Lab Exercise 11
CS 2334

November 5, 2015

Introduction

In this lab, you will extend your knowledge of creating graphics in Java. Specifically, you will experiment with using **KeyListeners** and **KeyEvents** to construct graphics programs that react to keyboard button presses that are made by a user.

The game that you are completing requires the player to move through a shifting maze by pressing the right and left keys arrow keys. The player must stay within the unoccupied area of the screen. If the player is caught by a moving wall, then the player loses the game.

Learning Objectives

By the end of this laboratory exercise, you should be able to:

1. Create event-driven graphics
2. Use a **KeyListener** to update graphics based on **KeyEvents**
3. Read existing code and documentation in order to complete an implementation

Proper Academic Conduct

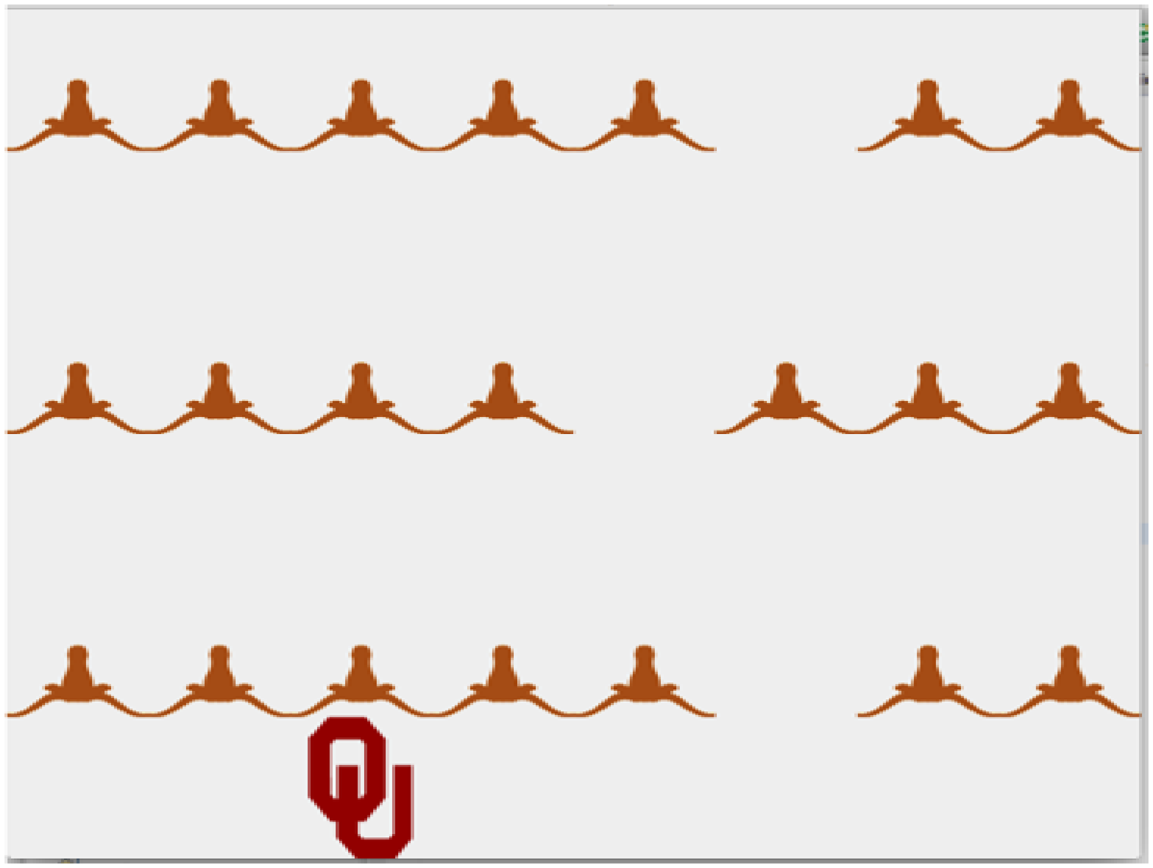
This lab is to be done individually. Do not look at or discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.

Preparation

1. Import the existing lab11 implementation into your eclipse workspace.
   (a) Download the lab11 implementation:
   `http://www.cs.ou.edu/~fagg/classes/cs2334/labs/lab11/lab11-initial.zip`
   (b) In Eclipse, select *File/Import*
   (c) Select *General/Existing projects into workspace*. Click *Next*
   (d) Select *Select archive file*. Browse to the lab11-initial.zip file. Click *Finish*

Game: OU vs UT

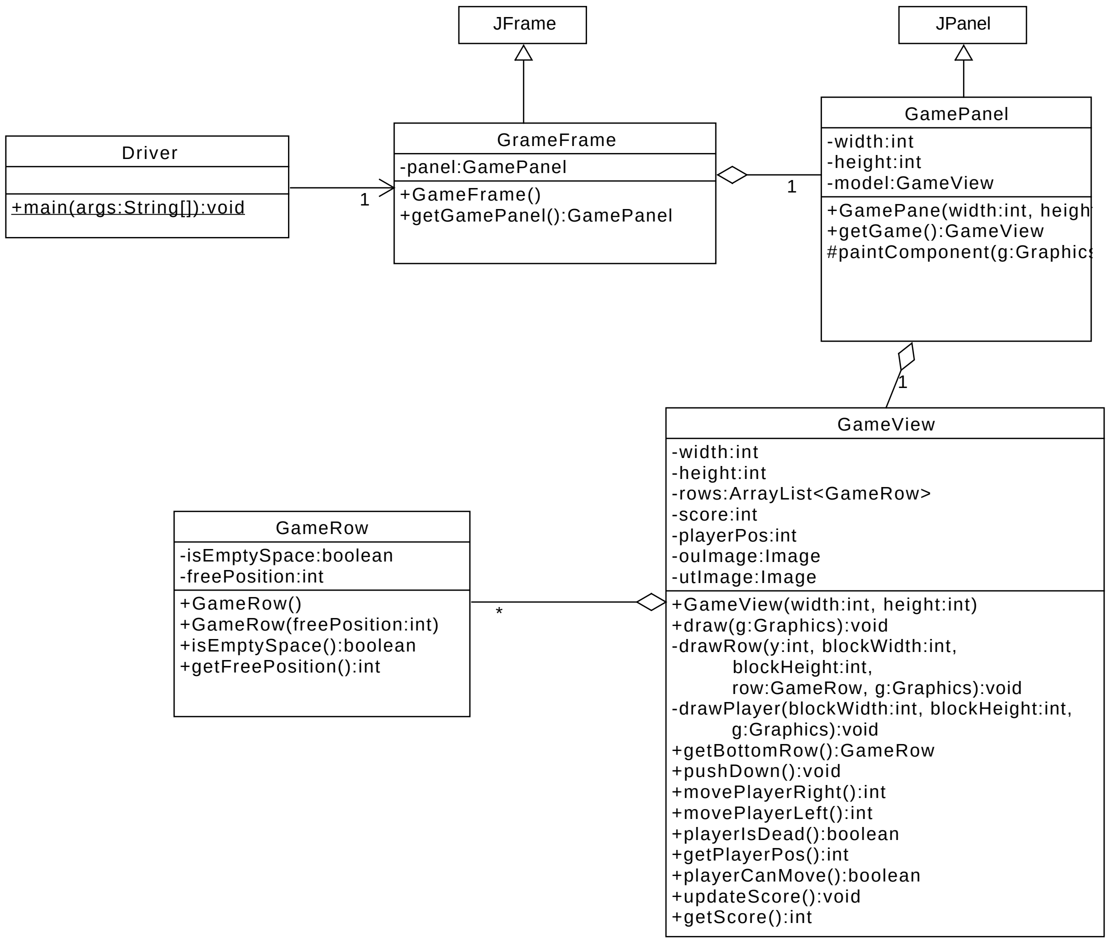
Below is an image of the graphical user interface for the game that we are creating.



The OU image represents the player. The rows of longhorns are walls through which the player cannot pass. These rows shift downwards at regular intervals. The goal is to move the player into a free position before the wall overtakes the player. This will be done by moving the player right or left using the right and left keys on the keyboard.

Note: the player can wrap around the window to get to the other side.

UML



Lab 11: Specific Instructions

All of the classes shown in the UML are provided in lab11-initial.zip.

1. Most of the classes are implemented. We want you to implement the graphics, not the logic of the game. However, you need to analyze and understand the game logic to implement the graphics.

- **Driver**, **GamePanel** and **GameRow** have been fully implemented. Read and understand these classes before moving on.
- Implement a **KeyListener** in **GameFrame**
  - When you implement the **KeyListener**, Java will require you to create handler methods for three events types. Since you only need one event type, it is okay to leave the other two methods with no body.
  - Alternatively, you may implement a **KeyAdapter**, for which you only need to override the one method of interest.
- Complete the implementation of **GameView**
  - Most of the logic occurs in this class. Fully analyze and understand the code before moving on.

2. Do not add functionality to the classes beyond what has been specified

3. Don't forget to document as you go!

Final Steps

1. Generate Javadoc using Eclipse.
   - Select *Project/Generate Javadoc...*
   - Make sure that your project is selected, as well as all of the Java source files
   - Select *Private* visibility
   - Use the default destination folder
   - Click *Finish*

2. Open the *lab11/doc/index.html* file using your favorite web browser or Eclipse (double clicking in the package explorer will open the web page). Check to make sure that that all of your classes are listed and that all of your documented methods have the necessary documentation.

3. If you complete the above instructions during lab, you may have your implementation checked by one of the TAs.

Submission Instructions

- All required components (source code and compiled documentation) are due at 11:59pm on Friday, November 6th.
- Prepare your submission file:
  1. Select the project in the *Package Explorer* window.
  2. Right-click. Select *Export*
  3. Select *General/Archive File*
  4. Expand your project and verify that both the *src* and *doc* folders are selected, as well as all of their contents
  5. Enter the archive file name: *lab11.zip* (note that you may want to browse to a different destination folder)
  6. Select *Save in zip format*
  7. Click *Finish*
- Submit your zip file to the lab11 folder on D2L.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 10 points

(10) The program is properly formatted (including indentation, curly brace and semicolon locations).

(5) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 15 points

(15) The program is using proper data types and method calls.

(10) There is one error in data type or method call selection.

(5) There are two errors in data type or method call selection.

(0) There are multiple errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented.

(7) A component of one required method is missing.

(4) A method is not implemented or components are missing from two methods.

(0) Two or more required methods are not implemented or components from three or more methods are missing.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.

(10) The program fails one test.

(5) The program fails two tests.

(0) The program fails three or more tests.

Execution: 15 points

(15) The program executes with no errors.

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements.

(0) The method documentation is missing more than two of the required documentation elements.

Inline Documentation: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used and has the correct contents.

(5) The correct zip file name is used, but one required component is missing.

(0) An incorrect zip file name is used or more than one required component is missing.