Team Programing Assignments PA3

Monopoly Game

This assignment was originally developed by Dr. Sudipto Ghosh at Colorado State University and was adopted for ESOF 322.

This is an iterative programing project. There will be three deliverables scheduled at various points in the semester. The deliverables correspond to assignments PA2—PA4.

You will work on the teams that you created during the start of the semester. **Read the whole document before starting the assignment.**

---

1. Problem Statement: Monopoly Game

In this project you will develop a Monopoly Game implementation. For a quick intro to Monopoly, take a look at the article on Wikipedia.

1.1. Version 1

You will develop this version in PA2 and PA3. PA2 will involve the analysis of requirements for version 1 with the help of use case models and the creation of a class model that describes the concepts in the monopoly game. PA3 involves implementation of the requirements for version 1 with a simple Graphical User Interface.

We don't expect you to implement the whole game in this version. We do expect you to implement the following items (please refer to the "Equipment" section on Wikipedia"):

1. Deeds: You will implement the positions on the board for all of the twenty-two streets divided into eight color groups, four railroads, and two utilities, with all their ownership rules as described on wikipedia. You can create a MSU version or any fictitious version of your choice.
2. Dice: A pair of normal six-sided dice.
3. Houses and hotels: Choose simple displays (color icons will suffice).
4. Money: Standard US currency as described.
5. Tokens: Represent a player using an icon (your choice). At least two and up to four players can play.

The following items are not required.

1. While there will be positions on the monopoly board for Chance and Community Chest, you do not need to implement the corresponding cards.

Follow the rules as described for taking turns, income tax, luxury tax, Jail (except for the go-to-jail card), Landing on properties, and mortgaging properties. There is no need to handle bankruptcies, house rules, etc. For simplicity, let us assume that there will be a configurable time limit in the game. When the time is up, the game will be over. The player with the most amount of cash and properties at that time is considered to be the winner.

1.2. Version 2

In version 2, you will choose to implement any one of the following three items:

1. Implement the cards for Chance and Community Chest, including the "Go-To-Jail" and "Get Out of Jail Free" cards.
2. Implement an AI player so that humans can play with the computer.
3. Implement a distributed version of the game so that humans can play from remote sites on the same game. The server keeps track of the game state and also controls the player's turn.

---

2. Deliverables

The project will be submitted in stages. Submit an electronic copy of the deliverables at each stage. For PA2 the deliverable will be a PDF document each. For PA3, you will submit both a PDF document and code as a jar file. For PA4, you will turn in code as a single jar file.

You must embed all the images (i.e. jpg, bmp, etc.) in the PDF file as your submission. Any tool specific model files will NOT be accepted. Do not submit images as separate files. Please make sure that you include your names on the front page of all submitted documents.

PA3. Design and Implement Version 1

- Due: Tuesday, November 07, 2017, 11:59 PM via D2L assignment submission.
- 50 points.
- Tasks
    1. Use GitHub repository based version control system for all types of documents (design diagrams, tests, implementation code).
    2. Prepare sequence diagrams as part of the design process. Submit three key/difficult/interesting ones.
    3. Update the UML class diagrams developed in PA2 based on the sequence diagrams if needed.
    4. Make a list of system test cases.
    5. Write JUnit test cases.
    6. Implement the system for version 1. Use Swing or JavaFX for the user interface.
    7. Test your system.
- Submission:
    1. Submit a document containing the following:
        1. Three sequence diagrams (9 points)
        2. Class diagram (5 points)
        3. System test cases (4 points)

    4. Commit log showing checkins, checkouts, etc from your repository based version control system (3 points). This log will help us determine a number of things, such as (1) level of individual participation, (2) adequate use of a version control system. If the document shows an inadequate amount of checkin and checkout, the team may lose points. If an individual has no corresponding entry in the log, this individual may lose a lot of points for not participating in the project (i.e., the penalty is not limited to the 3 points for submitting the commit log).

2. Submit the following code in a single jar file (the test and implementation should be in separate Java packages). Note that the code must implement the design you provided above. Points will be deducted if it doesn't.
    1. JUnit test code (4 points)
    2. System implementation and document describing how to run the system (25 points). This is broken down as follows:
        - Features: 20 points
        - Design quality: 3 points
        - Programming style: 2 points