CSSE 374: Exam 1

Policy

This is an open-book, open-note, open-moodle, take-home exam. You are also allowed to refer to any documents on the internet provided that you are not communicating or collaborating with another individual. Asking questions in online-forums, newsgroups, or via chat applications and emails are also forms of collaboration and are not allowed in this exam.

This part of the exam contributes 80% to your Exam 1 grade and 8% to your overall course grade. The exam is due by 5:10 pm, Friday, 12/19/2014 on Moodle.

Problem

We want to build a **G**eneral **P**urpose **F**ramework for **C**ommand-**L**ine **A**pplications (**GPFCLA**) as a part of this exam. Here is the specification for the framework:

The framework allows registering any number of extensions. An extension is a module (a class or a set of classes) that performs user-defined actions. Some command-line extensions to this framework could be as simple as echoing input in uppercase, concatenating supplied strings, coping supplied file to another destination, and so on.

After the framework is configured with such extensions, it should be able to delegate a user typed command to the right extension for further processing. Here is how the framework should work:

The framework will read a line from the command prompt. A line is expected to have the following format:

```
GPFCLA>[extension-name] [command] [arg0, arg1, …, argn]

where, [extension-name] is the registered unique extension name,

[command] represents one of the supported operations in the registered extension, and

[arg0, arg1, …, argn] represents an unbounded argument list and can be empty as well.
```

Here is an example command-line to echo input in uppercase:

```
GPFCLA>echo uppercase Hello, World

Executing App Extension: echo
---------------------------
HELLO WORLD
---------------------------
```

Here, **extension-name** = echo, **command** = uppercase, **arg0**=Hello, and **arg1**=World. When the first line is read by the framework, it should delegate further processing to the registered echo extension. The echo extension then sees the uppercase command and thus, prints the final output in uppercase.

Here is the concatenation example:

```
GPFCLA>StringOp concat Hello, Beautiful, World!

Executing App Extension: StringOp
--------------------------
Hello Beautiful World!
--------------------------
```

The framework should display an error message for a line that is not formatted correctly. It should also report user with an error message if an extension is supplied with an unknown command. Also, user should be able to exit the framework by typing "**quit**" in the command prompt. The partial code supplied to you handles the parsing logic and creates a **CmdModel** object with the parsed data for further use.

The framework should support any number of extensions. However, as a proof of concept, please implement just the aforementioned two extensions with the support for the following commands:

Extension: echo **Command**: uppercase
Behavior: Should display the arguments in uppercase as shown in the previous page.

Extension: StringOp **Command**: concat
Behavior: Should concatenate the arguments but separating them with spaces as shown above.

Extension: StringOp **Command**: find
Behavior: Should show the starting index of the first occurrence of arg0 in arg1, otherwise -1. This command only takes two arguments. You can use String's **arg1.indexOf(arg0)** method to get the index for this command.
Sample Usage:

```
GPFCLA>StringOp find key, My key is lost!

Executing App Extension: StringOp
--------------------------
Found at index: 3
--------------------------
```

```
GPFCLA>StringOp find key, My book is lost!

Executing App Extension: StringOp
--------------------------
Found at index: -1
--------------------------
```

It is critical that each extension should further allow addition of any number of commands in the future. So, in summary, **we want a framework that allows any number of extensions and each extension should itself allow any number of commands at a low/no maintenance overhead**.

Design

Create **Exam1/docs/Answer.pdf** with answers to the following problems:

Q1. Create a UML Class Diagram to present your design idea and explain it in a few lines. [**20 points**]

Implementation

Q2. Implement your solution in the **Exam1/src/problem** package. [**50 points**]

Testing

Q3: Implement necessary test cases in the **Exam1/test/problem** package that tests all of the important aspects of the framework and extensions. [**10 points**]

Deliverables

Bundle your project in the **zip** format [**not rar**] and turn it in on Moodle. Please check your **zip** file to see if it **includes** the **docs** folder before turning it in.