Lab 10

Due 8 May

Handout 13
CSCI 136: Spring 2017
1 May

Hex-a-Pawn

This week you may again choose to work with a partner. If do you work with a partner, you must both attend the same lab section(s) and together submit one shared assignment.

1 Pre-lab

Before arriving to your scheduled lab, prepare a design document for `GameTree`. It will not be collected, but **it will be worth 2 points of your lab grade**. For those who choose to work with a partner, each partner must prepare their own design document; you should collaborate on a final design after the lab has started.

We will post an example design document in the handouts section of the course webpage.

2 Lab Program

This week's lab uses trees to play Hex-a-Pawn, a small chess-like game. The lab is outlined in **Section 12.11** of your textbook. You will build a tree that starts from the initial game state and representing all possible states that the game board can be in. You will then implement several different players that consult this tree to make moves as they play Hex-a-Pawn. The players include: 1) a human player that asks the user for moves to make; 2) a random player that picks possible moves at random; and 3) a computer player that improves its strategy by learning from past mistakes. In the end, you will be able to run the different players against each other.

2.1 Details

- There are three starter files available from the handouts page: `HexBoard.java`, `HexMove.java`, and `Player.java`. A copy also appears in the starter directory:

  ```
  cp /opt/mac-cs-local/share/cs136/labs/hexapawn/* .
  ```

  Familiarize yourself with these classes. They are described briefly in the book, and the javadoc documentation is also available on the class handouts page.

- A significant part of your task is to design and implement the `GameTree` class. This is a tree structure with potentially many children instead of just two. Think about the methods you will need in this class and how you can represent the structure as you construct your design document.

- You can play a game of Hex-a-Pawn by running `java HexBoard` after compiling the starter files.

- To use your three `Player` classes as described in the book, implement the Hex-a-Pawn game as the main method of a class `HexaPawn.java`. This program should take four command-line arguments (i.e., specified in args[]). The first two specify the number of rows and columns the board should have. The third and fourth should be "human", "comp", or "random" to indicate what type of player should be used for the white and black pieces, respectively. Remember that you can convert a String to an integer with the `Integer.parseInt(str)` method.

2.2 Deliverables

Create and lab folder in the usual way, named `<unix>-lab10` (or `<unix1>-<unix2>-lab10` if working with a partner). Include the following items:

- Your well-documented source code for all Java files that you write.
- Answers to thought questions 1 and 2 from the book (pg 314). When working on the thought problems, your program may run out of memory. You may increase the size of the heap for your program by running it as, for example:

```
java -Xmx500m HexaPawn 3 3 comp human
```

  The option "`-Xmx500m`" indicates that your program is allowed to allocate up to 500 MB of memory using the `new` keyword.

As in all labs, you will be graded on design, documentation, style, and correctness. Be sure to document your program with appropriate comments, including a general description at the top of the file, a description of each method with pre- and post-conditions where appropriate. Also use comments and descriptive variable names to clarify sections of the code which may not be clear to someone trying to understand it.

Have fun!