

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-number-guesser-4/grade/zb64>

IT114-004-S2024 - [IT114] Number Guesser 4

Submissions:

Submission Selection

1 Submission [active] 2/12/2024 4:34:35 PM

Instructions

^ COLLAPSE ^

-
-
-
1. Create the below branch name
Implement the NumberGuess4 example from the lesson/slides
<https://gist.github.com/MattToegel/aced06400c812f13ad030db9518b399f>
Add/commit the files as-is from the lesson material (this is the base template). You may want to push this commit so you can open the pull request and keep it open.
Pick two (2) of the following options to implement
 - Display higher or lower as a hint after a wrong guess (only after a wrong guess that doesn't roll back the level)
 - Implement anti-data tampering of the save file data (reject user direct edits)
 - Add a difficulty selector that adjusts the max strikes per level (i.e., "easy" 10 strikes, "medium" 5 strikes, "hard" 3 strikes)
 - Display a cold, warm, hot indicator based on how close to the correct value the guess is (example, 10 numbers away is cold, 5 numbers away is warm, 2 numbers away is hot; adjust these per your preference) Only display this when the wrong guess doesn't roll back the level
 - Add a hint command that can be used once per level and only after 2 strikes have been used that reduces the range around the correct number (i.e., number is 5 and range is initially 1-15, new range could be 3-8 as a hint)
 - Implement separate save files based on a "What's your name?" prompt at the start of the game (each person gets their own save file based on user's name)
-
-
-
-
-
-
-
-
10. Fill in the below deliverables
Save changes and export PDF
Git add/commit/push your changes to the HW branch
Create a pull request to main
Complete the pull request (don't forget to locally checkout main and pull changes to prep for future work)
Upload the same PDF to Canvas

Branch name: M3-NumberGuesser-4

Tasks: 7 Points: 10.00

^ COLLAPSE ^

Task #1 - Points: 1

Text: Chosen Option and Details

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Mention which option you picked
<input checked="" type="checkbox"/> #2	1	Explain the logic of how you solved/implemented the chosen option (concrete details). Explain how the code works, don't just paste code snippets

Response:

I completed option 4. To solve option 4, I considered how to determine if the number entered by the user would be close to the generated number. To do this, I calculated the difference between the statement to check this difference and determine if it was very close, moderately close, or far from the random number. Once it's checked, it will be followed by a print statement indicating if the user is close, warm, or cold with their guess.

Task #2 - Points: 1

Text: 2+ Screenshots of code and demo

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Show implementation working by running the program
<input type="checkbox"/> #2	1	Clearly caption the screenshot of what you're showing
<input type="checkbox"/> #3	1	The code screenshot(s) clearly show the code specific to the feature
<input type="checkbox"/> #4	1	A comment with the UCID/date is visible near the code change(s)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
NumberGuesser4.java > NumberGuesser4 > processGuess(int)
120 level--;
121 if (level < 1) {
122     level = 1;
123 }
124 } //124-132 private void lose()
133
134 private void processGuess(int guess) {
135     if (guess <= 0) {
136         return;
137     }
138     System.out.println("You guessed " + guess);
139     if (guess == number) {
140         win();
141         pickNewRandom = true;
142     } else {
143         System.out.println("That's wrong");
144         strikes++;
145         if (strikes >= maxStrikes) {
146             lose();
147             pickNewRandom = true;
148         } else {
149             int diff = Math.abs(number - guess);
150             if (diff <= 2) {
151                 System.out.println("You are hot. You are really close");
152             } else if (diff <= 5) {
153                 System.out.println("You are warm! Keep going.");
154             } else if (diff <= 10) {
155                 System.out.println("You are cold.");
156             }
157         }
158     } //206-212/21
159 } //148-158 else
160 } //142-159 else
161 saveState();
162 } //134-161 private void processGuess(int guess)
163
164 private int strToNum(String message) {
165     int guess = -1;
166     try {
167         guess = Integer.parseInt(message.trim());
168     } catch (NumberFormatException e) {
169         System.out.println("You didn't enter a number, please try again");
170     } catch (Exception e2) {
171         System.out.println("Null message received");
172     }
173 }
```

Missing Caption

Checklist Items (0)



Implementation 2 (4 pts.)

^ COLLAPSE ^



Task #1 - Points: 1

Text: Chosen Option and Details

^ COLLAPSE ^

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Mention which option you picked
<input type="checkbox"/> #2	1	Explain the logic of how you solved/implemented the chosen option (concrete details). Explain how the code works, don't just paste code snippets

Response:

I completed option 5. For option 5, I initially created a message similar to the quit statement that was in the constructor and then introduced a new method to outline steps on how to determine the beginning and ending range so the user can utilize calculates the lower and set 10 as the calculating the upper limit of the range. I also set

in the constructor to indicate that the hint was being used once the range was method, I followed the same format as the quit statement and specified that the hint should be displayed when it hasn't been used and there are 2 or more strikes.

^ COLLAPSE ^

Task #2 - Points: 1
Text: 2+ Screenshots of code and demo

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Show implementation working by running the program
<input type="checkbox"/> #2	1	Clearly caption the screenshot of what you're showing
<input type="checkbox"/> #3	1	The code screenshot(s) clearly show the code specific to the feature
<input type="checkbox"/> #4	1	A comment with the UCID/date is visible near the code change(s)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.io.FileWriter;
4 import java.io.IOException;
5 import java.util.Random;
6 import java.util.Scanner;
7
8 public class NumberGuesser4 {
9     private int maxLevel = 1;
10    private int level = 1;
11    private int strikes = 0;
12    private int maxStrikes = 5;
13    private int number = -1;
14    private boolean pickNewRandom = true;
15    private Random random = new Random();
16    private String fileName = "ng4.txt";
17    private String[] fileHeaders = { "Level", "Strikes", "Number", "MaxLevel" }; // used for demo readability
18    //zb64 2/12/24
19    private boolean hintUsed = false;
20    private int range = 0;
21
22    private void saveState() {
23        String[] data = { level + "", strikes + "", number + "", maxLevel + "" };
24        String output = String.join(delimiters, data);
25        // Note: we don't need a file reference as FileWriter creates the file if it
26        // doesn't exist
27        try (FileWriter fw = new FileWriter(fileName)) {
28            fw.write(String.join(delimiters, fileHeaders));
29            fw.write(String.join(delimiters, data));
30            fw.write(output);
31        } catch (IOException e) {
32            // TODO Auto-generated catch block
33            e.printStackTrace();
34        }
35    } // <- #22-35 private void saveState()
36
37    private void loadState() {
38        File file = new File(fileName);
39        if (!file.exists()) {
40            // Not providing output here as it's expected for a fresh start
41            return;
42        }
43        try (Scanner reader = new Scanner(file)) {
44            int lineNumber = 0;
```

Checklist Items (0)

```

44  int lineNumber = 0;
45  while (reader.hasNextLine()) {
46      String text = reader.nextLine();
47      // System.out.println("Text: " + text);
48      if (lineNumber == 1) {
49          String[] data = text.split(regex);
50          String level = data[0];
51          String strikes = data[1];
52          String number = data[2];
53          String maxLevel = data[3];
54          int temp = strToNum(level);
55          if (temp > -1) {
56              this.level = temp;
57          }
58          temp = strToNum(strikes);
59          if (temp > -1) {
60              this.strikes = temp;
61          }
62          temp = strToNum(number);
63          if (temp > -1) {
64              this.number = temp;
65              pickNewRandom = false;
66          }
67          temp = strToNum(maxLevel);
68          if (temp > -1) {
69              this.maxLevel = temp;
70          }
71      } <- #48-71 if (lineNumber == 1)
72      lineNumber++;
73  } <- #45-73 while (reader.hasNextLine())
74  } catch (FileNotFoundException e) { // specific exception
75      e.printStackTrace();
76  } catch (Exception e2) { // any other unhandled exception
77      e2.printStackTrace();
78  }
79  System.out.println("Loaded state");
80  int range = 10 + ((level - 1) * 5); //zb64 2/12/24
81  System.out.println("Welcome to level " + level);
82  System.out.println(
83      "I picked a random number between 1-" + (range) + ", let's see if you can guess.");
84  } <- #37-84 private void loadState()
85
86  /***
87   * Gets a random number between 1 and level.
88   * @param level (level to use as upper bounds)
89   * @return number between bounds
90   */
91
92  private void generateNewNumber(int level) {
93      int range = 10 + ((level - 1) * 5);
94      System.out.println("Welcome to level " + level);
95      System.out.println(
96          "I picked a random number between 1-" + (range) + ", let's see if you can guess.");
97      number = random.nextInt(range) + 1;
98  } <- #92-98 private void generateNewNumber(int level)
99
100  private void win() {
101      System.out.println("That's right!");
102      level++; // level up!
103      strikes = 0;
104  } <- #100-104 private void win()
105
106  private boolean processCommands(String message) {
107      boolean processed = false;
108      if (message.equalsIgnoreCase("quit")) {
109          System.out.println("Tired of playing? No problem, see you next time.");
110          processed = true;
111      }
112      // TODO add other conditions here
113      return processed;
114  } <- #106-114 private boolean processCommands(String message)
115
116  private void genHints(int range) {
117      int startRange = Math.max(1, number - range/2); //this calculates the minimum range
118      int endRange = Math.min(10, number + range/2); //this calculates the maximum range
119      System.out.println("Hint is that the number is between " + startRange + "-" + endRange);
120      hintUsed = true;
121      //zb64 2/12/24
122  } <- #116-122 private void genHints(int range)
123
124  private void lose() {
125      System.out.println("Uh oh, looks like you need to get some more practice.");
126      System.out.println("The correct number was " + number);
127      strikes = 0;
128      level--;
129      // TODO add other conditions here
130  } <- #124-130 private void lose()

```

Checklist Items (0)

```

86  /***
87   * Gets a random number between 1 and level.
88   * @param level (level to use as upper bounds)
89   * @return number between bounds
90   */
91
92  private void generateNewNumber(int level) {
93      int range = 10 + ((level - 1) * 5);
94      System.out.println("Welcome to level " + level);
95      System.out.println(
96          "I picked a random number between 1-" + (range) + ", let's see if you can guess.");
97      number = random.nextInt(range) + 1;
98  } <- #92-98 private void generateNewNumber(int level)
99
100  private void win() {
101      System.out.println("That's right!");
102      level++; // level up!
103      strikes = 0;
104  } <- #100-104 private void win()
105
106  private boolean processCommands(String message) {
107      boolean processed = false;
108      if (message.equalsIgnoreCase("quit")) {
109          System.out.println("Tired of playing? No problem, see you next time.");
110          processed = true;
111      }
112      // TODO add other conditions here
113      return processed;
114  } <- #106-114 private boolean processCommands(String message)
115
116  private void genHints(int range) {
117      int startRange = Math.max(1, number - range/2); //this calculates the minimum range
118      int endRange = Math.min(10, number + range/2); //this calculates the maximum range
119      System.out.println("Hint is that the number is between " + startRange + "-" + endRange);
120      hintUsed = true;
121      //zb64 2/12/24
122  } <- #116-122 private void genHints(int range)
123
124  private void lose() {
125      System.out.println("Uh oh, looks like you need to get some more practice.");
126      System.out.println("The correct number was " + number);
127      strikes = 0;
128      level--;
129      // TODO add other conditions here
130  } <- #124-130 private void lose()

```


Missing Caption

Checklist Items (0)

```
185 NumberGuesser4.java > NumberGuesser4 > processGuess(int)
186     pickNewRandom = false;
187     } <- #182-186 if (pickNewRandom)
188     System.out.println("Type a number and press enter");
189     // we'll want to use a local variable here
190     // so we can feed it into multiple functions
191     String message = input.nextLine();
192     // early termination check
193     if (processCommands(message)) {
194         // command handled; don't proceed with game logic
195         break;
196     }
197     // this is just to demonstrate we can return a value and pass it into another
198     // method
199     int guess = strToNum(message);
200     processGuess(guess);
201     // the following line is the same as the above two lines
202     // processGuess(getGuess(message));
203     if (message.equalsIgnoreCase("hint") && !hintUsed && strikes >= 2) {
204         getHints(range);
205         continue;
206     }
207     //206-212 2/12/24
208     } while (true); <- #181-207 do
209     } catch (Exception e) {
210         System.out.println("An unexpected error occurred. Goodbye.");
211         e.printStackTrace();
212         System.out.println(e.getMessage());
213     } <- #208-212 catch (Exception e)
214     System.out.println("Thanks for playing!");
215     } <- #175-214 public void start()
216
217 Run/Debug
218 public static void main(String[] args) {
219     NumberGuesser4 ng = new NumberGuesser4();
220     ng.start();
221 } <- #8-220 public class NumberGuesser4
222
```

Missing Caption

Checklist Items (0)

Misc (2 pts.)

^ COLLAPSE ^

Task #1 - Points: 1

Text: Reflection

^ COLLAPSE ^

Checklist		*The checkboxes are for your own tracking
#	Points	Details
<input type="checkbox"/> #1	1	Example prompts: Learn anything new? Face any challenges? How did you overcome and issues?
<input type="checkbox"/> #2	1	At least a few logical sentences related to the assignment.

Response:

One of the things that I learned was how to create a quit message for users if they are tired of repeatedly playing the game. Additionally, I learned about filewriters and their functionality. One challenge that I faced was determining the range around the guessed number. I overcame this

challenge by searching online and asking for assistance on how to find the range of a number generator. This helped me understand what each part of the code does and how to implement it effectively.

COLLAPSE

Task #2 - Points: 1

Text: Pull Request URL

Details:
URL should end with /pull/# where the # is the actual pull request number.

URL #1
Missing URL

COLLAPSE

Task #3 - Points: 1

Text: Waka Time (or related) Screenshot

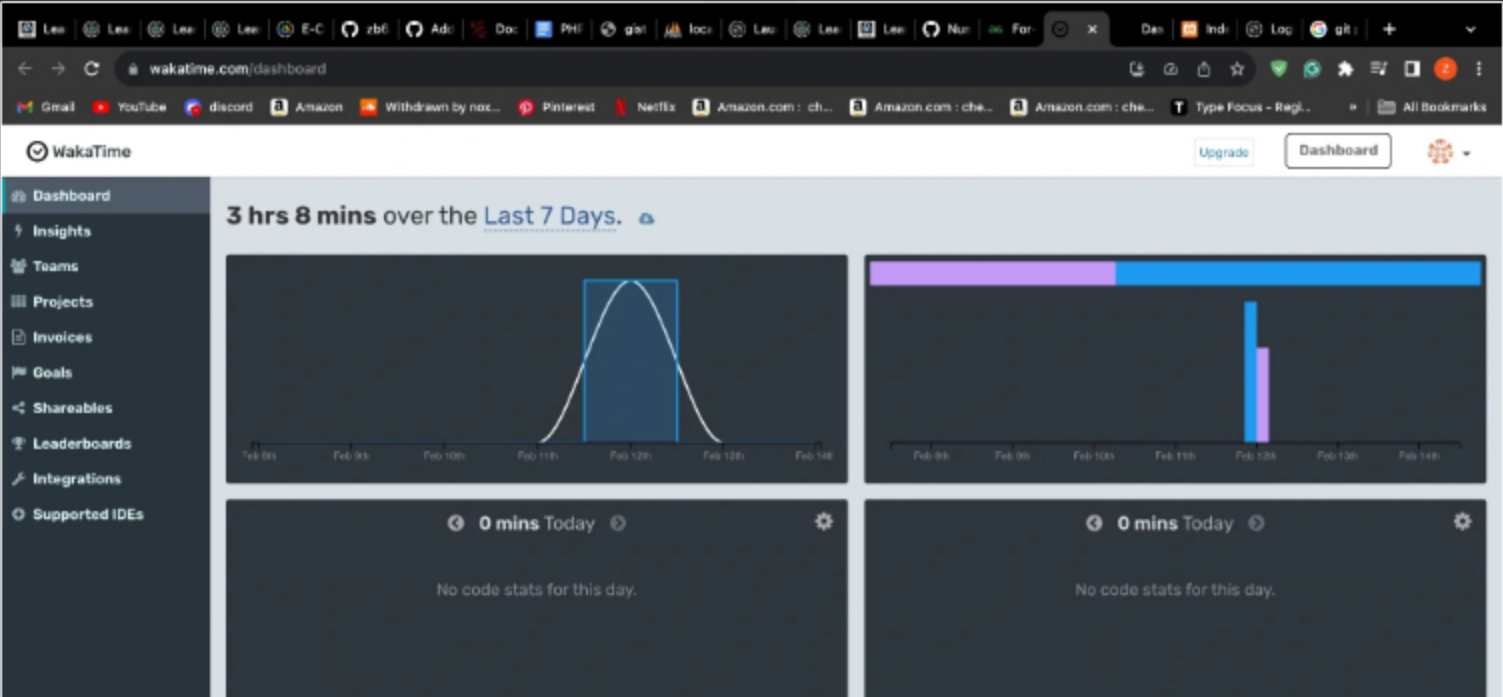
Checklist *The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

Task Screenshots:

Gallery Style: Large View

Small Medium Large





This is showing how long I've been working on this assignment.

Checklist Items (1)

#1 Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

End of Assignment