The most likely smallest set of Object-Oriented Programming terminology ever written

Version 1.0

Aaron Ponti

Object-Oriented Programming

A type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure: in this way, the data structure becomes an object that includes both data and functions.

Abstraction

Abstraction is the process by which data and programs are defined with a representation similar to its meaning (semantics), while hiding away the implementation details.

Class

A **class** is a construct that is used as a blueprint to create instances of itself – referred to as class *instances*, class *objects*, *instance objects* or simply *objects*.

A class defines constituent **members** which enable these class instances to have state and behavior:

- **Data** field members (*member variables*, *instance variables*, *properties*) enable a class object to maintain state.
- Other kinds of members, especially **methods**, enable a class object's behavior.

Class instances are of the *type* of the associated class.

Encapsulation

Encapsulation is used to refer to one of two related but distinct notions, and sometimes to the combination thereof:

- a language mechanism for restricting access to some of the object's components;
- a language construct that facilitates the bundling of data with the methods (or other functions) operating on that data

Inheritance

Inheritance is a way to compartmentalize and reuse code by creating collections of attributes and behaviors called objects which can be based on previously created objects.

In *classical inheritance* where objects are defined by classes, classes can **inherit** other classes. The new classes, known as **subclasses** (or **derived classes**), inherit attributes and behavior (i.e. previously coded algorithms) of the preexisting classes, which are referred to as **superclasses** (or **ancestor classes**, or **base classes**). The inheritance relationships of classes gives rise to a **hierarchy**.

Message passing

Objects communicate with each other by sending and receiving **messages** (e.g. by calling each other's methods, or by broadcasting **events**).