# Introduction to Classes and Objects

In this video, I'm going to start talking about **Object-Oriented Programming.**

# What is Object Oriented Programming

What is Object-Oriented Programming?

Object-Oriented Programming is a way to model real-world objects as software objects which contain both data and code.

OOP is a common acronym for Object-Oriented Programming.

# Class-based Programming

Class-based programming starts with classes which become the blueprints for objects.

But what does this really mean?

To start, we need to understand what objects are.

They're really the key to understanding this object-oriented terminology.

# Real World Object Exercise

What I'd like you to do is just have a look around in the area you're sitting in right now.

And if you do that, you'll find that there's many examples of real-world objects.

For example, I'm sitting here, and I can see:

- A computer.
- I can see a keyboard.
- I can see a microphone.
- I can see shelves on the wall.
- I can see a door.

All of these are examples of real-world objects.

# State and Behavior

Real-world objects have two major components:

- state.
- and behavior.

# State (computer)

State, in terms of a computer object, might be:

- The amount of RAM it has.
- The operating system it's running.
- The hard drive size.
- The size of the monitor.

These are characteristics about the item that can describe it.

# State (ant)

I could also describe animate objects like people, or animals, or even insects like an ant.

For an ant, the state might be:

- The age.
- The number of legs.
- The conscious state.
- Whether the ant is sleeping or is awake.

# Behavior (computer)

In addition to state, objects may also have behavior or actions that can be performed by the object, or upon the object.

Behavior, for a computer, might be things like:

- Booting up.
- Shutting down.
- Beeping or outputting some form of sound.
- Drawing something on the screen, and so on.

All of these could be described as behaviors for a computer.

# Behavior (ant)

For an ant, behavior might be:

- Eating.
- Drinking.
- Fighting.
- Carrying food, those types of things.

# State and Behavior

Modelling real-world objects as software objects is a fundamental part of Object-Oriented Programming.

A software object stores its state in fields, which can also be called variables or attributes.

Objects expose their behavior with methods which I've talked about before.

So, where does a class fit in?

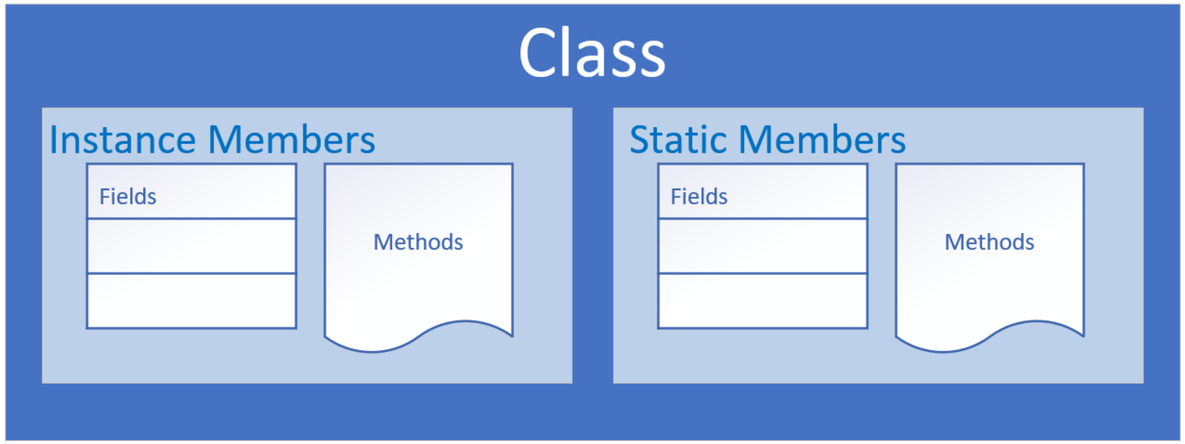
Well, think of a class as a template or a blueprint for creating objects.

Let's take another look at the class.

# The class as the blueprint

The class describes the data (fields), and the behavior (methods), that are relevant to the real-world object we want to describe.

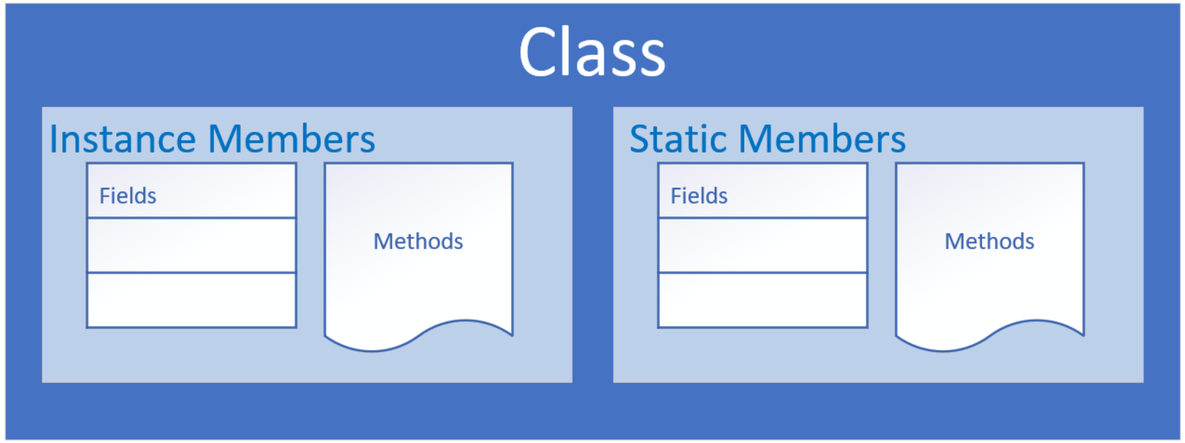
These are called class members.




A class member can be a field or a method, or some other type of dependent element.

If a field is static, there is only one copy in memory, and this value is associated with the class or template itself.

# The class as the blueprint




If a field is not static, it's called an instance field, and each object may have a different value stored for this field.

A static method can't be dependent on any one object's state, so it can't reference any instance members.

In other words, any method that operates on instance fields needs to be non-static.

# Organizing classes

Classes can be organized into logical groupings which are called packages.

You declare a package name in the class using the package statement.

If you don't declare a package, the class implicitly belongs to the default package.

# Access modifiers for the class

A class is said to be a top-level class if it is defined in the source code file and not enclosed in the code block of another class, type, or method.

A top-level class has only two valid access modifier options: public or none.

| Access keyword | Description |
|---|---|
| public | `public` means any other class in any package can access this class. |
| | When the modifier is omitted, this has special meaning, called package access, meaning the class is accessible only to classes in the same package. |

# Access modifiers for class members

An access modifier at the member level allows granular control over class members.

The valid access modifiers are shown in this table from the least restrictive to the most restrictive.

| Access keyword | Description |
|---|---|
| public | `public` means any other class in any package can access this class. |
| protected | `protected` allows classes in the same package, and any subclasses in other packages, to have access to the member. |
| | When the modifier is omitted, this has special meaning, called package access, meaning the member is accessible only to classes in the same package |
| private | `private` means that no other class can access this member |

# Encapsulation

Encapsulation in Object-Oriented Programming usually has two meanings.

One is the bundling of behavior and attributes on a single object.

The other is the practice of hiding fields and some methods from public access.