Assign, Copy & Destroy

You may assign one object to another, just as you can assign one `int` variable to another, even though the data members are `private`. As with the built-in types, the result is a **copy** of the data in the original members.

With class types, this is carried out by the overloaded **assignment operator**.

```
Time& operator=(const Time& rhs);
```

The assignment operator **is not used** when you pass an object by value, or **initialize** a new object variable with another. Instead, the **copy constructor** is called:

```
Time(const Time& rhs);
```

When an object is destroyed, its **destructor** is called. If your class allocates dynamic memory in the constructor, for instance, you would free it in the destructor. The destructor looks like the default constructor preceded by the **tilde**:

```
~Time();
```

C++ automatically writes a *synthesized* assignment operator, a *synthesized* copy constructor, and a *synthesized* destructor which work for simple types such as those in this course. In CS 250 you'll learn how to write your own assignment operators, copy constructors, and destructors to create more dynamic types.

You can **prevent** pass by value or assignment by adding the following to the definitions.

```
Time& operator=(const Time& rhs) = delete;
Time(const Time& rhs) = delete;
```