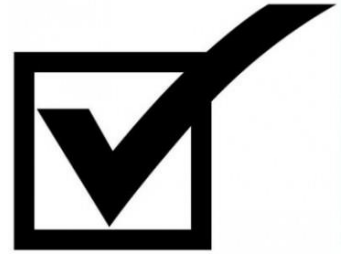




**BUSINESS
PROFESSIONALS**
of **AMERICA**
Giving Purpose to Potential



JAVA Programming

(340)

REGIONAL 2025

PRODUCTION:

AIChatBotGenerator_Regional

_____ (600 points)

Test Time: 90 minutes

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: AChatBotGenerator_Regional class <i>setChatbots()</i> : method randomly creates GPA using DecimalFormat object		15 points
SC13: Flag Comment Present		5 points
SC13: AChatBotGenerator_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: AChatBotGenerator_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: AChatBotGenerator_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

```

1 //create AI Chatbots
2 //test market for which chatbots are most popular and which token rate is the most profitable
3
4 import java.io.FileNotFoundException;
5 import java.io.File;
6 import java.util.*;
7 import java.text.DecimalFormat;
8
9 public class AIChatBotGenerator_Regional
10 {
11
12     static Scanner sc = new Scanner(System.in); //SC1
13     static DecimalFormat df_Currency = new DecimalFormat(pattern:"#,##0.00"); //SC7
14     Run | Debug
15     public static void main (String args [])
16     {
17         ArrayList<Chatbots> bots = new ArrayList<Chatbots>(); //SC2
18         System.out.println(x:"How many test bots do you want to create?");
19         int numberOfBots = integerInputManager();
20
21         //SC3
22         String [] languages ={"English", "French", "Chinese", "Spanish", "Arabic", "Russian", "German", "Japanese", "Portuguese", "Hindi", "Italian", "Korean", "Turkish",
23                               "Vietnamese", "Indonesian", "Greek", "Hebrew", "Thai"};
24
25         bots = setChatbots(bots, languages, numberOfBots);
26         printChatbots(bots, df_Currency);
27     }
28     private static String[][] compatibleLanguagePairs = {
29         {"English", "Spanish"}, {"English", "French"}, {"English", "German"}, {"Japanese", "English"}, {"Japanese", "Hindi"}, {"Portuguese", "French"},
30         {"Spanish", "Portuguese"}, {"Spanish", "Italian"}, {"French", "Italian"}, {"Russian", "Chinese"}, {"Russian", "Korean"}, {"Russian", "Japanese"},
31         {"Russian", "German"}, {"Hindi", "Arabic"}, {"Japanese", "Korean"}, {"Russian", "Arabic"}, {"Spanish", "Arabic"}, {"Russian", "French"},
32         {"Chinese", "Korean"}, {"Japanese", "Vietnamese"}, {"Arabic", "Hebrew"}, {"Greek", "Hebrew"}, {"Arabic", "Indonesian"}, {"Russian", "Indonesian"},
33         {"Chinese", "Thai"}, {"Japanese", "Vietnamese"}, {"English", "Chinese"}, {"Japanese", "Thai"}, {"Indonesian", "Thai"}, {"Chinese", "Spanish"},
34         {"Arabic", "Turkish"}, {"Greek", "Turkish"}, {"Vietnamese", "Thai"}, {"Vietnamese", "French"}, {"Vietnamese", "Indonesian"}, {"Turkish", "Spanish"},
35     };
36
37     //This will create the random bots
38     private static ArrayList<Chatbots> setChatbots(ArrayList<Chatbots> list_of_Bots, String[] languages, int noB) {
39         Random rand = new Random();
40         Chatbots chatBots;
41         ArrayList<Chatbots> bots = list_of_Bots;
42         DecimalFormat df = new DecimalFormat(pattern:"#.0000");
43
44         int numberOfBots = noB;
45         double tokenRate;
46         int complexity;
47         String primaryLang;
48         String secondaryLang;
49         for (int i = 0; i < numberOfBots; i++) {
50             do {
51                 primaryLang = languages[rand.nextInt(languages.length)];
52                 secondaryLang = languages[rand.nextInt(languages.length)];
53             } while (primaryLang.equals(secondaryLang));
54
55             complexity = rand.nextInt(16) + 1;
56             double tempTokenRate = rand.nextDouble() * 7.5 + 0.1;
57             String tempString = df.format(tempTokenRate);
58             tokenRate = Double.parseDouble(tempString);
59             chatBots = new Chatbots(primaryLang, secondaryLang, complexity, tokenRate);
60             bots.add(chatBots);
61         }
62         return bots;
63     }
64     private static boolean isCompatible(String primaryLang, String secondaryLang) { //SC15
65         for (String[] pair : compatibleLanguagePairs) {
66             if ((pair[0].equals(primaryLang) && pair[1].equals(secondaryLang)) ||
67                 (pair[1].equals(primaryLang) && pair[0].equals(secondaryLang))) {
68                 return true;
69             }
70         }
71         return false;
72     }
73 }

```

```

74 private static void printChatbots(ArrayList<Chatbots> bots, DecimalFormat df_Currency) {
75     System.out.println(x:"-----");
76     System.out.printf(format:"| %-4s | %-15s | %-15s | %-20s | %-10s | %-15s | %-14s | \n", //SC13
77         ...args:"No.", "Primary Lang.", "Secondary Lang.", "Algorithm Complexity", "TokenRate", "Estimated Cost", "Compatibility");
78     System.out.println(x:"-----");
79     int i = 1;
80     for (Chatbots b : bots) {
81         String compatibility = isCompatible(b.getPrimaryLanguage(), b.getSecondaryLanguage()) ? "Compatible" : "Not Compatible";
82         System.out.printf(format:"| %-4d | %-15s | %-15s | %-20d | %-10.2f | $%-14s | %-14s | \n", //SC14
83             i, b.getPrimaryLanguage(), b.getSecondaryLanguage(), b.getComplexity(), b.getTokenRate(), df_Currency.format(b.getEstimatedCost()), compatibility);
84         i++;
85     }
86     System.out.println(x:"-----");
87 }
88
89 private static int integerInputManager()
90 {
91     int temp;
92     while(true){
93         try{ //SC4
94             do{
95                 System.out.print(s:"Please enter in a value between 5 and 20: ");
96                 temp = sc.nextInt();
97                 if (temp > 20 || temp < 5) //SC5
98                     System.out.println(x:"Your entry is out of range.\n");
99                 sc.nextLine();
100             }while(temp > 20 || temp < 5); //SC5
101             return temp;
102         }
103         catch(InputMismatchException e) //SC4
104         {
105             sc.next();
106             System.out.println(x:"\nPlease enter a correct value.");
107         }
108     }
109 }
110
111 }
112
113
114 }
115 //Given
116 //////////////////////////////////////////////////
117 class Chatbots
118 {
119     String primaryLang;
120     String secondaryLang;
121     int complexity_Level;
122     double tokenRate;
123     double estimatedCost = 0;
124
125     public Chatbots()
126     {
127         primaryLang = "ASCII";
128         secondaryLang = "ASCII";
129         complexity_Level = 0;
130         tokenRate = 0.0;
131     }
132
133     public Chatbots(String primaryLang, String secondaryLang, int complexity_Level, double tokenRate)
134     {
135         this.primaryLang = primaryLang;
136         this.secondaryLang = secondaryLang;
137         this.complexity_Level = complexity_Level;
138         this.tokenRate = tokenRate;
139         setEstimatedCost();
140     }
141
142

```

```
143     public String getPrimaryLanguage()
144     {
145         | return primaryLang;
146     }
147
148     public String getSecondaryLanguage()
149     {
150         | return secondaryLang;
151     }
152
153     public double getTokenRate()
154     {
155         | return tokenRate;
156     }
157
158     public int getComplexity()
159     {
160         | return complexity_Level;
161     }
162
163     public void setEstimatedCost()
164     {
165         | estimatedCost = getComplexity() * getTokenRate();
166     }
167
168     public double getEstimatedCost()
169     {
170         | return estimatedCost;
171     }
172
173 }
```