Redefining Functions

A derived class may replace an inherited member function, which the class designer wanted left alone. In the `Person` class, `getName()` returns the name of the person. **This works fine as is**; there's no reason for a derived class to change it.

However, **nothing prevents a derived class from redefining it** like this:

```
class Imposter : public Person
{
public:
    string getName() const
    {
        return "Emperor " + Person::getName();
    }
};
```

The derived class has no access to the `private` member `name`. However, because the `getName()` member function can be **redefined**, the derived class was able to effectively gain access to this field. And, because of the **principle of substitutability**, the `Person` that your function receives as a parameter may actually be an `Imposter`.