**Member ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Rank: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**JAVA PROGRAMMING**

(340)

**REGIONAL 2024**

**PRODUCTION:**

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

**Test Time: 90 minutes**

**GENERAL GUIDELINES:**

*Failure to adhere to any of the following rules will result in disqualification:*

1. Member must hand in this test booklet and all printouts if any. Failure to do so will result in disqualification.
2. No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests (handwritten, photocopied, or keyed) are allowed in the testing area.
3. Electronic devices will be monitored according to ACT standards.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

1. Create a folder on the flash drive provided using your contestant number as the name of the folder.
2. Copy your entire solution/project into this folder. The project folder for you has already been provided: Java\_Regional
3. Submit your entire solution/project so that the graders may open your project to review the source code.
4. Ensure that the files required to run your program are present and will execute on the flash drive provided.
5. You will need to use a local Java IDE to complete this exam.

\*Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

\*It is recommended that you use relative paths rather than absolute paths to ensure that the program will run regardless of the flash drive letter. It is HIGHLY recommended that you place all of the files into one folder.

The graders will *not* compile or alter your source code to correct for this.   
Submissions that do *not* contain source code will *not* be graded.

**Assumptions to make when taking this assessment:**

* The goal of the program is to create a random student record generator. Each Record will have a random first and last name, grade level, and GPA.
* One Java file is provided in the contest folder (SeatingChartBuilder).
* The user will enter a number of students to create.
* The first and last name will be generated from a list of names; the first and last name must be different; however, names can be repeated and used in multiple records.
* The GPA will be randomly created on a range of 1 to 4. The GPA will be formatted to the hundredths place value (#.00).
* The grade level will be randomly generated 9-12.
* The program will terminate once the list is generated.
* The Students class is complete and does not need any adjustments.
* All text for names and messaging is already provided and placed in comments.

**Development Standards:**

* Your Code must use a consistent variable naming convention.
* All subroutines (if any), functions (if any), and methods (if any) must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any). Readability is a goal of good code.

**Commenting for Source Code Review (see the rubric):**

* Certain sections of your code will be graded. These gradable blocks of code can range from creating data structures, method algorithms, exception handling, and class construction.
* The grading rubric contains a section called Source Code Review: in this section are listed a description all of the graded programming concepts.
* Each gradable item must have a comment listed at its beginning, and the comment must be prefixed with the comment flag. The flag helps the graders easily locate the code to increase the effectiveness of grading.
* The flag will always use this naming convention: **SC#** (NOTE: the # symbol will be replaced with sequential numbering, i.e. **SC1, SC2, SC3**, etc.
* No explanation in the comment with the flag is required, only the comment flag; however, any information placed in the comment could help the grader better understand and avoid any costly errors.
* The comment flag needs to be place in close proximity to the block of code it represents.
* If a comment flag is not present, you will not receive credit.
* In this example the Source Code Review has a gradable section of code for printing to the console (remember these are non-related examples):
  + SC12: ***print*** method in the ***main*** classis printing the correct object \_\_\_\_ 10 pts
  + The user will place the code above the method call:

***//SC12 printing the car object***

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

**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**

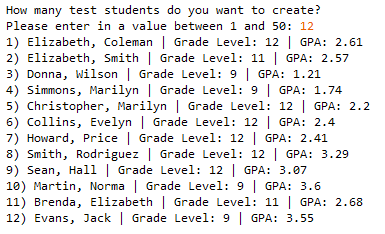
The user will enter in the number of student names to create in a range of 1 to 50 (inclusive). The information entered needs to be whole numbers.

Image shows prompt to user for creating student records. 

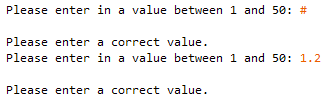
**Output**

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:

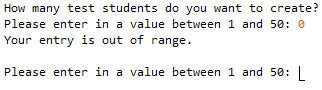
LastName, FirstName | GradeLevel | GPA



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.

**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.

**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 is prompted to enter in number 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** |