Concordia University
Comp 249 – Winter 2020
Object-Oriented Programming II
Assignment # 1 – Due 11:59 PM –
Friday, January 31, 2020

Purpose: Practically, this assignment should be considered as *Assignment # **0***! The purpose of this assignment is to help you review some of the main topics covered in previous courses, including classes, loops, arrays, arrays of objects, static attributes and static methods.

General Guidelines When Writing Programs:

- Include the following comments at the top of your source codes
  // ----------
  // Assignment (include number)
  // Question: (include question/part number, if applicable)
  // Written by: (include your name and student id)
  // ----------
- In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display a welcome message which includes your name(s).
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy to read format.
- End your program with a closing message so that the user knows that the program has terminated.

JavaDoc Documentation:

Documentation for your program must be written in **JavaDoc**.
In addition, the following information must appear at the top of each file:

```
Name(s) and ID(s)   (include full names and IDs)
COMP249
Assignment #        (include the assignment number)
Due Date            (include the due date for this assignment)
```

Part I)

An **appliance** object has four attributes, a type (String), a brand (String), a serial number (long), and a price (double). The type of an appliance can be any of the following: Fridge, Air Conditioner, Washer, Dryer, Freezer, Stove, Dishwasher, Water Heaters, and Microwave. The brand indicates the vendor/manufacturer of the appliance (such as LG, Samsung, Bosch, etc.), and can be any string. The serial number represents the serial number of the appliance, and must be unique. For simplicity, we assume that the serial numbers start at 1000000 and it is common/shared among all different brands and types (for example, if the creation of objects starts by a Washer created by Maytag, then it will be assigned serial number 1000000. If a Microwave by Breville is created next, then it will be assigned serial number 1000001, and so on afterwards). The price of an appliance cannot be less than 1$. (Hint: Static – You are allowed to add other attributes to the class.)

For this part, you are required to design and implement the **Appliance** class according to the following specifications:

- Upon the creation of an appliance object, the object must immediately be initialized with valid values; that is type, brand, serial number and price. (Hint: Constructors.)

- The design should allow enough flexibility so that the value of any of these attributes can be modified later on (with the exception of the serial number, which can never be changed once assigned). For example, it should be possible to create an appliance object with a given price then change its price later on. Notice that changes to brand or type is allowed (although this is unusual in reality!). The design should also allow the user to obtain the value of any of the attributes. (Hint: Accessors & Mutators.)
- The design should allow all information of an object to be displayed at once through the utilization of **System.out.print()** method. (Hint: **toString()** method)

- It is required to know how many appliance objects have been created. For that, you need to add a method, called **findNumberOfCreatedAppliances**(), to the class. This method must return the number of created appliance objects prior to the time this method is called. The method would simply return 0 if no appliances have been created by the time the method is called.

- It is required to compare two Appliance objects for equality. Two appliances are considered equal if they have the same brand, type and price. (Hint: **equals()** method)

- It is required to display any Appliance object (all info of that object) using System.out.println() method. (Hint: **toSting()** method)

Part II)

You are hired by a major appliance dealer to write a software that helps the store owner in acquiring and keeping track of appliances at the store. Write a driver program that will contain, at least, the following methods. **Note**: You can have the main function in a separate driver file, or in the same file if you prefer.

1. **a main() method**, that will:
    a. Display a welcome message;
    b. Prompt the store owner for the maximum number of appliances (maxAppliances) his/her store can contain (or wish to aquire; this is not a focus here). Create an empty array, called inventory, that will have the potential of keeping track of the created Appliance objects.
    c. Display a main menu (Figure 1) with the following choices and keep prompting the user until they enter a number between 1 and 5 inclusive (i.e invalid number will result in repeating the display of the main menu):

```
What do you want to do?
    1. Enter new appliances (password required)
    2. Change information of an appliance (password required)
    3. Display all appliances by a specific brand
    4. Display all appliances under a certain a price.
    5. Quit
Please enter your choice >
```

Figure 1. Main

d. When option 1 is entered:
   i. Prompt the store owner for his/her password. (Make sure you have a constant variable containing the password "c249" – do not use any other password as it will be easier for the marker to check your assignments). The store owner has a maximum of 3 attempts to enter the correct password. After the 3rd illegal entry, the main menu in Figure 1 is re-displayed again. Additionally after this process is repeated 4 consecutive times (i.e. total failed attempts is 12 consecutive attempts by now), the program must display the following messages: "Program detected suspicious activities and will terminate immediately!", then the program must exits.

   ii. If the correct password is entered (which will reset the counter of failed attempts), ask the owner how many appliances he/she wants to enter. Check that there is enough space in the store (inventory array) to add these many appliances. If so, add them; otherwise inform the owner of the maximum remaining places in the array. (How the appliance information is input/entered by the user is up to you).

e. When option 2 is entered:
   i. Prompt the store owner for his/her password. (Still, you should make sure the password is "c249" – do not use another password). Again the store owner has 3 attempts to enter the correct password. However, after the 3rd illegal entry, the main menu in Figure 1 is simply re-displayed again (notice the different behaviour in that case from the previous one above).

   ii. Ask the user of the serial number of the appliance he/she wishes to update. If there is no object with that serial number in the array inventory, display a message indicating that and ask the user if he/she wishes to re-enter another serial number, or quit this operation and go back to the main menu. If the entered serial number exists, display the current information of that appliance in the following format:

Appliance Serial # *xxxxxxx (*where *xxxxxxx* is the serial number)
Brand: *brand of the appliance*
Type: *type of the appliance*
Price: *price*

Then ask the user which attribute he/she wishes to change by displaying the following menu.

```
What information would you like to change?
    1. brand
    2. type
    3. price
    4. Quit
Enter your choice >
```

Figure 2. Update menu

Once the user has entered a correct choice, ask the user for the new information and make the changes to the object attribute then display again all of the attributes on the screen to show that the change has taken place. Keep prompting the user for additional changes until the user enters 4 (indicating that he/she wishs to stop the update process). Each time the user is prompted for a choice, make sure that a number from 1 to 4 is entered, otherwise keep prompting until a valid number is entered. You should pay a special attention to the

change when it comes to type, as the different types are specific and limited (cannot be different than the types indicated in the initial description in Part I above).

f. When **option 3** (in the main menu shown in Fig. 1) is entered, prompt the user to enter a brand name. You then need to display the information of all appliances in the inventory with that brand. (Hint: You may use a static method, for instance called findAppliancesBy, which accepts a string for an brand name then performs the needed search).

g. When **option 4** (in the main menu shown in Figure 1) is entered, prompt the user to enter a value (representing a price). You then need to display all appliances in the store that have a price smaller than that entered value. (Hint: You may use a static method, for instance called findCheaperThan, which accepts a double value, for a price, then performs the needed search).

h. When **option 5** (in the main menu shown in Fig. 1) is entered, display a closing message and end the program.

Submitting Assignment 1

- For this assignment, you are allowed to work individually, or in a group of a maximum of 2 students (i.e. you and one other student). You and your teammate must however be in the same section of the course. Groups of more than 2 students = zero mark for all members!
- Only electronic submissions will be accepted. Zip together the source codes. (Please use WINZIP).
- Students will have to submit their assignments (one copy per group) using the Moodle/EAS system (please check for your section submission). Assignments must be submitted in the right DropBox/folder of the assignments. **Assignments uploaded to an incorrect DropBox/folder will not be marked and result in a zero mark. No resubmissions will be allowed. Your instructor will indicate whether you should upload your assignment to EAS (**https://fis.encs.concordia.ca/eas/**) or to Moodle.**
- Naming convention for zip file: Create one zip file, containing all source files and produced documentations for your assignment using the following naming convention:

  The zip file should be called *a#_StudentName_StudentID*, where # is the number of the assignment and *StudentName/StudentID* is your name and ID number respectively. Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. For example, for the first assignment, student 12345678 would submit a zip file named like: *a1_Mike-Simon_123456.zip.* if working in a group, the name should look like: *a1_Mike-Simon_12345678-AND-Linda-Jackson_98765432.zip.*
- Submit only ONE version of an assignment. If more than one version is submitted the first one will be graded and all others will be disregarded.
- If working in a team, only one of the members can upload the assignment. Do **NOT** upload the file for each of the members!

⇨ Important: Following your submission, a demo is required (please refer to the courser outline for full details). The marker will inform you about the demo times. Please notice that failing to demo your assignment will result in zero mark regardless of your submission.

Evaluation Criteria for Assignment 1 (10 points)

Documentations	**1 pts**
JavaDoc documentations	1 pt
Part I (Class Appliance)	**3 pts**
Default & copy constructors	1 pt
Accessor/mutator method for static attribute	1 pt
equals, toString and static attributes/methods	1 pt
Part II (Driver & other static methods)	**6 pts**
Handling of password	1 pt
Handling of option 1	1 pt
Handling of options 2 & 3	1 pt
Handling of options 4 & 5	1 pt
General quality and correctness of code & functionality	2 pts