# Final Project

Eugene Zhuravel

Gitgub: https://github.com/ezhuravel/FinalProject.git

## Introduction

For the final project, I decided to write a simple Java game using the swing framework. I used this website as a starting point for my game: <http://zetcode.com/tutorials/javagamestutorial/basics/> The game has a character sprite moving around the screen with blue health pick ups and red obstacles that hurt the player. All the sprites are randomly generated during the start of the game. The game is over when the player health, which is shown in the upper left-hand corner of the screen, falls below zero.

To complete the project, I used 4 design patterns that we learned about in class: the factory design pattern, command pattern, singleton, and state.

## Singleton Pattern

The singleton pattern is used when you can only have one instance of the class. I thought the player object was a perfect candidate for this design pattern. Since this project is a single player game, the singleton pattern made sure there could not be two instances of the player object.

## Factory Pattern

Factory patterns help us instantiate several objects during run time. Although, the use of this patter in this pattern was light, I was able to easily randomly place game objects on the game board. With the factory, I can also easily extend it to create more objects with more complex logic such as increasing the number of items by difficulty levels.

## Command Pattern

Command patterns are used to delegate commands away from the target item. I thought a good use case was to use this pattern for collision detection. Based on what kind of collision is made, I decide what command to call. In this case I have two simple commands: take hit or take hit. Using this command pattern will allow me to extend the game easily by adding more commands, or switching them out entirely without needing to refactor the rest of the project.

## State

The state pattern keeps track of state of the object. I thought it was a perfect candidate to keep track of current state and transition to game over state. For this project, I let the states determine what game state to display to the user and whether a player can move or not.

## Challenges

For this project I had a more ambitious plan but ran into some issues which required me to scale back. I wanted to make the player object more dynamic by facing the direction the user is traveling. I wanted to use the state design pattern to handle the orientation of the sprite. My first approach was to rotate the image based on where the player was going but found myself spending to much time on this issue. I then decided to use 4 different sprites instead of rotating one. This also started taking too much time because the sprites where streaking and not fully clearing. Given the time constraints, and I was not focusing on design patterns I abandoned the idea all together.

The second challenge I faced was scaling the design patterns to a bigger project. The assignments were small and self-contained, so they were easier to wrap my head around. The final project was bigger, and I found myself frequently refactoring. The two design patterns that gave me the most problems were command and state. I had a hard time wrapping my head on where to declare them and how to call them. After reading through the book and working through the examples I think I figured it out.