<u>Project 7- Final Project Report</u>

Name of project- Snake App
Team members- Shaily Goyal and Shreya Maitra

<u>Final State of System Statement</u>

For this project, we implemented the classic Snake game using Java and JavaFX. Compared to Projects 5 and 6, we added two new features: "View Scores" and "Game Settings". Firstly, there is a "View Scores" button on the main menu that allows users to view all the scores saved in the scores.txt file. Clicking on the button opens a new window that displays all the scores by all users. This feature provides users with the ability to compare their scores with other players and adds a competitive element to the game.

Secondly, the "Game Settings" button was added to the main menu, which allows users to change the color of the snake. The user can choose from green, blue, and red color options. This feature provides users with a way to customize their gaming experience and adds an element of personalization to the game.

We chose not to implement the "New Game" button feature since the game already starts when the user presses any key, and adding a separate button for this feature was unnecessary. Additionally, we did not make any changes to the audio and high score features from Project 6. Instead, we implemented the feature to display all scores rather than just the high scores. Our focus was to improve the game's overall functionality and user experience with the new features we added.

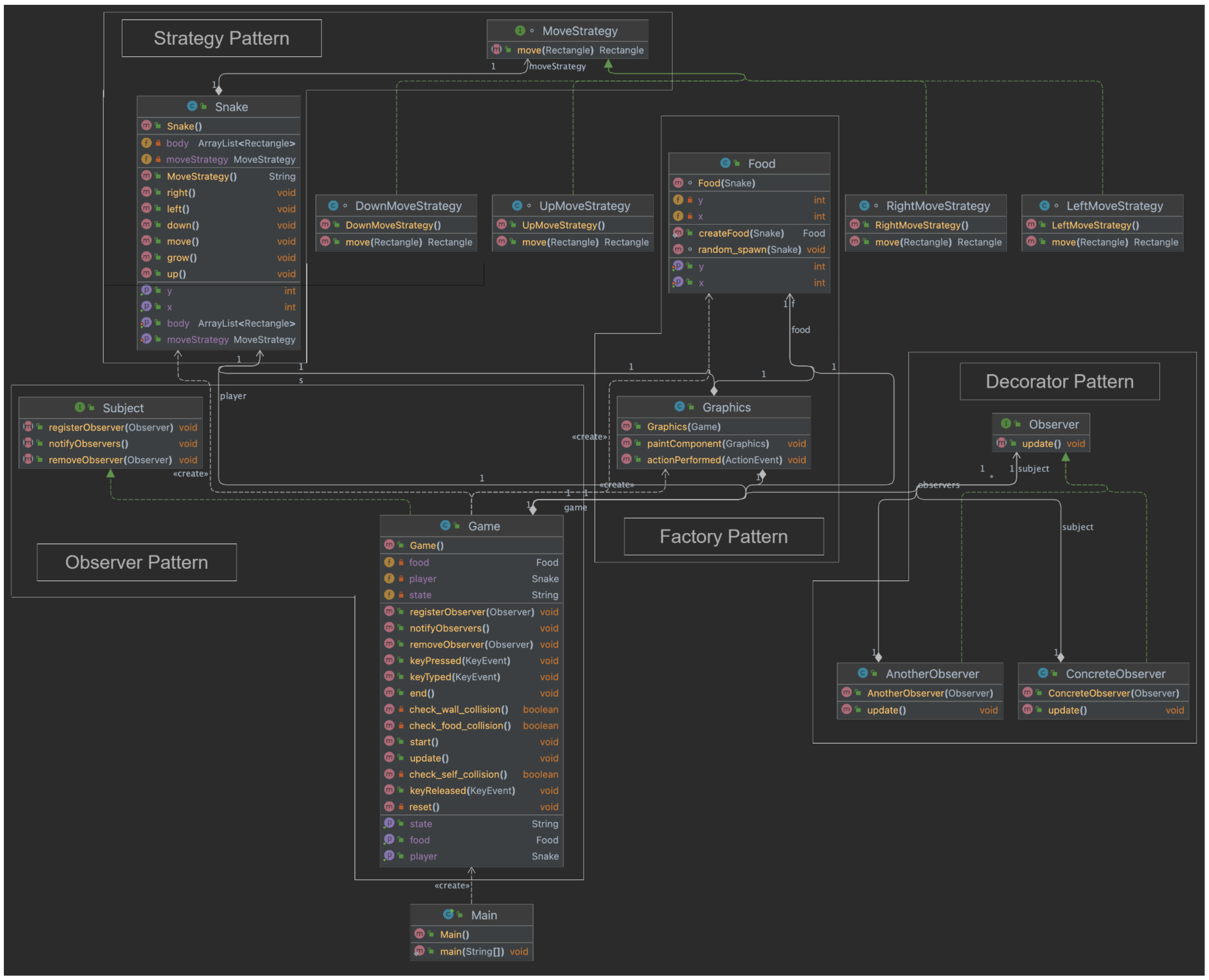
This is a simple snake-game application. Here the snake moves based on the arrow pressed. Below are the keys that we need to press for the snake to press:

- "w" to move up
- "d" to move right
- "a" to move left
- "s" to move down

When the snake touches the food its length increases and score increases. For every food a snake eats the score increases by 2. If the snake touches any of the walls then the game expires.

Final Class Diagram and Comparison Statement

Project 5 UML Diagram:

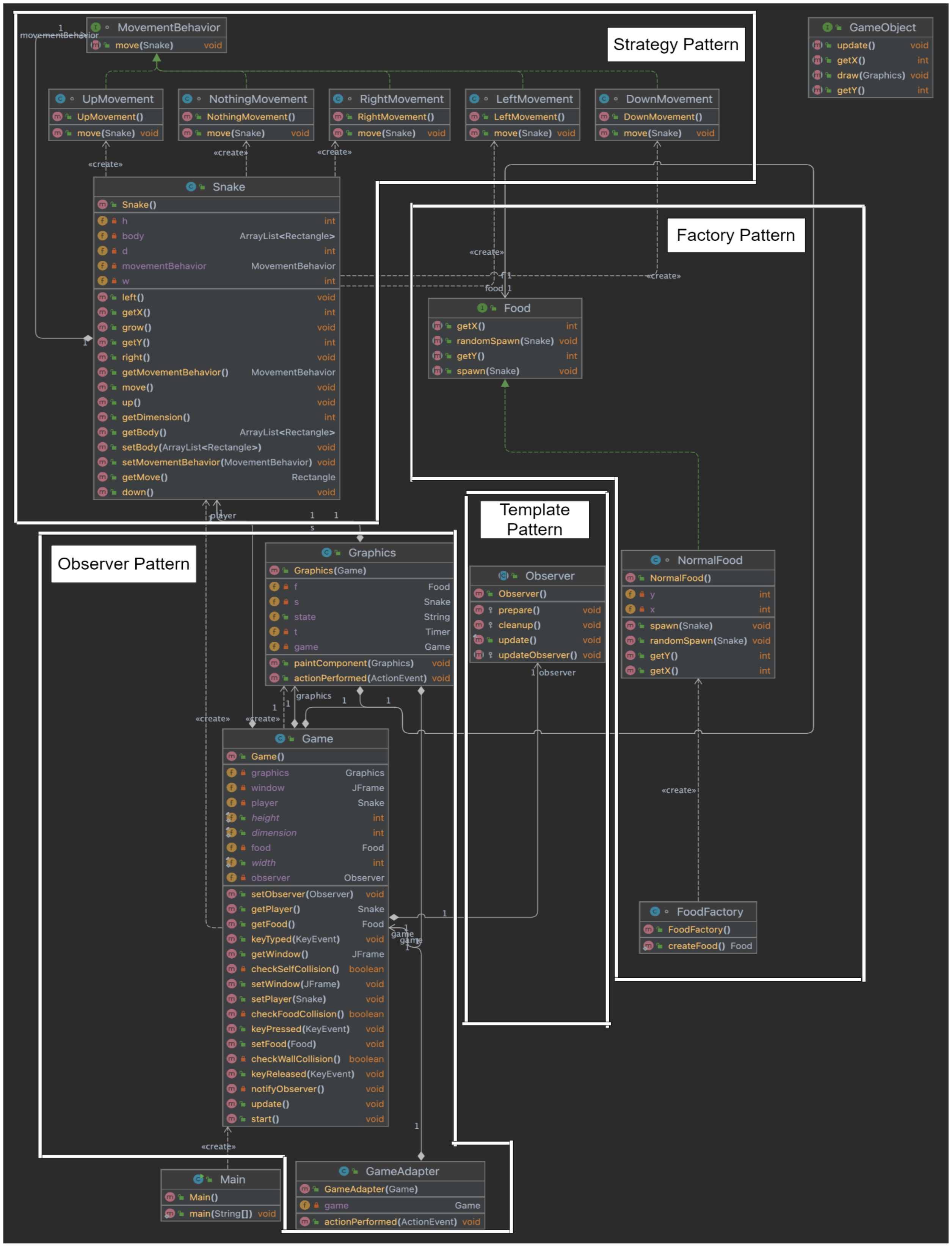


Changes in UML diagram for project 6 from project 5:

For project 5 no design pattern was explicitly used for Graphics.java class but for project 6 Observer pattern has been implemented . With this modification, the Game class is now able to alert the Graphics class whenever the game's state changes so that it can update the game objects.

For project 5 we had thought of implementing Decorator pattern for Observer.java class. With specific steps carried out by the subclasses, an algorithm's broad outline can be created using the template design pattern. So for project 6 we finally went ahead with using Template pattern for the Observer.java class.

Project 6 UML Diagram:

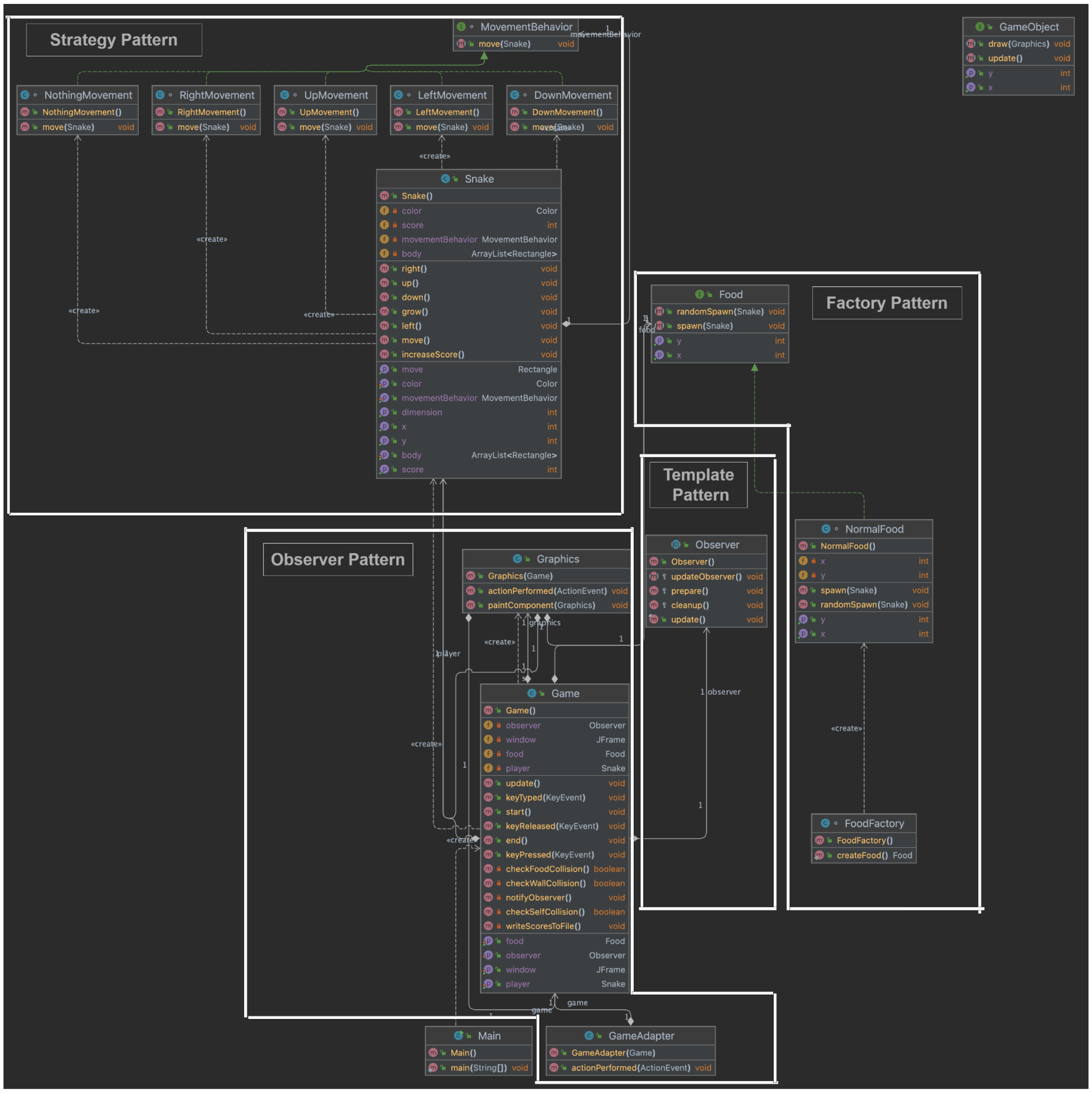


Changes in UML diagram for project 7 from project 6:

No changes have been made with respect to the design patterns used in project 6. Food.java still uses the factory pattern, Snake.java still uses the strategy pattern, and Game.java still uses the observer pattern and Observer.java class still uses Template Pattern.

For project 7 we have made in the code with reference to the addition of two new buttons "View Scores" and "Game Settings". “View Scores” shows all the scores scored by all the players and using “Game Settings” the color of the snake can be changed.

Project 7 UML Diagram:



Third-Party code vs. Original code Statement

We have taken below code reference of its functions use from URL:
https://www.programiz.com/java-programming/bufferedwriter

```
private void writeScoresToFile() {
   try {
      BufferedWriter writer = new BufferedWriter(new
FileWriter("scores.txt", true));
      for (int i = 0; i < scores.size(); i++) {
         writer.write("player score"  + ":" + scores.get(i)  +
"\n");
      }
      writer.close();
   } catch (IOException e) {
      System.out.println("Error writing scores to file: " +
e.getMessage());
   }
}
```

We have taken below code reference of using Math.random function from URL:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

```
x = (int)(Math.random() * Game.width - 1);
y = (int)(Math.random() * Game.height - 1);
```

Other links used for reference for third-party elements:
https://www.educba.com/java-tools/
https://www.theserverside.com/feature/
https://stackify.com/top-java-tools/
https://www.edureka.co/blog/java-frameworks/

Statement on the OOAD process for your overall Semester Project

Listing three key design process elements or issues:

1. Design pattern selection: Selecting an appropriate design pattern was a challenging process. Since the project involved multiple classes and interactions between them, it was crucial to choose a design pattern that could handle this complexity. Ultimately, we implemented the Observer pattern to handle events and updates between the different components of the game. The Factory pattern was used to create objects for the different types of food in the game. The Template pattern was used to define the structure of the game and allow for easy customization of different aspects, such as the game settings. The Strategy pattern was used to encapsulate the different algorithms for calculating the score based on the game mode.

2. File management: We encountered challenges when it came to managing files in the project. One such challenge was storing and retrieving scores for the game. We had to develop a file management system that could handle this task effectively. So, we created a separate file to store the scores, and implemented a method to read and write to the file.

3. UI design and implementation: Designing and implementing the user interface (UI) was also a task to be mentioned. We had to ensure that the UI was intuitive, user-friendly, and visually appealing. We spent time researching and experimenting with different UI elements, such as buttons, labels, and graphics, to create an engaging and immersive gaming experience. We also had to ensure that the UI was responsive and could handle user input effectively. Overall, we were able to create a UI that met the needs of the project and enhanced the gameplay experience.