Software Architecture
Patterns
Kacper Multan, Alicja Jonczyk

Part 1: Speak about the implemented program: The chosen program from the first exercise (Technology Introduction) or the new code.

We chose to work on this exercise using the Pong Game. Our game was based on two players competing against each other - Player One who was able to move their paddle up and down with keys W and S and Player Two who did it with the Up Key and Down Key. Besides the main rules of the game we also added another functionality - the ball increased its velocity after 5 unsuccessful bounces from the paddle - so the velocity increased 1,5 times when no player scored a goal in 5 tries. We also added a restart button that allowed the players to begin a new game.

Our decision to choose Pong was based on the size of the program (it was more complex than the Helicopter Exercises), which made it easier for us to implement new and more diverse patterns.

Part 2: Speak about the implementation of the Singleton pattern:

The singleton pattern is used to have one instance of a class used in the whole program that is also easily accessed everywhere in the project.

Secondly, Speak about the implementation of the Singleton pattern in the chosen program. Comparing the original code and the singleton pattern implemented code in your report is recommended.

We decided to use the Singleton Pattern for the class GameState. This class in our code contains the current state of the game - players points and information whether the game is running or not. As the game Pong has only one game running at a time using the Singleton Pattern makes it easy to get the current game status at any point in the program.

In this particular situation the implementation of the singleton pattern in our program does not change a lot in its logic and performance. Taking into consideration that everything in our code happens in one class - MyGdxGame, we only really use the pattern partially - to have one instance of the class GameStatus used in the whole program.

Part 3: Speak about the implementation of the pattern(s) from the lists of the pattern: Speak about how you modify the program by using pattern(s).

For this part of the exercise we used the Singleton Pattern, the Entity Component System Pattern and the Observer Pattern. We used the Singleton Pattern for the class GameState in the same way we described in Part 2 above.

The most important pattern in our program is the Entity Component Pattern. As an Entity we described everything - the paddles, the ball, the restart button. We decided to exclude the text as it is only displayed on the screen and does not have enough of the necessary

components that the others share. Our Entities have the following components: rectangle (which includes the position of the entity on the screen, its height and width - we decided to place those properties in one component as it makes it easier to implement an already existing library element used in our code), velocity and shape (whether it is a rectangle like the paddles or button, or circle like the ball, and the color of the entity).

We also added an extra field to describe the entities - entity type (whether it is interactive or static so whether the player can control it or not).

To keep the game and entities up to date and control the game flow we use the classes EntityManager, GameManager, PaddleInputManager.

We also added two systems - RenderSystem and Movement System. The first one is used to render the entities and text on the screen. The second one updates the positions of the entities depending on their velocity.

The Observer Pattern is used to take up action depending on the notifier's reaction to the changes in the program (such as game over, restart and updated entity).

Part 4: Answer these theoretical questions in your report:

A. For the patterns listed in Step 3, which are architectural patterns, and which are design patterns? What are the relationships and differences between architectural patterns and design patterns?

Architectural Patterns:

1. Model View Controller (MVC)
2. Entity Component System (ECS)
3. Pipe and Filter

Design Patterns:

1. Observer
2. State
3. Template Method
4. Abstract Factory

Architectural patterns dictate the overall structure and behavior of the system, while design patterns address specific problems within a particular context in software design. When it comes to flexibility and reusability architectural patterns are harder to change because they're the foundation of the system, design patterns, on the other hand, are easier to reuse and adjust because they solve specific issues. There is also a difference in dependency: sometimes, architectural patterns need design patterns to work smoothly, but design patterns can stand alone and be used in different architectural setups.

B. How is your chosen pattern realized in your code? (Which class(es) works as the pattern you chose?)

The Singleton Pattern is implemented in the class GameState.

The Entity Component System Pattern in the classes:

- Components: Component, RectangleComponent, ShapeComponent, VelocityComponent
- Entities: Entity
- Systems: RenderSystem, MovementSystem
- Managers: EntityManager, GameManager, PaddleInputManager

The Observer Pattern in the classes:

- Observers: RenderSystem, MovementSystem, EntityManager
- Notifiers: EntityManager, GameManager

C. **Are there any advantages to using this pattern in this program? (What are the advantages/disadvantages?)**

Advantages:

- the use of the Entity Component System Pattern gives the code better structure and improves its readability
- the ECS Pattern makes it easier to maintain and expand the project
- the Observer Pattern allows to reduce tight coupling between classes that should not be strictly connected

Disadvantages:

- more code (we compared the size of our projects from the Exercise 1 - Technology Introduction and this exercise)
- lower efficiency - slower computing time

Part 5: Comment: Speak about if you completed the exercise individually or with a pair. If in the pair, say his/her name.

We worked as a pair, our team was Kacper Multan and Alicja Jonczyk.