Exercise: Dependency Inversion Principle - Java

Exercise - Spelling Checker (20 minutes)

The accompanying zip file contains the code for a simple spelling checker program. The program accepts a single command-line argument, which is a URL that references a document on the web that is to be spell checked. (You can run it against https://pastebin.com/raw/t6AZ5kx3 to prove that it works before and after your modifications) The program downloads the document from the web, parses the words from the document, runs the words through a dictionary to determine which words are misspelled, and returns a data structure containing all misspelled words and how many times each one appeared in the document.

The core functionality of the program is contained in the SpellingChecker class. This class violates the Dependency Inversion Principle, because it has hard-coded dependencies on several lower-level concrete classes that it uses to do its work. Specifically, SpellingChecker uses the URLFetcher class to download the document from the web, the WordExtractor class to parse words from the document, and the Dictionary class to load a dictionary text file and determine if words are in the dictionary. This implementation has several disadvantages:

1. Any changes to the method interfaces of the URLFetcher, WordExtractor, or Dictionary classes will require changes to the SpellingChecker class.
2. SpellingChecker is hard-coded to obtain the document being spell-checked from the web. Documents can come from many places other than the web. Why should SpellingChecker be limited to only web documents? It shouldn't.
3. SpellingChecker is hard-coded to parse only plain-text documents. Different file formats will require different parsing algorithms. Why should SpellingChecker be limited to only plain-text documents? It shouldn't.
4. SpellingChecker is hard-coded to use a dictionary that is stored in a text file. There are other ways to store a dictionary. Why should SpellingChecker be limited to only dictionaries stored in a text-file? It shouldn't.

Refactor this program by applying the Dependency Inversion Principle to invert the dependencies of the SpellingChecker class to remove the disadvantages listed above.

Submit your code on Canvas.

Exercise - Flight Monitor (20 minutes)

In a previous exercise you modified the Flight Monitor program to implement the Observer pattern.

1. Explain how the Observer pattern is an example of the Dependency Inversion Principle.
2. The core functionality of the Flight Monitor program is implemented in the FlightFeed class, which monitors the status of a selected flight and notifies its observers when the flight's status changes. In addition to monitoring flight status, the FlightFeed class also contains the code for retrieving flight information from the Open Sky web service. This is a violation of the Single Responsibility Principle, because retrieving flight information is a separate responsibility from monitoring flight status. Refactor the code to separate the flight status retrieval responsibility into another class, but use Dependency Inversion so that FlightFeed does not have a type dependency on the new class.

Submit your work on Canvas.