

Submission Worksheet

CLICK TO GRADE

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

Course: IT114-003-F2024

Assignment: [IT114] Module 3 Number Guesser 4

Student: Jimmy S. (jns)

Submissions:

Submission Selection

1 Submission [submitted] 9/28/2024 5:55:34 PM

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 ^

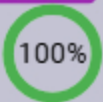
i Details:

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 decided to implement options 1 2 3 and 5 1: simple, in the process guess function, I added an additional two conditions that checked if the guess was higher or lower then the number, and then had it diplay to the console 2: to create some layer of anti data tampering, I first remove the headers in the number guesser text file, and then added and additional function, meanShuffle, which would convert the saved information into base64 before saving to the text file, and then decoding it before using to load the save. The result is a file that is just some base64 and some comments, which hopefully will deter people from messing with the file 3: For the difficulty selector, I chose to add it on the condition if the user loses a level, in which I implemented the changeDifficulty function, which would ask the

user to enter e, m, or h to select between difficulties hard, medium, and easy, which would change the number of maxStrikes from 10 all the way down to 3. 5: For the hint system, I created a new function, hint(), which would give a random range of where the number could be, with a max of two hints being given per level. Then, I created the condition that after two strikes, the user can ask the game for a hint via typing help.

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

```
//TASK 7
//To avoid data tampering, will remove file headers and turn numbers into Base64
private String[] meanShuffle(String[] data, boolean state){
    if(state){
        for(int i =0;i<data.length;i++){
            data[i] = new String(Base64.getDecoder().decode(data[i]));
        }
    }else{
        for(int i =0;i<data.length;i++){
            data[i] = Base64.getEncoder().encodeToString(data[i].getBytes());
        }
    }
    return data;
}
}
}
//TASK 8
```

```
private void onStrike() {
    //TASK 3
    changeDifficulty();
    //TASK 4
    // -> everytime private void level()
    //task 4: asks user to change difficulty after loading, sub's strikes...
    private void changeDifficulty() {
        Scanner s = new Scanner(System.in);
        System.out.println("Want to change difficulty? (type y/n)");
        String n = s.nextLine();
        if(n.equals("yes") || n.equals("y")) {
            System.out.println("You can choose, all strikes, no strikes, no strikes");
            new.nextLine();
            if(n.equalsIgnoreCase("anotherString":"?")) {
                newStrikes = 10;
            }
            if(n.equalsIgnoreCase("anotherString":"?")) {
                newStrikes = 5;
            }
            if(n.equalsIgnoreCase("anotherString":"?")) {
                newStrikes = 0;
            }
        } else {
            System.out.println("OK, continue playing");
        }
    }
    // -> everytime private void changeDifficulty()
    //TASK 5
```

ucid:jns date:9/28/2024 task 2

task 3

```
//TASK 5: Hint system; When user asks for help after two strikes, they can get up to a max of two hints  
private void hints(){  
    //random range so user can not identify the number by looking at the middle of the range  
    int range = 10 * ((level - 1) + 5);  
    int mod = (int)(Math.random()*range);  
    int mod2 = (int)(Math.random()*6);  
    System.out.println(mod);  
  
    if(hints == 0){  
        int i = number - mod < 0 ? 0 : number - mod;  
        int h = number + mod > range ? range : number + mod;  
        System.out.printf("Answer is in between " + (i) + " and " + (h));  
        hints++;  
    }else if(hints == 1){  
        int i = number - mod2 < 0 ? 0 : number - mod2;  
        int h = number + mod2 > range ? range : number + mod2;  
        System.out.printf("Answer is in between " + (i) + " and " + (h));  
        System.out.println("Last Hint for this level!");  
        hints++;  
    }else{  
        System.out.println("No hints for you!!");  
    }  
}  
}endGame();  
}< /END-200 private void hints()
```

```
private void processGuess(int guess) {
    } else {
        System.out.println("Computer's wrong");
        strikes++;
        if (strikes == maxStrikes) {
            lose();
            checkEndGame = true;
        }
        // TASK 3: IMPLEMENT HIGHER LOWER SYSTEM
        if (guess < number && strikes != maxStrikes && strikes != 0) {
            System.out.println("Higher...");
        }
        if (guess > number && strikes != maxStrikes && strikes != 0) {
            System.out.println("Lower...");
        }
        // TASK 3
    }
}

// CH 2210-2211 class
```

task 5

task 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

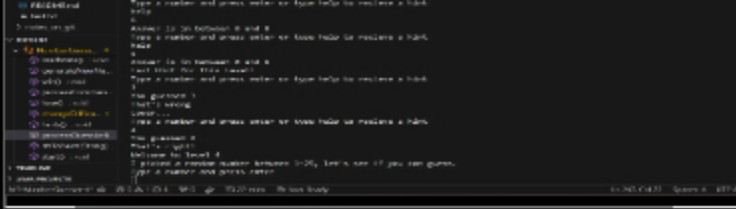
Gallery Style: 2 Columns

4

2

1

[illegible]



number guesser in action

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:

I decided to implement options 1 2 3 and 5 1: simple, in the process guess function, I added an additional two conditions that checked if the guess was higher or lower then the number, and then had it diplay to the console 2: to create some layer of anti data tampering, I first remove the headers in the number guesser text file, and then added

and additional function, `meanShuffle`, which would convert the saved information into base64 before saving to the text file, and then decoding it before using to load the save. The result is a file that is just some base64 and some comments, which hopefully will deter people from messing with the file 3: For the difficulty selector, I chose to add it on the condition if the user loses a level, in which I implemented the `changeDifficulty` function, which would ask the user to enter e, m, or h to select between difficulties hard, medium, and easy, which would change the number of `maxStrikes` from 10 all the way down to 3. 5: For the hint system, I created a new function, `hint()`, which would give a random range of where the number could be, with a max of two hints being given per level. Then, I created the condition that after two strikes, the user can ask the game for a hint via typing help.

Sub-Task

Group: Implementation 2

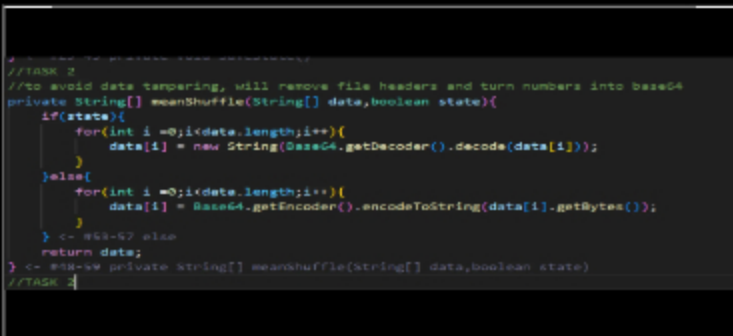
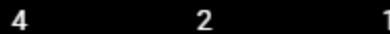
Task #1: Implementation Evidence

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



Task Screenshots

Gallery Style: 2 Columns



just look at images above, same stuff

Caption(s) (required) ✓

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

Sub-Task

Group: Implementation 2

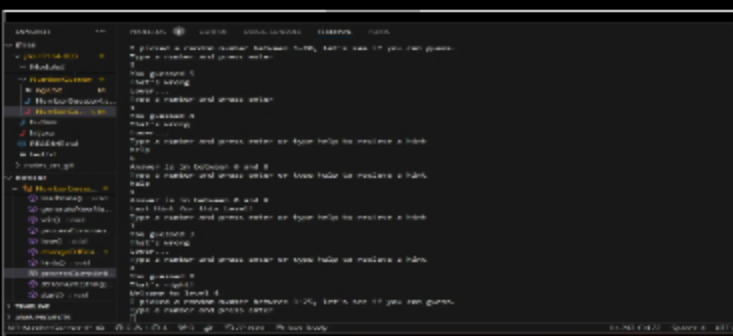
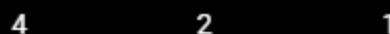
Task #1: Implementation Evidence

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



Task Screenshots

Gallery Style: 2 Columns



again, look above

Caption(s) (required) ✓

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

End of Task 1

End of Group: Implementation 2

Task Status: 1/1

Group



Group: Misc
Tasks: 3
Points: 2

^ COLLAPSE ^

Task



Group: Misc
Task #1: Reflection
Weight: ~33%
Points: ~0.67

^ COLLAPSE ^

Sub-Task



Group: Misc
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:

did a LOT of debugging, actually sat down to learn how to use the java debugger. code definitely is not perfect but I learned how to create alternate solutions of stuff that i may have imagined differently. Taking breaks with this assignment helped a lot.

End of Task 1

Task



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/jnsnjit/jns-IT114-003/pull/9>

URL

<https://github.com/jnsnjit/jns-IT114-003/pull/9>

End of Task 2

Task



Group: Misc

Task #3: Waka Time (or related) Screenshot

Weight: ~33%

Points: ~0.67

COLLAPSE

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

4 2 1



waka waka aye aye

End of Task 3

End of Group: Misc

Task Status: 3/3

End of Assignment