Final Classes

To prevent `Imposter` classes, when you design a base class, consider which functions you want to allow others to extend and which ones should be "set in stone". No one should ever change `getName()`, so you can **seal it using `final`** like this:

```
class Person
{
public:
    virtual std::string getName() const final;
};
```

When a member function is marked `final` then derived classes are **prevented** from overriding it and we would see an error message like this:

```
Person.cpp:33:12: error: virtual function 'virtual
      std::string Imposter::getName() const'
      string getName() const
      ^
Person.cpp:21:8: error: overriding final function
      'virtual std::string Person::getName() const'
      string Person::getName() const { return name; }
      ^
```

Only `virtual` functions can be marked `final`, (which is really annoying). When designing a collection of classes, you normally **won't want all of the classes to be extensible**. To prevent others from extending your class, add `final` to the class header, just like you did for the method. If you make a class `final`, then there is no reason to make the methods `final` as well.