Introduction to Comp. Sci., Homework 9

Due at 10pm on Wednesday, May 1

Readings from Liang

Read chapter 14, "Exception Handling and Text I/O."

Optional readings and exercises from HFJ

Read chapter 11 of Head First Java, "Risky Behavior," and do the exercises.

FileFinder.patternMatches

Create a class, `hw9.FileFinder`. Write a static method `patternMatches` that takes a pattern and a `String` to match and returns true if the string matches the pattern. Its signature is:

```
public static boolean patternMatches(String pattern, String s);
```

A `*` at the beginning or end of a pattern should match any sequence of zero or more characters. Some examples:

```
FileFinder.patternMatches("foobar", "foobar")  // true
FileFinder.patternMatches("*bar", "foobar")    // true
FileFinder.patternMatches("foo*", "foobar")    // true
FileFinder.patternMatches("*oob*", "foobar")   // true
FileFinder.patternMatches("foo", "foobar")     // false
FileFinder.patternMatches("*foo*", "foobar")   // true
FileFinder.patternMatches("foobar", "foo*")    // false
FileFinder.patternMatches("*", "")             // true
FileFinder.patternMatches("**", "")            // true
FileFinder.patternMatches("***", "")           // false
```

FileFinder.findMatches

In this section, you'll build a tool to search for files by name in the current directory, similar to the Unix `find` command. Add a static method to `hw9.FileFinder`, `findMatches`, that returns a list of `File` objects representing the files and directories in a specified directory whose filename parts match a specified pattern. `File.getName` can be used to extract the filename part of a full pathname, ignoring the directory part.

`findMatches` should be declared like this:

```
public static ArrayList<File> findMatches(File directory, String pattern)
    throws FileNotFoundException;
```

`findMatches` should throw a `FileNotFoundException` if the `directory` argument is nonexistent, not a directory, or not readable.

For example, if the `foo` directory contains files `a.txt`, `b.txt`, `a.doc`, and `b.doc`, then this call:

```
ArrayList<File> matched =
    hw9.FileFinder.findMatches(new File("foo"), "a.*");
```

should return a list with elements equal to `new File("foo/a.txt")` and `new File("foo/a.doc")`.

main

Write a `main` method for `FileFinder`. The program should take two command-line arguments. The first argument (`args[0]` in `main`) is the path to a directory in which to search. The second (`args[1]` in `main`) is a pattern to match filenames against. `main` should display a useful error message and return if there are not exactly two arguments or if the first argument is not a readable directory; this will require catching exceptions thrown by `findMatches`. If there are no errors, `main` should print the full pathname of every file matching the specified pattern in the specified directory or its subdirectories.

For example, this command:

```
java hw9.FileFinder hw9 "*.java"
```

should generate output like this:

```
hw9/FileFinder.java
```

No solution to this problem (nor any other Java code you are likely to ever write) should have the text "`catch (Exception ...)`."

As usual, there are automated tests, this time called `Hw9Test`.