AP(IT), Assessed Exercise 1

Jan 2022

Description

- **Title**: Composite Pattern File System
- **Deadline**: 4:30pm on Friday February 11th 2022
- **Contribution to final course mark**: 25%
- **Solo or Group**: Solo work
- **Anticipated Hours**: 3

Specifcation

You are to build a simplified file management system using the composite design pattern. Within your system, you can store files and directories. Directories can include other directories and/or files. All files and directories are within one top directory. The file system needs to be able to do the following:

- Add a file/directory to a given directory, using method `void add(Component)`.
- Remove a file/directory from a given directory, using method `void remove(Component)`.
- Display a directory and all of its contents: name and size of each of its contents, whether file or directory.
- Get the size of a directory as the total size of the files within it.
- Search for a directory containing a file with a certain name. If more than one is found, only the first directory is to be returned. File name is to be matched exactly and is case-sensitive. You can only search for file names and not directory names.

You are **required** to use the following interface:

```
public interface Component {
    public String getName();
    public int getSize();
    public int getCount();
    public String display(String prefix);
    public Component search(String name);
}
```

Example:

```
root: (count=5, size=886)
    Settings (10)
    pictures: (count=1, size=120)
        portrait (120)
    music: (count=3, size=756)
        jazz: (count=2, size=335)
            Kind of Blue (201)
            Giant Steps (134)
        classical: (count=1, size=421)
            Beethoven, Symphony no 6 (421)
```

In the above example, there are 5 directories (`root`, `pictures`, `music`, `jazz` and `classical`) and various files. *Please carefully note the format of the output*: the filesize in brackets next to each file, the directory summary in count of files and total size, and that directory contents are indented with a tab (“`\t`”). Note that directories do *not* count towards the count or size; only files do.

Using the previous example, the following search parameters and their results are given:

- `search("Giant Steps")` -> return: reference to the `jazz` directory
- `search("pictures")` -> return: null reference
- `search("portrait")` -> return: reference to the `pictures` directory
- `search("Take Five")` -> return: null reference

Tasks

- Start by understanding the given interface. You will need to implement the interface where appropriate or you will be deducted marks.
- Sketch out the components (hint: two concrete classes) that you will need to build.
- Create the components. Remember to ensure that directories should be able to include other directories.
- Create a driver class: i.e. another class with a main method to test your system. This class will *not* be marked. We provide an example of this for your own testing purposes, but a different driver with unseen input will be used for marking.
- Make sure that the output from your `display` method matches the specification above and similar to the given example. Otherwise, the automated marker might not identify your output as correct. Note: the marker will identify changes in whitespace, but will deduct marks accordingly.
- Add all your classes to a package called `file1234567`, where 1234567 is your student ID. Failure to do this will prevent you from scoring full marks.
- Submit your classes to Moodle as a compressed ZIP file. We expect at least 3 files within the ZIP file, one of which is the given interface **unedited**. You do not have to submit your test driver class.

Marking Scheme

Your submission will be marked on a 100 point scale, broken down as indicated below. Note that there is substantial emphasis on working submissions, as reflected by the large fraction of the points reserved for this aspect. It is to your advantage to ensure that whatever you submit compiles and runs correctly.

- 25: Compilation
  - 5: Submitted files do not compile but due to a simple error
  - 10: Submitted files compile successfully on their own
  - 10: Submitted files compile successfully with unseen test drivers (i.e. interface and public methods match the specification)
- 40: Functionality (using unseen input)
  - 10: Add files/directories
  - 10: Remove files/directories
  - 10: Correct display of file system
  - 10: Search for directory containing a filename
- 15: Definitions (use of given interface)
  - 10: Implementation of interface
  - 5: Encapsulation
- 20: Design pattern
  - 20: Designing the appropriate concrete components