Theory: Introduction to OOP

6 minutes 3 / 12 problems solved

Start practicing

Fundamentals

Object-oriented programming (OOP) is a programming paradigm based on the concept of **objects** that interact with each other to perform the program functions. Each object can be characterized by a state and behavior. An object keeps the current state in the **fields** and the behavior in the **methods**.

Basic principles of OOP

There are four basic principles of OOP. They are **encapsulation, abstraction, inheritance,** and **polymorphism.**

- **Data encapsulation** is the mechanism of hiding the internal data of objects from the world. All interaction with the object and its data are performed through its public methods. Encapsulation allows programmers to protect the object from inconsistency.
- **Data abstraction** means that objects should provide the simplified, abstract version of their implementations. The details of their internal work usually aren't necessary for the user, so there's no need to represent them. Abstraction also means that only the most relevant features of the object will be presented.
- **Inheritance** is a mechanism for defining parent-child relationships between classes. Often objects are very similar, so inheritance allows programmers to reuse common logic and at the same time introduce unique concepts into the classes.
- **Polymorphism** literally means *one name and many forms*, and it concerns the inheritance of the classes. Just as the name suggests, it allows programmers to define different logic of the same method. So, the name (or interface) stays the same, but the actions performed may be different. In practice, it is done with overloading or overriding.

These are the key concepts of OOP. Each object-oriented language implements these principles in its own way, but the essence stays the same from language to language.

Objects

The key notion of the OOP is, naturally, an **object.** There are a lot of real-world objects around you: pets, buildings, cars, computers, planes, you name it. Even a computer program may be considered as an object.

It's possible to identify some important characteristics for real-world objects. For instance, for a building, we can consider a number of floors, the year of construction and the total area. Another example is a plane that can accommodate a certain number of passengers and transfer you from one city to another. These

characteristics constitute the object's attributes and methods. Attributes characterize objects' data or states, and methods — its behavior.

In OOP, everything can be considered an object. Programs are made from different objects interacting with each other. An object's state and behavior are usually placed together, but it's not always so. Sometimes, we will see objects without a state or methods. This, of course, depends on the purpose of the program and the nature of an object.

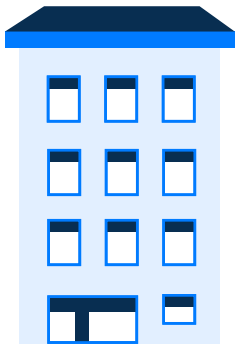
Classes

Often, many individual objects have similar characteristics. We can say these objects belong to the same **type** or **class**.

A class is another important notion of OOP. A class describes a common structure of similar objects: their fields and methods. It may be considered a template or a blueprint for similar objects. An object is an individual **instance** of a class.

Let's look at some examples below.

Example 1. The building class



An abstract building for describing buildings as a type of object (class)

Each building has the same attributes:

- the number of floors (an integer number);
- area (a floating-point number, square meters);
- year of construction (an integer number).

Each object of the building type has the same attributes but different values.

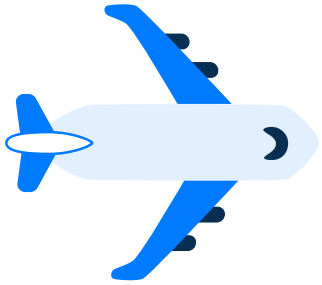
For instance:

- Building 1: the number of floors = 4, area = 2400.16, year of construction = 1966;
- Building 2: the number of floors = 6, area = 3200.54, year of construction = 2001.

It's quite difficult to determine the behavior of a building. But this example demonstrates attributes pretty well.

Example 2. The plane class

Unlike the building, it is easy to define the behavior of a plane. It can fly and transfer you between two points on the planet.



An abstract plane for describing all planes as a type of object (class)

Each plane has the following attributes:

- a name (a string, for example, "Airbus A320" or "Boeing 777");
- passengers capacity (an integer number);
- standard speed (an integer number);
- current coordinates (they are needed to navigate).

Also, it has a behavior (a method): transferring passengers from one geographical point to another. This behavior changes the state of a plane, namely, its current coordinates.

Conclusion about objects and classes

To put it concisely, you should remember the following:

- an object-oriented program consists of a set of interacting objects;
- as a rule, the internal state of an object is hidden;
- an object may have characteristics: fields and methods;
- an object is an instance of a class (type);
- a class is a more abstract concept than an individual object; it may be considered a template or blueprint that describes the common structure of a set of similar objects.