Tut-Lab Week 2

Aims:

- Develop familiarity with object-oriented programming concepts
- Learn about the representation of objects and inheritance in Java
- Gain familiarity with Java development using Eclipse
- Practise documenting code using Javadoc

Preparation:

- Review the API documentation for basic library classes: **String, Calendar, LocalDate**
- Install Eclipse on your own machine

Setting Up Eclipse on Lab Machines

- From a terminal, execute **eclipse**: this should start up a version of eclipse in a new window – make sure it is Eclipse 4.7.2 Oxygen
    - **Do this at the start of the tut-lab as this will take a long time if everyone starts up eclipse at once!**
- Follow the prompts, and set up a workspace directory for eclipse in your home (or sub-) directory
    - Follow any recommended instructions concerning the Scala IDE plugin
- If you see a welcome screen, close it to reveal a Java perpective
    - If (instead) you see a Java EE perspective, open a Java perspective (click the Open Perspective button just to the left of the Java EE button and choose Java from the list), then close the Java EE perspective (right click on it and select Close)
- Make sure Java SE 8 is the default execution environment: check this via Window > Preferences > Java > Installed JREs
    - The JRE should be called java-8-openjdk-i386; if it is not there, add the JRE which should be located at /usr/lib/jvm/java-8-openjdk-i386
- You can open a new Java project (via New > Java Project or using the New button drop-down): choose the default JRE for the execution environment
- Eclipse should be set up to automatically (re-)compile your projects when editing (check via the Project menu that Build Automatically is ticked): saves time!
- Run programs in Eclipse via Run > Run As ... Java Application or hit the Play button after setting up a Run Configuration

- Look at the Problems tab for any bugs, Javadoc to see your own documentation, and Console to see the output

Object-Oriented Programming

- Create an **Employee** class which has private fields for an employee's name and salary
- Create a **Manager** class that is a subtype of **Employee**, which also has a private field for a manager's hire date
- Define constructors for the classes, and public getter and setter methods for the fields: document them using Javadoc
- Define the standard methods toString, equals and clone for **Employee** and **Manager**
    - The methods for **Manager** should call those for **Employee** using the **super** construct
    - What output do you expect when getClass().getName() is called in the toString method of **Employee** with a **Manager** object?
- Create a class for testing the **Employee** and **Manager** classes and define some tests for your methods
    - What do you expect when you test whether an **Employee** is equal to a clone of the **Employee**?
    - What do you expect when you test whether a **Manager** is equal to an **Employee** with the same name and salary (and vice versa)?
    - What do you expect when you test whether the name of an **Employee** is equal (or ==) to the name of a clone of the **Employee**?
    - If you change the hire date of a clone of a **Manager**, is the hire date of the original **Manager** also changed?
    - If a **Manager** object is declared as an **Employee**, e.g. `Employee e = new Manager(...)`, which methods of **Manager** can be called on this object?