Bangladesh University of Engineering and Technology
Department of Computer Science and Engineering
CSE308: Software Engineering Sessional
July 2023 Semester

Assignment-3 on Structural Design Patterns

Deadline : 06 January 2024, 11:55 PM

Problem - I [10 marks]

In a spaceship, roaming from one galaxy to another, there are two types of passengers onboard. Some of these passengers are called Crewmates and the other passengers are called Imposters.

- Crewmates are the individuals who study the interstellar objects and sometimes do basic maintenance tasks of the spaceship.
- On the other hand, imposters are actually space monsters in disguise who are attempting to sabotage this voyage by poisoning the crewmates and damaging the spaceship.

Recently, some of the crewmates have started to notice strange behavior from some of the other passengers (who are imposters, actually) while doing the maintenance tasks together. To avoid suspicion, imposters have developed a device that helps them damage the spaceship without looking **suspicious** while doing normal work.

Now, implement the above scenario by writing the necessary classes and using an *appropriate design pattern*.

You have to show the above scenario in one of your implemented classes by creating objects and calling methods.

Take input from the user. A sample input has been attached:

INPUT	OUTPUT
login crew1	Welcome Crewmate!
repair	Repairing the spaceship.
work	Doing research
logout	Bye Bye crewmate.
login imp1	Welcome Crewmate! We won't tell anyone; you are an imposter.
repair	Repairing the spaceship. Damaging the spaceship.
work	Doing research. Trying to kill a crewmate. Successfully killed a crewmate.
logout	Bye Bye crewmate. See you again Comrade Imposter.

Hint:

- Crewmates have `repair()` and `work()` functions. Suppose you don't have access to Crewmates source code.
- Imposters have the same functions, but the functions also perform damaging actions.
- Don't think much about the login and killing parts.

Problem - II [10 marks]

You have to implement a hierarchical file system (similar to Linux). In a hierarchical file system, there are different drives, folders, and files. A drive can contain both folders and files. Under a folder, there can be folders and files. Any folder can also be empty. There are common properties:

- name
- size
- type
- directory (path starting from drive)
- component_count
- creation_time

Let's have a sample hierarchy:

```
C:\
—- Music
    —-mp3
        —--rainbow.mp3
        —--moon.mp3
    —-mp4
        —--moonlit.mp4
        —--shunshine.mp4
   —- dream.flv
   —- sing.mkv
```

Required functionalities:

1. **Changing Directory**: user command 'cd <name>'
   Change the current directory to 'name'.
   If the name is a file, show an error.
   If the folder or drive doesn't exist in the current directory, it will show an error.
2. **Details**: user command 'ls <name>'
   Lists the details of the file, folder, or drive with 'name'.
   For example, 'ls Music' would print:

   ```
   Name: Music
   Type: Folder
   Size: 50 kB
   Directory: "C:\Music"
   Component Count: 4
   Creation time: 13 December, 2023 5:12 PM
   ```

3. **Listing**: user command 'list'
   Lists all the files and folders under the current directory.
   For example, 'list' in mp4 folder would print:

```
Moonlit.mp4        34 kB        13/12/2023 17:12:45
shunshine.mp4      50 kB        10/12/20223 4:23:45
```

4. **Delete:** user command 'delete <name>'
   If it is a file, delete it.
   If it is a folder or drive, delete it only if it is empty.
5. **Recursive Delete**: user command delete -r <name>
   If it is a file, delete it with a warning.
   If it is a folder, first delete all its child folders and files.
6. **Jump to root:** user command 'cd ~'
   Set the current directory to root (at the top of all drives).
7. **Makedir**: user command 'mkdir <name>'
   Create a folder in the current directory named <name>.
   Check if you are under any drive.
8. **Touch**: user command 'touch <names> <size>'
   Create a file with <size> kB.
   Check if you are under any drive or folder.
9. **MakeDrive**: user command 'mkdrive <name>'
   Create a drive with <name>

Now, implement the above scenario by writing the necessary classes and using an *appropriate design pattern*.

Take input from the user. A sample input has been attached:

```
mkdrive C
cd C:\
mkdir Music
cd Music
mkdir mp3
touch rainbow.mp3
list
cd ~
cd C:\
ls Music
delete Music
delete -r Music
```

Special Instructions

- Please first make a UML diagram of the classes.
- You are encouraged to discuss the design with your peers.
- You should implement the code in Java.
- Don’t copy solutions from anywhere.

Submission Guidelines

- Create a folder that is named after your 7-digit student ID.
- Place all the essential files (with .java extension) inside the folder, and then zip that folder.
- Submit the zipped file in Moodle.