Concordia University
Department of Computer Science and Software Engineering

SOEN341 Fall 2020 - Project A - Part 2

Purpose

The main goal of part 2 of this project is divided in twofold:

1. make a good analysis and design of all command-line programs by extracting a common architecture for them:
   (a) design an administrator class to be responsible of parsing all options and arguments and making them available as inputs to the executable core of a command-line program;
   (b) be able to get the process's result of the executable core to be printed to the standard output.
2. develop a small object-oriented framework (called wcOO) by decentralizing the `wc` C command-line program into a small of hierarchy of counters. Classes such as `CharCounter`, `LineCounter`, and `WordCounter` designed in order to independent of the framework. In other words, these classes should be configurable and are not part of the framework. They could be replaced by new ones, such as a `WhitespaceCounter`, `KeywordCounter`, and so on.

Requirements

In the part 2 of this project, the overall main design requirement is:

- to use interfaces at the root of all abstractions: command-line program, administrator, option, configurator, counter, and so on.
- to integrate three basic design patterns: strategy, template method, and factory.

All command-line programs (copy, charcount, linecount, wordcount, and wc) must support the following common options: help, verbose, and banner.

The notation below defines a command-line syntax for part 2:

```
CommandLine   = CommandName { Option } { Argument }

Option        = HelpOption | VerboseOption | BannerOption .

HelpOption    = "-?" | "-h" | "-help"
VerboseOption = "-v" | "-verbose"
BannerOption  = "-b" | "-banner"
```

An option has a short form, a long form, and a description. Each option is disabled by default. An option that is explicitly specified on the command line becomes automatically enabled.

There is no checking on duplication in specifying the option more than once, it is its last appearance that is retained in the option's cache.
where:

Short version	Long version	Meaning
`-h` or `-?`	`-help`	Print the usage of the program
`-v`	`-verbose`	Verbose during the execution of the program
`-b`	`-banner`	Print the banner of the program

Table 1: All common options

When the verbose option is enabled, the execution of:

- `copy` prints a dot (`.`) for every character copied from the source to the destination file.
- `wc` prints `c` for every character found, `w` for every word found, and `l` for every line found.

Here is an example of a typical usage of the banner option: The following command-lines are examples of using the above C programs:

```
C:\> copy -banner
Copy Version 1.42b
Copyright (C) ABC Inc 2020. All Rights Reserved.
Written by John Smith

C:\>
```

Submission of Sprint 2 in Project A

For this sprint, **you must work individually. Only electronic submissions will be accepted.** Zip together the Java source codes only. Naming convention for zip file is:

```
ProjectA_Sprint2_StudentID_StudentName.zip
```

Example:

```
ProjectA_Sprint2_12345678_JohnSmith.zip
```

Following your submission, a demo is required. The marker will inform you about demo times. Please notice that failing to demo your sprint will result in zero mark regardless of your submission.

Deadlines and Grading Scheme

Due dates for each Sprint:

- **Week 3**: Sprint 2 - **Due 11:59 PM - Monday, September 28st, 2020**
- **Week 4**: Sprint 3 - **Due 11:59 PM - Monday, October 5th, 2020**

The first two sprints are 3% as individual marks. The remaining 4% for Sprint 3 is team marks. There is 1% for review contribution and peer-evaluation (more details in Week 4).