Final project report.md

CSCI 5448: Object-Oriented Analysis & Design

Prof. Bruce Montgomery

Project 7: July 20, 2022

Final project report

1. Name of project and names of all team members

Pong With Me by Tyler Walker and Ivane Gamkrelidze

2. Final state of system statement

Every feature outlined in Project 5 was implemented. Every stretch goal outlined in Project 6 was met. We deviated from the patterns proposed in Project 5 as we found more appropriate patterns. We believe that we accomplished a good final state of the system and the game seems to work very well. We refactored the code several times, and provided plenty of comments so that the code is easily readable and maintainable.

Primary features implemented were:

- Main menu
- Top5 leaderboard
- Gameplay
  - Paddle motion
  - Ball motion
  - Game timer
  - Collisiion detection
  - Increasing game difficulty

Stretch goals implemented were:

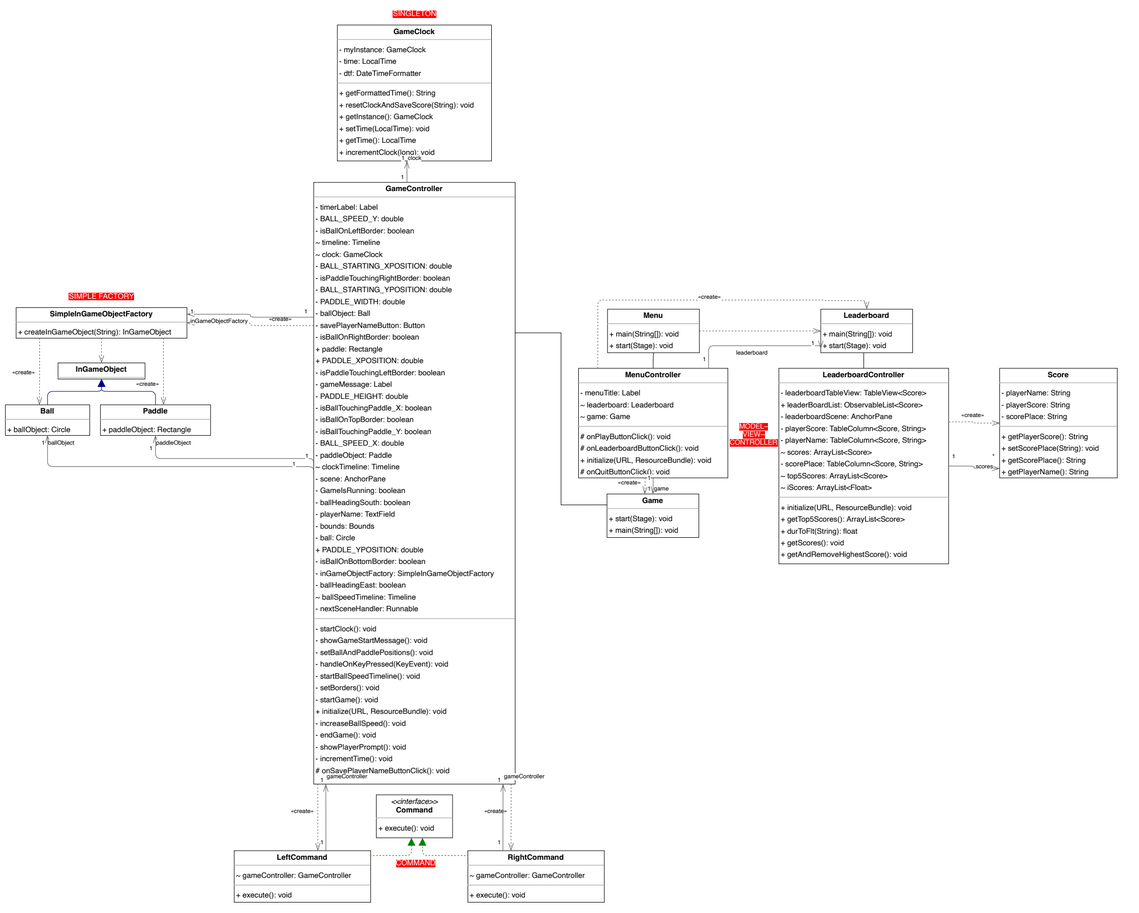
- Make the game work despite window size
- Custom player name

3. Final class diagram and comparison statement

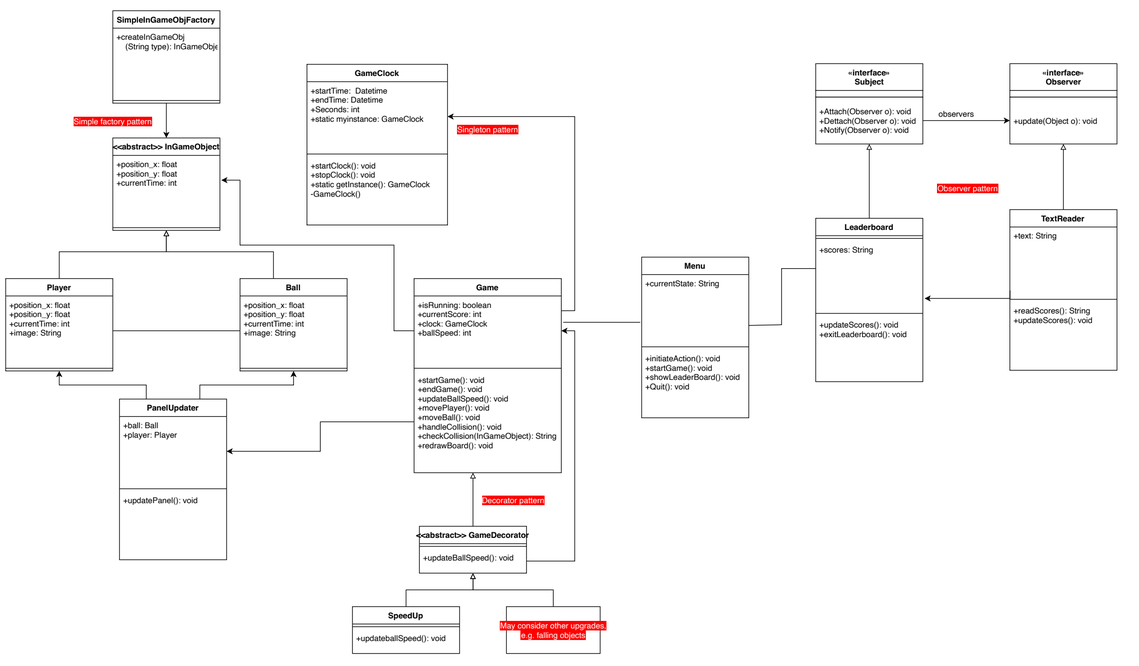
Final class diagram

Legend

+	public access modifier
#	protected access modifier
~	package-private access modifier
-	private access modifier
🟥	design patterns



Original class diagram



Key changes in system since original design

Since the original design and analysis work, we found the observer and decorator patterns unnecessary and thus did not implement them. Instead, we decided for model–view–controller and command. We found many modern JavaFX program examples use MVC models where the View was FXML, which is a really nice markup to build GUIs. The FXML is essentially markup that was connected to the Controller. The command pattern made sense for handling the movement of the paddle in the game. Another thing that we didn't realize was that we would have to handle a lot of the logic in the respective controller classes (`LeaderboardController` , `MenuController` , and `GameController`). This is why the Controller classes have a lot more properties and methods. Overall, the UML class diagram of our final product looks much different than our UML class diagram from Project 5.

4. Third-party code *vs* original code statement

The code in the project is original, except for the following sources, which were utilized and adapted to our needs:

- Add CSS files to scene: Add an external CSS file to a JavaFX Application | Engineering Education (EngEd) Program | Section
- Ball motion: JavaFX and Scene Builder - bouncing ball - YouTube
- Ball motion: JavaFX with Scne Builder - Bouncing Ball
- Button styling: intellij idea - Styling button in javaFX using CSS - Stack Overflow
- Game controller: AbdelrahmanBayoumi/StopwatchFX: Stopwatch desktop application made with JavaFX
- Game controller: Gaspared/pong: simple pong game in javaFX
- JavaFX library: JavaFX
- Keyboard listener: java - setOnKeyPressed event not working properly - Stack Overflow
- Leaderboard: java - Javafx GUI scoreboard - Stack Overflow
- Leaderboard: javafx : TableView Example
- Pong mechanics: PONG Game, Java (fx) Programming Tutorial - YouTube
- Project set-up: Create a new JavaFX project | IntelliJ IDEA
- Stopwatch: Stopwatch With Source Code | JavaFX - YouTube

We used several of these sources as a good starting point to develop our project. Our code significantly deviates from the original sources, which merely served as guidance.

5. Statement on the OOAD process for your overall semester project

1. Implementing patterns pre-determined during the design stage proved challenging. While some were obvious (MVC), in other cases we had to learn as we go which patterns were appropriate as we developed the program.
2. We found the UML class diagram to be a helpful starting point. Briefly, we started the project by setting up the structures identified in the UML class diagram. However, we found this was not the way forward for this project (too many depenedencies for GUI to figure out). So, instead we began with a simple JavaFX `HelloWorld` -type project and built up from that. We still found the majority of the structures identified in the UML class diagram instructive.
3. Working with a GUI to build a game, we had to think a lot about the state of objects over time. JavaFX Timelines helped us manage objects over time, such as the positioning of the paddle or ball every 10 milliseconds. The logic to determine the interaction between objects heavily depended on using JavaFX Timelines, which proved to be much more efficient than using our `RedrawBoard` method that we had originally envisioned.