Theory: Introduction to OOP

7 minutes 11 / 11 problems solved

Start practicing

23348 users solved this topic. Latest completion was **13 minutes ago**.

§1. 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**.

§2. Basic principles of OOP

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

- **Encapsulation** ensures the bundling (=encapsulating) of data and the methods operating on that data into a single unit. It also refers to the ability of an object to hide the internal structure of its properties and methods.
- **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 this is about class inheritance. It allows programmers to define different logic for the same method. Thus, the name (or interface) remains the same, but the actions performed may differ.
  Example: the site needs three types of publications - news, announcements, and articles. They are somewhat similar - they all have a headline and text, news, and announcements have a date. In some ways they are different - articles have authors, news have sources, and announcements have a date after which it becomes irrelevant. It is convenient to write an abstract class with general information for all publications, so as not to copy it every time. And store what is different in the appropriate derived classes.

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.

§3. 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 of 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 viewed as an object, a class, for example, is also an object. Programs are made up of different objects that interact with each other. Object state and behavior are usually combined, but this is not always the case. Sometimes we see objects without state or methods. This, of course, depends on the purpose of the program and the nature of the object.

Current topic:

✓ Introduction to OOP Stage 3 3★ ···

Topic is required for:

Encapsulation ···

Interfaces ···

Inheritance ···

✓ The concept of patterns ···

Data and object mapping ···

Unit testing ···

✓ Class 3★ ···

✓ Defining classes Stage 3 ···

Declaring classes ···

Table of contents:

For example, there is such a thing as an interface. Not a user interface, but a class that only serves to be inherited from, in order to guarantee an interface to its descendant classes. It is a stateless class. Structures exist in C++ for historical reasons. This is such a data type. Now a structure is also a class, but once upon a time a structure had only properties and did not have any methods. A type for storing data and nothing else. These are special cases and are sometimes useful.

§4. Classes

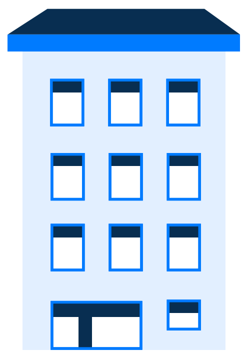
Often, many individual objects have similar characteristics. We can say these objects belong to the same **type** or **class**.

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.

In accordance with the principle of encapsulation mentioned above, any class should be considered a black box, i.e. the user of the class should see and use only the interface part of the class, namely, the list of declared properties and methods, and should not delve into the internal implementation.

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.

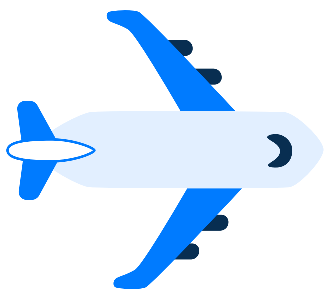
For instance:

- 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.

§5. 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;
- according to the principle of encapsulation, the internal implementation of the object is not accessible to the user;
- 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.