**OOD Lab 3 – Inheritance**

* Put your name and student number in a comment at the top of each file.
* Ensure that your code is well formatted e.g. lines after an opening brace should be indented. Each closing brace should be at the same indentation as the statement which opened it. (Use Ctrl + Shift + F if using Eclipse)
* It is compulsory to comment your code appropriately. For each of the exercises ensure that your code is sufficiently commented to demonstrate your understanding of the software you are writing. You do not have to comment every line however.
* Marks will be lost if the above instructions are not followed.
* You may discuss assignments with you classmates but you must write each assignment yourself. Copying another student’s code will be considered plagiarism. Outcomes vary from receiving zero in assessments to failing the course.

1. Create a class called **Account**. This class should have
   * An attribute for the account number.
   * An attribute for the balance of the account.
   * A one argument constructor which takes an account number and which sets the balance to zero.
   * A method called **depositFunds** which accepts an amount to deposit into the account. If the amount is less than or equal to zero an error message should be printed.
   * A method called **withdrawFunds** which accepts an amount to withdraw from the account. If the amount is less than zero, or if there aren’t enough funds in the account, then a message should be printed.
2. Write two classes which derive from the **Account** class called **SavingsAccount** and **CurrentAccount**.
   * A **SavingsAccount** class, in addition to the attributes of an Account class, should have an interest variable and a method which applies this interest to the balance.
   * A **CurrentAccount** class, in addition to the attributes of an Account class, should have an overdraft limit variable. This should allow a user to withdraw more than the present balance of the account, up to the overdraft limit e.g. a deposit account with an overdraft of 200 can have a balance of -200, but not -201.
   * Ensure that you have overridden methods of the Account class as necessary in both derived classes.
3. Write another class called Bank containing a main method in order to fully test your account classes. Create an object for each account class (total 3). Using the objects call the various methods defined within these classes to illustrate that they are functioning correctly.

**What do I submit?**

Submit the Account.java, SavingsAccount.java, CurrentAccount.java,

and Bank.java files on Moodle.