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

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

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



**JAVA PROGRAMMING**

(340)

**STATE 2024**

**PRODUCTION:**

Java\_State \_\_\_\_\_\_\_\_\_\_\_ (520 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\_State
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:**

* One Java file is provided in the contest folder (SeatingChartBuilderState).
* The goal of the test is to create a program that can create random student record, allow the user to manually enter a new student, find the highest GPA, lowest GPA, and allow the program to keep running until the user chooses to exit.
* Each student record will be organized last name, first name, grade level, and GPA.
* The random student generation and printing methods have already been completed for you.
* The user will press 1 to create random students (1 to 50), press 2 to create a single student, or press 3 to exit the program.
* The Max GPA (highest) and Min GPA (lowest) appear for the list of randomly created students, and it will appear for the manually entered students.
* The manually created students will not populate the randomly created students.
* The program must detect data value entry errors and range violations.
* When the user presses three, the program must exit.
* All randomly and manually students created will be added to the same list.
* Manually created students need their names adjusted so the first letter is capital and all following letters are lowercase, i.e.: Lastname, Firstname

**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);***

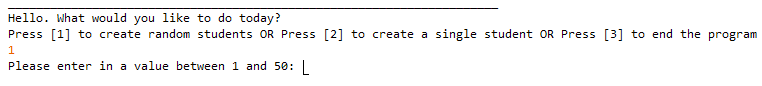
**Java\_State**

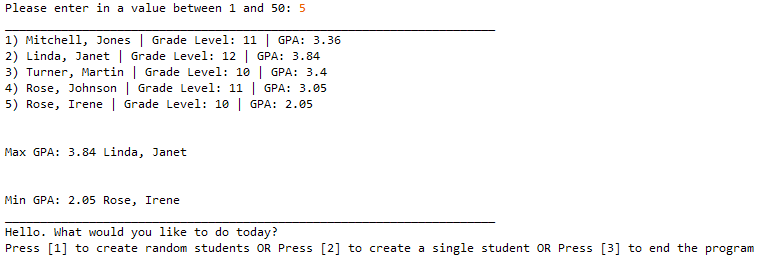
In this test you will be creating another prototype seating chart software for your local school. The program will allow the user to be able to have control options through key board entries. There are three major actions of the program: (1) Randomly create student list; (2) Manually create students; and (3) Exit the program.

**Input/Output Option [1]**

All user input and output will take place in the console.

Image shows step 1.STEP 1: The program begins by asking the user what they would like today.

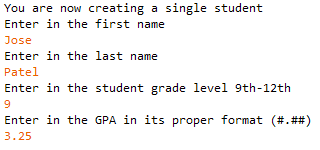
STEP 2 Option [1]: After entering 1, the program will ask the user to enter in a value between 1 and 50 (this number is how many students the program will create).

STEP 3 Option [1]: After entering the amount of student records to create, the program will list out the results. In addition, the program also finds the records with the Max GPA and Min GPA. After the list is created, the program returns to STEP 1.

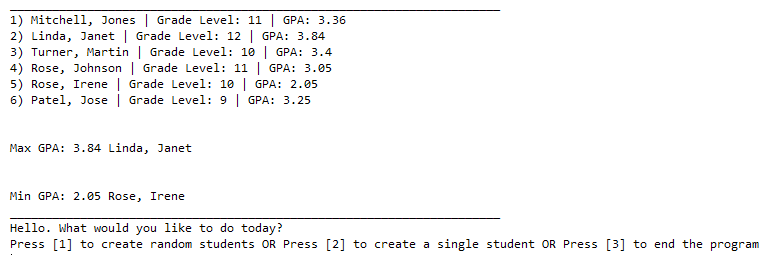
**Input/Output Option [2]**

STEP 1: This the same as before. The user will be entering option [2].

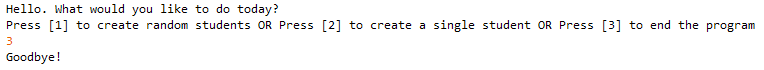
STEPS 2-5 Option [2]: The user will enter in all the student information in this sequence.



* STEP 2: First Name
* STEP 3: Last Name
* STEP 4: Grade Level
* STEP 5: GPA

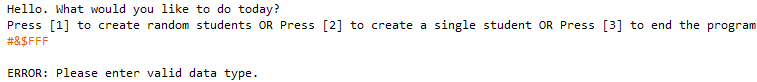
STEP 6: This is the output for the list. Notice that the record created is added to the end of the list that was previously created (the new student created is the 6th student). The program will still need to find the highest and lowest GPA. After listing, the program returns to STEP 1.

**Input/Output Option [3]**

STEP 1: This the same as before. The user will be entering option [3]. This will exit the software.

**Error Messages:**

**Input/Output STEP 1 Invalid Data Error**

STEP 1: the user attempts to enter invalid data. Error message is given. The program needs to return to STEP 1.

**Input/Output STEP 1 Range Violation**

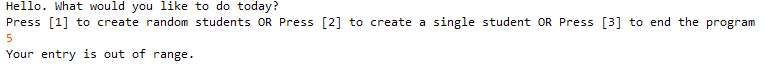
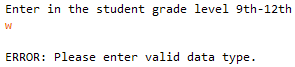
STEP 1: the user attempts to enter value outside of the range. “Your entry is out of range.” message is given. The program needs to return to STEP 1.

Image shows input/Output STEP 2 Option [1] Invalid Data Error**Input/Output STEP 2 Option [1] Invalid Data Error**

STEP 2: the user attempts to enter invalid data. Error message is given. The program needs to return to STEP 2.

**Input/Output STEP 2 Option [1] Range Violation**

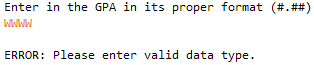
Image shows input/Output STEP 2 Option [1] Range ViolationSTEP 2: the user attempts to enter value outside of the range. “Your entry is out of range.” message is given. The program needs to return to STEP 2.

**Input/Output STEP 4 Option [2] Invalid Data Error**

STEP 4: the user attempts to enter invalid data. Error message is given. The program needs to return to STEP 4.

**Input/Output STEP 4 Option [2] Range Violation**

Image shows input/Output STEP 4 Option [2] invalid data errorSTEP 4: the user attempts to enter value outside of the range. “Your entry is out of range.” message is given. The program needs to return to STEP 4.

**Input/Output STEP 5 Option [2] Invalid Data Error**

STEP 5: the user attempts to enter invalid data. Error message is given. The program needs to return to STEP 5.

**Input/Output STEP 5 Option [2] Range Violation**

Image shows input/Output STEP 5 Option [2] Range ViolationSTEP 5: the user attempts to enter value outside of the range. “Your entry is out of range.” message is given. The program needs to return to STEP 5.

|  |  |  |
| --- | --- | --- |
| **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)*** | | |
| Program starts STEP 1 |  | 10 points |
| STEP 1 Invalid Data Error is handled appropriately |  | 20 points |
| STEP 1 Range Violation is handled appropriately |  | 10 points |
| Option [1] is selected, program moves to appropriate STEP 2 |  | 10 points |
| STEP 2 Option [1] Invalid Data Error is handled appropriately |  | 20 points |
| STEP 2 Option [1] Range Violation is handled appropriately |  | 10 points |
| STEP 3 Option [1] Random students matches output & correct Max/Min GPA displayed |  | 30 points |
| After STEP 3 Option [1] Program returns immediately to active STEP 1 |  | 20 points |
| Option [2] is selected, program moves to appropriate STEP 2 |  | 10 points |
| STEP 4 Option [2] (Grade Level) Invalid Data Error is handled appropriately |  | 20 points |
| STEP 4 Option [2] (Grade Level) Range Violation is handled appropriately |  | 10 points |
| STEP 5 Option [2] (GPA) Invalid Data Error is handled appropriately |  | 20 points |
| STEP 5 Option [2] (GPA) Range Violation is handled appropriately |  | 10 points |
| STEP 6 Option [2] Student created in STEPS 2-5 appears in auto generated student original formatted list with proper formatting (first letter of names UC; Grade level is whole number; GPA #.##) |  | 30 points |
| After STEP 6 Option [2] Program returns immediately to active STEP 1 |  | 20 points |
| STEP 1 Option [3] program exits with goodbye message. |  | 10 points |
|  |  |  |
|  |  |  |
|  |  | 20 points |
| **Subtotal** |  | **/280 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 **SeatingChartBuilderState.java** file |  | 10 points |
| SC1:**Students.java** class: All constructors, methods, and variables are appropriately created; no partial credit |  | 40 points |
| SC2: **SeatingChartBuilderState** class: create Scanner and ArrayList objects (place SC2 by bother instances); no partial credit |  | 10 points |
| SC3: **SeatingChartBuilderState** class ***printHighestGPA( ):*** code that compares GPA’s for all students |  | 20 points |
| SC4: **SeatingChartBuilderState** class ***printLowestGPA( ):*** code that compares GPA’s for all students |  | 20 points |
| SC5: **SeatingChartBuilderState** class ***inputManager( ):*** code with try/catch exception for invalid data error and range violation of STEP 1 (place SC5 by both instances) |  | 20 points |
| SC6: **SeatingChartBuilderState** class ***inputManager( ):*** code with try/catch exception for invalid data error and range violation of STEP 2 Option[1] (place SC6 by both instances) |  | 20 points |
| SC7: **SeatingChartBuilderState** class ***inputManager( ):*** code with try/catch exception for invalid data error and range violation of STEP 4 Option[2] (place SC7 by both instances) |  | 20 points |
| SC8: **SeatingChartBuilderState** class ***inputManager( ):*** code with try/catch exception for invalid data error and range violation of STEP 5 Option[2] (place SC8 by both instances) |  | 20 points |
| SC9: **SeatingChartBuilderState** class ***inputManager( ):*** code that formats GPA with proper decimal format object (place SC9 by all uses of this object) |  | 10 points |
| SC10: **SeatingChartBuilderState** class ***inputManager( ):*** code that formats first and last name to required convention (first letter is UC, all other letters are LC) |  | 20 points |
| SC11: **SeatingChartBuilderState** class ***inputManager( ):*** code that exits the program STEP 1 Option [3] |  | 10 points |
| **Subtotal** |  | **/240 Points** |
| **Total Points** |  | **/520 Points** |