Private Base-Class Members

Private base-class data members are not directly accessible to the derived class object member functions. To see how this works, in the editor, change the `Student` constructor so that it attempts to set the `private name` data member directly, (instead of using the `setName()` member function).

```
Student::Student(const string sname, long sid)
{
    // setName(sname);  // comment this out
    name = sname;       // add this;
    studentID = sid;
}
```

Type `make` in the console. You'll see an error message that looks something like this:

```
student.cpp:8:3: error: string Person::name is private here:
  name = sname;
  ^~~~
```

Even though you can't access the `private name` data member from the derived `Student` class, the data member **exists inside the `Student` object nonetheless**. You know that is true, because the **inherited** `setName()` and `getName()` member functions work correctly, and without the existence of the `private` data member, that would not be possible.

> *Private data members **are** inherited, in the sense that each derived class object contains a copy of each `private` variable defined in the base class, even though the member functions in the derived object are not free to directly access that variable.*