Calling the Base Constructor

Just as you can initialize data members before the constructor runs, you can initialize your object's "base part" by **calling the base class constructor in the initializer list** .

When you do this, the `Student` constructor invokes the `Person(String)` constructor, instead of using the `setName()` member function, as you've done up until now. This is the **normal way to write derived class constructors**.

```
Student::Student(const string sname, long sid)
    : Person(sname)   // this initializes name
{
    // setName(sname); // Remove this!!!
    studentID = sid;
    cout << "Calling Student(" << name << ", "
        << sid << ")\n";
}
```

Now, when you run the sample program, instead of **implicitly** calling the `Person` default constructor (which no longer exists), you **explicitly** chain to the `Person(string)` constructor to initialize the `name` data member.