Java Practical with Lambdas

**Objectives**

The objectives of this practical session are to:

* Experiment with Lambdas and Streams for Data Processing

**Overview**

This uses Lambdas to process a large number of Employees in a variety of ways.

**Practical**

|  |  |
| --- | --- |
| Recreating the Project Code and Using Lambdas |  |

1. Create a new project in IntelliJ, and create (or copy) classes for Employee and Manager, just like the ones you had earlier for the Java Practicals, when we were using Collections of Employees.
2. Create an application class that creates lots (!) of Employee objects with random data, storing them in a List. Again, you can copy this code from previous work.
3. Now that you have a List of Employee objects, you can call the stream() method with will return a Stream<Employee>. This then allows you to do processing using the Stream API, with accept Lambda expressions for doing optimal functional processing.
4. Recall that Streams are immutable, and that any operations you do on them return new Streams, possibly of different type, if you do a mapping for example.

In the following steps, use a single line of code, chaining Stream API code, starting from the original List.

1. Print out all the Employees (hint: use the forEach() method!).
2. Print out the names of all the Employees (use the map() method).
3. Filter the Employees based on age (e.g. only those under 50), and print them out.
4. If an employee is over 50, promote them to be a Manager, and return a List with all the Employees, reflecting the changes. Then create another Stream, and print out the class names of each of the Employees in turn (use the getClass() method, which returns a Class object, which itself has a getName() method).