Programming for Engineers Portfolio : Quarter 3, Set 2

These exercises are intended to directly follow the tasks from the second lab of Q3. If you haven't done the lab, you are strongly recommended to do so first, as it shows how to do all the tasks, and also puts it in a more realistic context: lab2 spec

These exercises are very artificial and simply require you to demonstrate that you can complete the same steps on an unseen code-base. The lab and lectures try to give context as to why these are useful, but here they just need to be done.

1 - Merge changes from a remote repository

Merge this repository (elec40004-2019-q3-portfolio-es1219-master) into your private repository.

(All changes are in a seperate directory, so you should not get any merge conflicts.)

2 - Overloading << and >>

Provide definitions for the `<<` and `>>` operators of `Nugget`. Example input is given in `test_Nugget_io_in.txt`.

3 - Testing << and >>

Write a script called `test_Nugget_io.sh` which checks that your implementation of `<<` and `>>` can correctly read and write all the values from `test_Nugget_io_in.txt`.

(You can (and need to) create helper files of various types, ideally with the prefix `test_Nugget_` (though this is not checked).)

4 - Fix const-correctness

The class file `Proxy_Dst1.hpp` contains an implementation of the `Proxy` member variables, but the methods are not yet const-corrrect to match the class declaration. Make it so.

5 - Compile the controllers

Create a script called `compile_and_run_Generators_to_Dst1.sh`, which compiles each of the generator source files into an executable, and then runs and captures the output of the program in a file. The two generator files and associated executables/output names are:

- `GenA.cpp`: executable is `GenA_to_Dst1`, output is `GenA.sink1.txt`
- `GenB.cpp`: executable is `GenB_to_Dst1`, output is `GenB.sink1.txt`

Capture the state of the repository in a commit with the word `s2e5FixedDst1` in the commit message.

6 - Convert the source files to use the Dst2 implementation

Modify the two generator source files so that they include the `Proxy_Dst2.hpp` implementation.

7 - Compile the source files

Create a script called `compile_and_run_Generators_to_Dst2.sh`, which compiles each of the source files into an executable, and then runs and captures the output of the program in a file. The two generator source files and their assocated executables/output names are:

- `GenA.cpp`: executable is `GenA_to_Dst2`, output is `GenA.sink2.txt`
- `GenB.cpp`: executable is `GenB_to_Dst2`, output is `GenB.sink2.txt`

Capture the state of the repository in a commit with the word `s2e7SelectedDst2` in the commit message.

8 - Converting the implementation headers to source files

Rename the implementation headers from `.hpp` to `.cpp`

(Sometimes things are that simple. If you want to, you can also remove the `#ifndef` guards, but that sin't checked.)

9 - Compile generator sources against channel implementation sources

Remove the `#include "Proxy_Dst2.hpp"` line from both Generator sources.

Split the file `Nugget.hpp` up, by moving operator overload definitions into a source file called `Nugget.cpp`.

10 - Update the two build scripts

Update `compile_and_run_Generators_to_Dst1.sh` and `compile_and_run_Generators_to_Dst2.sh` so that they select the correct functionality for each executable by compiling different combinations of source files.

Add all the new files, commit your final version with the message `s2e10SeperateSources`, and then push the final results to your private repo.