Homework 5

Due Wednesday, December 5th at 11:59 pm. No late submissions

100 Points

For this assignment we will continue to work on our ConnectX game, but we will be changing the user interface to be a graphical user interface. You are provided with the Graphical user interface, but you need to complete the controller class to make the GUI work with your model. You must follow the Model View Controller architectural pattern.

I have provided much of the code for this assignment, but you will need to add it all you your project correctly, and understand the code to use it correctly. You will also need to add in your IGameBoard interface and both implementations.

Code files:

ConnectXApp.java

This class is the entry point of the program. It calls the SetupView and SetupController class and turns the control of the program over to the event dispatch thread which will wait for an event to occur. You do not need to make any changes to this file.

SetupView.java

This class contains the code to create and layout the GUI for the setup screen. It also is the observer of the submit button. When someone clicks on submit, the action performed method is called, which then calls the controller Object.

You do not need to make any changes to this file, but it is a good example of a java swing GUI.

SetupController.java

This class is the controller for our setup screen. The processButtonClick method is called by SetupView when someone clicks on the submit button. It is passed in the rows, cols, players and the number to win by the view, but it still needs to validate that input. If there are any errors it can use the displayError method in the SetupView class to inform the player of the error, then wait for them to fix it and resubmit.

If there are no errors it will create a new IGameBoard object (the implementation will depend on the size of the game board) to serve as the model, and the ConnectXController and ConnectXView. Control is then passed over the the event dispatch thread that will wait for an event to occur

No changes need to be made to this class.

ConnectXView.java

This class is the view of our Connect X game. Our view has a message area, and a list of buttons. The buttons will be arranges in a ROWS_IN_BUTTON_PANEL x COLUMNS_IN_BUTTON_PANEL Grid. Players will use another set of buttons above the grid to select the column to place the marker in. All events will be passed to the controller by calling the processButtonClick method.

This class also provides many methods that your code will need to call to interact with the user. It is important that you fully understand this class to be able to complete this assignment.

You do not need to make any changes to this file.

ConnectXController.java

This class is the Controller for the game. It uses the ConnectXView object to interact with the user and the IGameBoard object as it's model to track what's happening in the game. This is the file you will need to complete.

You may need to add private data variables, and make changes to the constructor. You will need to complete the processButtonClick method, which handles everything about the process when a use clicks on a column to place a token, including input validation. You will need to use the methods in the ConnectXView class and the methods in the IGameBoard object to accomplish this.

A newGame method is provided to send the program back to the setup stage. You do not need to make any changes to this method, but you will need to call it. After someone wins the game, you will display to the players who won, and tell them to click any button to start a new game. If they click a button, you will then start a new game. Remember that clicking the button will call processButtonClick and return to the top of the method. The same should happen for a tie game.

While there will be some similarities to the Connect4Game class in previous submissions, there will also be some key differences. Remember that in our controller we are now waiting for events to occur and then we respond to them, while in previous assignments we had control over when the user could accomplish certain tasks.

Contracts and Comments

You should already have contracts in your IGameBoard interface and the implementations. You should make any corrections needed to those contracts. You do not need to provide contracts for the new methods in our view or controller classes as I have already included those.

You do need to comment any code that you add or edit in this program.

The Program Report

Along with your code you must submit a well formatted report with the following parts:

Requirements Analysis

Fully analyze the requirements of this program. Express all functional requirements as user stories. Remember to list all non-functional requirements of the program as well. Your user stories should not have specified using a command line interface, so they should be largely unchanged. Your non-functional requirements may be affected.

Design

You need to create a UML class diagram for the ConnectXController class, and an Activity diagram for each of it's methods. You do not need to create any diagrams for any other classes, or submit diagrams from previous homework assignments

Testing

No test cases are required for this assignment, as that was covered thoroughly for Homework 4

Deployment

You do not need to provide anything for the deployment of this program.

Additional Requirements

- Remember, this class is about more than just functioning code. Your design and quality of code is very important. Remember the best practices we are discussing in class. Your code should be easy to change and maintain. You should not be using magic numbers. You should be considering Separation of Concerns, Information Hiding, Encapsulation, and Programming to the interface when designing your code. You should also be following the idea and rules of design by contract.
- You class files should be in the cpsc2150.connectX package
- Your code should be well formatted and consistently formatted. This includes things like good variable names and consistent indenting style.
- You should not have any dead code in your program. Dead code is commented out code.
- Code that does not compile will receive a 0.
- Your UML Diagrams must be made electronically. I recommend the freely available program draw.io, but if you have another diagramming tool your prefer feel free to use that.

Tips and Reminders

- Do your design work before you write any code. It will help you work through the logic and help you avoid writing code just to delete it later.
- When writing your code, try to consider how things may change in the future. How can you design your code now to be prepared for future changes and requirements?

Submission

You will submit your program on Handin. You should have one zipped folder to submit. Inside of that folder you'll have your package directory (which will contain your code), your report (a pdf file) and your make file. Make sure your code is your java files and not your .class files.

Academic Dishonesty

This is an INDIVIDUAL assignment. You should not collaborate with others on this assignment. See the syllabus for further details on academic dishonesty.

Late Policy

Your assignment is due at 11:59 pm. Even a minute after that deadline will be considered late. Make sure you are prepared to submit earlier in the day so you are prepared to handle any last second issues with submission.

Since we are at the end of the semester and the TAs need time to be able to grade this assignment, no late submissions will be accepted for this assignment.

Checklist

Use this Checklist to ensure you are completing your assignment. Note: this list does not cover all possible reasons you could miss points, but should cover a majority of them. I am not intentionally leaving anything out, but I am constantly surprised by some of the submissions we see that are wildly off the mark. For example, I am not listing "Does my program play Connect 4?" but that does not mean that if you turn in a program that plays Checkers you can say "But it wasn't on the checklist!" The only complete list that would guarantee that no points would be lost would be an incredibly detailed and complete set of instructions that told you exactly what code to type, which wouldn't be much of an assignment.

- Can I change the size of my game board?
- Can I set the number in a row needed to win?
- Can I change the number of players in my game?
- Did I create an interface for my Game Board?
- Did I provide my interface Specification?
- Did I move my contracts from my GameBoard class to my Interface?
- Did I implement all secondary methods as default methods in my interface?
- Do I have both implementations of my game board?
- Does my game allow for players to play again?
- When players start a new game can they change the size of the board or the number to win?
- When my players start a new game, can they change the number of players?
- Does my game behave exactly the same for both implementations?
- Did I correct my contracts?
- Are my implementations efficient?
- Does my game take turns with the players?
- Does my game correctly identify wins?
- Does my game correctly identify tie games?
- Does my game run without crashing?
- Does my game validate user input?
- Did I protect my data by keeping it private, except for public static final variables?
- Did I encapsulate my data and functionality and include them in the correct classes?
- Did I follow Design By Contract?
- Did I provide contracts for my methods?
- Did I provide correspondences to tie my private data representation to my interface?
- Did I follow programming to the Interface?
- Did I comment my code?
- Did I avoid using magic numbers?
- Did I use good variable names?
- Did I follow best practices?
- Did I remove "Dead Code." Dead Code is old code that is commented out.
- Did I make any additional helper functions I created private?
- Did I use the static keyword correctly?
- Does my Game use the provided Graphical User Interface?
- Does my program follow the Model View Controller Design Pattern?
- Does my game remove old error messages after the player has corrected the issue?