

Member ID: _____

Time: _____

Rank: _____



JAVA Programming

(340)

REGIONAL 2025

PRODUCTION:

AIChatBotGenerator_Regional

_____ (600 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 chatbots with two spoken languages; the program will randomly assign each chatbot two languages.
- You will not actually be programming an AI chatbot.
- Each record will have a random primary language, secondary language, algorithm complexity, token rate, an estimated cost, and a compatibility. These variables will be explained in more detail later.
- The program will terminate once the list is generated.
- One Java file is provided in the contest folder (AIChatBotGenerator_Regional).
- There is one class inside the Java source code. You do not need to add any additional classes or files to the project folder
- All text for languages and output 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 of 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 placed 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* class is 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_AIChatBotGenerator_Regional

In this test, you will be creating a prototype software that will create parameter information for a chatbot by assigning it two spoken language and algorithmic quantifiers that will be used to create an estimated cost for deploying the bot based upon its token rate. Additionally, the program will check to see if the paired languages are compatible, meaning a bot that speaks Chinese and Korean would be considered compatible since these two languages have a geographical proximity to one another and there will be a high probability that the bot will speak either language frequently; conversely, a bot that speaks Chinese and German would not be compatible since the geographic locations of the highest concentration of people who speak these languages are not in proximity. The compatibility pairs have been predetermined but you will need to program the comparison.

Input

The user will enter in the number of chatbots to create in a range of 5 to 20 (inclusive). The information entered needs to be whole numbers. There can be no negative values, double values, values out of range, or letters-symbols.

Output

Once the program receives the user input it will automatically generate the output and print it to the console. The order of the variables of the data and the format of the output is shown in the image below. All output is randomized so your results will not be exactly the same, however they should be very similar. The program will not be allowed to have a bot with the same languages.

| No. | Primary Lang. | Secondary Lang. | Algorithm Complexity | TokenRate | Estimated Cost | Compatibility |
|-----|---------------|-----------------|----------------------|-----------|----------------|----------------|
| 1 | Arabic | Vietnamese | 9 | 0.23 | \$2.04 | Not Compatible |
| 2 | Turkish | Greek | 6 | 1.75 | \$10.51 | Compatible |
| 3 | Spanish | French | 7 | 1.18 | \$8.29 | Not Compatible |
| 4 | Korean | Hebrew | 16 | 5.80 | \$92.74 | Not Compatible |
| 5 | Hebrew | Chinese | 8 | 5.25 | \$42.03 | Not Compatible |

Numerical Data Values

- **TokenRate:** this value will be randomly assigned. It will be a double number in the range of 0.1 to 7.5. and will need to be formatted 0.00.
- **Algorithm Complexity:** will be a randomly generated integer value of 1 to 16 (inclusive).
- **Estimated Cost:** will be calculated with the formula **TokenRate * Algorithm Complexity**. This value will need to be formatted to USD.

Input Error

If the user enters in a non-positive whole number, double values, letters, or symbols, the program will inform the user to “Please enter in a correct value”. The focus of the program will return back to prompt to enter in the information. If the value entered is out of range, the message will be “Your entry is out of range.” Again the focus of the program will return to the prompt for creating the chatbots. The image on the right demonstrates all of the scenarios of input errors.

```
How many test bots do you want to create?
Please enter in a value between 5 and 20: tttt

Please enter a correct value.
Please enter in a value between 5 and 20: 5.55

Please enter a correct value.
Please enter in a value between 5 and 20: -222
Your entry is out of range.

Please enter in a value between 5 and 20: $$$

Please enter a correct value.
Please enter in a value between 5 and 20: 333
Your entry is out of range.

Please enter in a value between 5 and 20: 1
Your entry is out of range.

Please enter in a value between 5 and 20: |
```

Requirements

1. Your contestant number must appear as a comment at the top of the **AChatBotGenerator_Regional.java** source code file.
2. You will be programming all sections of the **AChatBotGenerator_Regional** class. This class will perform all of the functionality of the program and will also create the **Chatbots** objects. The **Chatbots** class has been created for you and it is located immediately below the **AChatBotGenerator_Regional** class.
3. Code elements and methods in the **AChatBotGenerator_Regional** class:
 - a. All of the **Chatbots** objects will need to be stored in a data structure.
 - b. You will program all of the **setChatbots** method. This method will create all of the chatbot objects by randomizing the primary and secondary language selection, complexity, and token rate.
 - c. You will program all the **printChatbots** method which oversees formatting and printing the randomly generated chatbots to the console. In this method you will need to use the **PrintStream** class **printf()** formatting method when creating the table. Listed below are the format values for the record's width, alignment of each field header (the title at the top of the column) and placeholder data type:
 - i. No. (this counts the number of bots): 4 for a string with left alignment
 - ii. Primary Lang: 15 for a string with left alignment
 - iii. Secondary Lang: 15 for a string with left alignment
 - iv. Algorithm Complexity: 20 for a string with left alignment
 - v. TokenRate: 10 for a string with left alignment
 - vi. Estimated Cost: 15 for a string with left alignment
 - vii. Compatibility: 14 for a string with left alignment

The printing of the records will be similar, you will need to use the **PrintStream** class *printf()* formatting method. The values are similar but slightly different since there are numerical values being entered. Listed below are the format values for the record's width, alignment, and placeholder data type (remember these are the values for the data that is printed from the records):

- i. No. (this counts the number of bots): 4 for an integer with left alignment
 - ii. Primary Lang: 15 for a string with left alignment
 - iii. Secondary Lang: 15 for a string with left alignment
 - iv. Algorithm Complexity: 20 for an integer with left alignment
 - v. TokenRate: 10 for a float with two decimal places and left alignment
 - vi. Estimated Cost: 14 for a string with left alignment
 - vii. Compatibility: 14 for a string with left alignment
- d. The *integerInputManager()* method will need to be completely programmed by you. This method has the responsibility of gathering the user's entry for how many chatbots to create and it also checks the user entry for wrong data type and for range violations.
- 4. Your program must use an **ArrayList** object to store the chatbots.
 - 5. Your program must use a **Scanner** object to get the user input.
 - 6. Your program must use a **Random** object to generate the random integers and double values.
 - 7. Your program must use a **DecimalFormat** object to format the TokenRate.
 - 8. Your program must use a **CurrencyFormat** object to format the Estimated Cost.
 - 9. Your output should look as close as possible to the examples provided.
 - 10. The Chatbots class is complete and does not need to be altered.

Methods List

- 1. **main(String args[]):** This is the main method of the class where the program execution begins. It prompts the user to specify the number of test bots to create, creates them, and then prints their details.
- 2. **setChatbots(ArrayList<Chatbots> list_of_Bots, String[] languages, int noB):** This method generates random chat bots with specified primary and secondary languages, algorithm complexity, and token rate.
- 3. **isCompatible(String primaryLang, String secondaryLang):** This method checks if the primary and secondary languages of a chat bot are compatible based on predefined compatible language pairs.
- 4. **printChatbots(ArrayList<Chatbots> bots, DecimalFormat df_Currency):** This method prints the details of the generated chat bots, including primary language, secondary language, algorithm complexity, token rate, estimated cost, and compatibility.
- 5. **integerInputManager():** This method manages user input to ensure that the specified number of test bots falls within the range of 5 to 20.

| | | |
|---|--|--------------------|
| 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 | | 10 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 chatbots to create | | 10 points |
| Format of the prompt matches the provided sample exactly with NO deviations | | 10 points |
| Program accepts appropriate data in the prompt | | 10 points |
| Program produces exact number of chatbots | | 20 points |
| Output of chatbot records is numbered in sequential order in the “No.” column | | 20 points |
| No chatbots have the same Primary and Secondary Languages | | 20 points |
| Algorithm Complexity is randomized and in the range of 1-16 | | 20 points |
| TokenRate is randomized and formatted #.00 and in the range of 0.01 to 7.50 | | 20 points |
| Program stops running after list is printed to the console | | 10 points |
| Estimated Cost is the correct calculation and formatted for USD | | 20 points |
| Compatibility value is correct for the language pairs | | 20 points |
| The format of the entire output table matches the sample exactly | | 50 points |
| After entering wrong data values and warning message, program keeps running and prompts user to enter number of chatbots to create | | 20 points |
| After entering data values out of range and warning message, program keeps running and prompts user to enter number of chatbots to create | | 20 points |
| Subtotal | | /280 Points |

| | | |
|--|--|-----------|
| Source Code Review No credit will be given for missing flagged comments. Code segments must work to receive full credit. | | |
| A comment containing the contestant number is present at the top of the AIChatBotGenerator_Regional.java file | | 10 points |
| SC1: Flag Comment Present | | 5 points |
| SC1: AIChatBotGenerator_Regional class: Place the comment by the code that creates the Scanner object | | 10 points |
| SC2: Flag Comment Present | | 5 points |
| SC2: AIChatBotGenerator_Regional class: Place the comment by the code that creates the data structure that stores the Chatbots objects | | 10 points |
| SC3: Flag Comment Present | | 5 points |
| SC3: AIChatBotGenerator_Regional class <i>integerInputManager()</i> : Place the comment by the code that gathers user input | | 10 points |
| SC4: Flag Comment Present | | 5 points |
| SC4: AIChatBotGenerator_Regional class <i>integerInputManager()</i> : method properly uses the <i>try/catch</i> code for data type input error | | 25 points |
| SC5: Flag Comment Present | | 5 points |
| SC5: AIChatBotGenerator_Regional class <i>integerInputManager()</i> : method uses strategy for detecting range violations | | 15 points |
| SC6: Flag Comment Present | | 5 points |
| SC6: AIChatBotGenerator_Regional class: Place the comment by the code that creates the Random object | | 10 points |
| SC7: Flag Comment Present | | 5 points |
| SC7: AIChatBotGenerator_Regional class: Place the comment by the code that creates the DecimalFormat object | | 10 points |
| SC8: Flag Comment Present | | 5 points |
| SC8: AIChatBotGenerator_Regional class <i>setChatbots()</i> : method randomly selects chatbots names using Random object and properly checks that objects are not equal | | 15 points |
| SC9: Flag Comment Present | | 5 points |
| SC9: AIChatBotGenerator_Regional class <i>setChatbots()</i> : method randomly selects chatbots grade levels using Random object | | 10 points |
| SC10: Flag Comment Present | | 5 points |
| SC10: AIChatBotGenerator_Regional class <i>setChatbots()</i> : method randomly creates GPA using Random object. | | 15 points |
| SC11: Flag Comment Present | | 5 points |
| SC11: AIChatBotGenerator_Regional class <i>printChatbots()</i> : method gets all of chatbot object information using get methods, and formats the printout | | 15 points |

| | | |
|---|--|--------------------|
| SC12: Flag Comment Present | | 5 points |
| SC12: AIChatBotGenerator_Regional class <i>setChatbots()</i> : method randomly creates GPA using DecimalFormat object | | 15 points |
| SC13: Flag Comment Present | | 5 points |
| SC13: AIChatBotGenerator_Regional class <i>printChatbots()</i> : formats record-headers with proper <i>printf()</i> values per the instructions. | | 25 points |
| SC14: Flag Comment Present | | 5 points |
| SC14: AIChatBotGenerator_Regional class <i>printChatbots()</i> : formats the records with proper <i>printf()</i> values per the instructions. | | 25 points |
| SC15: Flag Comment Present | | 5 points |
| SC15: AIChatBotGenerator_Regional class <i>isCompatible()</i> : compares primary and secondary languages using proper string comparison (equals() or compareTo()) | | 25 points |
| Subtotal | | /320 Points |
| Total Points | | /600 Points |