COMP2404B (Fall 21) -“Introduction to Software Engineering”

Tutorial 8 of 10 (Double mark tutorial)

- No late tutorials will be accepted.
- If there are specific instructions for making a function, please follow them exactly. That means that
  - function names
  - function return types
  - parameter types and order

  should all be **EXACTLY** as described. If the script can’t read it, you will receive 0 for that part.
- Your **Tutorial 8** code will be marked by a Python script. You are being given the script so that you may make sure your code runs correctly. As such, submissions with improper files, configuration, or function signatures will not be accepted.

1 Submission Instructions

Download “tutorial8.zip”. Unzip it into your working directory. There is a directory “tutorial8” and the test file “t8test.py”. In the “tutorial8” folder are “Date.h”, “Date.cc”, “test.cc” and “defs.h” to get you started. To the “tutorial8” folder you should add the following files.

1. “Part.h” containing headers for the `Part`, `FH_Part`, `IT_Part`, and `FHIT_Part` classes from Assignment 4 Section 4.2-4.5 (you may break them into separate files if you prefer).
2. “Part.cc” containing source for the `Part`, `FH_Part`, `IT_Part`, and `FHIT_Part` classes from Assignment 4 Section 4.2-4.5 (you may break them into separate files if you prefer).
3. A Makefile.

You will zip the “tutorial8” directory into a file “tutorial8.zip”. If you are doing this in the course VM you must do this from the command line. Open a terminal in the folder that contains “tutorial8”. Use the command `zip -r tutorial8.zip tutorial8`. This will zip the `tutorial8` folder, or update it if you change the contents. Submit `tutorial8.zip` to Brightspace by the deadline. DO NOT USE .tar OR .tar.gz FILES. Use .zip only please.

2 Testing Your Tutorial With `t8test.py`

`t8test.py` is a test script that is very similar to what will be used to mark your tutorial (basically I will change the input and expected output ... unless I get lazy, then I won’t change anything). So the mark you see here should be the mark you receive (as long as you did not hard code output. If you try a shortcut you might get burned). To run `t8test.py`, open a command line in the directory containing `t8test.py`. You may have to make it executable, so type `chmod +x t8test.py`. You may run the script as is, in which case it will look for a file to unzip. Or, if you have not zipped your files yet you may supply a “-nozip” argument, in which case it looks for the “tutorial8” folder.

To have the script unzip `tutorial8.zip` and then test your code, run `./t8test.py`. To skip the unzip step use `./t8test.py -nozip`. When your tutorial is being officially marked we expect a zipped file.

Running this script will generate a file “results.txt” just outside of the “tutorial8” folder. This will have some useful output as well as the mark.

3 Learning Outcomes

In this tutorial you will learn about multiple inheritance and polymorphism.

4 Instructions

4.1 Overview

In this tutorial you will write the abstract `Part` class from Assignment 4, Section 4.2. You will write the `FH_Part` and `IT_Part` classes from Sections 4.3 and 4.4 which should subclass `Part`. Finally write the `FHIT_Part` class from Section 4.5 which should be a subclass of both `FH_Part` and `IT_Part` (and thus you must solve the diamond problem of multiple inheritance).

IMPORTANT NOTE: The assignment asks you to override the `ostream <<` operator. However we (probably) have not learned that yet. As such this tutorial WILL NOT test the `ostream <<` operator, so you may leave it out of these classes for now if you wish.

4.2 Part Class

Complete Section 4.2 in Assignment 4.

4.3 FH_Part Class

Complete Section 4.3 in Assignment 4.

4.4 IT_Part Class

Complete Section 4.4 in Assignment 4.

4.5 FHIT_Part Class

Complete Section 4.5 in Assignment 4.

4.6 Makefile

A Makefile is provided, should you wish to use it. It assumes all `Part` class definitions are in the same .h file, and all `Part` class implementations are in the same .cc file. It compiles two object files, `Date.o` and `Part.o`, and links these object files to the `test` executable. In addition any Makefile you provide should contain an `all` command that creates the `test` executable and a `clean` command that removes all executables and object files.

4.7 t8test.py

Run this python script to test the classes described above. Correct all errors.