Department of Computer Science
University of Pretoria

Programming Languages
COS 333

Practical 7:

Object-Oriented Programming

October 20, 2022

1 Objectives

This practical aims to achieve the following general learning objectives:

- To gain and consolidate some experience writing object-oriented programs in Ruby;
- To consolidate a variety of basic concepts related to object-oriented programming languages, as presented in the prescribed textbook for this course.

2 Plagiarism Policy

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.ais.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the `Library` quick link, and then click the `Plagiarism` link). If you have any form of question regarding this, please consult the lecturer, to avoid any misunderstanding. Also note that the principle of code re-use does not mean that you should copy and adapt code to suit your solution. Note that all assignments submitted for this module implicitly agree to this plagiarism policy, and declare that the submitted work is the student's own work. Assignments will be submitted to a variety of plagiarism checks. Any typed assignment may be checked using the Turnitin system. After plagiarism checking, assignments will not be permanently stored on the Turnitin database.

3 Submission Instructions

Upload your practical-related source code file to the appropriate assignment upload slot on the ClickUP course page. You must implement and submit your entire Ruby program in a single source file named `s99999999.rb`, where `99999999` is your student number. Multiple uploads are allowed, but only the last one will be marked. The submission deadline is **Monday, 31 October 2022, at 23:00**.

4 Background Information

For this practical, you will be writing a program in Ruby 2.5. The course website contains documentation related to Ruby [1]. You will be implementing all classes in a single source file.

5 Practical Task

This practical requires you to build a simple system that represents shops and stock items. The following details must be included in your implementations:

A `StockItem` has a name and a price. Each `StockItem` should provide at least the following behaviour:

- An accessor method that sets the stock item's name to a provided name.
- An accessor method that retrieves the stock item's name.
- An accessor method that sets the stock items's price to a provided floating point number.
- An accessor method that retrieves the stock item's price, which does nothing (it will be overridden in subclasses of `StockItem`). The `StockItem` class would therefore be an abstract class in a language like C++. Ruby, however, does not support abstract classes, so you should raise an exception if the stock item's price is changed.

Ruby provides a shorthand mechanism for allowing access to member variables, called attribute readers and writers. You should use these wherever possible. Refer to the Ruby documentation for information on attribute readers and writers.

Write a `Shop` class. A `Shop` object is composed of an array of `StockItem` objects. Additionally, the `Shop` class should provide at least the following behaviour:

- `addStockItem(newStockItem, aName)`, which adds a new `StockItem` object (denoted `newStockItem`) to the shop, and sets its name to `aName`.
- `setPrice(aName, aPrice)`, which sets the price of the `StockItem` object with the name `aName` to `aPrice`.
- `getAveragePrice()`, which returns the average price of all the `StockItem` objects stored in the `Shop`.
- `print()`, which prints out, for each of the `StockItem` objects stored in the `Shop`, the item's name and price.

Write two new classes, each of which is a subclass of `StockItem`. These classes should be called `NormalPriceStockItem` and `DiscountedStockItem`. In the case of a `NormalPriceStockItem`, the price is simply returned by the appropriate accessor method. In the case of a `DiscountedStockItem`, the price is reduced by 10%. Override the accessor method for the stock item's price to accomplish this.

Be sure to utilise as much functionality from the parent class as possible, and do not re-implement functionality that exists in parent class.

Write program code that creates a shop and allows the user to type in the name (as a string), price (as a floating point number), and type (indicated by an "n" for a `NormalPriceStockItem` or a "d" for a `DiscountedStockItem`) for three stock items. It should then print out all the stock items using the `print()` method, and print the average price of the three stock items rounded to two digits after the decial point. Use the functionality in the `Shop` class to achieve this.

An example of the type of input and output for this program is as follows:

```
Enter name for item 1: Apple
Enter price for item 1: 2.5
Enter type of item 1: n

Enter name for item 1: Pear
Enter price for item 1: 3.0
Enter type of item 1: d

Enter name for item 1: Carrots
Enter price for item 1: 1.5
Enter type of item 1: n

Items:
Apple: 2.5
Pear: 3.3
Carrots: 1.5

Average price: 2.43
```

6 Marking

Both the implementation and the correct execution of the program will be taken into account during marking. Your program code will be marked offline after the submission deadline. Make sure that you upload your complete program code to the appropriate assignment upload slot, and include comments at the top of each program, explaining how your program should be compiled and executed, as well as any special requirements that your program has.

References

[1] Huw Collingbourne. The book of Ruby. 2009.