**Project Sprint #5**

The main tasks of this assignment are:

1. Adding the feature of recording a game into a text file. The user story and acceptance criteria of both record and replay are required, but the implementation of replay is for extra credit (up to 2 points in the weighted total).
2. Conducting a code review exercise.
3. Summarizing the lessons learned from Sprint 0 through Sprint 5.

The following is a sample GUI layout of the final product, where “Replay” is optional.

|  |  |  |
| --- | --- | --- |
| SOS Icon  Description automatically generated Simple game Icon  Description automatically generated General game Board size  8 | | |
| Blue player  Icon                              Description automatically generated Human  Icon  Description automatically generated S  Icon  Description automatically generated O  Icon                              Description automatically generated Computer | Chart, line chart  Description automatically generated | Red player  Icon  Description automatically generated Human  Icon  Description automatically generated S  Icon  Description automatically generated O  Icon  Description automatically generated Computer  Replay |
| Record game | Current turn: blue (or red) | New Game |

Figure 1. Sample GUI layout of the final product

**Total points: 16**

1. **Demonstration (6 points)**

Submit a video of no more than 8 minutes, clearly demonstrating that you have implemented all the features in the following table. In the video, you must explain what is being demonstrated.

|  |  |
| --- | --- |
|  | **Feature** |
| 1 | A complete simple game of two human players is recorded |
| 2 | A complete general game of two human players is recorded |
| 3 | A complete simple game of human-computer players is recorded |
| 4 | A complete general game of human-computer players is recorded |
| 5 | A complete simple game of computer-computer players is recorded |
| 6 | A complete general game of computer-computer players is recorded |

If you have implemented the “replay” feature for extra credit, you should include its demonstration in the video.

1. **User Stories and Acceptance Criteria for the Record/Replay Requirements (1 points)**

* **User Story Template**: As a <role>, I want <goal> [so that <benefit>]

Add or delete rows as needed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **User Story Name** | **User Story Description** | **Priority** | **Estimated effort (hours)** |
| 20 | Record | As a player, I need to record my game so that I can review my moves. | 1 | 4 |
| 21 | Replay | As a player, I need to replay a recorded game so that I can visualize and review my previous game. | 2 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **User Story ID and Name** | **AC**  **ID** | **Description of Acceptance Criterion** | **Status (completed, toDo, inProgress)** |
| 20 Record | 20.1 | AC 20.1 Recording a Simple Game  Given a new Simple Game  When the record button is selected  And New Game is pressed  Then all moves are recorded | Completed |
| 20.2 | AC 20.2 Recording a General Game  Given a new General Game  When the record button is selected  And New Game is pressed  Then all moves are recorded | Completed |
| … |  |  |
| 21 Replay | 21.1 | AC 21.1 Replaying a Simple Game  Given a recorded Simple Game  When the replay button is hit  Then the prior Simple Game will be replayed for the user | Completed |
| 21.2 | AC 21.1 Replaying a Simple Game  Given a recorded Simple Game  When the replay button is hit  Then the prior Simple Game will be replayed for the user | Completed |

1. **Code Review (2 points)**

Apply source code review to one or two most important classes (and other classes if time permits) and report the findings. In addition to looking for bugs, the review should check: (1) whether the entire project has followed the coding standard in a consistent manner, (2) whether the project has followed the design principles introduced in class, and (3) whether there are code smells that indicate the need for refactoring. The following checklists provide basic guidelines. You may add new items to each of the checklists.

Make sure your answers resulted from the code review exercise. If there is no finding for an entry, you should provide an explanation. For example, if your answer to “Are the naming conventions violated?” is no, you should describe a naming convention and present an example. You will receive no credit for this part if your answers are simply yes or no without additional information.

Classes that have been reviewed: GUI, Board

Date/time duration of the code review exercise: 12/7/2023

|  |  |  |  |
| --- | --- | --- | --- |
| **Checklist** | **Checklist Item** | **Findings** | |
| Coding Standards | Are the naming conventions violated? | Yes, some method names and class names have not adhered to the naming conventions. (Ex: cpuRED, GUI, checkSOS) | |
| Is the ordering convention of method arguments violated? | No, I ordered my methods in a way where either relevant variables were nearby, or the method is near relevant nested methods. This made adjusting and adding to my code easy. | |
| Any comments meaningless or inconsistent with the code? | Yes, there exist comments that are irrelevant. Such comments are commented out code I tested manually. Most comments are meaningful. Such comments either describe what a method does or are useful notes I used to develop methods. | |
| Any code block has an inconsistent formatting style? | No, my style is very consistent. It is mostly built from habit. All my methods and classes are formatted the same. I did that deliberately. | |
| Any indentations inconsistent? | No, but I did not follow the Google Style Guide’s conventions on this (+2 spaces for each new block). | |
| … |  | |
| Design Principles | Any class/method not well-modularized? | My subclasses, SimpleActionListener and GeneralActionListener are very, very long. I reduced it down as much as I felt reasonable. The reason they are long is because each version of the game had special requirements and I needed to ensure each game met those requirements. | |
| Any class with poor abstraction? | I didn’t use any private classes, as I was not 100% certain when and when not to use a private method. Board Class methods should mostly remain public as they are called by the GUI class. | |
| Is the visibility of any variable, method, and class inappropriate? | Since most methods are called in the GUI class and not in any of the four action listener subclasses, I feel most methods may need to remain public. | |
| Is design by contract (pre/post-condition) violated? | I would say no. I designed in a way where the client methods must adhere to strict conditions. I made many conditions though, and have traced my code to ensure it accounts for any said scenario. Most have been verified through testing. | |
| Is the Open-Closed Principle violated? | Yes, I have modified methods such as makeSMove. In fact I did this to simple count how many moves have been made. | |
| Is the Single Responsibility Principle violated? | I would say no. I have subclasses of GUI dedicated only to a single game mode and it is easier to enforce the game rules with the game logic separated in the board class. Other subclasses serve a single purpose: One is starting a game and the other is replaying a recorded game. | |
| Code Smells | Are there magic numbers? | Yes, I use numbers 0, 1, 2, and 3 in my code to represent the state of cells on the board. (0 = empty, 1 = has S, 2 = has O, 3 = restricted cell because a Simple Game is over) | |
| Are there unnecessary global / class variable? | I use certain global variables to help manage specific requirements with games, such as signaling when a CPU is in a Simple Game and must follow the Simple Game rules. | |
| Is there duplicate code? | I have methods that look similar with deviations. I have makeSimpleGrid and makeGeneralGrid, which are very similar. The only difference is which ActionListener is used. | |
| Are there long methods? | Both Simple and General ActionListeners have long methods that are that way because they account for different player scenarios for their game modes (Human vs Human, Human vs CPU, CPU vs Human, CPU vs CPU). | |
| Is there any long parameter list? | I do not use very long parameters; this is because I use many nested methods and I do not require so much in one method. | |
| Is there over-complex expression? | I do not use complex expressions, I either flat out declare a number or increment when needed. | |
| Is there switch or if-then-else that needs to be replaced with polymorphism | This would most likely need to happen in my Simple and General ActionListeners, as there is a lot of if statements in each. A good example of separation is my checkSOS method, which checks the array in all directions (each direction is separated into methods). | |
| Any variable or method name whose intent is unclear? | I would say no as I purposefully name my methods/variables to describe its purpose. | |
| Any similar methods in different classes? | My Simple and General Action Listeners have very similar methods that deal with game requirements. Minor differences between both of them. | |
| … |  | |
| **Bugs** | **Buggy code snippet** | **What is the bug?** | **Why is it a bug?** |
| N/A by far |  |  |
|  |  |  |
|  |  |  |

1. **Summary of All Source Code (1 points)**

**https://github.com/DKarr393/CS-449/tree/main/Sprint%205/SOS\_Game/src**

|  |  |  |
| --- | --- | --- |
| Source code file name | Production code or test code? | # lines of code |
| Board.java | Production | 951 |
| BoardTest.java | Test | 365 |
| GUI.java | Production | 1964 |
| GUITest.java | Test | 18 |
| Total lines of code | | 3298 |

**You will receive no credit for this assignment unless your complete source code is submitted.**

1. Summarize the lessons learned from the entire project by answering the following questions from the perspectives of development processes, coding, design, refactoring, and testing **(6 points)**:

* What did you personally gain from the project?
* What does your project do well, and what could your project do better?
* How could you improve your development process if you develop a similar game from scratch?

Minimum requirement for (5): One full page single spaced, font size no bigger than 12 points.

What I have gained from this project is the fundamental understanding of the process of software engineering. This was my first experience with GUIs and Unit Testing. This was also my first experience with Java. This project is one of my flagship items on my resume and LinkedIn page. I am proud of this project because this is the biggest and most complex project I have ever undertaken. Is it perfect? No. This is my first Software Engineering Project. I still have a lot to learn, but I have gained a process where I can learn better. I use my test-driven environment to help ensure that my program meets the objectives. I have learned problem-solving techniques and how to teach myself. For example, back in Sprint 3, I was working on getting the board to detect an SOS. I was creating my checkSOS method, which was originally much longer than it is now. I used Eclipse’s code coverage feature to help me realize that there were parts of that method that were never reached. Thus, I created 12 nested methods to be called in checkSOS. 8 of them were to check for SOSes whenever an S is placed, and the other 4 were used for when an O is placed. I was proud of myself for figuring it out. I felt that I had become a better programmer because I had itemized objectives to achieve, and my work was targeted in that fashion. Extra touches were to ensure the maximum user experience, such as adding a delay to computer moves. I did that so it felt as if you were playing against a human who had to think of their next move.

I feel my project meets all the requirements and does so with a nice presentation for a rookie. I’m proud of my GUI design. A majority of the lines of code in the GUI were extra touches of detail I wanted to add. I was showing my girlfriend my program, and she complained that it was too bright. I thought, “Maybe I should address that”. I made the GUI mostly dark so that it was easier on the eyes. I’d bring my project with me to Boy Scout Meetings and let young scouts try out the game. If there was one thing that I think I did a great job with, it was making a great user experience. It helped me to put a face in those user stories, having people test my program out and receive feedback from a non-programming perspective. However, despite the good, I could’ve used better coding practices. I have long methods and some parts of the code do repeat. I am unfortunately guilty of some bad coding practices. That’s to be expected though. I was never going to be perfect, especially not while learning a new programing language while working on the project. I will be honest, if I had chosen C++ (a language I am more familiar with) I would have likely still used the same bad practices.

Let’s not dwell on the bad. What am I going to do about it going forward? Now that I know how to properly run a test environment, I will continue to test my code. However, I want to start with a class diagram. I think mapping out the basic components of my code before I write it for any sprint will make me a more effective programmer. It will allow me to adhere to Coding Standards and write cleaner code. Here’s my step-by-step process that I believe is a better way to write code going forward:

Plan sprints, Create user stories, Create acceptance criteria, Try to predict code time, Adjust sprint plans as needed, Draw class diagram, create unit tests, Write code to pass tests, Run manual tests, Review acceptance criteria, Review user stories, Repeat.

I think this will allow me to capture all requirements and do so in a better way than I have done for SOS. Though my SOS program isn’t perfect, I will always and truly be proud of my work on it. I am grateful for the opportunity to work on such a project. I was initially intimidated by the sound of a Software Engineering project. I had no illusions that I was a spectacular coder, especially with my Imposter Syndrome. Which is the main reason I am grateful for this project. I proved my biggest detractor wrong, myself. Not only did I do that, but I have something I can keep and treasure forever. This project is symbolic for me. I didn’t expect to have so much fun doing this either. I sometimes stayed up late because I was so excited to complete the next Sprint. It is why my time management for Sprints seemed to be better, it was because my motivation increased tenfold after Sprint 2. My newfound attitude towards programming will help me follow through on the process I described above. I cannot wait to try making fun personal projects with all that I have learned. I will be using the same processes I have learned in this class. I am very lucky to have had this opportunity to be challenged, because I feel I have achieved an extreme amount of growth. Especially comparing my negative attitude going into the semester, versus my attitude now.