Background of Object-oriented Programming (OOP)

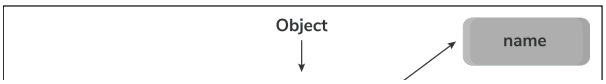
Get a brief introduction to object-oriented programming and its building blocks.

Definition

We use programming to solve real-world problems, and it won't make much sense if one can't model real-world scenarios using programming languages. This is where object-oriented programming comes into play. **Object-oriented programming**, also called OOP, is a programming model that is dependent on the concept of *objects* and *classes*.

OOP is a programming style, not a tool, so despite being old, it's vastly popular and established. This programming style involves dividing a program into pieces of objects that can communicate with each other. Every object has its own unique set of properties. These properties are later accessed and modified through the use of various operations.

Check out the illustration below. It shows a real-life example of an employee record, where every employee can be considered an "object". Since every employee has a *name*, *age*, *salary*, and *designation*, all these can be considered as the properties of each employee.



Real-life example of an employee record

Building blocks of OOP

The following are the essential concepts of object-oriented programming:

- Attributes
- Methods
- Classes
- Objects

Classes and objects

In the real world, we can find many objects around us like cars, buildings, and humans. *What are the characteristics of these objects?* All these objects have some *state* and *behavior*.

Let's take an example of a calculator. It has a state, i.e., either it is *on* or *off*. It also has behaviors, i.e., we can perform addition, subtraction, multiplication, division, and many other operations on numbers. Therefore, we can say that **objects** have state(s) and behavior(s).

Interesting, isn't it? However, the question is, "*where do the objects come from?*"

The answer to the question above is **classes**. A **class** can be thought of as a *blueprint* for creating objects.

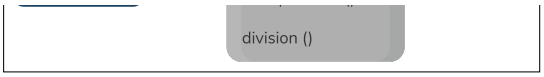
Attributes

Attributes are variables that represent the state of the object. In other words, if you were to implement the calculator object below in a computer program, variables could represent its state.

Methods

Methods are like functions that represent the behavior of the object. In other words, if you were to implement the calculator object below in a computer program, functions could represent its behavior. Methods have access to a class's attributes (and other methods). They can accept parameters, return values, and are used to perform an action on an object of a class.

The illustration below shows what a `Calculator` class should look like:



Calculator class with attributes and methods

Principles of OOP

The following are the four principles of object-oriented programming:

- Encapsulation

- Abstraction
- Inheritance
- Polymorphism

In the next few lessons, we will explain these four principles in detail.

Back lesson

Completed

Next

Introduction to the Course

Encapsulation