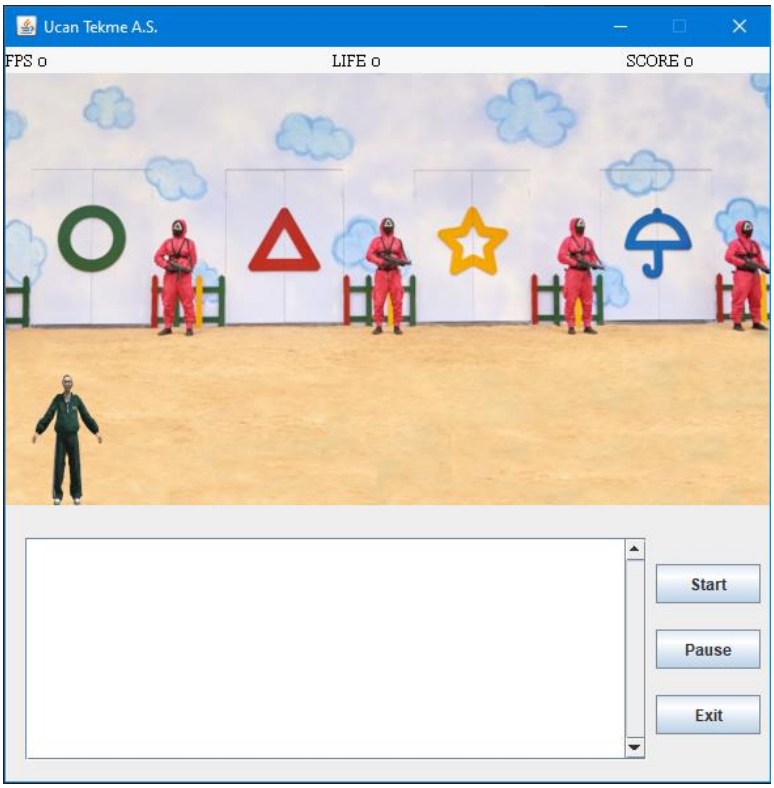
# Object Orıented Analysıs and Desıgn

# Homework-1 Report


# Muhammed Özkan 151044084


## Game Explanation

1. The game screen will be modeled as an empty 2d canvas of size 600x350 pixels.
   Game screen is 600x350 pixels. Frame is 600x600 pixels.
   Fps, lıfe and score are displayed on the screen.
   In the lower right part, there are start, pause and exıt buttons.
   In the lower left part, there is the part where the events in the game will be written as a message.




2. He is the main character that we will manage in the game. Jumps with space key, moves right with d key.

3. These figures show the power-ups we will receive in the game. To gain these power-ups, we must collide with them. The triangular candy represents 2x, the star one means 5x, and the round one means 10x points. Score power-ups increase cumulatively. The one with the umbrella unlocks the low or high jump power-up.

4. These characters represent enemies. If we collide with them, it takes one of our lives. If we jump over them we get points based on our power-ups amount.

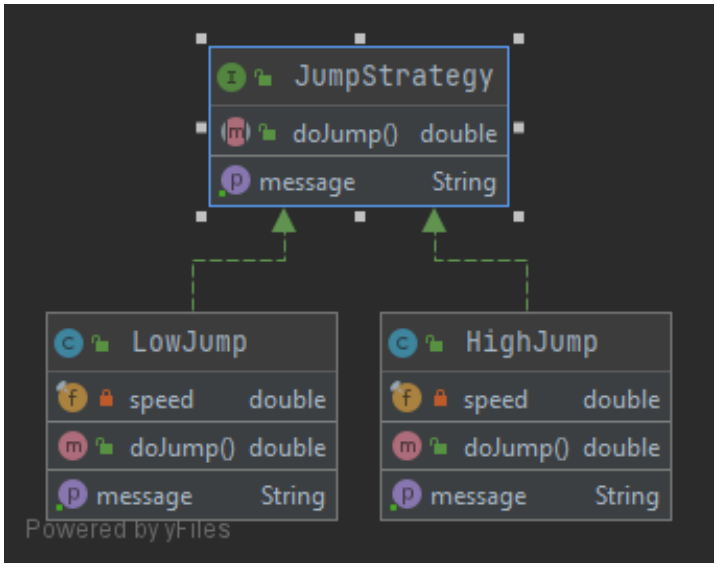
## Design Decision

Strategy and decorator design patterns were used as desired in the power-ups and jump systems in the game. In both systems, it returns the jump amount or the score multiplier with the help of a method, following the conventions. Each object in the game is represented by a class and their positions are recalculated when the update method is called. After each update method is called, the drawing method is also called and the changes are shown to the user. The game is made with runnable implementation to avoid freezing on the screen. With the help of thread, methods such as update, draw, collision control are run with an endless loop. In this way an video effect is given to the user. The thread is also controlled by a timer that releases the relevant mutex every 10ms. The reason ı do this is not to cause busy waiting like sleep. I got a speed like 100fps in 10 ms.

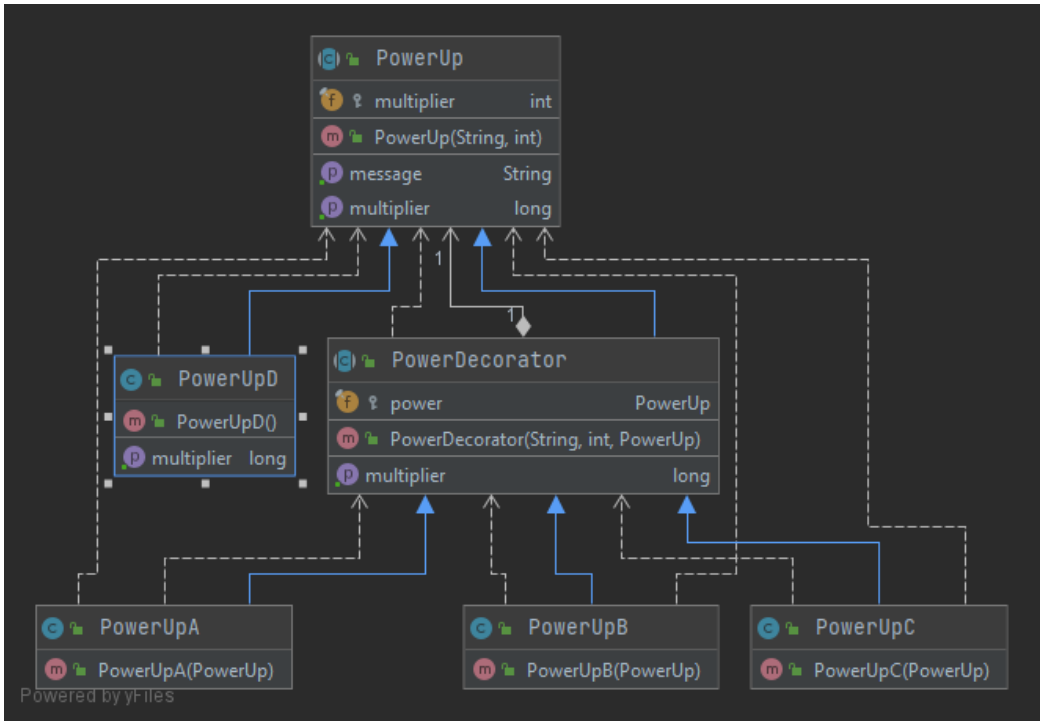
The reason why ı use mutex is to control the running of the threads in the desired time. Functions corresponding to keys captured from the keyboard by listeners are called by the game class. The logic in all parts of the game is the same. The next step is calculated each time the related functions are called by the thread. When draw function is called, related drawings are made to these calculated coordinates. When this starts to do more than 24 times per second, an animated video emerges. I have set up a mutex structure between the timer and the thread, so that I can perform the operations over and over in a period of 10ms.

## Uml Diagram

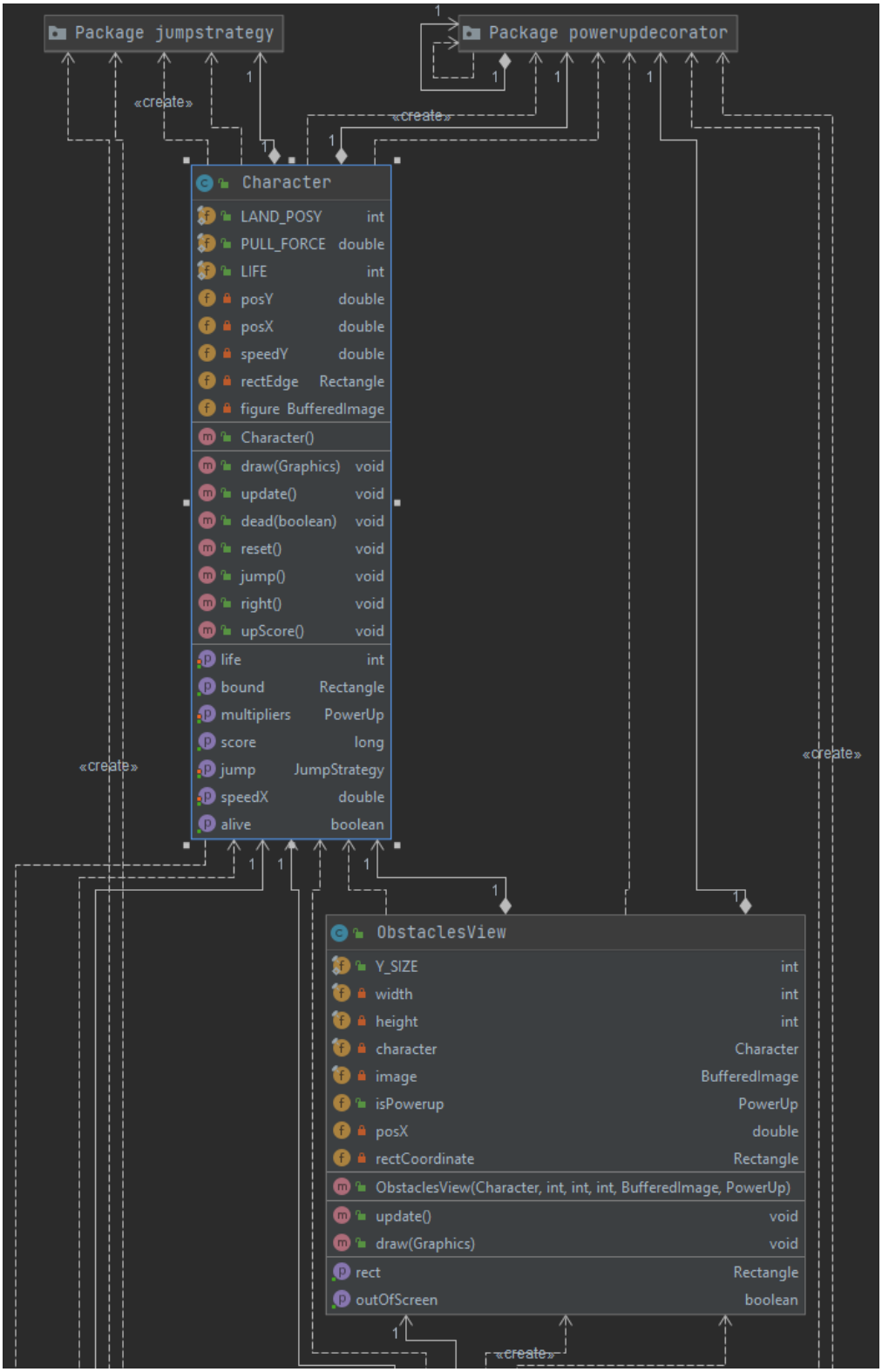
In accordance with the jump system strategy pattern, the dojump() method is designed to be preserved.



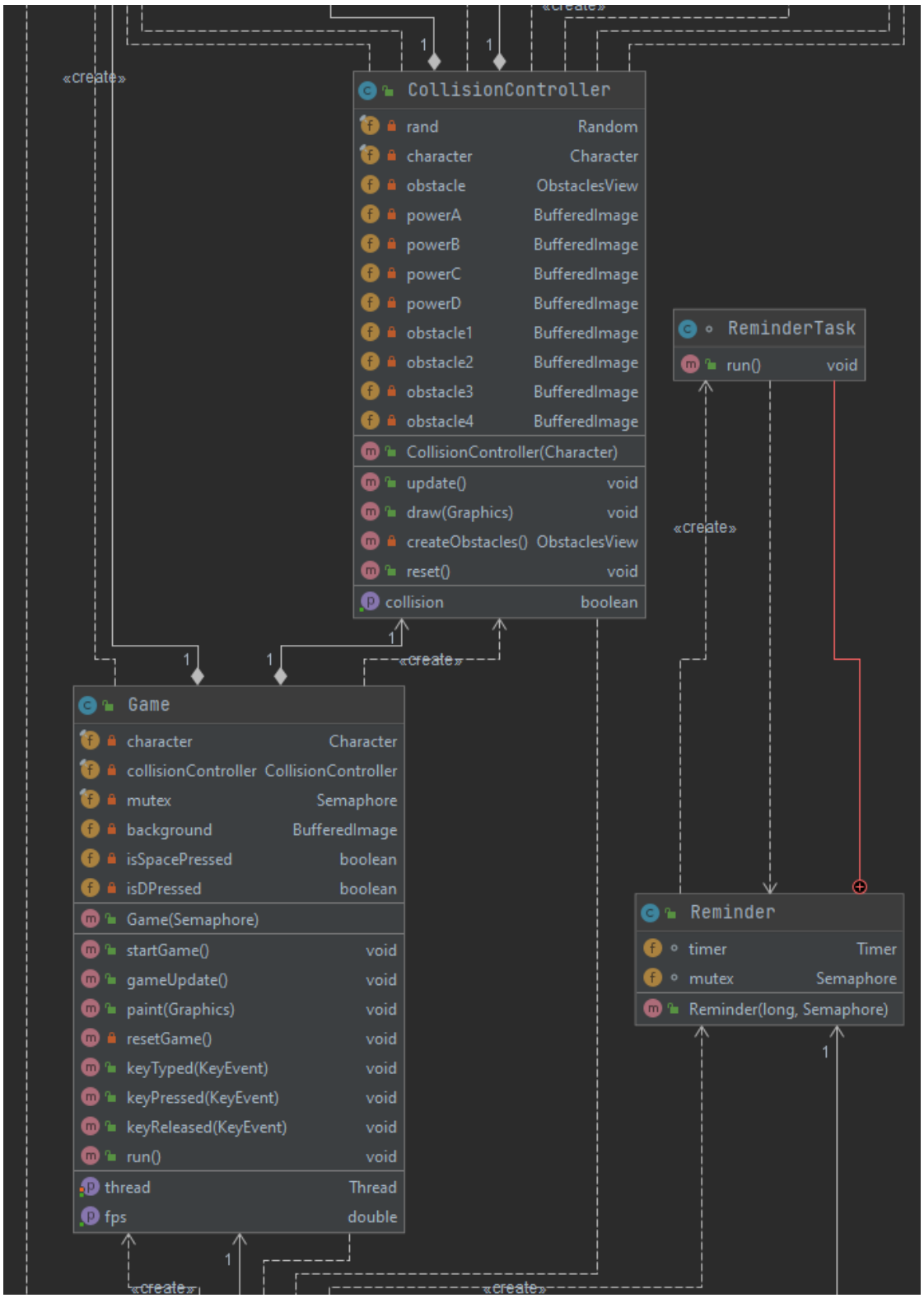

Decorator design pattern was applied in the power-ups system. A b c power-ups derive from an intermediate decorator, while d power-up derive directly from the main class. The intermediate decorator class also includes a power-up element to hold other power-up. At the same time, the intermediate decorator class also implements the main abstract class.



Enemies and power-ups appearing on the screen are represented by a class. Also, our game character is represented by a separate class. Classes represent shapes with properties of position, velocity. It represents functions such as collision and drawing with methods.



It is the collisioncontroller class where the enemies and rewards in the game are created and controlled.



The game class is the part responsible for the game screen and thread mechanism that appears on the screen.

The mainwindow is the part where the controls and other objects that appear on the whole screen are defined.