Lab 8

Due 11:59pm, 18 April

Handout 10
CSCI 136: Spring, 2005
11 April

Hex-a-Pawn

1 Short Answers

Complete the following questions and turn them in at the start of lab:

1. Bailey, 10.6
2. Bailey, 10.12
3. Under what circumstances would a `get` method as described in question 10.13 be more useful than the existing `contains` method? Write a `get` method for `OrderedVector`.
4. The `OrderedList` implementation described in the text has methods that include the line

   ```
   Comparable cValue = (Comparable)value;
   ```

   Are these casts necessary? Do they restrict the usage of the structure in any way?

Also bring a written design to lab this week (see below).

2 Lab Program

This week's lab uses trees to play Hex-a-Pawn, a small chess-like game. You will build a tree 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: a human player that asks the user for moves to make; a random player that picks possible moves at random; and 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.

The lab is outlined in Section 11.11 of *Bailey*.

2.1 Details

- There are three starter files available from the handouts page: `HexBoard.java`, `HexMove.java`, and `Player.java`. 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. **Bring a handwritten design of `GameTree` to lab.**
- 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. 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. You can convert a String to an integer with the `Integer.parseInt(str)` method.

- Gardner's original paper describing this game is under `labs/hexapawn` in the CS136 course directory area on cortland. Take a look, if you're interested. Notice the ad for a "comparator" on the third page.

2.2 Deliverables

Create and submit a `tar` file containing the following:

- Your well-documented source code for all Java files that you write. You do not need to submit source code for starter files, as you should not need to modify them.
- A `README` file that contains your answers to thought questions 1 and 2.

To create a `tar` file, use the "`tar`" command to archive the full contents of a directory into a single file. For example, the command `tar -cf lab8.tar lab8dir` creates a file called `lab8.tar` containing the full contents of the directory `lab8dir`. You can then run `turnin -c 136 lab8.tar` to submit the tar file.