CS 112 - Week 13 Lab Exercise - 2022-11-18

Deadline

Due by the end of lab on 2022-11-18.

How to submit

Submit your `.cpp` and `.h` files for the problems below on https://canvas.humboldt.edu.

IF you prefer, you may instead compress your `.cpp` and `.h` files to be submitted into a single `.zip` file and submit that `.zip` file to Canvas.

(I'll also accept the `.zip` file created when one downloads a folder from the CS50 IDE, as long as it includes all of your lab's `.cpp` and `.h` files -- I suspect it will also contain your resulting executables, but that's OK.)

Purpose

To practice writing a derived class.

Important notes

- Be sure to put BOTH of your names and today's date in each of the files for this lab exercise.
- When you are done, or before you leave lab, the driver/whoever's account has the lab exercise files should e-mail a copy of all of the files to BOTH/ALL of you, and EACH of you should submit these files on Canvas.

Lab Exercise Setup

- FIRST: in the CS50 IDE, in your folder for today's lab exercise, create copies of the following:
  - `Point.h` and `Point.cpp` from **Week 13 Lecture 2** (so you get the revised `to_string` method now using an `istringstream`! 8-))

Practice writing a derived class

Create a derived class `ThreeDPoint`, in files `ThreeDPoint.h` and `ThreeDPoint.cpp`, derived from base class `Point`.

You will NOT modify the `Point` files for this!

Perform the following tasks:

- (Of course, make sure all of your names are in `ThreeDPoint.h` and `ThreeDPoint.cpp`!)
- The `ThreeDPoint` class should inherit all the characteristics of a `Point` object, and add the following data field (placed in the `private` area of the `ThreeDPoint` class definition):
  - `z` - which is the z coordinate of a point (x, y, z) in three-dimensional space
- The `ThreeDPoint` class should have two constructor methods:
  - a no-argument constructor (that should initialize each of the data fields `x`, `y`, and `z` to 0) and

  - a three-argument constructor (which allows the caller to specify initial values for the data fields `x`, `y`, and `z`)
  - Each should fully initialize the data fields of both `Point` and `ThreeDPoint` classes.
  - These constructors should call the corresponding `Point` constructor to set the initial values of the `Point` data fields, and will also set the initial value of the `z` data field.
- The `ThreeDPoint` class should have one accessor method `get_z`

  and one mutator method `set_z`

  ...for the new data field `z`.
- Write a REDEFINED method `display`, that writes a `ThreeDPoint`'s info to the screen, including all the data fields in both `Point` and `ThreeDPoint` classes.
- Write a REDEFINED method `to_string`, that returns a `string` depiction of a `ThreeDPoint`'s info, including all the data fields in both `Point` and `ThreeDPoint` classes.
- Write a version of the == operator that expects a `ThreeDPoint` object and returns whether the calling `ThreeDPoint` object has the same `x`, `y`, and `z` values, respectively, as the given `ThreeDPoint` object.
- Write a version of method `dist_from` that expects a `ThreeDPoint` object and returns the distance between the calling `ThreeDPoint` object and the given `ThreeDPoint` object.
  - The mathematical formula for the distance between two points (x1, y1, z1) and (x2, y2, z2) is:
    $$\sqrt{(x2-x1)^2+(y2-y1)^2+(z2-z1)^2}$$
  - Note: the `cmath` library includes the following methods:
    - `sqrt` - expects `double` number, returns the square root of that number
    - `pow` - expects a `double` number to raise to some power, and a `double` power to raise the first number to, and returns the result of raising that first number to the given power.

THEN, in a `main` function in a file `three-d-play.cpp`,

- Declare instances of `ThreeDPoint` using both of its constructors
- Call each of its (inherited AND newly created!) accessors at least once, printing to the screen the result of comparing their results to what they SHOULD return
- Call each of its (inherited AND newly created!) mutators at least once,

  and then print to the screen the results of comparing the resulting values of those changed data fields to what they SHOULD return
- Compare the results of `dist_from` to what it should return when called with at least your `ThreeDPoint` instances, and when called from one of your `ThreeDPoint` instances with that `ThreeDPoint` instance as its argument (to determine the distance between a three-d point and itself... 8-))
- Compare the results of `to_string` to what it should return for at least one of your `ThreeDPoint` instances
- Call `display` on each `ThreeDPoint`, preceded by printing a description of what one SHOULD see

Submit your resulting `three-d-play.cpp`, `ThreeDPoint.h`, and `ThreeDPoint.cpp`

- When you are done, or before you leave lab, use Gmail to
  - MAIL a copy of ALL of the resulting files for these programs to BOTH of you, and
  - EACH of you should SUBMIT the required files on Canvas