Discipline of Computing and IT
University of Newcastle

SENG1120/6120 – Semester 2, 2017
Week 3

Video guide: https://www.youtube.com/watch?v=vHHSGAhl1mY

1. Download the provided files (`account.h`, `account.cpp` and `bank.cpp`) on Blackboard to the folder in which you will perform this week's laboratory exercises.

2. As shown in lectures, modify the code to include a macro guard compiler directive for the class account.

3. Now create a namespace entitled <your_surname>_lab02 that restricts the scope of the names used in class account.

4. Ensure that your program compiles and runs as it should. Modify the `makefile` provided and put the two `.cpp` files under "SOURCES". There is an initialization problem in the member variable for the account's balance. Find it and fix it.

5. Now, in the file bank.cpp, introduce the variable `ptr` that is a pointer to the account object `my_account1`. Change all calls to `my_account1` so that they are achieved using `ptr->`.

6. Just for interest, use a `cout` statement to display the content of `ptr`.

7. Use the = operator to create a copy, called `acct_copy`, of `my_account1`. Verify that `acct_copy` is really a copy by calling its mutator methods and confirming that the calls do not alter `my_account1`.

8. Now create a reference type called `acct3` that implements an alias for `my_account1`. Verify that `acct3` is a reference type by demonstrating that calls on the mutator methods of `acct3` also mutate `my_account1`.

<u>Extension questions for SENG6120 (or if you want to learn MORE)</u>

9. Extend class `account` to include a method called `compare` that takes, as parameters, pointers to two instances of account, and returns a pointer to the instance that contains the bigger balance. Verify that the returned value is a pointer by using it to query the instance's balance.

10. Now extend class `account` to include a method called `merge` that takes, as parameters, references to two instances of `account`, and returns a new instance of `account` that contains the sum of the balances of the parameter instances. Verify that the returned value is a new object by using it to access the balance.

11. Create a new class called `portfolio` that has, as instance data, two `account` objects representing, respectively, savings and cheque accounts. After defining and implementing the constructor for portfolio, define and implement mutator methods `savings_zero, cheque_zero, savings_deposit, cheque_deposit, savings_withdraw and cheque_withdraw`. Then define and implement the query methods `savings_bal` and `cheque_bal`. Ensure that `portfolio` defines a `namespace`, and is protected by a guard compiler directive.

12. Write a program that creates an instance of `portfolio`, and then exercises its methods to demonstrate the success of your implementation.

Good Luck!