OBJECT ORIENTATION

What is object-oriented programming?

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained. This includes programs for manufacturing and design, as well as mobile applications; for example, OOP can be used for manufacturing system simulation software.

The organization of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups. Additional benefits of OOP include code reusability, scalability and efficiency.

The first step in OOP is to collect all of the objects a programmer wants to manipulate and identify how they relate to each other -- an exercise known as data modeling.

Examples of an object can range from physical entities, such as a human being who is described by properties like name and address, to small computer programs, such as widgets.

Once an object is known, it is labeled with a class of objects that defines the kind of data it contains and any logic sequences that can manipulate it. Each distinct logic sequence is known as a method. Objects can communicate with well-defined interfaces called messages**.**

What is the structure of object-oriented programming?

The structure, or building blocks, of object-oriented programming include the following:

Classes are user-defined data types that act as the blueprint for individual objects, attributes and methods.

Objects are instances of a class created with specifically defined data. Objects can correspond to real-world objects or an abstract entity. When class is defined initially, the description is the only object that is defined.

Methods are functions that are defined inside a class that describe the behaviors of an object. Each method contained in class definitions starts with a reference to an instance object. Additionally, the subroutines contained in an object are called instance methods. Programmers use methods for reusability or keeping functionality encapsulated inside one object at a time.

Attributes are defined in the class template and represent the state of an object. Objects will have data stored in the attributes field. Class attributes belong to the class itself.

What are the main principles of OOP?

Object-oriented programming is based on the following principles:

Encapsulation. This principle states that all important information is contained inside an object and only select information is exposed. The implementation and state of each object are privately held inside a defined class. Other objects do not have access to this class or the authority to make changes. They are only able to call a list of public functions or methods. This characteristic of data hiding provides greater program security and avoids unintended data corruption.

Abstraction. Objects only reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. The derived class can have its functionality extended. This concept can help developers more easily make additional changes or additions over time.

Inheritance. Classes can reuse code from other classes. Relationships and subclasses between objects can be assigned, enabling developers to reuse common logic while still maintaining a unique hierarchy. This property of OOP forces a more thorough data analysis, reduces development time and ensures a higher level of accuracy.

Polymorphism. Objects are designed to share behaviors and they can take on more than one form. The program will determine which meaning or usage is necessary for each execution of that object from a parent class, reducing the need to duplicate code. A child class is then created, which extends the functionality of the parent class. Polymorphism allows different types of objects to pass through the same interface.

What are examples of object-oriented programming languages?

While Simula is credited as being the first object-oriented programming language, many other programming languages are used with OOP today. But some programming languages pair with OOP better than others. For example, programming languages considered pure OOP languages treat everything as objects. Other programming languages are designed primarily for OOP, but with some procedural processes included.

For example, popular pure OOP languages include:

- Ruby
- Scala
- JADE
- Emerald
- Programming languages designed primarily for OOP include:
- Java
- Python
- C++
- Other programming languages that pair with OOP include:
- Visual Basic .NET
- PHP
- JavaScript

What are the benefits of OOP?

Benefits of OOP include:

Modularity. Encapsulation enables objects to be self-contained, making troubleshooting and collaborative development easier.

Reusability. Code can be reused through inheritance, meaning a team does not have to write the same code multiple times.

Productivity. Programmers can construct new programs quicker through the use of multiple libraries and reusable code.

Easily upgradable and scalable. Programmers can implement system functionalities independently.

Interface descriptions. Descriptions of external systems are simple, due to message passing techniques that are used for objects communication.

Security. Using encapsulation and abstraction, complex code is hidden, software maintenance is easier and internet protocols are protected.

Flexibility. Polymorphism enables a single function to adapt to the class it is placed in. Different objects can also pass through the same interface.