Week 8 Assignment - Design Patterns

Overview

This week, we reviewed Design Patterns and learned how they can make your OOP programs more structured and more easily readable by fellow programmers. Last week's assignment you designed a program to implement the game of Pig. This week, we will expand on that game by using a few patterns that we have learned, specifically the Proxy and Factory patterns.

Make sure to create a github repository for this assignment named **IS211_Assignment8**. All development should be done in this repository.

Useful Reminders

1. Read the assignment over a few times. At least twice. It always helps to have a clear picture of the overall assignment when understanding how to build a solution.
2. Think about the problem for a while, and even try writing or drawing a solution using pencil and paper or a whiteboard.
3. Before submitting the assignment, review the "Functional Requirements" section and make sure you hit all the points. This will not guarantee a perfect score, however.

Extending our Design

For this assignment, we will expand on last week's assignment in two ways:

1. You will now be able to play against the computer, where the computer will follow some predefined strategy
2. You can now play a timed version of the game: if the game lasts longer than one minute, the winner is whoever had the most points at the time.

You will implement the computer player as an inherited class from the base `Player` class. This will help differentiate the computer player from a human one, which will be in its own class. You will also design a PlayerFactory class, that will instantiate the correct `Player` class. For the timed version of the game, you will implement this as a Proxy pattern on the `Game` class. This class will use all the same methods as the `Game` class, but will keep track of time and when time is up, determine a winner.

Playing Against the Computer

In order for the computer to make decisions in this game, the computer must have some kind of strategy. For this assignment, have the computer use the following strategy: given the computer's score `x`, the computer will hold at the lesser of 25 and 100 - `x`; otherwise, the computer will roll. This strategy should be implemented in a new class, called `ComputerPlayer`, that inherits from the `Player` base class.

To determine what kind of players to use, your program should now accept two arguments, a `--player1` and `--player2` argument, each of which can be "human" or "computer". Yes, your computer can potentially have two computer players playing against one another.

Now that you have a new class, you should design a Factory class that will instantiate either a human or computer player, depending on the input. The Game class should use this Factory when initializing the game

Timed Version of Pig

Your next task is to write a Proxy to the `Game` class, called `TimedGameProxy`, which will follow all the same exact rules of Pig as before, but will introduce a timed aspect: the game will continue until either someone scores 100, or one minute has elapsed since the start of the game. This proxy should keep track of the time the game starts, and should check that no more than one minute has gone by since then at every step. In order to activate this feature, your program should take in a `--timed` parameter.