CSCI 345–Assignment 2

Implementation

Dr. Moushumi Sharmin (lots of materials borrowed from Dr. Aran Clauson)

Due: Nov 17, 2017 (Completed Assignment)

Introduction

This part of the assignment is asking you to implement! You analyzed the problem, designed a solution, now it is time to implement the design.

Outcomes

Upon successful completion of this assignment you will

- Convert your software design into a working program
- Demonstrate how the following class relationships are realized in Java
  - Association, Aggregation, Composition, Implementation, and Inheritance
- Have a simple (text-based) implementation of DeadwoodTM

Problem Statement

Implement your DeadwoodTM design as a console application. This version of your game is indented to be playable, but may not be pretty. The point of the program is to verify that your design correctly models the game play. For example, players can only move to adjacent rooms and your program must enforce this restriction.

The program will accept a very limited vocabulary. The following table is the complete language:

Command	**Action**
who	The software identifies the current player and any parts that the player is working.
Where	The software describes the current player's room and any active scenes.
move *room*	The current player moves to the indicated room.
work *part*	The current player takes the indicated role.
upgrade $ *level*	Upgrade the current player to the indicated level
upgrade cr *level*	Upgrade the current player to the indicated level.
Rehearse	The current player rehearses
act	The current player performs in its current role.
end	End the current player's turn

Emphasized words (*italics*) denote arguments. The *room* and *part* are string arguments. *Level* denotes an integer from two to six.

The **end** command must be provided even if the current player has no legal actions.

Example Interactions:

Example 1:

If the current player is working a part, the interaction could be the following:

```
> who

red ($15, 3cr) working Crusty Prospector, "Aww, peaches!"

> where

in Train Station shooting Law and the Old West scene 20

> act

success! you got 1$

> end
```

Example 2:

If a player is not working a part. The interaction might look like the following:

```
> who

blue ($1, 5cr)

> where

Jail wrapped

> move Train Station

> where

Train Station shooting Law and the Old West scene 20

> work Talking Mule

> end
```

I'm showing additional information like the player's money and credits. This would be nice (and earn you bonus points), but is not required. Remember, the point is to test the design and to have a working model, it is not to have a friendly user interface. In addition, you may find that your design (the one you submitted) has weaknesses. This is typical of any significantly sized software project. It is difficult to anticipate every possible detail of the system. Feel free to modify your design. However, you **must update your design document (class diagram)** as you modify the design. I am going to use this document to understand the software.

Your program should accept a single parameter: the number of players. Like the rule book says, the game is designed for two to eight players. The requirement is to design the 2~3 player version.

If you design for more users, great! (but it is not required). While it does not really matter, I suggest naming the players with the dice colors: blue, cyan, green, orange, pink, red, violet, and yellow.

Deliverables

You can submit your code via canvas or give us access to your git repository. We will download your implementation from the repository, so please make sure we have full access to your repository. The file **Deadwood.java** contain should contain the main program.

Due Dates

Similar to Assignment 1, Assignment 2 also has three different due dates. There are different expectations associated with the due dates:

Due Date 1: Saturday, Nov 4 at 10:00pm

> Requirement: Skeleton of the program. You must have all major classes with the needed attributes and methods defined. Method body is not a requirement. Your program must compile.

Due Date 2: Friday, Nov 10 at 10:00pm

> Requirement: Complete code for at least two~three major classes with all the methods implemented. Your program must compile.

Due Date 3: Friday, Nov 17 at 10:00pm

> Requirement: Complete code of your program.

Grading

20pt Your software correctly plays the game of DeadwoodTM.

5pt Quality of your code (e.g., meaningful comments, well defined methods, descriptive symbols, etc.).

5pt How well your design matches your implementation. (If your implementation is very different from your design, you MUST submit a revised design and a brief description stating what changed and why)

5pt Contribution Summary (email individually)

Major Deductions

Case 1: Your program does not compile. If it does not run, we can't give you any point.

Case 2: Your program compiles but break repeatedly. Every time we try to interact with it, it breaks, we won't be able to test the correctness of the program.

Case 3: Your program runs, but does not work correctly. For example, players can move to non-adjacent rooms, players can act in roles above their rank, etc.