A picture containing text, clipart

Description automatically generated



**JAVA PROGRAMMING**

(340)

**REGIONAL 2024**

**PRODUCTION:**

Java\_Regional \_\_\_\_\_\_\_\_\_\_\_ (390 points)

**Test Time: 90 minutes**

**Regional\_Java**

In this test you will be creating a prototype seating chart software for your local school. The program at this point is solely focused on generating random student information (names, GPA, and grade level).

**Input**

The user will enter in the number of student names to create in a range of 1 to 50 (inclusive). The information entered needs to be whole numbers.



**Output**

The output will be in the format below. The print format has not been created and will need to be completed in the SeatingChartBuilder.java file. For format for the output is as follows:

LastName, FirstName | GradeLevel | GPA

Text

Description automatically generated

The image to the right is an example of the output. NOTE: yours will be different since all the information is randomly generated. First and Last names come from the same list, so there could be instances where a name is used for a First Name in one record, and then is used as a Last Name in another record: this is acceptable.

Text

Description automatically generated**Input Error Wrong Data Type**

If the user enters in a non-positive whole number, the program will inform the user to enter in a correct value. The program will prompt the user to enter in the correct value.

Text

Description automatically generated**Input Error Range Violation**

If the user enters a positive whole number that is beyond the acceptable range of 1 to 50. The program will prompt the user to enter in the correct value.

**Requirements**

1. Your contestant number must appear as a comment at the top of the **SeatingChartBuilder.java** source code file.
2. You will be programming all sections of the **SeatingChartBuilder** class. This class will perform all of the functionality of the program and will also create the **Students** objects. The **Students** class has been created for you and it is located immediately below the **SeatingChartBuilder** class.
3. Code elements and methods in the **SeatingChartBuilder** class:
   1. All of the **Students** objects will need to be stored in a data structure.
   2. You will program all of the ***setStudents*** method. This method will create all the student objects by randomizing the name selection and data generation (grade level and GPA).
   3. You will program all the ***printStudents*** method which oversees formatting and printing the randomly generated students to the console.
   4. You will program all the ***intergerInputManager*** method which is in charge gathering the user’s entry for how many students to create and it also checks the user entry for wrong data type and for range violations.
4. Your program must use a **Scanner** object to get the user input.
5. Your program must use a **Random** object to generate the random integers and double values.
6. Your program must use a **DecimalFormat** object to format the GPA.
7. Your output should look as close as possible to the examples provided.

|  |  |  |
| --- | --- | --- |
| **Solution and Project (There is NO partial credit)** *(NOTE: UC represents uppercase and LC represents lowercase)* | | |
| The Java source file is present on the flash drive in a single folder with your contest ID |  | 20 points |
| **Program Execution (*If the program does not execute, then the remaining items in this section receive a score of zero)*** | | |
| When program starts user it prompt to enter in amount of students to create |  | 20 points |
| Program accepts appropriate data in the prompt |  | 10 points |
| Output of student records is numbered in sequential order |  | 10 points |
| No student names have the same first and last name |  | 20 points |
| Grade levels are randomized and in the range of 9-12 |  | 20 points |
| GPA is randomized and formatted #.00 |  | 20 points |
| Program stops running after list is printed to the console |  | 10 points |
| When entering wrong data values, program give warning message |  | 20 points |
| After entering wrong data values and warning message, program keeps running and prompts user to enter number of students to create |  | 20 points |
| When entering data values out of range, program give warning message |  | 20 points |
| After entering data values out of range and warning message, program keeps running and prompts user to enter number of students to create |  | 20 points |
| **Subtotal** |  | **/210 Points** |

|  |  |  |
| --- | --- | --- |
| **Source Code Review (There is NO partial credit)** *NOTE: you must place the comment flag in front of the comment in your code in order to get credit. The comment flag will precede the explanation. For example, if the flag is SC1, your comment must read as “#SC1…” in front of the part of the code being reviewed. Code must work to get credit. There is no partial credit.* |  |  |
| A comment containing the contestant number is present at the top of the **SeatingChartBuilder.java** file |  | 10 points |
| SC1:**SeatingChartBuilder** class: Place the comment by the code that creates the **Scanner** object |  | 10 points |
| SC2: **SeatingChartBuilder** class: Place the comment by the code that creates the data structure that stores the **Students** objects |  | 10 points |
| SC3: **SeatingChartBuilder** class ***integerInputManager( )***: Place the comment by the code that gathers user input |  | 10 points |
| SC4: **SeatingChartBuilder** class ***integerInputManager( )***: method properly uses the ***try/catch*** code for data type input error |  | 30 points |
| SC5: **SeatingChartBuilder** class ***integerInputManager( )***: method uses strategy for detecting range violations |  | 20 points |
| SC6: **SeatingChartBuilder** class: Place the comment by the code that creates the **Random** object |  | 10 points |
| SC7: **SeatingChartBuilder** class**:** Place the comment by the code that creates the **DecimalFormat** object |  | 10 points |
| SC8: **SeatingChartBuilder** class ***setStudents( )***: method randomly selects students names using **Random** object and properly checks that objects are not equal |  | 20 points |
| SC9: **SeatingChartBuilder** class ***setStudents( )***: method randomly selects students grade levels using **Random** object |  | 10 points |
| SC10: **SeatingChartBuilder** class ***setStudents( )***: method randomly creates GPA using **Random** object, and properly formats object using **DecimalFormat** object |  | 20 points |
| SC11: **SeatingChartBuilder** class ***printStudents( )***: method gets all of student object information using get methods, and formats the printout |  | 20 points |
| **Subtotal** |  | **/180 Points** |
| **Total Points** |  | **/390 Points** |

Machine generated alternative text:
  1 
  2 
  3 import java.io.FileNotFoundException;
  4 import java.io.File;
  5 import java.util.*; 
  6 import java.text.DecimalFormat;
  7 
  8 public class SeatingChartBuilder
  9    {
 10    
 11    static Scanner sc = new Scanner(System.in); //Point //SC1
 12    public static void main (String args [])
 13       {
 14       ArrayList<Students> students = new ArrayList<Students>(); //Point  //SC2
 15       System.out.println("How many test students do you want to create?");
 16       int nameCount = integerInputManager(); //Point
 17       
 18       //Name Only Given. They have to declare and assign  //SC3
 19       String [] allNames ={"Walter","Jones","Rose","Wilson" ,"Jack", "Rodriguez" , "Elizabeth" , "Smith", "
Earl", "Carter", "Linda", "Ward", "Christopher", 
 20          "Turner", "Martin", "Murphy", "Betty", "Garcia", "Shawn", "Taylor","Sean", "Simmons", "Joshua", "E
vans", "Norma", "Mitchell", "Brenda", "Johnson", "Donna", 
 21          "Clark", "Irene", "Diaz","Marilyn", "Coleman","Arthur", "Collins","Henry", "Hall","Howard", "Robin
son","Jerry", "Green","Maria", "Price", "Evelyn", "Bell", 
 22          "Janet", "Moore", "Susan", "Foster"};
 23          
 24       students = setStudents(students, allNames, nameCount);
 25       printStudents(students);
 26    
 27       }
 28    //Point
 29 
 30    private static ArrayList<Students> setStudents(ArrayList<Students> stuList, String [] allNames, int nc)
 31       {
 32       Random rand = new Random();  //SC6
 33       Students s;
 34       ArrayList<Students> students = stuList;
 35       DecimalFormat df = new DecimalFormat("#.00");  //SC7
 36       int nameCount = nc;
 37       double gpa;
 38       int grade;
 39       String fn; String ln; 
 40       for(int i =0; i<nameCount; i++)  //SC8
 41          {
 42          
 43          do{
 44          
 45             fn = allNames[rand.nextInt(allNames.length)];
 46             ln = allNames[rand.nextInt(allNames.length)];
 47             }while(fn.equals(ln)); //SC8
 48          grade = rand.nextInt(4)+9; //SC9
 49          double tempGPA = rand.nextDouble()*3.0 + 1.0; //SC10
 50          String  tempString =df.format(tempGPA); //SC10
 51          gpa = Double.parseDouble(tempString);
 52          s = new Students(fn,ln,grade,gpa);
 53          students.add(s);
 54          }
 55       return students;
 56       }
 57    //Point
 58    //SC11 
 59    private static void printStudents(ArrayList<Students> students){
 60       int i = 1;
 61       for (Students n : students)
 62          {
 63          System.out.println(i+") "+n.getWholeName() + " | Grade Level: "+n.getGradeLevel()+" | GPA: "+n.get


Machine generated alternative text:
GPA());
 64          i++;
 65          }
 66    
 67       }
 68  
 69               
 70    private static int integerInputManager()
 71       {
 72       int temp;
 73       while(true){   
 74          try{  //SC4
 75             do{
 76                System.out.print("Please enter in a value between 1 and 50: ");
 77                temp = sc.nextInt();
 78                if (temp > 50 || temp <1) //SC5
 79                   System.out.println("Your entry is out of range.\n");
 80                sc.nextLine();
 81                }while(temp >50 || temp <1); //SC5
 82             return temp;
 83             
 84             }
 85          catch(InputMismatchException e) //SC4
 86             {
 87             sc.next();
 88             System.out.println("\nPlease enter a correct value.");
 89             }
 90          }
 91       
 92       }  
 93 
 94 
 95    }
 96 //Given
 97 ///////////////////////////////////////////////////
 98 class Students
 99    {
100    String first_Name;
101    String last_Name;
102    int grade_Level;
103    double GPA;
104       
105    public Students()
106       {
107       first_Name = "Dee";
108       last_Name =  "Fault";
109       grade_Level = 0;
110       GPA = 0.0;
111       }
112 
113    public Students(String fn, String ln, int gl, double gpa)
114       {
115       first_Name = fn;
116       last_Name =  ln;
117       grade_Level = gl;
118       GPA = gpa;
119       }
120       
121       //last, first
122    public String getWholeName()
123       {
124       return last_Name + ", "+first_Name;
125       }
126       
127    public String getFirstName()
128       {


Machine generated alternative text:
129       return first_Name;
130       }
131       
132    public String getLastName()
133       {
134       return last_Name;
135       }
136       
137    public double getGPA()
138       {
139       return GPA;
140       }
141       
142    public int getGradeLevel()
143       {
144       return grade_Level;
145       }
146       
147    public void setFirstName(String fn)
148       {
149       first_Name = fn;
150       }
151     
152    public void setLastName(String ln)
153       {
154       last_Name = ln;
155       }
156       
157    }
