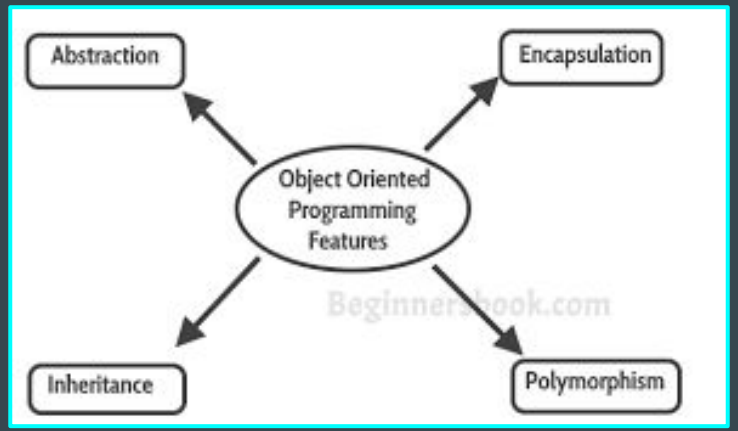
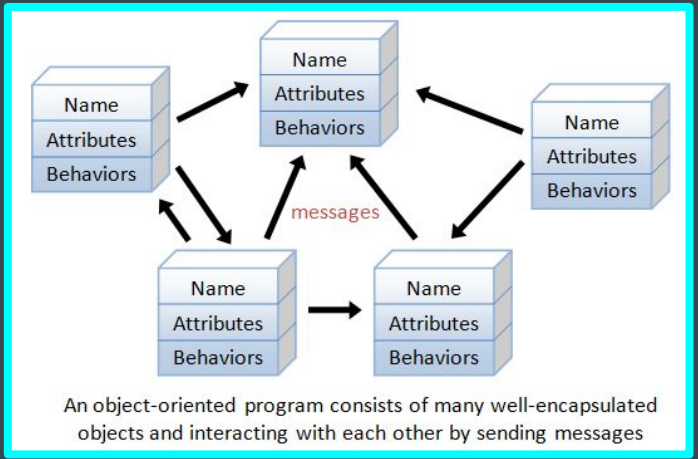
Object-Oriented Programming (OOP)



A tutorial by Joshua Rovner

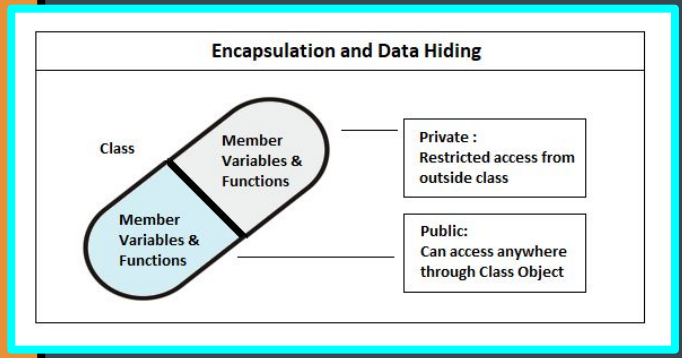
What is OOP?

- A style of programming in which classes serve as templates in order to create objects.
- Classes contain instance variables, which are the attributes of the object, and methods, which are its behaviours.
- As an example, a "human" class could contain variables such as height, weight, and eye color, while a method could age the human, adding 1 to their current age.



Abstraction + Encapsulation

- Abstraction refers to the hiding of a program's inner workings, so that important data cannot be altered by a client program.
- Instance variables must be declared private, and accessor and mutator methods must be made inside the class to alter the variables from the outside.
- **Encapsulation -** Accessors return the current value of a variable, and mutators set a new value specified by a parameter in the method call. Essentially, all variables and methods are put together into one class.



Inheritance

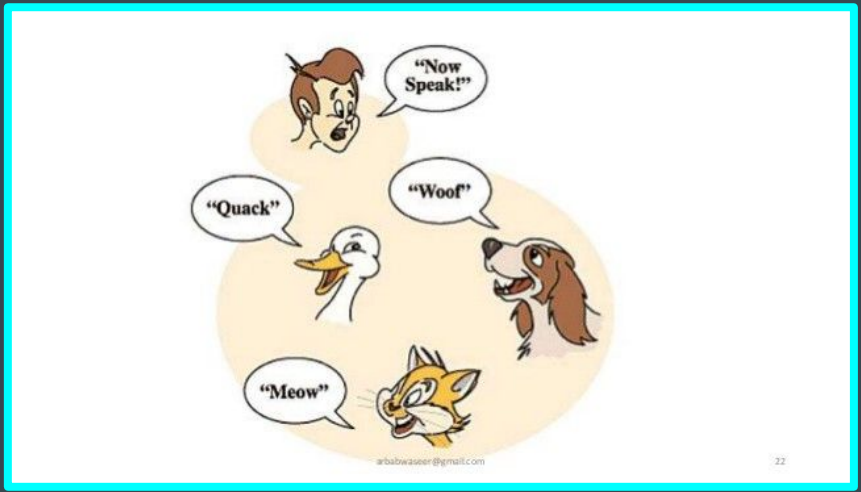
- Classes can extend other classes if one contains most of the properties of another. For example, a “dog” class could extend an “animal” class because a dog *is-an* animal. Animals can move, and since a dog *is-an* animal, it can move. But, a dog can bark, while a general animal cannot. In this case, the “dog” class would be considered a subclass, and the “animal” class a superclass.
- Subclasses inherit all non-private attributes and behaviours of their subclass.
- Keywords:
  - **Private** - no other class has access to the field or method.
  - **Public** - any other class has access to the field or method.
  - **Protected** - only subclasses have access to the field or method.

Types of Classes

- **Abstract classes** represent an abstract notion and require at least one abstract (un-implemented) method to be declared in the class. Subclasses that extend an abstract class must provide implementation for all the abstract methods in the abstract superclass.
- An **interface** contains only abstract methods, and can be thought of as a very basic layout for a general concept that can be used in many different ways, such as a board game grid. The grid can be altered for anything from chess to battleship.

Polymorphism

- Polymorphism is the ability of an object to convert between the type of a superclass and the type of a subclass at run-time. The compiler must make an immediate decision on which type to use.



- In this image, the human tells the animals to speak (method call). Each one makes their own unique sound, which illustrates the concept of polymorphism.