

# Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-module-3-number-guesser-4/grade/mcp62>

Course: IT114-003-F2024

Assignment: [IT114] Module 3 Number Guesser 4

Student: Michael P. (mcp62)

## Submissions:

Submission Selection

1 Submission [submitted] 10/2/2024 11:06:25 AM

## Instructions

^ COLLAPSE ^

Overview Video: <https://youtu.be/ej6lWrg9XjE>

1. Create the below branch name
2. Implement the NumberGuess4 example from the lesson/slides
  1. <https://gist.github.com/MattToegel/aced06400c812f13ad030db9518b399f>
  2. Add/commit the files as-is from the lesson material (this is the base template).
  3. Push the changes to the HW branch and create a pull request to keep open until this assignment is done
3. Pick two (2) of the following options to implement
  1. Display higher or lower as a hint after a wrong guess (only after a wrong guess that doesn't roll back the level)
  2. Implement anti-data tampering of the save file data (reject user direct edits)
  3. Add a difficulty selector that adjusts the max strikes per level (i.e., "easy" 10 strikes, "medium" 5 strikes, "hard" 3 strikes)
  4. 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
  5. 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)
  6. 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)
4. Fill in the below deliverables
5. Save changes and export PDF

6. Git add/commit/push your changes to the HW branch
7. Create a pull request to main (if not done so before)
8. Complete the pull request (don't forget to locally checkout main and pull changes to prep for future work)
9. Upload the same PDF to Canvas

Branch name: M3-NumberGuesser-4

#### Group



Group: Implementation 1

Tasks: 1

Points: 4

^ COLLAPSE ^

#### Task



Group: Implementation 1

Task #1: Implementation Evidence


Weight: ~100%

Points: ~4.00

^ COLLAPSE ^

#### Details:

Code screenshots must have ucid/date shown as a comment in the code.

Explanations must be your own words describing the logic and how the solution code solves the problem. 

Columns: 1

#### Sub-Task



Group: Implementation 1

Task #1: Implementation Evidence

Sub Task #1: Mention which option you picked and how you solved it

## ≡ Task Response Prompt

*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 implemented the first prompt by using a higher or lower number indicator, i did this by modifying the processGuess method, since its already stated in there that if Strikes exceeds the maximum number of strikes then you lose() however with this we can simply write an else statment to run each time that the player is still currently playing. Inside this else block I wrtie the two conditionals for the hint if guess is less than the current umber write guess higher, else if guess is greater than the current number guess lower, additonally with the way the code is positioned within that else block means I avoid sending the message if the user guesses the correct number since its only goin to run assuming the player is going to exceed max strikes.

### Sub-Task

Group: Implementation 1

Task #1: Implementation Evidence

Sub Task #2: Add screenshots of the coded solution (ucid/date must be visible)

100%

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```

// guessNumber.java
import java.util.*;

public class guessNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int target = 7;
        int guess;
        int strikes = 0;
        boolean pickedNewRandom = false;

        while (true) {
            guess = scanner.nextInt();

            if (guess == target) {
                System.out.println("That's right!");
                strikes = 0;
                pickedNewRandom = true;
            } else if (guess > target) {
                System.out.println("Guess higher! Your number is too low");
                strikes++;
            } else if (guess < target) {
                System.out.println("Guess lower! Your number is too high");
                strikes++;
            }

            if (strikes >= 10) {
                System.out.println("You've lost!");
                break;
            }

            if (pickedNewRandom) {
                target = (int)(Math.random() * 100) + 1;
                pickedNewRandom = false;
            }
        }
    }
}
// ucip62 9/20/2024
```

implementation 1

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

### Sub-Task

Group: Implementation 1

Task #1: Implementation Evidence

Sub Task #3: Show implementation working by running the program

100%

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```

// guessNumber.java
import java.util.*;

public class guessNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int target = 7;
        int guess;
        int strikes = 0;
        boolean pickedNewRandom = false;

        while (true) {
            guess = scanner.nextInt();

            if (guess == target) {
                System.out.println("That's right!");
                strikes = 0;
                pickedNewRandom = true;
            } else if (guess > target) {
                System.out.println("Guess higher! Your number is too low");
                strikes++;
            } else if (guess < target) {
                System.out.println("Guess lower! Your number is too high");
                strikes++;
            }

            if (strikes >= 10) {
                System.out.println("You've lost!");
                break;
            }

            if (pickedNewRandom) {
                target = (int)(Math.random() * 100) + 1;
                pickedNewRandom = false;
            }
        }
    }
}
// ucip62 9/20/2024
```

shows higher or lower messages

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 1

End of Group: Implementation 1

Task Status: 1/1

### Group

100%

Group: Implementation 2

Tasks: 1

Points: 4

^ COLLAPSE ^

### Task

100%

Group: Implementation 2

Task #1: Implementation Evidence

Weight: ~100%

Points: ~4.00

^ COLLAPSE ^

### Details:

Code screenshots must have ucid/date shown as a comment in the code.

Explanations must be your own words describing the logic and how the solution code solves the problem.



Columns: 1

### Sub-Task

100%

Group: Implementation 2

Task #1: Implementation Evidence

Sub Task #1: Mention which option you picked and how you solved it

## Task Response Prompt

*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:

For implementation 2 i selected the third option and made it so that you can select your difficulty, I did this within the start() method in order to prompt the user easy medium or hard depening on what they type in, depending on what the player types is what determines the max number of strikes, selecting hard will give the user a max of 3 tries, while selecting easy will give the player 10.

### Sub-Task

100%

Group: Implementation 2

Task #1: Implementation Evidence

Sub Task #2: Add screenshots of the coded solution (ucid/date must be visible)

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
public void start() {  
    try {Scanner input = new Scanner(System.in); {  
        ...  
    }  
}
```

**Weight: ~33%**

Points ~0.67

^ COLLAPSE ^

#### Sub-Task

Group: Misc

100%

Task #1: Reflection

Sub Task #1: Learn anything new? Face any challenges? How did you overcome any issues?

## Task Response Prompt

*Provide at least a few logical sentences*

Response:

I learn how to better read larger ammounts of code, originally i kept a massive amount of incredibly poor written code, but then went to office hours to see how I can simplify it with 1/10 the amount of redundant code. I had to over come alot of annoying technical problems like messages being sent when i dont want them to and having the strike amount to work properly when selecting difficult (also needing these two feture to work together since if you select hard then the amount of strikes change dynamically)

End of Task 1

#### Task

100%

Group: Misc

Task #2: Pull Request URL

Weight: ~33%

Points: ~0.67

^ COLLAPSE ^

#### i Details:

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



## Task URLs

URL #1

<https://github.com/Onervv/mcp62-IT114-003/pull/8>

URL

<https://github.com/Onervv/mcp62-IT114-003/pul>

End of Task 2

#### Task

100%

Group: Misc

Task #3: Waka Time (or related) Screenshot


Weight: ~33%

Points: ~0.67

^ COLLAPSE ^

**Checklist** \*The checkboxes are for your own tracking

**Checklist** \*The checkboxes are for your own tracking

#	Details
 #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: 2 Columns

waka time

End of Group: Misc  
Book Status: 0/0

Task Status: 3/3

End of Assignment