Homework 3: Object-Oriented Design

Instructor: Mehmet Emre

CS 32 Spring '22

Due: 4/13 12:30pm
Name & Perm #:
Homework buddy (leave blank if you worked alone):
Reading: Object Oriented Design, PS 10.2

1

According to Savitch in PS, the scope resolution operator `::` and the dot operator `.` have a similar purpose, but there is a major difference between them.

1. (5 pts) What do they have in common?

2. (5 pts) What is the difference between them?

In a class for an object representing a video game character:

2 (5 pts)

Would `setHealth` be an accessor or a mutator function? Circle one: accessor mutator

3 (5 pts)

Would `getName` be an accessor or a mutator function? Circle one: accessor mutator

4 (5 pts)

What does the term *encapsulation* refer to?

5 (5 pts)

According to Savitch in PS, it is normal practice to make member functions private under what circumstances?

6

On p. 579 and then again on p. 586 Savitch drives home a point about the syntax of invoking a constructor– what you SHOULD do when invoking a no-arg constructor, and what you should NOT do. Though he doesn't mention it, this is a "trap" that many C++ learners fall into if they learned Java first, because this is a place where C++ and Java syntax differ significantly.

Suppose you have a no-arg constructor for a class `Student`.

1. (5 pts) What is the correct syntax to declare a local variable inside a function or method called `s` that is of type `Student`, and is created with the no-arg constructor?

2. (5 pts) What is the "wrong" syntax for doing that same thing that Savitch specifically warns *against* doing?

3. (5 pts) In one of the passages where this is explained, Savitch indicates WHY this other syntax is wrong if your intention is to make `s` an instance of class `Student`. This syntax actually has a completely different meaning in C++. What does this alternate syntax mean–that is, what would `s` be declared to be under this alternate syntax?

4. (5 pts) Suppose you have a class `Student` with private data members as shown below. You could write a constructor like this:

```
class Student {
    private:
       int perm_;
       std::string name_;
  }

  Student::Student(int perm, std::string name) {
     perm_ = perm;
     name_ = name;
  }
```

However, there is an alternate way to initialize data members in the so called "constructor initialization section" described in Section 10.2 of PS. Rewrite this constructor so that the body is an empty set of braces, and the code is moved to the so-called "constructor initialization section".[1]

[1]Note that I am *not* referring to C++11's "constructor delegation" described on p. 587-588; this is a feature that has been in C++ for many years before C++11, and is described somewhere before p. 581-584.