CSCI 1913: Introduction to Algorithms, Data Structures, and Program Development
Computer Laboratory 5
October 11–12, 2016

0. Introduction.

Most modern operating systems, like Unix and GNU/Linux, store files in *directories*. They can have directories nested inside those directories, and still more directories nested inside those, etc. *Files* containing data can also reside in any directory, no matter how deeply it is nested. Also, files can have optional *types* that tell what kind of data they contain.

A *Pathname* gives a sequence of directory names that tells how to find a file in a nested series of directories. In this laboratory assignment, you will implement a Java class that represents a pathname. Among other things, your class will have a more interesting `equals` method than was discussed in the lectures.

1. Theory.

In Unix-like operating systems, a pathname starts with '`/`', followed by one or more directory names, separated by '`/`'s. The pathname ends with the name of a file. Following that, there can optionally be a '`.`', followed by the name of a type that tells what the file contains. Each pathname gives instructions about how to find a file in a series of nested directories. For example, the pathname `/home/jim/csci1913/final.txt` gives these instructions.

1. Start at a special root directory `/`.
2. In the root directory is another directory, `home`. Go inside it.
3. In `home` is yet another directory, `jim`. Go inside it.
4. In `jim` is still another directory, `csci1913`. Go inside it.
5. In `csci1913` is a file named `final`.
6. The file `final` has type `txt`, so it contains text.

Pathnames in a real operating system can have many directories, but the ones you will implement here can have at most 10.

2. Implementation.

In this laboratory assignment, you will write a Java class called `Pathname` that represents a pathname. Your class must have the following private variables, so its first few lines must look like this. To simplify grading, you must use the same names for these private variables that are shown here. (The three dots mean that some parts of the class are not shown.)

```
class Pathname
{
  private int       depth;
  private String [] directories;
  private String    name;
  private String    type;

  ⋮
}
```

The variable `directories` points to an array of 10 strings, the names of the directories. The variable `depth` is the number of directories in the pathname. The variable `name` is the file's name. The variable `type` is the file's type, or the empty string `""` if the file has no type.

Your class `Pathname` must also have the following public methods. To simplify grading, you must use the same names for methods and their parameters as are shown here. You can use any names you want for local variables within methods.

`public Pathname(String name)`

(5 points.) Constructor. Initialize `depth` to `0`, `directories` to an array of 10 strings, and `type` to the empty string. Initialize the variable `name` to the parameter `name`. Hint: you may need to use `this`.

`public Pathname(String name, String type)`

(5 points.) Constructor. Like the previous constructor, but you must also initialize the variable `type` to the parameter `type`. A class can have two constructors, or two methods, with the same name if they have different parameters. Hint: you may need to use `this`.

`public void addDirectory(String directory)`

(5 points.) If `depth` is greater than 10, then do nothing: there is no room for more directories. Otherwise, add `directory` to the end of the array `directories`, and increment `depth`. Hint: `depth` tells you where to add `directory` in the array.

`public boolean equals(Object object)`

Test if `object` is equal to this pathname, using the techniques discussed in the lectures. Return `true` if it is, and return `false` otherwise. Hint: you may need a loop. You may also need the operator `&&`, which means *and*.

`public String toString()`

Convert `directories`, `name`, and `type` to a string. See the example driver program below to find out how to do that. Return the string. Hint: you may need a loop.

If *a* and *b* are pointers to class instances, then you can test if they are not equal by writing `!` *a*`.equals(`*b*`)`, because the `!` operator means *not*. Never write *a*`.equals(`*b*`) == false`: you will lose points if you do that.

You should also write a driver class to test your `Pathname` class, even though you will not get points for it. Here's what a driver class might look like. The comments show what should be printed if everything works correctly. Note that if you print a pathname, then Java automatically calls its `toString` method.

```
class Pathfinder
{
  public static void main(String [] args)
  {
    Pathname p0 = new Pathname("coffee", "java");
    p0.addDirectory("home");
    p0.addDirectory("Desktop");
    p0.addDirectory("labs");
    System.out.println(p0);  //  Prints /home/Desktop/labs/coffee.java

    Pathname p1 = new Pathame("cola");
    p1.addDirectory("home");
    p1.addDirectory("hax");
    System.out.println(p1);  //  Prints /home/hax/cola
```

```
        Pathname p2 = new Pathname("tea");
        System.out.println(p2);  //  Prints /tea

        System.out.println(p0.equals(p0));                //  Prints true
        System.out.println(p0.equals(p1));                //  Prints false
        System.out.println(p1.equals(p2));                //  Prints false
        System.out.println(p0.equals("Not a pathname"));  //  Prints false
    }
}
```

This is not necessarily a complete set of tests! For best results, you should write your own driver class, with your own `main` method, and your own tests.

3. Deliverables.

This lab assignment is due October 18–19, 2016. Please submit only one file, containing the Java Source code for your `Pathname` class and your driver class. If you have output, such as test cases produced by your main method, then that should appear in a comment at the bottom of your file. If you do not know how to submit your work, then please ask your lab TA.