CSE 8B Spring 2022

Assignment 8

Recursion

Due Date: Wednesday, Jun 1, 11:59 PM

Hi again! Be sure to start this assignment as EARLY as possible! You got this!

Learning goals:

- Apply knowledge of recursive programming to enhance features of the Virtual File System.

NOTE: This programming assignment must be done individually. Paired programming is NOT allowed for this assignment.

Coding Style (10 points)

For this programming assignment, we will be enforcing the CSE 8B Coding Style Guidelines. These guidelines can also be found on Canvas. Please ensure to have COMPLETE file headers, class headers, and method headers, to use descriptive variable names and proper indentation, and to avoid using magic numbers.

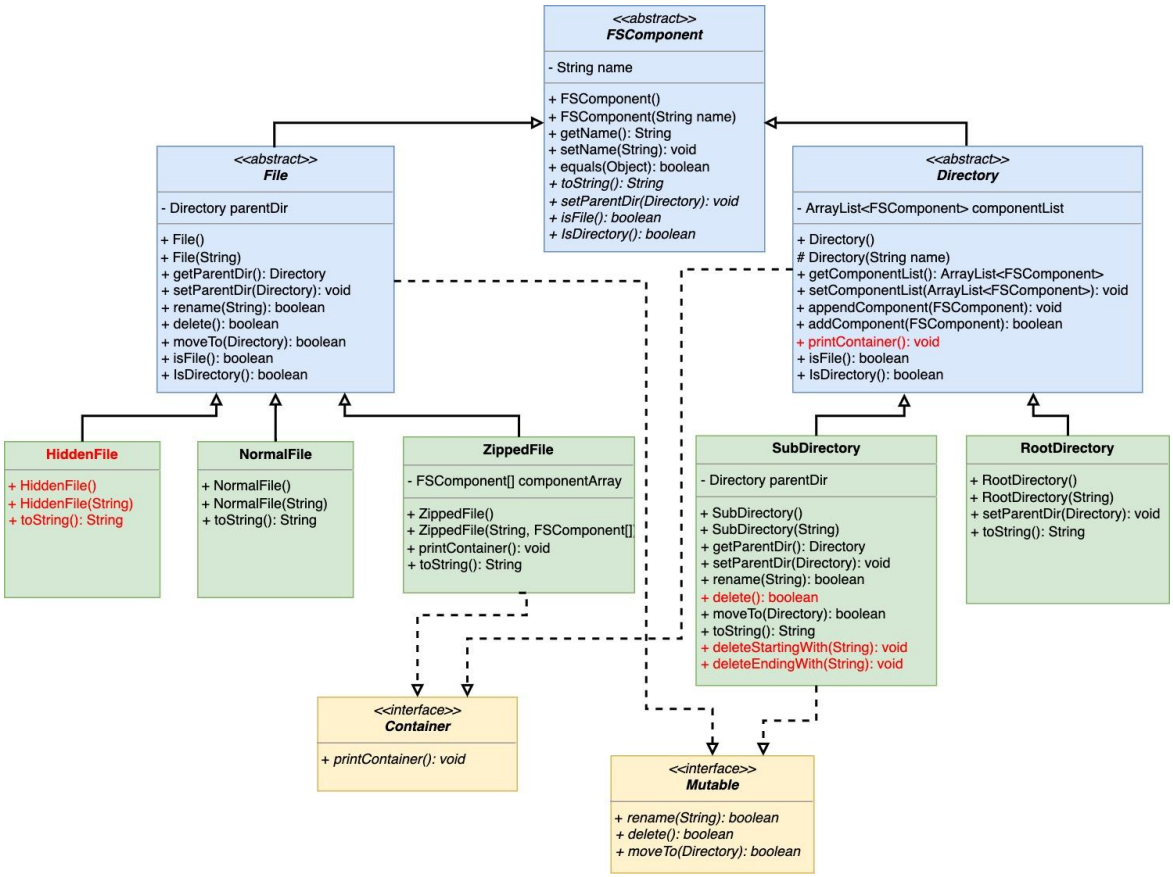
Part 0: Getting started (0 points)

This assignment does NOT have any starter code. Instead, it starts with your completed code from Assignment 7. However, you must create an `Assignment8.java` file to replace the `Assignment7.java` file used in Assignment 7 (Hint: copy the contents of `Assignment7.java` to `Assignment8.java` and replace all **Assignment7** with **Assignment8**). Similar to Assignment 7, `Assignment8.java` will contain your testers.

Part 1: Overview

Since Assignment 8 is an extension of Assignment 7, we will be reusing most of the UML (Unified Modeling Language) diagram for Assignment 7. The image below is the UML diagram for Assignment 8, showing the relationships between different classes. If the image looks blurry in the write-up, then open PA8_UML.pdf on Piazza.

There are some methods that need to be implemented or revised by you. Those methods are colored in red.



Part 2: `HiddenFile.java` (10 points)

You will have to create this file from scratch. Ensure the full file name (including the file extension) is `HiddenFile.java`.

Like its sibling `NormalFile`, the `HiddenFile` concrete class extends `File`. The functionality of this class will be covered in later parts. For now, add and implement the following methods:

1. `public HiddenFile()`
   This is the default no-arg constructor. You do not need to initialize anything in this constructor.

2. `public HiddenFile(String name)`
   Implement this constructor by initializing the `name` instance variables in its parent class.

3. `public String toString()`
   This method should return the string representation of the `HiddenFile` object. **To ensure full compatibility with the Gradescope Autograder, you should return the following string EXACTLY:**

   ```
   "Hidden file: " + this.getName();
   ```

   Don't forget the "`@Override`" annotation!!

Part 2: `SubDirectory.java` (50 points)

In Assignment 7, you implemented two constructors; setter and getter methods for the `parentDir` member variable; and `rename`, `delete`, `moveTo` and `toString` methods. All of those will not be modified for Assignment 8 **except** the `delete` method**. You will also additionally implement 2 new methods** - `deleteStartingWith` and `deleteEndingWith` methods.
Here is what you will do:

1. `public boolean delete()`

   Unlike the `delete()` method you did for Assignment 7, this time the method should return `false` if there exists any `HiddenFile` object under the current directory, or under subdirectories, sub-subdirectories, etc. That means for the figure below, you cannot

delete any of the subdirectories. You may need to write a ***recursive helper method*** to check the existence of `HiddenFile` instances**.**
If no `HiddenFile` instance is found, then implement ***two-way binding*** by removing the `SubDirectory` from its parent and set the parent of the current `SubDirectory` to `null`. **You can always assume that the parent exists.**

```
Root Directory: HOME
    Sub Directory: folder1
        Sub Directory: folder2
            Sub Directory: folder3
                Hidden file: .config
```

Note: It is mandatory to use recursion for this method.

Hint: All you need to do is add a check for hidden file existence using your recursive helper before the code you wrote for Assignment 7.

2. `public void deleteStartingWith(String prefix)`

   This is a new method you need to add and implement in Assignment 8. This method deletes all the files (only files, NOT directories) whose `name` starts with the `prefix`. For this, you may need to write a ***recursive helper method*** to recursively check the files under the current directory, or under subdirectories, sub-subdirectories, etc. You can use the `delete()` method in `File` class to delete the files or directories in the recursive calls. Do NOT delete `HiddenFile` instances even if their file name qualifies the `prefix` condition. **The only files you delete are those that start with the `prefix` and are of type `NormalFile` or `ZippedFile`**.
   Note: It is mandatory to use recursion for this method.

3. `public void deleteEndingWith(String suffix)`

   This is a new method you need to add and implement in Assignment 8. This method deletes all the files (only files, NOT directories) whose `name` ends with the `suffix`. For

this, you may need to write a ***recursive helper method*** to recursively check the files under the current directory, or under subdirectories, sub-subdirectories, etc. You can use the `delete()` method in `File` class to delete the files or directories in the recursive calls. Do NOT delete `HiddenFile` instances even if their file name qualifies the `suffix` condition. **The only files you would delete are those that end with the `suffix` and are of type `NormalFile` or `ZippedFile`**. Since `ZippedFile`s end with ".zip", remember to check the name of the file after stripping or ignoring the ".zip" extension from the file name. For example, if the zipped file name is "HowToMakeAPudding.zip", and the suffix is "ing", then you will have to delete this file. Similarly, for other `NormalFile`s that end with other extensions (eg: .png, .mp3), strip the extension from the file name and check the name of the file. In other words, look for the `'.'` character to strip the file name accordingly.

Hint: use substring() to strip the file extensions if any.

Note: It is mandatory to use recursion for this method.

Part 3: `Directory.java` (20 points)

The only method you need to revise in this part is the `printContainer()` method.

1. `public void printContainer()`

   This method will print out the all files and directories under the current directory hierarchically. This includes not only all elements in the `componentList`, but also all components in any subdirectories, sub-subdirectories, etc., if there are any. The printing format is shown below. Whenever digging into a deeper level subdirectory, add a tab to the front of the elements that need to be printed (**Hint**: the `toString` method of all concrete classes is already provided to you). The components of the directory must be printed in the linear order of the ArrayList (from the first element to the last element).

```
Root Directory: HOME
```

```
Normal file: cat.png
Sub Directory: secret
    Hidden file: .cse8b_final_exam
    Sub Directory: music
        Normal file: rice.mp3
    Sub Directory: lecture
        Zipped file: Lectures1to5.zip
```

You must use recursion to implement this method. Creating a recursive helper method is HIGHLY recommended. Use `print` or `println` to output contents; use "\t" for encoding a tab character.

Part 4: `Assignment8.java` (10 points)

You will have to create `Assignment8.java` in this assignment. (Hint: copy the contents of `Assignment7.java` to `Assignment8.java` and replace all **Assignment7** with **Assignment8**)
As always, this file contains all of your testers, a method `unitTests()` to run all of your testers, and the main method to run `unitTests()`. The `unitTests()` method should **return true** only when all the test cases are passed. Otherwise, you should **return false**.
To get full credit, **create at least 6 tester methods in Assignment8.java** testing the new methods that you have added as part of Assignment8. In other words, we expect to see a **total of at least 6 tester methods** being called by `unitTests()`. You will not receive credit for this section if you include tests from Assignment7. We also suggest making some print messages in each of your test cases so that you will know which test case is failing.

IMPORTANT NOTE: The Gradescope Autograder will be reading the output from `printContainer()` to ensure correctness, so make sure that you **do NOT** leave any other, unnecessary print statements inside the method `printContainer()`. Otherwise, it is OK if your unit tests print to standard output.

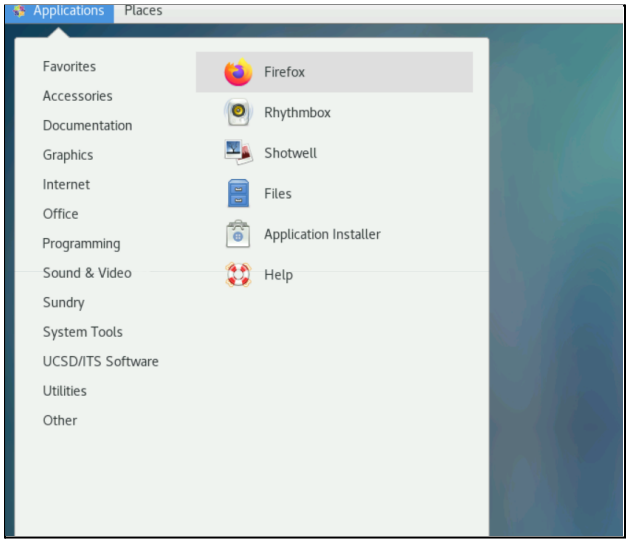
Remember that it is OK to have magic numbers in your unit tests.
You can compile and run your unit tests from `main()` using the following commands: (Make sure you are in the correct directory, else navigate to the starter code using `cd`).

```
> javac *.java
> java Assignment8
```

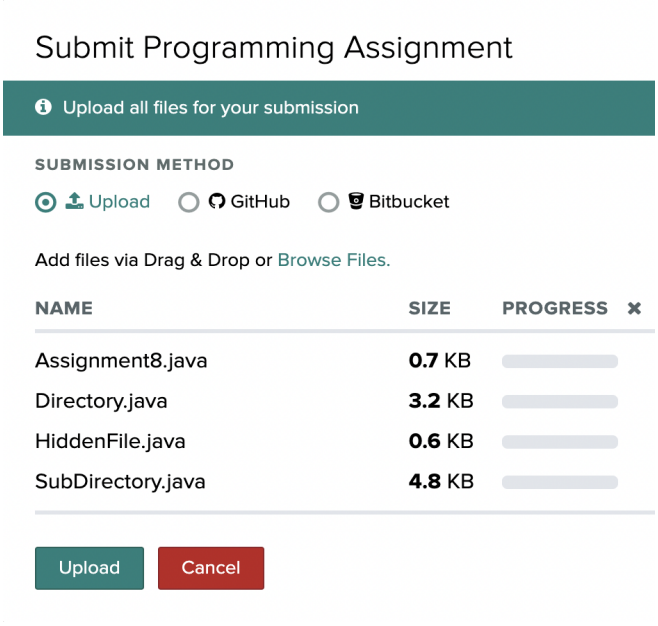
Submission

You're almost there! Please follow the instructions below carefully and use the **exact submission format**. Because we will use scripts to grade, **you may receive a zero** if you do not follow the same submission format.

1. Open a web browser on the Remote Desktop. You can open Firefox by going to Applications → Firefox.



2. Open Gradescope in Firefox and login. Then, select this course → PA8.
3. Click the DRAG & DROP section and directly select the **FOUR** required files: (`Assignment8.java, Directory.java, HiddenFile.java, and SubDirectory.java`). Drag & drop is fine. Please make sure you don't submit a zip, just the separate files in one Gradescope submission. Make sure the names of the files are correct.
4. You can resubmit unlimited times before the due date. Your score will depend on your final (most recent) submission, even if your former submissions have higher scores.
5. Your submission should look like the below screenshot. If you have any questions, feel free to post on Piazza!



Q&A

What if I did not get full points on the delete() method from SubDirectory.java in PA7?
You might want to fix the bug from your `delete` method of `SubDirectory.java`, because the same bug can happen to your `delete()` method in this programming assignment.

What if I did not get full points on the files that are not required to submit? Will I be penalized on PA8?
You will not get penalized for the mistake on those files that are not required to submit. We will use our solution code for `FSComponent.java`, `File.java`, `NormalFile.java`, `ZippedFile.java`, `RootDirectory.java`, `Container.java` and `Mutable.java`. This is also the reason why we do not ask you to submit those files on Gradescope. So you can assume for the methods from PA7, the solution code follows the writeup correctly. However, you need to understand those methods from PA7 in order to do PA8.

Can I get a copy of the solution code for PA7?
We cannot release the solution code for PA7. If you need help on your PA8 code, please visit any lab hours or set up appointments with any tutors/TAs.