

ECS510 Algorithms and Data Structures in an Object-Oriented Framework

Exercise Sheet 1: Using the **DrinksMachine** class

This is a set of exercises to support the “Using Objects” section.

Note that 1), 2) and 4) are not really “questions” as they do not have a specific programming problem to solve. For assessment, you do need to show you have downloaded the code as requested, and written some appropriate code to test how it works.

- 1) Download the files you need to run the Drinks Machines examples. You can find these by going to the Code Index in the “Using Objects” section of the module web site. In order to create and use `DrinksMachine` objects you will need the `.class` files `Can.class`, `DrinksMachine.class` and `EmptyCanException.class`. These are pre-compiled Java code, you do not need the `.java` files for them, and you are deliberately not given them. The constructors and methods for class `DrinksMachine` are given in the module notes.

The files ending in `.java` are “front end” code, they hold programs which make use of the classes given by the pre-compiled files ending in `.class`.

Make sure you can run the programs in the `.java` files. If you are confident with basic use of objects in Java, use the class `DrinksMachineB.class` instead of `DrinksMachine.class`. If you do that, you will then also need the classes `EmptyMachineException.class` and `NotEnoughMoneyException.class`. You would then complete the rest of this exercise using this class, which means you must write code to handle the exceptions.

You may organise your files through an Interactive Development Environment like BlueJ or NetBeans, or edit, compile and run directly through Linux.

- 2) Experiment with the downloaded Java programs for drinks machines. Try altering them in various ways to see what the effect is. Use them to make sure you understand what is covered in the notes.

- 3) Now try writing a program which simulates the following scenario:

I have a drinks machine. I insert some money into it. I keep on pressing the Coke button until when I press it no more Coke cans come, then I press the Change button.

The program should end by printing a message which tells me what I then have, for example:

I have 20p and 3 cans of Coke

- 4) The class `Can` has public methods with the following signatures:

```
Can(String cont)
String toString()
boolean isFull()
void drink()
```

The first of these is the constructor, it takes a string representing the particular drink, for example "Coke" and returns an object representing a can of that drink. The second is used to give a text representation of the `Can` object it is called on, you will already have used it when you “printed” a `Can` object (which causes it to be called automatically). The third method says whether a can is full or not. When a `Can` object is created, it is always full. Once the fourth method, `drink()`, is called on it (simulating drinking the can’s contents), it becomes empty.

Experiment with these methods by writing code which simulates getting a can from a machine and then drinking the contents of the can. Your code should print messages showing what you have before the can is drunk and what you have after.

Note for text input, the method with signature `String next()` in class `Scanner` reads and returns the next word typed in the command window, and the method `nextLine()` reads and returns the whole line typed in as a `String`. There is also the method `nextInt()` to read and return integers, which is used in some of the example code.

- 5) Write a static method which takes two `Can` objects as its arguments, and simulates taking two cans, drinking from the first can unless that is empty, and drinking from the second can if the first can is empty and the second can is not. Write a program which demonstrates this method being used, so that you can set its arguments to full or empty cans. The method may contain print statements saying what is being simulated.

For questions 6), 7) and 8), you should write a program to demonstrate the method being used, and to test that it works correctly. Note that the principal task in the question is to write the method requested, which should not contain print statements. However, writing appropriate test code is an important aspect of programming, and in this exercise you are asked to do it yourself, writing code which interacts with you to provide example arguments and to print the results.

- 6) Write a static method which takes two `DrinksMachine` objects and simulates the owner of the machines collecting the stored cash from both of them and identifying which has the most cash stored in it. The method should return a reference to whichever of the `DrinksMachine` objects is identified as the one with the most cash.
- 7) Write a static method which takes two `DrinksMachine` objects and an integer `p` as its arguments. It must raise the price charged by the cheaper machine to whichever is the lowest out of its existing price plus `p`, or the price of the more expensive machine.
- 8) Write a static method which takes a `DrinksMachine` object as its argument and returns the number of Coke cans stored in the drinks machine it represents without changing that number (it can do this by buying Cokes until they run out, then loading them back into the machine).

As a transition from the sort of programming you have done previously, question 9) asks you to write code that interacts with a user. However, in the rest of this module you will move on from thinking of programming primarily in those terms to concentrate on writing code in the form of classes which describe objects that interact with other objects through method calls, rather than code which interacts directly with a human user.

- 9) Write a program that simulates the following scenario:

Two drinks machine are set up. People buy cans from the machines. At the end of the day, the machine operator collects the cash that is left in the machines.

Remember that the method with signature

```
int collectCash()
```

in class `DrinksMachine` simulates collecting the cash from the machine. You should make sure people do not buy drinks if there are none of their required sort left. This can be tested using the methods in class `DrinksMachine` with signatures:

```
boolean cokesEmpty()
```

```
boolean fantasEmpty()
```

You will need to devise your own simple text-based human interface for this program, so that a “user” can type in commands representing the actions of the people using the machines.

Make a loop so that you can type in response to prompts over whether there are more people to come, whether each person wants Coke or Fanta, which machine they use, how much money they insert and so on.