Assignment 1.2 - Creating a GUI for Your Chess library

Overview

This week, we will be focusing on Model-View-Controller architecture. This means you will be implementing the graphical user interface (GUI), but GUI is only a part of MVC, so do not waste your time with small specifics. Your GUI should be clean and easy to use, but it doesn't have to be pretty and fancy.

For this assignment, you are *required* to use either Eclipse or IntelliJ IDEA. Both are free and have powerful refactoring tools available.

Eclipse vs. IntelliJ IDEA

You are likely already familiar with Eclipse from earlier programming courses here at UIUC. Some staff prefer IntelliJ. If you've never tried it out, consider using it for this project (Some people think it's better, and you can get the community edition for free).

Assignment Format

This course is likely very different previous courses you have taken, in that we typically **reuse** your code from the previous week for each assignment. As such, don't waste your time with messy code: focus on maintainability.

Read this entire page before beginning your assignment, and post on Piazza if anything is still unclear.

Part 0: Refactoring & Polishing Test Suite

You should be refactoring your code all the time. You do not need to have a big chunk of time dedicated to refactoring, refactoring should be part of your daily programming activities. You should refactor to make your code more maintainable, extensible, and understandable, as well as to incorporate any feedback from your moderator. Add tests (first) and fix them as you write your code.

Part I: Game Loop

The central component of any game, from a programming standpoint, is the game loop. The game loop allows the game to run smoothly regardless of a user's input or lack thereof. The game loop is the main loop that repeatedly gathers the user input, handles and computes them, and render to the screen. Implement the game loop for your chess game. Your custom pieces need to be included in the chess game. Some resources you may find useful are:

- Java Gaming - Understanding the Basic Concepts
- Understanding the Game Loop - Basix

- Java Game Programming

Required Features

As always, you will be graded on participation and implementing the guidelines for modular, readable code. To receive full requirements for the assignment, your GUI must fulfill several requirements:

- Start/restart/forfeit a game
- A player is allowed to forfeit a game at any point in time
- If both players agree, they can restart a game and the score will be tied
- Each player should be given a unique name
- Scores are recorded if a pair of players play several games
- Move a piece
- A player can only move their piece, not the opponent's
- Illegal moves should not be allowed, and a player does not lose his/her turn for attempting an illegal move
- Check, checkmate, illegal moves should provide a visual feedback to players
- Turns must alternate

Part II: MVC with GUI

For this part of the assignment we will be building a Graphical User Interface (GUI) for the chess libraries we have developed. Again, your custom pieces should be included in the chess game, e.g. you may give players options to play either the classic game of chess or some pieces replaced by your custom pieces. As a method of creating this GUI, we'll be turning to the model-view-controller (MVC) software architecture, and we suggest using Java's Swing to implement your GUI. (Alternatives like JavaFX may also be good resources.)

In an MVC architecture there are three components that comprise all the pieces of a GUI:

- The Model is used to hold data related to an application. In the context of this assignment, this might be the maze representation(s), state of the solution, state of the GUI, etc.

- The View component contains what the user will see. A view would have all buttons, labels, frames, pictures, and other graphical elements.

- The Controller handles communication between the user and the application. It will contain the functions to handle button clicks, for example.

The following link is an excellent description of the MVC design paradigm: http://www.cs.rice.edu/~cork/newBook/node89.html. I found excellent examples by using the following search string in Google: "mvc gui example". Of particular use was this from Pace: http://csis.pace.edu/~bergin/mvc/mvcgui.html

Part III: Undo

For this part of the assignment, we will implement the undo function that

- allows a player to undo their last move
- returns turn to the player who performed the undo

Implementation detail is left up to you - you may allow players to undo whenever, or only before the opponent makes a move, or only when the opponent allows for undo. Whichever option you choose to implement, it should be clearly documented and tested with your unit tests. Implementation of the undo action itself should use the Command Pattern. From wikipedia, "*If all user actions in a program are implemented as command objects, the program can keep a stack of the most recently executed commands. When the user wants to undo a command, the program simply pops the most recent command object and executes its undo() method.*"

Some resources you might find useful:

- Add an undo/redo function to your Java apps with Swing
- Let your players undo their in-game mistakes with the command pattern

Part IV: Manual Test Plan

Now that you have extended your once-static GUI, there should be more you need to test. Build on your manual test plan from last week to test more GUI functionality. Anything you can test with automated JUnit tests, you should write JUnit tests. Include screenshots and specific test steps for human testers to follow and observe.

Need help?

First, ask questions on Piazza. If you have a question, there is a pretty good chance someone else has the some one and an even better chance that someone else in the class or one of the TAs will be able to answer it for you. If you are still having a problem, email your moderator or one of the TAs to get advice. Remember, its best to ask questions early on so they have time to be answered. Don't wait until the last second to get started then realize that you are confused. In general, we are flexible with interpretations of the assignment, **as long as it does not trivialize any component of the assignment**.

Submission

This assignment is due at the **beginning of your discussion section the week of October 2nd, 2017**. Please be sure to submit in SVN, and ask your moderator or TA before the deadline if you have any questions. Also, make sure to place your work in a folder called **Assignment1.2**, matching this spelling exactly.

For example, you should be able to bring up https://subversion.ews.illinois.edu/svn/sp17-cs242/NETID/Assignment1.2/ in a web browser and see all of your code (if you replace NETID with your actual netid)

Objectives

- Clean up any problems in your code for Assignment 1.1, expand your test suite if necessary
- Implement the game loop
- Create a GUI interface for your Chess library

Resources

A collection of all resources mentioned in this writeup can be found here:

Java GUI (Swing)

- Crash Course in Java GUI
- Sun JAVA GUI Tutorial
- Java GUI Examples

MVC

- MVC Wikipedia Article
- Model/View/Controller GUI

Game Loop

- Java Gaming - Understanding the Basic Concepts
- Understanding the Game Loop - Basix
- Java Game Programming

Grading

We will bias clarity over cleverness in evaluating your code. You should adopt a *"teaching"* mindset, always asking yourself, *"How would I change what I have written to make it as easy as possible for someone else to understand my code without my direct assistance?"*

Refer to the standard Grading rubric if any portion of this rubric is unclear. _ Note that the standard rubric assumes a scale of 0-2 for each category_.

Category	Weight	Scoring	Notes
Basic Preparation	2	0-1	Ready to go at the start of section
Cleverness	2	0-2	The hardest points on the rubric
Code Submission	4	0-2	Submitted correct content on time and to the correct location in the repository
Decomposition	4	0-2	Project is adequately decomposed to different classes and methods
Documentation	4	0-2	Comments for each class and each function are clear and are following style guides
Effort	2	0-2	Perform considerable amount of work
Naming	2	0-2	Variable names and method names are readable and are following Java conventions
Overall Design	4	0-2	Have nice approaches and structures in overall
Participation	5	0-2.5	Interact with the group 2 times (ask a question, make a comment, help answer a question, etc.)

Category	Weight	Scoring	Notes
Presentation	4	0-2	Present the code clearly
Requirements - Game Loop	5	0-2.5	2 Points: Game loop runs without any errors and all features work including Start/restart/forfeit a game & players are able to move pieces in turns. 2.5 Points: Game loop includes additional features. Left to the discretion of the moderator.
Requirements -GUI Moving Pieces and Game Control	5	0-2.5	2 Points: The is a clear functional organization of the MVC as described in the assignment site. 2.5 Points: MVC organization has been implemented to perfection. Additional features such as highlighting of player’ s chosen piece, list of possible moves when a piece is selected shown on UI and aextremelyneat & professional UI with no glitches.
Requirements - Refactoring	5	0-2.5	2 Points: There is not much duplicate/redundant code and student has put effort to refactor code from last week. There is minor but not significant scope for improvement. 2.5 Points: Every piece of code is functional and overall design of the project is so well designed that itcannotbe refactored/improved any further.
Requirements - Undo	5	0-2.5	2 Points: Undo button is fully functional. Done by pushing entire game state into stack. (Not the most efficient way! Memory intensive.) 2.5 Points: Multiple undo steps are implemented. Implemented in a more efficient way such as pushing moves and killed pieces into stack.
Testing - Unit Tests for New Code	5	0-2.5	2 Points:80% of the new code is tested and the tests cover edge cases for each function. 2.5 Points:>= 95% of the new code has been tested

Category	Weight	Scoring	Notes
			and every possible edge caseshasbeen tested for.
Testing - Manual Test Plan	5	0-2.5	2 Points:Tests for new UI features added. Tests cover all new features added as part of Assignment 1.2. 2.5 Points:Every possible feature in the UI is tested for. (Manual Test Plan is more than 20 pages long).
Total	63		