ODL127 Algorithms 16 January 2006

Exercise 1: Using the `DrinksMachine` class

1) Download the files `Can.class`, `DrinksMachine.class` and `EmptyCanException.class`, and the files `UseDrinksMachines1.java`, `UseDrinksMachines2.java`, `UseDrinksMachines3.java` and `UseDrinksMachines1.java` from the web link. The files ending in `.java` hold programs which make use of the classes given by the pre-compiled files ending in `.class`. You have deliberately been given the pre-compiled files without the `.java` files used to create them.

Make sure you can run the programs in the `.java` files.

2) Experiment with the 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 has been taught in the lectures.

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

> I have a drinks machine. I insert some money into it. I press the Coke button, then I press the Fanta button, 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 a can of Coke and a can of Fanta
```

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 say 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.

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.

6) Write a program which simulates the following scenario:

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

Remember that the method with signature

```
int collectCash()
```

in class `DrinksMachine` simulates collecting the cash from the machine. You might also wish to 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()
```

Note also, for text input, the method with signature

```
String next()
```

in class `Scanner` reads and returns the next word typed in the Linux command window (you have seen `nextInt()` to read and return integers). Make a loop so that you can type in response to prompts over whether there are more people to come, and whether each person wants Coke or Fanta.

7) Write a program which simulates the following scenario:

> Two drinks machines are set up. People buy cans from the machines. They buy from the cheapest machine, unless that has run out of the drink they want, in which case they buy it from the other machine.