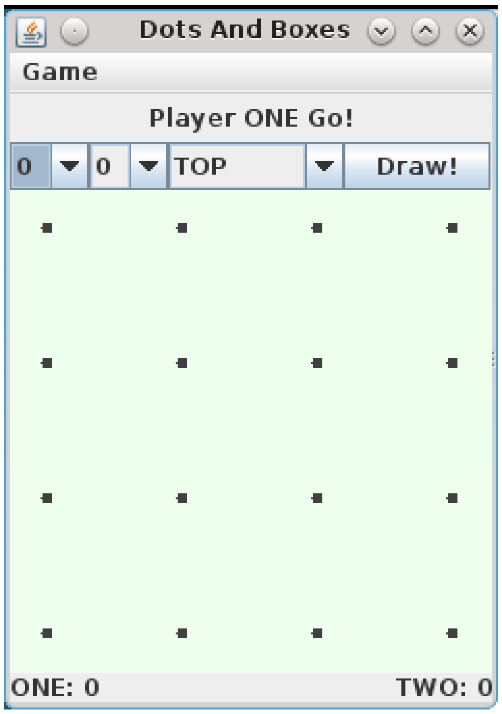
Project 6 - DAB GUI

Due Monday by 12pm **Points** 80

Overview

Your assignment is to create an application that provides a Swing-based GUI front-end to the Dots and Boxes implementation from Project 4. The interface should look something like this, although the exact layout is up to you:



The graphical rendering of the game grid is handled by a custom Swing component that you'll download below. You'll need to create all of the other interface elements, design a layout that's responsive to resizing, and ensure all your components work properly with the custom component and the underlying DABGame implementation.

Details

Your implementation must be named `edu.vt.cs5044.DABPanel` and your JUnit test suite must be named `edu.vt.cs5044.DABPanelTest`. The system relies upon your Project 4 DABGame implementation as the fundamental engine for this system, but see the next section for an alternative.

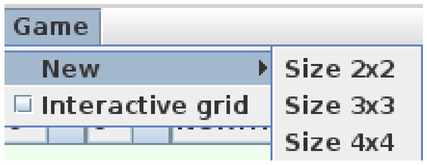
Reference Implementation

It is fully recognized that some of you may have outstanding bugs and/or missing features in your Project 4 DABGame implementation. That's not any problem at all. Everyone is free to download a fully operational implementation of Project 4 (as a JAR with class files only; no source code) that can be integrated within this project.

Graphical Interface Description

When your implementation is launched, it should begin with a newly-initialized 3x3 game displayed, along with a menu bar, status indicator components, and input components. The layout does NOT need to exactly match the sample above, but it should still appear at least reasonably neat and professional in nature. It should also be reasonably responsive to resizing, such that the interface neatly expands or contracts to fill the available space given to its window.

The menu bar will contain one "Game" menu with a sub-menu that must look exactly like this (there is some very small degree of flexibility in the test cases, but not much):



The menu includes items to initialize a 2x2, 3x3, and 4x4 grid, as well as an item to toggle on/off interactive mode (see the "DABGrid" section below). The status indicators must include the score for each player, along with a display of which player has the current turn (or a "game over" message, when applicable). The status indicators must always reflect the current state of the game. The input components must include drop-down boxes to specify x and y parts of a coordinate (with only valid options shown, meaning 0, 1, and 2 for a 3x3 grid), along with a direction drop-down, plus a button to attempt to draw the specified edge when clicked. The button should be disabled (grayed-out) when the game is over, but enabled at all other times.

DABGrid

DABGrid is the custom component responsible for drawing the grid. You must integrate this component within your overall layout. By default, this component simply displays the current state of the grid, by calling DABGame accessor methods. When this component is set to interactive mode, the grid will become interactive, responding to mouse movements and clicks. Hovering the mouse near an un-drawn edge will highlight the edge. Clicking a highlighted edge will cause this component to call the DABGame drawEdge() mutator to draw the edge, then notify your system via a callback that the status indicators must be refreshed. Interactive mode must be activated and deactivated by your code, in response to the menu system.

Testing

Testing of GUI applications is notoriously complicated. Because the Swing system actually takes primary control of application, we can't just call the application code directly as in our previous systems. In order for test code to interact with the application, we first need to be sure that each component has a name assigned. The test code must search through the component hierarchy looking for these names, in order to fetch references to the named components.

Once you have references to the Swing components, your test code needs to ask Swing to schedule your method calls. At that point you may query the state of the the GUI directly with accessors, and make assertions as normal.

This can all be done via standard JUnit test cases, but there a few additional issues that we won't have time to cover in detail. Thus a *complete* JUnit test suite is provided for download below. This is exactly the same test suite used by Web-CAT for this assignment. Assuming you don't have any redundant conditionals or other such problems in your code, this test suite should also achieve 100% coverage of your code.

Getting started with Eclipse

Configuring the Build Path

Create a new project in Eclipse and create the `edu.vt.cs5044` package within the src folder, and another new source folder called test. Right-click the project and select Build Path | Configure Build Path, then select the Libraries tab.

All of the following should be added to the *Classpath* section:

First, click "Add Library" then select JUnit | JUnit 4 and click Finish.

Next, use "Add External JARs" to navigate to and select the `dab5044.jar` library you downloaded during Project 4 (available below). Expand the library node, select Javadocs, and click Edit to verify and add the `dab5044-api.jar` file (also available below) from Project 4 as the Javadocs.

Next, you need to download the `dabgui.jar` and `dabgui-api.jar` files (both available below) and repeat the process above to add them to your project libraries as well.

Finally, you need to choose a Project 4 (DotsAndBoxes) implementation to use with Project 6. You have two options, and you can quite easily switch between them at any time:

Option 1 - Reference implementation (recommended)
To use the reference implementation, just download the `dabgame-ref.jar` file (available below) and save it somewhere convenient.

Option 2 - Your own implementation (advanced)
If Web-CAT assigned you 30 "correctness" points in Project 4, you should be able to use your own implementation with no problems. Be sure to open your Project 4 project in Eclipse, then select File | Export from the menu. Select the Java | "JAR file" wizard and click Next. Uncheck everything at first, then in the main top area expand your Project 4 project and check only its "src" folder (which causes the project to show a dash in the checkbox). Below that, check "Export generated class files and resources" and uncheck all the other checkboxes and options. Click the Browse button, navigate to a convenient location, and choose any reasonable file name that ends with the `.jar` extension. Click Finish to create and save the Jar file.

Once you've completed at least one of the options above, or if you later just want to switch between the two options, right-click your Project 6 project, click Build Path | Configure Build Path, then select the Libaries tab.

If you already have one of the implementations above listed in the libraries area, select it and cilck Remove. To add an implementation, use "Add External JARs" in the *Classpath* section, then navigate to the Jar file of your choice, select it, and click OK. You do not need to add Javadocs to this file.

Starter code and test code

Download the `DABPanel.java` and `DABPanelTest.java` files anywhere convenient that is outside of your project folder. Open the folder in your operating system, then drag these files to the `edu.vt.cs5044` package of the src and test source folders respectively. Be sure to select "Copy" when Eclipse asks how to import these files. If you configured the build path properly, both files should compile with no errors (although there are lots of Checkstyle issues with the starter code).

Please carefully review both the starter file and the test file before you begin. Note that you don't need to add to the test file, nor alter it in any way. The comments are there to help you understand how it works. The starter code is also commented to help guide your implementation approach. There's a "TODO" comment in each location where you must develop code.

Downloads

dabgui.jar - The DABGrid custom component, plus a utility class for component names
dabgui-api.jar - The API for the DABGrid component and utility class
dabgame-ref.jar - Reference implementation of the DotsAndBoxes interface (completed Project 4)
DABPanel.java - Starter code for your implementation
DABPanelTest.java - Complete JUnit test code for your implementation
dab5044.jar - The library from Project 4
dab5044-api.jar - The API for the library from Project 4

Submit to Web-CAT

Submit your solution to **Web-CAT** **(https://web-cat.cs.vt.edu/)** via Eclipse. You may resubmit your code as many times as you like (before the deadline) in order to improve your score.

Important Notes:

- See the **project grading rubric** for full details on the grading criteria applied to this assignment.