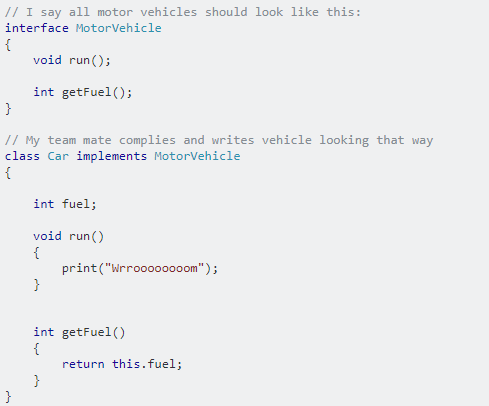
**Interfaces and Classes**

**An interface is an empty shell**. There are only the signatures of the methods, which implies that the methods do not have a body. The interface can't do anything. It's just a pattern.

For example (pseudo code):



Implementing an interface consumes very little CPU, because it's not a class, just a bunch of names, and therefore there isn't any expensive look-up to do. It's great when it matters, such as in embedded devices.

Abstract classes

Abstract classes, unlike interfaces, are classes. They are more expensive to use, because there is a look-up to do when you inherit from them.

Abstract classes look a lot like interfaces, but they have something more: You can define a behavior for them. It's more about a guy saying, "these classes should look like that, and they have that in common, so fill in the blanks!".

For example:

