Class Definition Syntax

▶ *The Time Class Definition*

To create a class definition for `Time`, similar to the structure from last week, follow these rules:

1. Instead of `struct`, use the `class` keyword. There is no `public` in front of this as with Java.
2. The `public` keyword, followed by a colon, indicates the start of the **public interface**. Here we **prototype** the member functions it contains.
3. The member functions `hours()`, `minutes()`, `sum()`, `difference()` and `write()`, all access the `hours` and `minutes` data members **without changing them**. When this is the case, add the `const` keyword **after the argument list**. We say these functions are **accessors**.
4. The `read()` member function does **modify** the `Time` object. This is called a **mutator**.
5. The class definition **ends with a semicolon**, just like a structure. This is not optional.

Data Members

Most of the implementation will appear inside a `.cpp` file. **Defining the data members** which store **object state**, is **written inside the header file instead**. A common practice is to use a special indicator like `m_` to show that it is a data member.

The `Time struct` used two individual data members: one for `hours` and one for `minutes`. This is fine; it allows you to store all of information needed. By adding `private`, you can **prevent clients of `Time` from accessing the data members directly.**