OOAD HW1
2D Scrolling Game

Sema Dilber

October 28, 2021

Abstract

A 2D Scrolling-Game, using Strategy and Decorator Patterns.

1 Design Decisions

Since I have never implemented a game before, I benefited from a tutorial and examples on the internet. I created the patterns' designs myself according to the book and what I understood from the OOAD lessons I listened to.

1.1 Design Patterns

These patterns contain many details about the project.

1.1.1 Strategy Pattern

For this pattern I created a JumpBehavior interface and it has the jump() method in there. And for 2 types of jump; LowJump and HighJump implements this interface. This methods basically prints the which type of jump to the terminal and returns the height for the jumping height player.
In this way if anyone wants to adding new jump type ex: rocket jump, she/he can do; and as customer asked it changeable dynamically.
In here since I was not successful in collision -So I don't get the A,B,C,D letters-, I associated this D powerup with L key in keyboard. So for changing jump behavior mode, you can press the **L** key.
I create highJumpPLayer and LowJumpPlayer that extends the main character's UcanTekmePlayer class, and this classes creates jump behaviors low and high.

1.1.2 Decorator Pattern

For this pattern I create a PowerUp abstract method and it contains an abstract method currentPowerUp() for calculating the current total powerup. At first I had trouble creating the base class, but then I found it logical to set the powerup to 1 for the base class. Because later, other powerups would be able to wrap it up. I named this classed as PurePowerUp and it returns 1 as currentPowerUp().
I named the Decorator class as PowerUpDecorator and all A,B,C,D powerups extens this class and they firs prints their powerup types in the screan as expected. For the wrapping each other and the base powerup a PowerUp objext was defined in these classes named powerUp and in constructer parameter PowerUps assigned these powerUps so we can wrap one to another in this way(other powerups and base powerup) In D something different because D was for changing jump mode so in currentPowerUp prints D and calls the changeJumpbehavior method.
In this way customer can add additional powerups in the future as asked without changing anything else and with wrapping calculatinf all combination of powerups is possible.
As I mentioned before, these letters are connected to the keys A, B, C and L(for D) since I can't do the collision.

Other Components

-Main character fixed somewhere in the screen and the background is moving to -the left. And screen is painted as expected.
-I created occasional/random immobile monsters of constant and jumpable height, standing.
-Our main character can be jump with Space key as expected.
-Our Main character can go to the right with D key.
-E key for the Exit the game.
-I couldn't make an adjacent TextArea with the game screen but I printed the Jump style, current score and when some powerUp occurs as expected. And I print the screen total life(fixed).
-For starting the press powerup keys, you should jump ones at least.
-I couldn't calculating score so I give 5 point for each jump move.

Detailed information about the code can be found on Javadoc.
BEcause of the size I added class diagram in zip.