# What are the main principles of OOP?

## Appropriate answers

### 4 principles

**Abstraction** is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user.

**Encapsulation** is the process of combining data and functions into a single unit called class

**Inheritance** enables you to create new classes that re-use, extend and modify the behaviour that is defined in other classes

**Polymorphism** is the ability to present the same interface for differing underlying forms (data types).

Polymorphism describes a pattern in object oriented programming in which classes have different functionality while sharing a common interface.

Both **overriding** and **overloading** are used to achieve polymorphism (some people say that they are not a polymorphism, others say that they are cases of poly, others say that overloading is a case of poly).

### 3 principles which describe language as OOP

There are 3 requirements for a language to be object-oriented:

* a language that supports only **encapsulation** (objects) is not object-oriented, but it is modular
* a language that supports just **encapsulation** (objects) and **message-passing** (polymorphism) is not object-oriented, but it is object-based
* a language that supports **encapsulation** (objects), **message-passing** (polymorphism), and **inheritance** (abstraction), is object-oriented

NOTE: message-passing enables polymorphism.

NOTE: Abstraction is a much more general concept; encapsulation et al are kinds of abstraction, just as a subroutine is a kind of abstraction. See [Abstraction](https://en.wikipedia.org/wiki/Abstraction_%28computer_science%29)

## 

## OOPS Principles Better way of explain in interview?

1. **Abstraction**

Show only necessary thing to user that he required, not extra information (use public private, protected). Abstraction is done when we need to inherit from certain class but do not instantiate the objects of that class.

2. **Encapsulation**

Group all relevant things together. I.e. encapsulation is wrapping/binding up of data and member functions in single unit. In simple, abstraction is hiding the implementation and encapsulation is to hide data.

3. **Inheritance**

If something already exist, why should I recreate it (same as re-inventing a wheel). Use inheritance to inherit all things of that class into your class. **Inheritance** enables you to create new classes that re-use, extend and modify the behaviour that is defined in other classes.

4. **Polymorphism**

When an object exhibits different behavior in different situation. In simple way, when a message can be processed in different ways/forms.

## 

## Answer 1 (22 upvotes)

* **Abstraction**We try to obtain abstract view, model or structure of real life problem, and reduce its unnecessary details. With definition of properties of problems, including the data which are affected and the operations which are identified, the model abstracted from problems can be a standard solution to this type of problems. It is an efficient way since there are nebulous real-life problems that have similar properties.
* **Encapsulation**Encapsulation is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible. (Data hiding: a language feature to restrict access to members of an object, reducing the negative effect due to dependencies. e.g. "protected", "private" feature in C++)
* **Inheritance**The idea of inheritance is simple, a class is based on another class and uses data and implementation of the other class. And the purpose of inheritance is Code Reuse.
* **Polymorphism**Polymorphism is the ability to present the same interface for differing underlying forms (data types). With polymorphism, each of these classes will have different underlying data. A point shape needs only two co-ordinates (assuming it's in a two-dimensional space of course). A circle needs a center and radius. A square or rectangle needs two co-ordinates for the top left and bottom right corners and (possibly) a rotation. An irregular polygon needs a series of lines.

Ref:

[1] <http://www.geeksforgeeks.org/commonly-asked-oop-interview-questions/>

[2] <http://gd.tuwien.ac.at/languages/c/c++oop-pmueller/tutorial.html>

[3] <https://stackoverflow.com/questions/16014290/simple-way-to-understand-encapsulation-and-abstraction>

[4] <https://stackoverflow.com/questions/1031273/what-is-polymorphism-what-is-it-for-and-how-is-it-used>

## Answer 2 (4 upvotes)

**Encapsulation**

Encapsulation is the mechanism of hiding of data implementation by restricting access to public methods.

**Abstraction**

Abstract means a concept or an Idea which is not associated with any particular instance. Using abstract class/interface we express the intent of the class rather than the actual implementation. In a way, one class should not know the inner details of another in order to use it, just knowing the interfaces should be good enough.

**Inheritance**

Inheritances expresses "is a" relationship between two objects. Using proper inheritance, In derived classes we can reuse the code of existing super classes.

**Polymorphism**

It means one name many forms. It is further of two types - static and dynamic. Static polymorphism is achieved using method *overloading* and dynamic polymorphism using method *overriding*.

## 

## Answer 3 (4 upvotes)

**Abstraction:** To represent the essential feature without representing the background details is abstraction. It lets you focus on what the object does instead of how it does it.

Abstraction provides you a generalized view of your classes or objects by providing relevant information. Mostly we make abstract class to generalized a concept.

Ex: generalized file upload, where you can upload any kind of file like word pdf or any other.

**Encapsulation:** Prevents access to implementation details i.e. Wrapping up a data member and a method together into a single unit. Example: Mobile Phones.

**Inheritance:** Inheritance is a process of object reusability. When a class includes a property of another class it is Inheritance. Example: parent-child relation.

**Polymorphism:** It means one form, many uses. Using Many forms of a single object.

Types:

* **Compile-time (static):** compiler identifies which polymorphism form it has to take and execute at compile time. (*Overloading*).
* **Run-time (dynamic):** compiler identifies which polymorphism form it has to take and execute at runtime but not at compile time. (*Overriding*).

**Abstraction** is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user. Consider your mobile phone, you just need to know what buttons are to be pressed to send a message or make a call, What happens when you press a button, how your messages are sent, how your calls are connected is all abstracted away from the user.

**Encapsulation** is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.

**Class** has attributes(i.e data) and behaviors (i.e methods that operate on that data).

