Final Project
Object-Oriented Programming and Personal Software Process
Points: 300

Description:
Choose your favorite game (such as a board game, card game, or other) and implement it in Java using Object-Oriented Programming Concepts. You will also follow the Personal Software Process - PSP1. You will work in teams of 3. Use PSP1 process script provided in Assignment 8 (Week 5). Your project submissions should be made via Blackboard before the specified deadline. You will work through this project in 4 iterations. There are deliverables specified at the end of each iteration section.

Iteration 1: Planning
Discuss with your project partners and decide on a game that you would like to implement. Understand the requirements for the game that you are building, determine the scope, and create a requirements statement. Also follow PSP1 Planning script provided in Assignment 8 (Week 5) and go through the planning process. Create all the scripts specified in the exit criteria of the PSP planning script **individually** for the tasks you are working on. Sample requirements statements are provided at the end of this document for the Checkers and TicTacToe. You are not allowed to choose Checkers and TicTacToe as the game for your project.

Due Date: Sep 24 (by 11:59pm)

Group Deliverables:
- None

Individual Deliverables:
- PSP1 Planning Scripts (all the documents listed in exit criteria of Planning phase)

Iteration 2: Design
Design the game during this iteration. Create a use case diagram and class diagram (use a software of your choice like GenMyModel, ArgoUML, Microsoft Visio, Powerpoint or any others that you might be familiar with). Also follow PSP1 Development script provided in class and go through the development process.

Due Date: October 1, 2014 (by 11:59pm)

Group Deliverables:

- UML Class Diagram

Individual Deliverables:
- None

Iteration 3: Implementation
Implement the game using object-oriented programming concepts and Java. Use a top-down approach to implementation. Break up the implementation into various modules that can be divided among your team members. Continue to follow the steps in the Development phase of PSP1 process script. Complete PSP1 scripts **individually** for the modules that your are working on.

Due Date: October 8, 2014 (by 11:59pm)

Group Deliverables:
- None

Individual Deliverables:
- PSP1 Development Scripts (documents specified in the exit criteria of Development Phase)

Iteration 4: Testing & Postmortem
Thoroughly test your software. Conduct the postmortem of the software development process. Follow PSP1 postmortem script provided in class and go through the process. Create all the scripts specified in the exit criteria of the PSP postmortem script individually.

Due Date: October 10, 2014 (by 11:59pm)

Group Deliverables:
- Thoroughly Tested Software game (as an executable jar)
- Source Program Listing (Java source code)
- Test Results (sample test screenshots)

Individual Deliverables:
- PSP1 Postmortem Scripts (documents specified in the exit criteria of Postmortem Phase)

Grading Rubric:
Iteration 1: 50 points
Iteration 2: 50 points
Iteration 3: 75 points
Iteration 4: 75 points

Peer-review: 50 points
Total: 300 points

Sample Requirements Statement

Checkers Game - Specifications:

Develop a checkers program controlled by a multithreaded server. The two users/clients should alternate making moves. Your server program should mediate the players moves, determining whose turn it is and allowing only valid moves.

Board and Pieces:

The board is a 8x8 checkered grid. The pieces are split into one darker and one lighter color. There are 2 classes of pieces: one is a regular piece and the other is a "king". Each player starts with 12 pieces placed on 3 rows as shown in the sample figures in the next page.

How to Move:

There are two ways to move a piece:

1. A simple move involves sliding a piece one space diagonally forward (also diagonally backwards in the case of kings) to an adjacent unoccupied square.
2. A jump is a move from a square diagonally adjacent to one of the opponent's pieces to an empty square on the directly opposite side, therefore "jumping over" the square containing the opponent's piece. A piece that is jumped is captured and removed from the board.

Kings: - Optional Feature

If a player's piece moves into the kings row (starting row) on the opposing player's side of the board, that piece is said to be "crowned" (or "kinged"), thereby gaining the ability to move both forwards and backwards. If a player's piece jumps into the kings row, the current move terminates; having just been crowned, the piece cannot continue on by jumping back out (as in a multiple jump), until the next move.

How the game ends:

When one of the following happens:

- A player captures all of the opposing player's pieces.
- A player leaves the other player with no legal moves.

Sample Requirements Statement

Tic-Tac-Toe Game - Specifications:

Develop a Tic-Tac-Toe game program (controlled by a multithreaded server-optional). Design the game such that the user plays with the computer or another user or both. The two users/clients should alternate making moves. In case of a multithreaded program, your server program should mediate the players moves, determining whose turn it is and allow only valid moves.

Features:

- Allow each player to enter his/her name and use the name to notify the player of his/her turn.
- Present each player with a list of symbols/graphics and allow the player to choose X or O.
- In a game against the computer, allow the human player to choose a skill level: beginner, intermediate, expert.
- Provide a graphical representation of the tic-tac-toe board.
- Allow the player to choose where to place an X or O by moving the mouse to a square and clicking in the square.
- Determine a game winner.
- Give the players an option to play again or quit.
- Track the number of wins and losses for each player and display the records at the user's request.
- *Human-Computer Interface Description:* The main interface consists of the traditional tic-tac-toe board. To place an X or O, the user moves the mouse over the desired square and clicks. The system will then fill the square with the correct symbol for that player. Beneath the board, the system notifies the player when it is his or her turn. The interface also consists of opening screens that prompt the user for game information, such as whether this game will be played against the computer, the names of the players, and the symbols the players will use to represent X or O. At the end of the game, the system uses a dialog to notify the player(s) of the winner and ask the player(s) if they wish to play another game or exit the program.