Object-Oriented Programming.

Ortiz Vega Angelo
Course Code: CE1103
Name: Algorithms and Data Structures I,
Academic Area of Computer Engineering.
Cartago, Costa Rica.

-Keywords: Object-Oriented Programming, Programming Paradigm.

-Content:

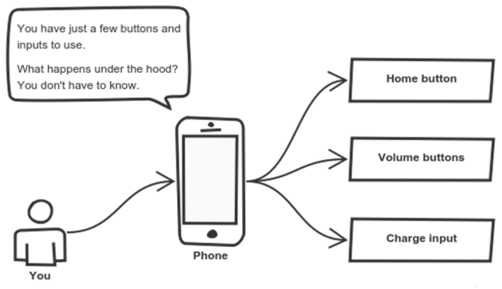
Principles of OOP:
~ Abstraction
~ Encapsulation
~ Classes
~ Attributes
~ Methods
~ Instances
~ Inheritance
~ Polymorphism

Thought process:
What are the objects? -> How do they interact? -> What are your attributes?

What is an object?
It is an abstraction of an object from real life. Like real objects, POO objects possess:
1. Attributes: Characteristics that identify the object [Adjective]
2. Behavior: Things that the object can do [Verbs]

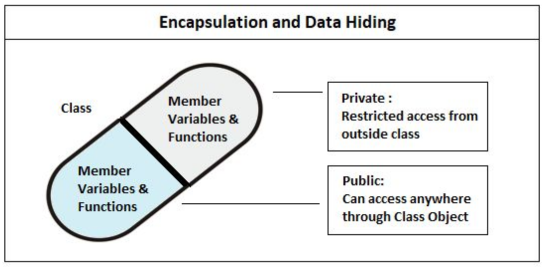
At a more technical level, an object is a block of memory that contains data and executable code. Each block is independent of each other.

Abstraction:
It is an imprecise representation of reality, it is an intrinsic characteristic of OOP. It refers to being able to construct objects that represent a concept of the real world, but focused on specific points. Programming is like an abstract painting.
Abstraction is a concept aiming to ease this problem. Applying abstraction means that each object should only expose a high-level mechanism for using it. This mechanism should hide internal implementation details. It should only reveal operations relevant for the other objects.


You have just a few buttons and inputs to use.
What happens under the hood? You don't have to know.
Home button
Volume buttons
Charge input
Phone
You

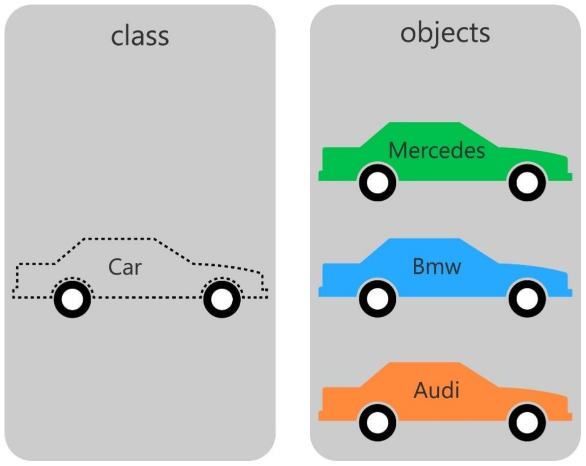
Encapsulation:
An object is a unit that contains data (attributes) and operations (methods), objects can decide which data and operations to expose and which to hide. The interaction between objects is done through well-defined interfaces.

Encapsulation is achieved when each object keeps its state private, inside a class. Other objects don't have direct access to this state. Instead, they can only call a list of public functions — called methods.So, the object manages its own state via methods — and no other class can touch it unless explicitly allowed. If you want to communicate with the object, you should use the methods provided. But (by default), you can't change the state.

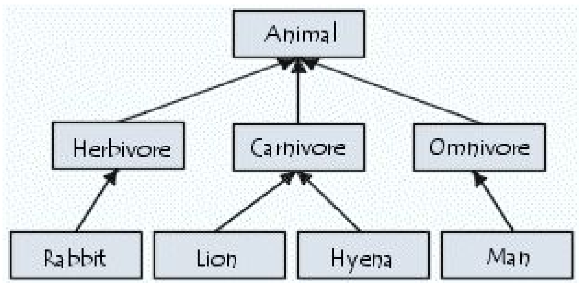

Encapsulation and Data Hiding
Class
Member Variables & Functions
Member Variables & Functions
Private : Restricted access from outside class
Public: Can access anywhere through Class Object

Classes:
Classes are the mold that allows to build instances of objects. The mold defines the characteristics of the resulting object.
An **attribute** is a data of the object that can have a specific value, they are independent between objects.

The **methods** are operations that interact with the object and its attributes, it allows to send messages to the objects.
The **instances** are the objects created from the class, each instance is independent of each other.



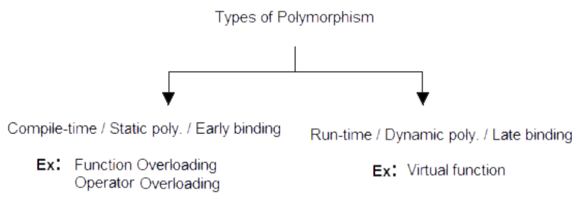
Inheritance:
In the real world, objects are part of a hierarchy. Each hierarchy shares attributes and behaviors as the hierarchy descends.
The objects are part of hierarchies that share attributes and methods with "children" objects. Objects enter into father-son relationships



Polymorphism:

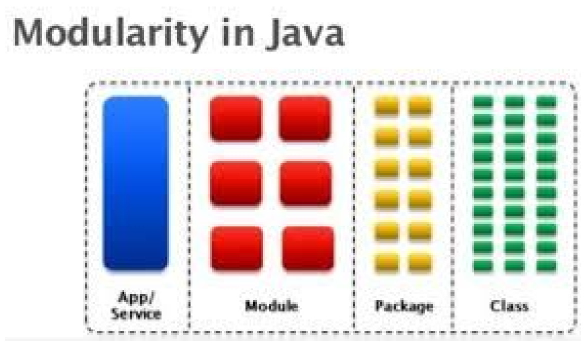
If you're wondering if an object is polymorphic, you can perform a simple test. If the object successfully passes multiple is-a or instanceof tests, it's polymorphic.

In computer science, it describes the concept that objects of different types can be accessed through the same interface. Each type can provide its own, independent implementation of this interface. It is one of the core concepts of object-oriented programming (OOP).



Modularity:

It is an intrinsic concept of OOP. A class or object can be seen as reusable modules. Modularity has to do with similar packages or namespaces, a package is a logical grouping of related classes. One or more packages can be exported as .jar files, to be included in different projects.



Criticism of OOP

The idea of object-oriented programming has been criticized by developers for multiple reasons. The largest concern is that OOP overemphasizes the data component of software development and does not focus enough on computation or algorithms. Additionally, OOP code may be more complicated to write and take longer to compile. Alternative methods to OOP include functional programming, structured programming and imperative programming, but most advanced programming languages give developers the option to combine them.