CSCI 345–Assignment 2

Implementation

Dr. Moushumi Sharmin and Dr. Aran Clauson

Due: May 18, 2018

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 text-based implementation of Deadwood

Problem Statement

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

Your program should accommodate the following tasks/actions:

- Identify the active player
  - Display this players information
- Location of the current players and all the other players on the board
  - Display location of all players and indicate the active player
- Moving from one location to another
  - Display the source and destination
- Working on a part
  - The active player works on a part
  - Display part-related information
- Upgrading level
  - Display current and target level
  - Support different types of upgrade
- Rehearsing
  - The active player rehearses
  - Display related information
- Acting
  - The active player acts on a role

    - Display related information
- Ending the current players turn
    - Active player can decide to end a turn
    - The **end** command is recommended even if the active player has no legal actions available

Example Interactions:

Example 1:

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

> Active player?

The active player is Jane Doe. She has $15, 3credits and 10 fames. She is working Crusty Prospector, "Aww, peaches!"

> show location of all players

The active player is in Train Station shooting Law and the Old West scene 20

Player 1 is Saloon

Player 2 is in Casting Office

> act

John Doe successfully acted and received $2

John Doe has $12, 3 credits and 4 fames.

> end turn

Jane Doe's turn ended.

Example 2:

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

> Active player?

The active player is John Doe. He has $10, 5 credits and 5 fames.

> show location of Active Player

John Doe is in Jail wrapped

> move to Train Station

Train Station shooting Law and the Old West scene 20

This is an adjacent room

Move successful!

> work Talking Mule

> end

This style of interaction is one way of showing information but this is not the only way. You are not required to display information in this manner, however, your implementation MUST support all the actions listed above. 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. We are going to use this document to understand your software design.

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).

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** should contain the main program.
- You must submit the Class Diagram that your group is implementing. Please briefly describe (1~2 paragraphs) rationale for choosing this design. Please discuss what types of Cohesion and Coupling you utilized in your design.

Due Date

May 18th at 10:00pm

Requirement:

- Class Diagram
- Complete code for your program.
- Read Me file with instructions about how to compile and run your code.

Grading

5 points Class Diagram and Rationale for choosing this design.

20 points Correctness of code (Your software correctly plays the game)

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

5 points Contribution Summary (email individually; Please list in which specific ways you and your partner contributed to the project with example. This is confidential. You should work on it 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.