CS1027 - Computer Science Fundamentals II
Summer 2019

Assignment 1

Learning Outcomes

By completing this assignment, you will gain skills in:
Using strings and reading text files in Java,
Writing some Java classes,
Using Java arrays and
Testing your code.

In this assignment, you will create a Java program that has two classes GroceryList and GroceryItemOrder. GroceryList class represents a list of items to buy from the grocery store, and another class named GroceryItemOrder represents a request to purchase a particular item in a given quantity (example: four boxes of cookies). The GroceryList class should use an array field to store the grocery items and to keep track of its size (number of items in the list so far).

You are required to implement two classes: GroceryList and GroceryItemOrder stored them in files GroceryList.java and GroceryItemOrder.java, respectively. Assume that a grocery list will have no more than ten items.

The GroceryItemOrder class

The GroceryItemOrder class should store an item quantity and a price per unit. A GroceryItemOrder object should have the following public methods:

- A constructor method, GroceryItemOrder that constructs an item order to purchase the item with the given name, in the given quantity, which costs the given price per unit.
- An accessor method, getCost, returns the total cost of this item in its given quantity. For example, four boxes of cookies that cost 2.30 per unit have a total cost of 9.20.
- A modifier method, setQuantity, set the grocery item's quantity to be the given value.

The GroceryList class

The GroceryList class holds information about the Grocery List, via the private variables:

- items: an array containing a list of grocery Items ordered (of type GroceryItemOrder)
- numItems: the number of items in the grocery list (an integer)

And public methods:

- A constructor method, GroceryList that creates an instance of the class (an object).

- An accessor method, addItem, adds the given item order to this list if the list has fewer than ten items.
- A method, getTotalCost: Returns the total sum cost of all grocery item orders in this list. You should print this number with a format that rounds it so that there are two decimal digits to the right of the decimal point. For example, use %7.2f to print this number with seven spaces, two spaces to the right of the decimal point (will do rounding for you), the decimal point as one space and four spaces to the left of the decimal point for the part of the number not fractional.

GroceryList.txt file has the data you need for this class.

The Main.java program

This file has one static method called main. Main is the method used to run all the code. The grocery data will be retrieved from the GroceryList.txt file; then, the data will be used to create GroceryItemOrder objects. These objects are inserted-in one at a time into the GroceryList array. Luckily for you, ***Main.java*** is supplied!

The InStringFile.java program

How do you do file I/O in Java? You are supplied with ***InStringFile.java***, which contains Java code to parse a file, reading one "token" at a time. A token is an alphanumeric character string with no blanks in it, such as "Canada" or "123abd456def?" The GroceryList.txt file has three tokens per line. ***Main.java*** uses this class when parsing the input file.

Testing

Testing is done simply by running the supplied Main.java.

Additional Specifications

- Add the following method to your GroceryList class:

  ```
  public boolean GroceryListQuery(String itemName)
  ```

  The method should return **true** if the given grocery item is listed on the grocery list. You should modify the ***main.java*** to query if the given grocery list contains “Bread” and “Eggs”.

- Also, add the following method to your GroceryList class:

  ```
  public String toString()
  ```

  The method should return a string containing the values of an instance of the GroceryList class, and this string is "pretty printed" (formatted). Use String.format to format your data and "\n" (new line character) to print new lines.

  For example:

  ```
  s = "String.format("The character string %-6s\n","62") +
      String.format("has inreger value %2d\n",62) +
      String.format("and double value %6.2f\n",62.0);
  System.out.println(s)
  ```

  prints:
  The character string 62
  has integer value 62
  and double value 62.00

Note that "+" does string concatenation. Method format is a method of class String and allows pretty printing of data (i.e. nice formatted output, with the appearance of numbers and strings controlled).

The format %-6s allows six characters to print a string left justified (format %6s means the string is printed right justified). The format %2d means two spaces are used to print an integer value. The format %6.2f means use 6 spaces in total to print a floating point number, with 2 digits to the right of the decimal point for the fractional part of the number, a decimal point counting as a one space and with up to 3 digits to the left of the decimal point for the part of the number greater than zero. Note that %s, %d, and %f format strings, integers, and doubles with a dynamic format depending on the specific data values. Only use the formatted output for this assignment.

- You are going to modify the ***Main.java*** again to make the program able to print out the complete grocery list in "pretty printed" as discussed above.

Non-functional Specifications:

- Your program does not have to perform exception handling.
- You can assume the data in the given file is correct. So, no error checking of any type is required.
- Include a comment identifying yourself (name and student number, uwo email address) and the assignment number, course and year at the beginning of all your class files. For example, the professor has:

```
////////////////////////////////////////////////////////////////
// Ahmed Ibrahim 123456789                                    //
// ahmed@csd.uwo.ca or ahmed@uwo.ca                           //
// Assignment 1, CS1027, Spring 2019                          //
////////////////////////////////////////////////////////////////
```

- Include comments describing key parts of your program well. Comments should be grammatically correct, concise, easy to understand and match the Javadoc syntax.
- Assignments are to be done **individually** and **must be your work**. Group work is not tolerated. A software tool may be used to detect plagiarism.
- Use Java coding conventions and good programming techniques. For example:
    - Use meaningful variable names
    - Use Java conventions for naming variables and constants.
- Make sure your code runs using Eclipse's Java, even if you do not use Eclipse to write your code.
- Zip all the files used by your program. Use your first and last names and student number and assignment number and year (with linking of the tokens in the zip filename done by underscores). For example, your professor's files would be in

Ahmed_Ibrahim_123456789_Assignment_1_Spring_2019.zip.

Grading Criteria:

Functional specifications:

- Does the program behave correctly, according to specifications?
    - Does it run when tested with the main program provided?
    - Are your classes appropriately created?
    - Are you using appropriate data structures?
- Non-functional specifications: as described above
- Assignment submission via OWL by 11:55 pm on June 27, 2019 (a Thursday)