# What does Encapsulation Mean?

In Java, encapsulation means hiding things by making them private or inaccessible.

# Why hide things?

Why would we want to hide things in Java?

- To make the interface simpler, we may want to hide unnecessary details.
- To protect the integrity of data on an object, we may hide or restrict access to some of the data and operations.
- To decouple the published interface from the internal details of the class, we may hide actual names and types of class members.

# What do we mean by interface here?

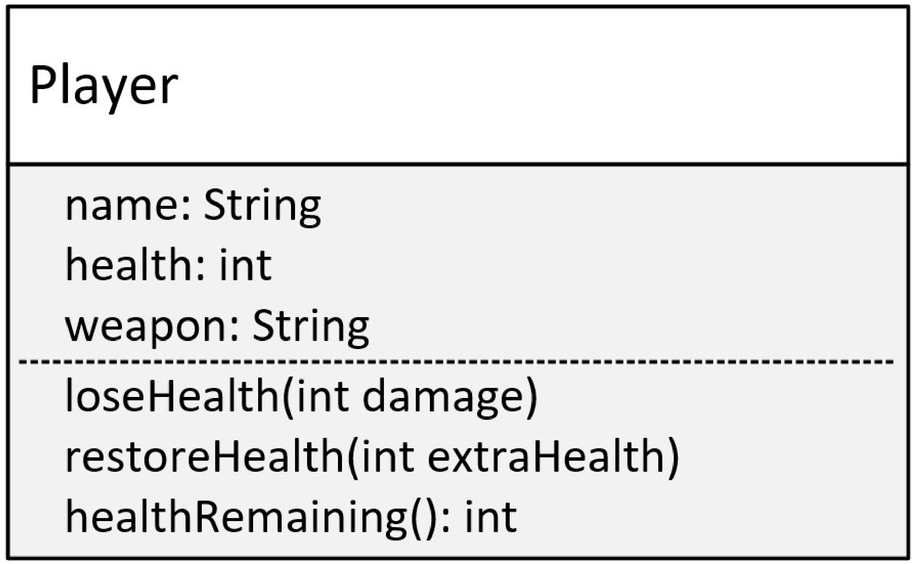
Although Java has a type called interface, that's not what I'm talking about here.

When I talk about a class's public or published interface, I'm really talking about the class members that are exposed to or can be accessed by the calling code.

Everything else in the class is internal or private to it.

An application programming interface or API is the public contract that tells others how to use the class.

# The Player Class




This is the model for a Player class.

The Player has three fields: name, health, and weapon.

This class has three methods: loseHealth(), restoreHealth(), and healthRemaining(), which I'll explain shortly.

I'm going to create this class without using encapsulation.

# Problem One

Allowing direct access to data on an object can potentially bypass checks and additional processing your class has in place to manage the data.

# Problem Two

Allowing direct access to fields means calling code would need to change when you edit any of the fields.

# Problem Three

Omitting a constructor that would accept initialization data means the calling code is responsible for setting up this data on the new object.