## Lab 8 : More Classes

**Topics covered** creating classes, constructors

**Exercise 1**: Designing a class can be tricky business, and will generally require you to design and redesign the contents a few times. To make the process as easy as possibly, it is necessary to plan the class before you get right into writing the code.

**Part A:** Identify and list what kinds of information is attached to, and/or exists in the following (not writing code!):

- One train ticket gate
- One supermarket register
- One landline telephone (digital)
- One staff member of an ice cream company
- One digital book

**Part B:** For each of the above real-world objects, identify suitable attributes and data types needed to store the information you came up with.

**Part C:** For each attribute, explain which would be public and which would be private.

**Part D:** Would the access modifier depend on the software system being built? For each class described, give an example of a software system that would require the attributes be public, and one that would require them to be private. Think about who is using the system, and what they need to know about the class.

**Exercise 2**: Create a class for each of the following objects. You classes should each contain instance variables (fields), appropriate get/set methods and at least one constructor. Do not worry about the implementation of the rest of the class.

a) An `OrderedPair` object contains two real numbers which are stored in numerical order. The constructor should take in two numbers and store them in the correct order, regardless of input order.

b) A `PasswordGenerator` object which stores a list of 10 pre-generated passwords. The `PasswordGenerator` has two constructors (listed below).

The first one takes in three `boolean` flags that indicate whether the passwords generated should contain letters, numbers and symbols. If none of the values are `true`, the constructor will `throw` an exception.

The second constructor will use all three (letters, numbers and symbols) by default.

```
1  public PasswordGenerator(boolean useLetters, boolean useNumbers,
2          boolean useSymbols) {
3     // First constructor
4  }
5  public PasswordGenerator() {
6     // Second constructor
7  }
```

**Exercise 3**: Implement the following methods for the classes listed in exercise 2:

a) `OrderedPair`

- A `toString` method that turns the object into a readable `String`; e.g. `(3, 4.5)`
- An `equals` method that checks if one `OrderedPair` is the same as another.

b) `PasswordGenerator`

- A `generatePasswords` method that generates 10 new passwords. This method will only work once – if it is called a second time, it should just leave the already generated passwords alone. This method should only be called from the constructor to set up the passwords.
- A `generateExtraPasswords` method that generates $n$ extra passwords, where $n$ is given as an argument, and then stores them in a file called `extraPasswords.txt`.

---

**Extensions**

**Extension 1:** Extend the `generatePasswords` class so that it can set ratios for the amount of letters, numbers and symbols in a password. For example, we can set the generated passwords always contain 60% letters, 30% numbers and 10% symbols.