|  |  |
| --- | --- |
| **Name** | **Hatim Yusuf Sawai** |
| **UID no.** | **2021300108** |
| **Experiment No.** | **9** |

|  |  |
| --- | --- |
| **AIM:** | To implement user-defined exception handling and catching errors & exceptions in java. |
| **Program 1** | |
| **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 the list of cricketers in ascending order of their batting average  and also display the cricketer stats in this order.  If the average of the batting average of the entire team is less than 20 runs then throw a user-defined exception.  Note- handle errors like ArrayIndexOutOfBoundsException, ArithmeticException,ArrayStoreException, NumberFormatException, etc |
| **PROGRAM:** | import *java*.*util*.*\**;  *class* low\_avg *extends* Exception  {      low\_avg() {          super("Team Average is too low!");      }  }  *public* *class* Cricketer {      String player\_name;      int runs\_hit,innings\_count,not\_out\_count;      double batting\_avg;  *public* Cricketer(String player\_name,int runs\_hit,int innings\_count,int not\_out\_count) {          this.*player\_name* = player\_name;          this.*runs\_hit* = runs\_hit;          this.*innings\_count* = innings\_count;          this.*not\_out\_count* = not\_out\_count;          batting\_avg = 0;      }      void get\_avg() {          try {              batting\_avg = (double)runs\_hit/(innings\_count-not\_out\_count);          }          catch(ArithmeticException e) {              System.*out*.println("batting avg is invalid!");          }      }      void sort\_team(Cricketer [] players) {          Arrays.sort(players,new Comparator<Cricketer>() {              @Override  *public* int compare(Cricketer o1, Cricketer o2) {                  return o1.*batting\_avg*>o2.*batting\_avg*?1:-1;              }          });      }      void print\_team(Cricketer [] players) {          double avg=0;          System.*out*.println("Player\tRuns\tInnings\tN/Os\tBat. Avg");          for(int i=0;i<players.*length*;i++) {  avg+=players[i].*batting\_avg*;          }          try {              avg = avg/players.*length*;              if(avg<20)                  throw new low\_avg();              else                  System.*out*.println("Team Average is "+avg);          }          catch(low\_avg e) {              System.*out*.println(e.getMessage());          }      }  *public* *static* void main(String[] args) {          Scanner sc = new Scanner(System.*in*);          Cricketer[] players = new Cricketer[3];          String player\_name = new String();          int runs\_hit=0,innings\_count=0,not\_out\_count=0;          for(int i=0;i<players.*length*;i++) {              System.*out*.println("\nPlayer "+(i+1));              try {                  System.*out*.print("Enter the name of the player: ");                  player\_name = sc.next();                  System.*out*.print("Enter the number of runs hit: ");                  runs\_hit = sc.nextInt();                  System.*out*.print("Enter the number of innings: ");                  innings\_count = sc.nextInt();                  System.*out*.print("Enter the number of not outs: ");                  not\_out\_count = sc.nextInt();                  players[i] = new Cricketer(player\_name, runs\_hit, innings\_count, not\_out\_count);                  players[i].get\_avg();              }              catch(InputMismatchException e) {                  System.*out*.println("Invalid input!");                  i--;              }          }          players[0].sort\_team(players);          players[0].print\_team(players);          sc.close();      }  } |
| **RESULT:**  **Case: Avg too low (test run with 3 players)** | |
| **Program 2** | |
| **PROBLEM STATEMENT:** | **Write a program to accept distance 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* MyException *extends* Exception {  *public* MyException(int days) {          super();      }  }  *public* *class* Vaccine {  *public* *static* void main(String[] args) {          int days,flag=0;          Scanner sc = new Scanner(System.*in*);          while(flag==0) {              System.*out*.println("Enter days(1-84) between 2 Vaccine doses:");              try {                  days = sc.nextInt();                  if(days>100 || days<0) {                      throw new MyException(days);                  }                  else {                      flag = 1;                  }              }              catch (InputMismatchException e) {                  System.*out*.println("Invalid input(Nust be an integer!)");                  sc.nextLine();                  flag=0;              }              catch (MyException ex) {                  System.*out*.println("Days cannot be more than 100 or negative!");                  flag=0;              }          }          sc.close();      }  } |
| **RESULT:** | |
| **Program 3** | |
| **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 (in both**  **deposit and withdraw transaction) and insufficient-amount-**  **exception ( in withdraw transaction).**  **For the above scenario write an interactive program in Java. Also, show output for different use cases.** |
| **PROGRAM:** | import *java*.*util*.*\**;  *class* negative\_amount *extends* Exception {  *public* negative\_amount(double amt) {          super();      }  }  *class* insufficient\_balance *extends* Exception {  *public* insufficient\_balance(double amt) {          super();      }  }  *abstract* *class* Account {      String name;      long account\_no;      double balance;  *abstract* void deposit(double amt);  *abstract* void withdraw(double amt);  *abstract* void display();  }  *public* *class* SavingAccount *extends* Account {      Scanner sc = new Scanner(System.*in*);      double in\_rate=3.5,minbal=0;      SavingAccount(String name,long account\_no,double balance) {          this.*name* = name;          this.*account\_no* = account\_no;          this.*balance* = balance;      }      void setMinBal(double minbal) {          this.*minbal* = minbal;      }      void addInterest() {          balance = balance + (balance\*in\_rate/100);      }      void deposit(double amt) {          balance += amt;      }      void withdraw(double amt) {          balance -= amt;      }      void display() {          System.*out*.println("Name: "+name);          System.*out*.println("A/c No: "+account\_no);          System.*out*.println("Current Balance: "+balance);      }  *public* *static* void main(String[] args) {          Scanner sc = new Scanner(System.*in*);          String name = new String();          long ac\_no;          double bal,minbal;          System.*out*.print("Enter the name of the account holder: ");          name = sc.nextLine();          System.*out*.print("Enter the account number: ");          ac\_no = sc.nextLong();          System.*out*.print("Enter the initial balance: ");          bal = sc.nextDouble();          SavingAccount sa = new SavingAccount(name,ac\_no,bal);          System.*out*.print("Enter the minimum balance: ");          minbal = sc.nextDouble();          sa.setMinBal(minbal);          double amt;          while(true) {              System.*out*.println("\nWelcome to the Savings Account of "+name+"\nSelect 1 option:\n1.Deposit\n2.Withdraw\n3.Display");              int choice = sc.nextInt();              switch(choice) {                  case 1:                      System.*out*.println("Enter the amount to be deposited:");                      try {                          amt = sc.nextDouble();                          if(amt<0) {                              throw new negative\_amount(amt);                          }                          else {                              sa.deposit(amt);                          }                      }                      catch (negative\_amount ex) {                          System.*out*.println("Amount cannot be negative!");                      }                      break;                  case 2:                      System.*out*.println("Enter the amount to be withdrawn:");                      try {                          amt = sc.nextDouble();                          if(amt>sa.*balance* || sa.*balance*-amt<sa.*minbal*) {                              throw new insufficient\_balance(amt);                          }                          else if(amt<0) {                              throw new negative\_amount(amt);                          }                          else {                              sa.withdraw(amt);                          }                      }                      catch (negative\_amount e) {                          System.*out*.println("Amount cannot be negative!");                      }                      catch (insufficient\_balance e) {                          System.*out*.println("Insufficient balance!");                      }                      break;                  case 3:                      sa.display();                      break;                  default:                      System.*out*.println("Invalid choice!");              }              if(choice!=3) {sa.display();}              System.*out*.println("Do you want to continue?(y/n)");              char ch = sc.next().charAt(0);              if(ch=='n') {                  break;              }          }          sc.close();      }  } |
| **RESULT:**  **Case: Deposit**  **Case: Withdraw**  **Case: Min balance** | |
| **CONCLUSION:** | In this experiment, we learnt how to raise errors and exceptions using try-catch block in java. We also learnt how to throw new User-defined Exceptions which are custom made for that particular program. |