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
2. Implement the NumberGuess4 example from the lesson/slides
    1. https://gist.github.com/MattToegel/aced06400c812f13ad030db9518b399f
3. 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.
4. 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)
5. Fill in the below deliverables
6. Save changes and export PDF
7. Git add/commit/push your changes to the HW branch
8. Create a pull request to main
9. Complete the pull request (don't forget to locally checkout main and pull changes to prep for future work)
10. Upload the same PDF to Canvas

Branch name: M3-NumberGuesser-4

Tasks: 7 Points: 10.00

COLLAPSE

Implementation 1 (4 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Chosen Option and Details

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention which option you picked
#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.

COLLAPSE

Task #2 - Points: 1

Text: 2+ Screenshots of code and demo

Checklist *The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Missing Caption

Checklist Items (0)

Implementation 2 (4 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Chosen Option and Details

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Mention which option you picked
☐ #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
☐ #1	1	Show implementation working by running the program
☐ #2	1	Clearly caption the screenshot of what you're showing
☐ #3	1	The code screenshot(s) clearly show the code specific to the feature
☐ #4	1	A comment with the UCID/date is visible near the code change(s)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Missing Caption

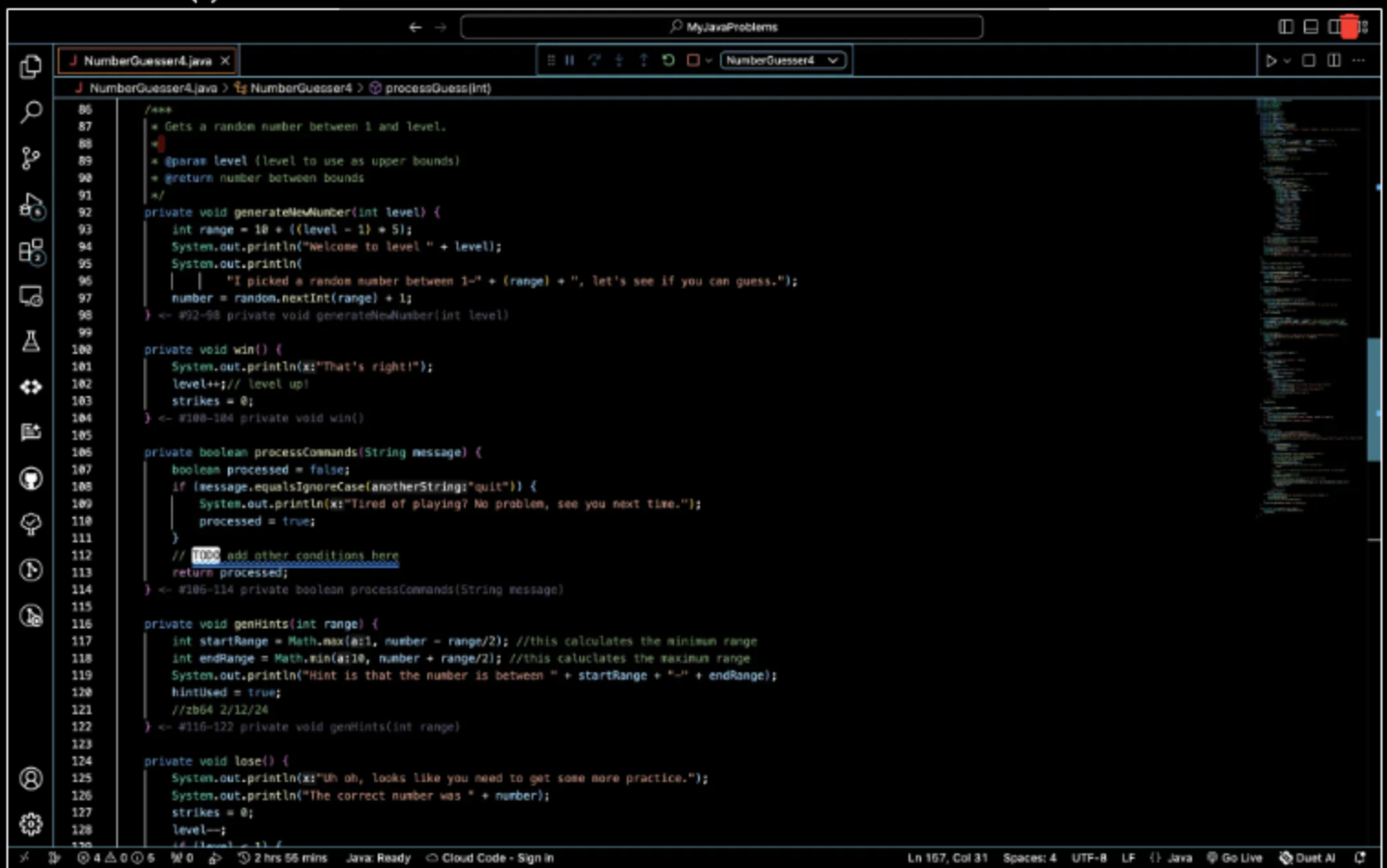
Checklist Items (0)

```
            int lineNumber = 0;
            while (reader.hasNextLine()) {
                String text = reader.nextLine();
                // System.out.println("Text: " + text);
                if (lineNumber == 1) {
                    String[] data = text.split(regex:",");
                    String level = data[0];
                    String strikes = data[1];
                    String number = data[2];
                    String maxLevel = data[3];
                    int temp = strToNum(level);
                    if (temp > -1) {
                        this.level = temp;
                    }
                    temp = strToNum(strikes);
                    if (temp > -1) {
                        this.strikes = temp;
                    }
                    temp = strToNum(number);
                    if (temp > -1) {
                        this.number = temp;
                        pickNewRandom = false;
                    }
                    temp = strToNum(maxLevel);
                    if (temp > -1) {
                        this.maxLevel = temp;
                    }
                } <- #48-71 if (lineNumber == 1)
                lineNumber++;
            } <- #45-73 while (reader.hasNextLine())
        } catch (FileNotFoundException e) {// specific exception
            e.printStackTrace();
        } catch (Exception e2) {// any other unhandled exception
            e2.printStackTrace();
        }
        System.out.println(x:"Loaded state");
        int range = 10 + ((level - 1) * 5); //zb64 2/12/24
        System.out.println("Welcome to level " + level);
        System.out.println(
            "I picked a random number between 1-" + (range) + ", let's see if you can guess.");
    } <- #37-84 private void loadState()

    /***
     * Gets a random number between 1 and level.
```

Missing Caption

Checklist Items (0)



```
    /***
     * Gets a random number between 1 and level.
     *
     * @param level (level to use as upper bounds)
     * @return number between bounds
     */
    private void generateNewNumber(int level) {
        int range = 10 + ((level - 1) * 5);
        System.out.println("Welcome to level " + level);
        System.out.println(
            "I picked a random number between 1-" + (range) + ", let's see if you can guess.");
        number = random.nextInt(range) + 1;
    } <- #92-98 private void generateNewNumber(int level)

    private void win() {
        System.out.println(x:"That's right!");
        level++;// level up!
        strikes = 0;
    } <- #100-104 private void win()

    private boolean processCommands(String message) {
        boolean processed = false;
        if (message.equalsIgnoreCase(anotherString:"quit")) {
            System.out.println(x:"Tired of playing? No problem, see you next time.");
            processed = true;
        }
        // TODO add other conditions here
        return processed;
    } <- #106-114 private boolean processCommands(String message)

    private void genHints(int range) {
        int startRange = Math.max(a:1, number - range/2); //this calculates the minimum range
        int endRange = Math.min(a:10, number + range/2); //this caluclates the maximum range
        System.out.println("Hint is that the number is between " + startRange + "-" + endRange);
        hintUsed = true;
        //zb64 2/12/24
    } <- #116-122 private void genHints(int range)

    private void lose() {
        System.out.println(x:"Uh oh, looks like you need to get some more practice.");
        System.out.println("The correct number was " + number);
        strikes = 0;
        level--;
```

Missing Caption

Checklist Items (0)

Missing Caption

Checklist Items (0)

COLLAPSE

Misc (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Reflection

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Example prompts: Learn anything new? Face any challenges? How did you overcome and issues?
☐ #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

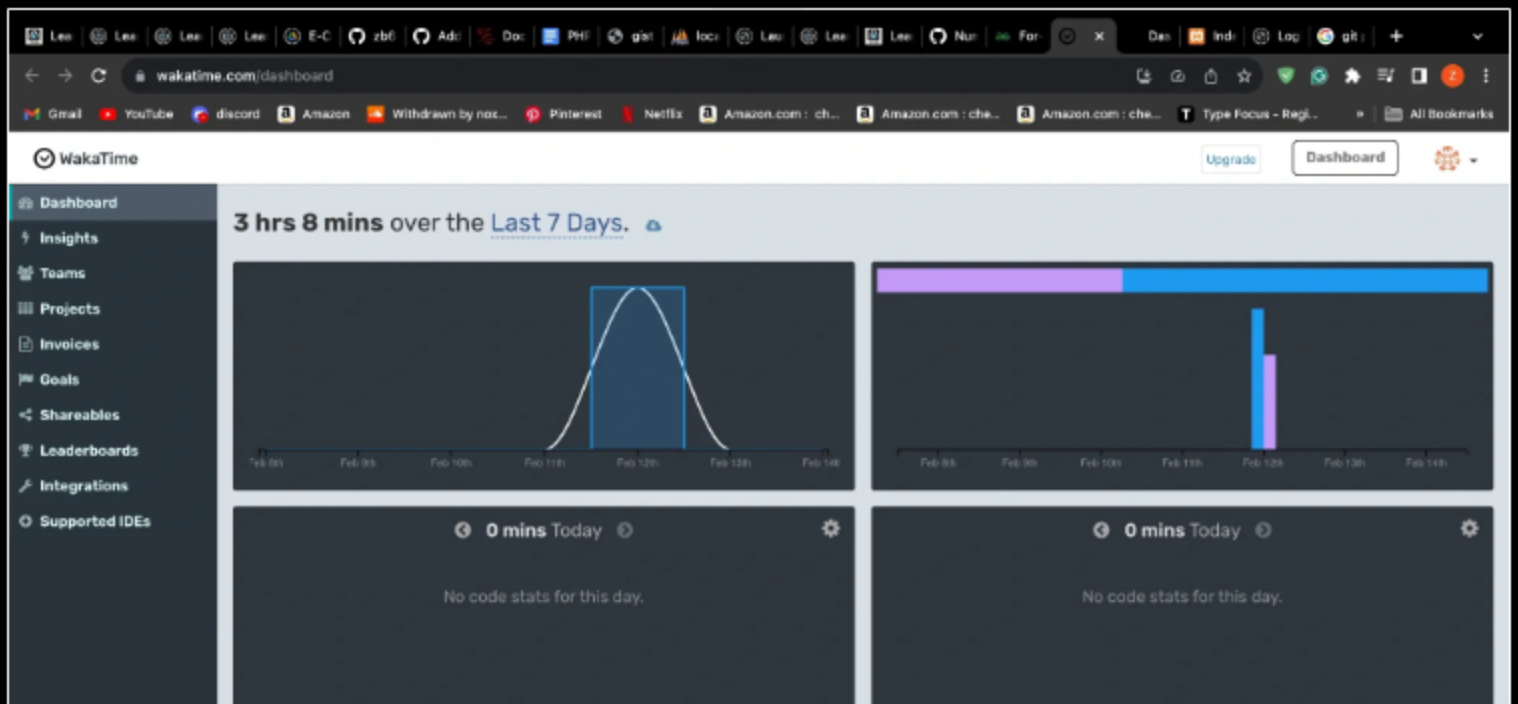
Checklist — *The checkboxes are for your own tracking

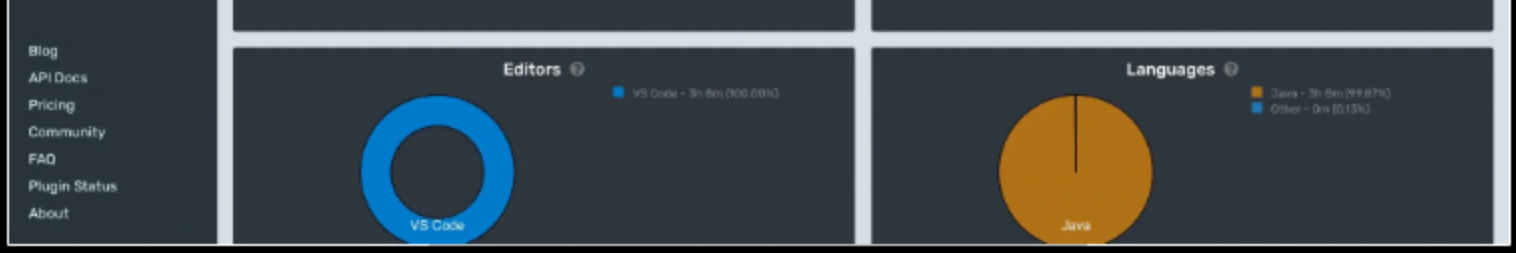
#	Points	Details
☐ #1	1	Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

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