Tut-Lab Week 3

Aims:

- Understand Programming by Contract
- Understand inheritance in object-oriented programming
- Understand how class invariants work with inheritance
- Learn to use JUnit for unit testing

Preparation:

- Review the concepts of *encapsulation*, *programming by contract*, *pre-* and *postcondition*, and *class invariant*
- Review the JUnit tutorial

Programming by Contract

- Draw a UML class diagram that incorporates the following requirements
  - A **BankAccount** class for maintaining a customer's bank balance
    - Each bank account should have a current balance and methods implementing deposits and withdrawals
    - Money can only be withdrawn from an account if there are sufficient funds
    - Each account has a withdrawal limit of $800 per day
  - A subclass of **BankAccount** called **InternetAccount**
    - In addition to the constraints on **BankAccount**, there is a limit of 10 Internet payments per month
    - Note that Internet payments count as withdrawals, so are subject to the daily limit on withdrawals
- Give Java implementations of both classes
  - Explain how the limits on withdrawals are enforced within your system
- Define pre- and postconditions for each method in the **BankAccount** and **InternetAccount** classes
  - Explain how your code is consistent with your pre- and postconditions
  - Verify that your pre- and postconditions respect contravariance and covariance (respectively)
  - Provide a rigorous argument that $balance \geq 0$ is a class invariant for both classes

- Define a JUnit test case to test your implementations
    - Especially construct tests for the postconditions and class invariants
- (Advanced) Are your class definitions consistent with the Liskov Substitution Principle?
    - Consider both the informal definition and those in the paper by Liskov & Wing
    - If not, explain why. Is it their formulations of the principle that are wrong, or your specifications?