CM2204: Advanced Programming Laboratory Worksheet (Week 11)

Prof. D. Marshall

Aims and Objectives

After working through this worksheet you should be familiar with:

- Be able to define new dynamic classes and C++'s (cleaner) use of pointers
- Be able to compile C++11 programs
- Be able to use C+11's smart pointers
- Understand the differences in approach between C++ and Objective-C:
- Be able to write and compile simple classes in Objective-C
- Understand the purpose of categories and explain the difference from inheritance

None of the work here is part of the assessed coursework for this module.

- Follow the web links for files highlighted and underscored to get code listings
- All lecture and lab class code is a available on the CM2204 Web page
- Solutions to the exercises will be released on the CM2204 Web page after this lab class.

Smart Pointers & Objective-C

Smart Pointers

1. **Reference Counting**: Download the template (Template.zip) for a *reference counted* `String` implementation. Complete the code as follows:

   - Add **two** extra functions to the class
     - each should convert the `MyString` object to upper case, but
       - one should *modify* the `MyString` object itself,
       - while the other should return a new MyString with the result.
     - Verify that the *reference counting* works correctly in **both** cases.
   - The example uses the `const keyword` applied to functions and arguments.
     - What happens if you modify the body of the `getChar` function so that it changes the `char` array (e.g. sets the first character to something)?
     - What happens if you try to modify i within this function?

2. **Smart Pointers**: Download, compile and run the smart pointer code mention in the lectures

   - SharedPtr.cpp — Shared Pointer Example
   - WeakPtr (Zip) (Header.h + WeakPtr.cpp) — Weak Po

   Open the source files in a suitable editor and study how the respective smart pointer are defined, created and used.

   - Recall: To compile C+11 code in the the **Linux lab**:
     ```
     g++ -std=c++0x .....
     ```
   - or adapt the Makefiles supplied with the example code.
   - See lecture notes/Makefiles for Mac OS X compilation.

3. **Smart Pointers**: Using appropriate smart pointers, write some code that can *swap* two smart pointers. Hint:

   - You could implement this with a function you create yourself (good practice), **or**
   - Look at the http://www.cplusplus.com/reference/memory/shared_ptr/ reference for a suitable function to achieve the task.

4. (Advanced) **Smart Pointers**: Create a simple singly linked list structure using Smart Pointers. You may adapt the *vanilla* C++ pointer linked list code: List.cpp

Objective-C

1. Compile and run the HelloWorld.m example from the CM2204 Week 11 Code Web page.

   - Modify the code so that it uses `NSLog` for output *instead* of `printf`.

   Recall: to **compile** Objective-C in the **Linux lab**:

   - You must execute the command:

     ```
     ./usr/share/GNUstep/Makefiles/GNUstep.sh
     ```

     once to configure your environment.
   - Use the command:

     ```
     gcc -o HelloWorld HelloWorld.m -I `gnustep-config --variable=
         GNUSTEP_SYSTEM_HEADERS` -L `gnustep-config --variable=
         GNUSTEP_SYSTEM_LIBRARIES` -lgnustep-base -fconstant-string-
         class=NSConstantString -D_NATIVE_OBJC_EXCEPTIONS
     ```

     for compilation.
   - or adapt the Makefiles supplied with the example code.
   - See lecture notes/Makefiles for Mac OS X compilation.

2. Write an Objective-C class which represents a *circle*, with member variables to store the *radius* and the value of *pi* (use 3.1419).

   - Add methods to get the value of the radius, set the value of the radius and calculate the circumference of the circle.
   - Write code to test this class.
   - Write a category that adds the functionality to calculate the *area* of the circle.

Further Practice

1. (Advanced) **Smart Pointers**: Revisit the Stack exercise from Week 10 Lab Class and create a smart pointer version of a Stack.