  } catch (UserException e) {  System.out.println("Negative amount can't be deposited");  }  }  else{  account.deposit(depamt);  }  break;  case 2: System.out.print("Enter amount to be withdrawn : ");  double withamt = sc.nextDouble();  if(withamt>account.balance){  try {  throw new UserException("User Defined Exception");  } catch (UserException e) {  System.out.println("Transaction failed. Insufficient funds!!");  }  }  else if(withamt<=0){  try {  throw new UserException("User Defined Exception");  } catch (UserException e) {  System.out.println("Negative/Zero amount can't be withdrawn.");  }  }  else{  account.withdraw(withamt);  }  break;  case 3: account.display();  break;  case 4: System.out.println("Thank you!!");  break;  default:  System.out.println("Invalid choice!!");  }  }  while (x!=4);  }  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | Define a class Cricketer which has:-  Attributes:-  ● player\_name  ● runs\_hit  ● innings\_count  ● not\_out\_count  ● batting\_avg  Methods:-get\_avg  Make a cricket team with 11 cricketers. For each cricketer, find his batting average. Handle all different errors while calculating this. Also, make a method which will find he list of cricketers in ascending order of their batting average and also display the cricketer stats in this order.  Note- handle errors like ArrayIndexOutOfBoundsException, ArithmeticException,ArrayStoreException, NumberFormatException, etc. |
| **PROGRAM:** | import java.util.\*;  class Cricketer{  String player\_name;  int runs\_hit, innings\_cnt, not\_out\_cnt;  double batting\_avg;  Cricketer(String player\_name, int runs\_hit, int innings\_cnt, int not\_out\_cnt){  this.player\_name = player\_name;  this.runs\_hit = runs\_hit;  this.innings\_cnt = innings\_cnt;  this.not\_out\_cnt = not\_out\_cnt;  }  double get\_avg(){  if(innings\_cnt == not\_out\_cnt){  batting\_avg = runs\_hit;  }  else{  batting\_avg = runs\_hit/(innings\_cnt-not\_out\_cnt);  }  return batting\_avg;  }    void print(){  System.out.printf("%-25s %-15s %-15s %-15s %-15s \n", player\_name, batting\_avg, runs\_hit, innings\_cnt, not\_out\_cnt);  }  }  class UserException extends Exception{  public UserException(String str){  super(str);  }  }  class cricket{  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  String player\_name;  int runs\_hit, innings\_cnt, not\_out\_cnt, x=0;  double batting\_avg;    System.out.print("Enter no of players : ");  x = sc.nextInt();    sc.nextLine();  Cricketer c[] = new Cricketer[11];    for(int i=0; i<x; i++){  try{  System.out.println("Enter details of player " + (i+1) + " -");    System.out.print("Name : ");  player\_name = sc.nextLine();    do{  System.out.print("Runs Hit : ");  runs\_hit = sc.nextInt();  if(runs\_hit<0){  try{  throw new ArithmeticException();  }  catch(ArithmeticException e){  System.out.println(e + " - Number of runs scored can't be negative.");  }  catch(NumberFormatException e){  System.out.println(e + " - Number format exception occured. Enter a valid input!!");  }  }  }  while(runs\_hit<0);    do{  System.out.print("Innings Count : ");  innings\_cnt = sc.nextInt();  if(innings\_cnt<0){  try{  throw new ArithmeticException();  }  catch(ArithmeticException e){  System.out.println(e + " - Innings count can't be negative.");  }  catch(NumberFormatException e){  System.out.println(e + " - Number format exception occured. Enter a valid input!!");  }  }  }  while(innings\_cnt<0);    do{  System.out.print("Not Out Count : ");  not\_out\_cnt = sc.nextInt();  if(not\_out\_cnt<0){  try{  throw new ArithmeticException();  }  catch(ArithmeticException e){  System.out.println(e + " - Not out count can't be negative.");  }  catch(NumberFormatException e){  System.out.println(e + " - Number format exception occured. Enter a valid input!!");  }  }  else if(not\_out\_cnt>innings\_cnt){  try{  throw new UserException("User Defined Exception");  }  catch(Exception e){  System.out.println("Not out count can't be greater than innings count.");  }  }  }  while(not\_out\_cnt<0 || not\_out\_cnt>innings\_cnt);  c[i] = new Cricketer(player\_name, runs\_hit, innings\_cnt, not\_out\_cnt);  }  catch (ArrayIndexOutOfBoundsException e){  System.out.println(e);  }  catch(ArrayStoreException e){  System.out.println(e);  }  sc.nextLine();  if(i==12){  break;  }  }    sort(c);    System.out.println("Sorted List of Players in ascending order of their batting average -");  System.out.printf("%-25s %-15s %-15s %-15s %-15s\n", "Name", "Batting Average", "Runs Hit", "Innings Count", "Not Out Count");  for(int k=0; k<c.length; k++){  c[k].print();  }  }  static void sort(Cricketer c[]){  for(int i=0; i<c.length; i++){  for(int j=0; j<c.length-i-1; j++){  if(c[j].get\_avg()>c[j+1].get\_avg()){  Cricketer temp = c[j];  c[j] = c[j+1];  c[j+1] = temp;  }  }  }  }  } |
| **RESULT:** | |
| **Program 3** | |
| **PROBLEM STATEMENT :** | Write a program to accept gap between two vaccine dose from 1-84 as input from user. If the user enters <84 days as an input or if user enters any negative number, or >100 user defined exception should be generated. |
| **PROGRAM:** | import java.util.\*;  class UserException extends Exception{  public UserException(String str){  super(str);  }  }  class vaccinegap {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int days;  do {  System.out.print("Enter gap between two vaccine doses : ");  days = sc.nextInt();  if(days<0){  try{  throw new ArithmeticException();  }  catch(Exception e){  System.out.println(e + " - Gap between two vaccine doses can't be negative.");  }  }  else if(days==0){  try {  throw new UserException("User Defined Exception");  } catch (Exception e) {  System.out.println(e.getMessage() + " - Can't take two vaccines on the same day!!");  }  }  else if(days>100){  try{  throw new UserException("User Defined Exception");  }  catch(Exception e){  System.out.println("Gap between two vaccine doses can't be greater than 100.");  }  }  else if(days>=1 && days<84){  System.out.println("You're eligible for the second dose.");  }  }  while (days<=0 || days>100 || days>84);  }  } |
| **RESULT:** | |
| **CONCLUSION :** Implemented exception handling in the given problems. | |