# Getter & Setter Patterns

**The pattern of pairing a "getter" along with a "setter" function is common,** and you will see it in any major C++ project you work on. Unlike Java, the actual `get*` and `set*` name pattern is not as common. Instead, what programmers often do is write **a pair of overloaded functions**.

Instead of the name `getHours()` or `setHours()`, use the name `hours()` for both of them.

- The accessor is `const` and returns a value.
- The non-`const` mutator returns a reference, which can be assigned to.

```
Time Time::hours() const { return m_hours; }  // accessor
Time& Time::hours()                           // mutator
{
    if (h < 0 || h > 23) throw out_of_range("...");
    return m_hours;
}
```

In general, it is safer to allow clients to **read the values** of the data members than it is allow them to **change** those values. As a result, "setters" are far less common than "getters" in class design. Classes with no mutators at all, are called **immutable classes**.