Object-oriented programming (OOP) is a computer programming paradigm in which a software system is modeled as a set of objects that interact with each other.

A programming paradigm is a paradigmatic style of programming (compare with a methodology which is a paradigmatic style of doing software engineering).

A programming paradigm provides (and determines) the view that the programmer has of the execution of the program.

Procedural programming is a programming paradigm based upon the concept of the modularity and scope of program code (i.e., the data viewing range of an executable code statement).

A main procedural program is composed of one or more modules (also called packages or units), either coded by the same programmer or pre-coded by someone else and provided in a code library.

Procedural programming offers many benefits over simple sequential programming since procedural code:
- is easier to read and more maintainable
- is more flexible
- facilitates the practice of good program design

Procedural programming languages facilitate the programmer's task in following a procedural programming approach.

Each module is composed of one or more subprograms (which may consist of procedures, functions, subroutines or methods, depending on programming language).

It is possible for a procedural program to have multiple levels or scopes, with subprograms defined inside other subprograms.

Each scope can contain names which cannot be seen in outer scopes.

An object is fundamental concept in object-oriented programming.

Generally, something is an object if it has a name, properties associated with it, and messages that it can understand.

Objects are the basic units of modeling in object-oriented design and the basic units of programs in object-oriented programming.
**Modelling + coding**

- OOP involves:
  - modeling the software system as a set of interacting (conceptual) objects in object-oriented design,
  - then implementing (coding) the design using an object-oriented programming language with (programming language) objects.

**Object characteristics**

- Objects have three defining characteristics:
  - identity, state, behavior.
- An object has an identity if that object can be distinguished from other objects (usually with a name).
- The state of an object describes the properties (or data stored) in the object.
- The behavior of an object describes the messages the object can understand (methods in the object's interface).

**Object characteristics**

- Objects have three defining characteristics:
  - identity, state, behavior.
- An object has an identity if that object can be distinguished from other objects (usually with a name).
- The state of an object describes the properties (or data stored) in the object.
- The behavior of an object describes the messages the object can understand (methods in the object's interface).

**Objects**

- The state of an object can be referred to using a number of terms: properties, attributes, members, or data members.
- The messages that an object understand can be referred to as methods or member functions.
- The terminology depends on the context of use.
  - For example, in C++ objects have data members and member functions, while in Java objects have attributes and methods.
- Objects are instances of a class.

**Example**

- Assume we have a class called Dog that is capable of having certain properties and can understand certain messages.
- Then, we can create an instance of that class, an object named "Fido".
- The object Fido can have specific properties and can perform certain actions based on the messages it understands.
  - For instance, Fido can be male, brown, and can stay, sit, and roll over.

```java
Class Dog
define class Dog with {
    attribute gender in {male, female};
    attribute color in {white, brown, black, tan};
    ... other attributes ... 
    method fetch() returns Ball;
    method stay() returns nothing;
    method rollover() returns nothing;
    method eat(Food) returns nothing;
    ... other methods ... 
}
```

```java
Objects
create Fido;
Fido.gender = male;
Fido.color = brown;
create Princess;
Princess.gender = female;
Princess.color = white;
```
Methods

```java
if not Fido.isSitting() then
    Fido.sit();
    Fido.stay();
    Fido.eat(treat);
Princess.rollOver();
Princess.eat(treat);
```

Abstraction

- The ability for a program to ignore some aspects of the information that it is manipulating, i.e. the ability to focus on the essential.
- Each object in the system serves as a model of an abstract "actor" that can perform work, report on and change its state, and "communicate" with other objects in the system, without revealing how these features are implemented.
- Processes, functions or methods may also be so abstracted, and when they are, a variety of techniques are required to extend an abstraction:

Encapsulation

- Also called information hiding:
- Ensures that users of an object cannot change the internal state of the object in unexpected ways;
  - only the object's own internal methods are allowed to access its state.
- Each class exposes an interface (of an API – Application Programming Interface) that specifies how other classes may interact with it.

Encapsulation

- This prevents users from breaking the invariants of the class, which is useful because it allows the implementation of a class of objects to be changed for aspects not exposed in the interface without impact to user code.
- The definitions of encapsulation focus on the grouping and packaging of related information (cohesion) rather than security issues.
- OOP languages do not normally offer formal security restrictions to the internal object state.
- Using a method of access is a matter of convention for the interface design.

Polymorphism

- Different things or objects can have the same interface or answer the same message (based on message name) and respond appropriately depending on the thing's nature or type.
- This potentially allows multiple things to be interchangeable with each other:
  - If a bird receives the message "move fast", it will flap its wings and fly.
  - If a lion receives the same message, it will run with its legs.
  - Both answer the same request, but in ways appropriate to each creature.

Polymorphism

- Imagine, if you will, an operator + that may be used in the following ways:
  1. 1 + 2 → 3
  2. 3.14 + 0.0015 → 3.1415
  3. 1 + 3.7 → 4.7
  4. [1, 2, 3] + [4, 5, 6] → [1, 2, 3, 4, 5, 6]
  5. [true, false] + [false, true] → [true, false, false, true]
  6. "foo" + "bar" → "foobar"
Overloading

- To handle these six function calls, four different pieces of code are needed—or three, if strings are considered to be lists of characters:
  - In the first case, integer addition must be invoked.
  - In the second and third cases, floating-point addition must be invoked.
  - In the fourth and fifth cases, list concatenation must be invoked.
  - In last case, string concatenation must be invoked, unless this too is handled as list concatenation.
- Thus, the name + actually refers to three or four completely different functions. This is an example of overloading.

Inheritance

- Organizes and facilitates polymorphism and encapsulation by permitting objects to be defined and created that are specialized types of already-existing objects - these can share (and extend) their behavior without having to reimplement that behavior.
- This is typically done by grouping objects into classes, and defining classes as extensions of existing classes, thus and grouping classes into trees or lattices reflecting behavioral commonality.
- Although the use of classes is the most popular technique for inheritance, another well-known technique is Prototype-based programming.

C++

- In C and C++, the expression x++ increases the value of x by 1.
- The name "C++" is a play on this, suggesting an improvement upon C.
- Bell Labs' Bjarne Stroustrup developed C++ (originally named "C with Classes") during the 1980s as an enhancement to the C programming language.
- Enhancements started with
  - the addition of classes, followed by, among many features, virtual functions, operator overloading, multiple inheritance, templates, and exception handling.
- C++ is a general-purpose computer programming language.
- It is a statically typed free-form multi-paradigm language supporting procedural programming, data abstraction, object-oriented programming, and generic programming.
- During the 1990s, C++ became one of the most popular commercial programming languages.