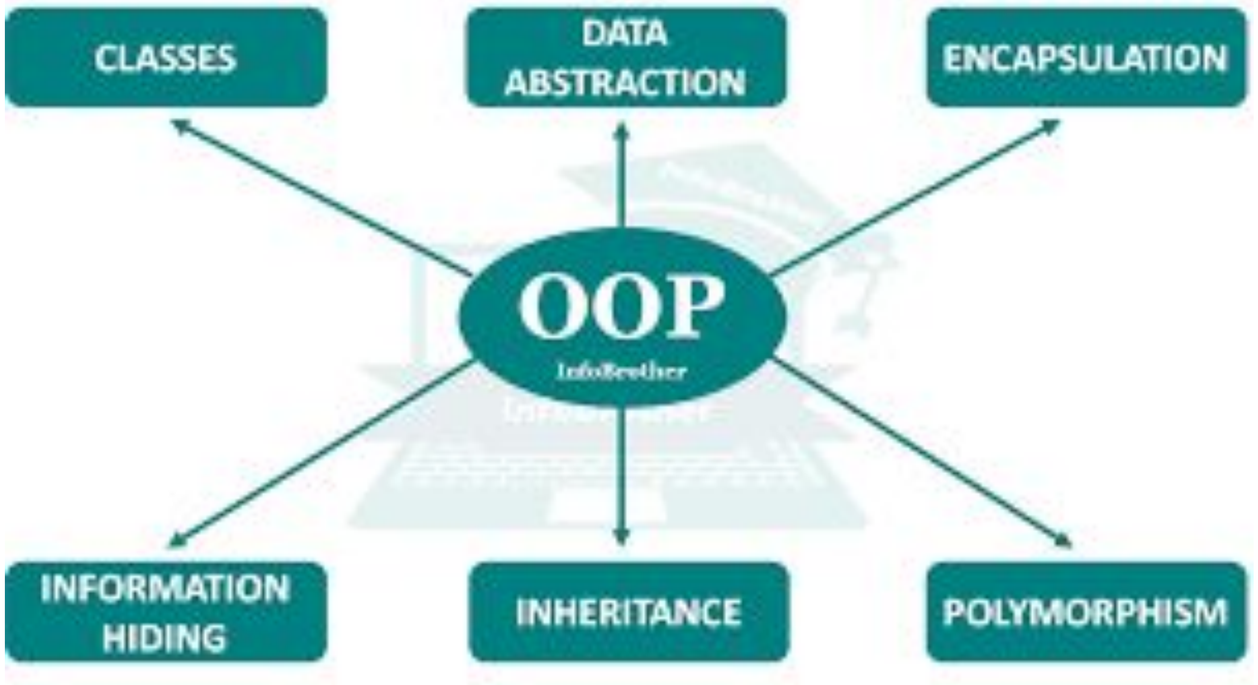
Object-Oriented Programming 1

Jerry Zhu

What is OOP?

- Literally, it is a style of programming based off of real life objects that contain data and do things on each other.



The above image delineates the key components that constitute OOP.

Classes

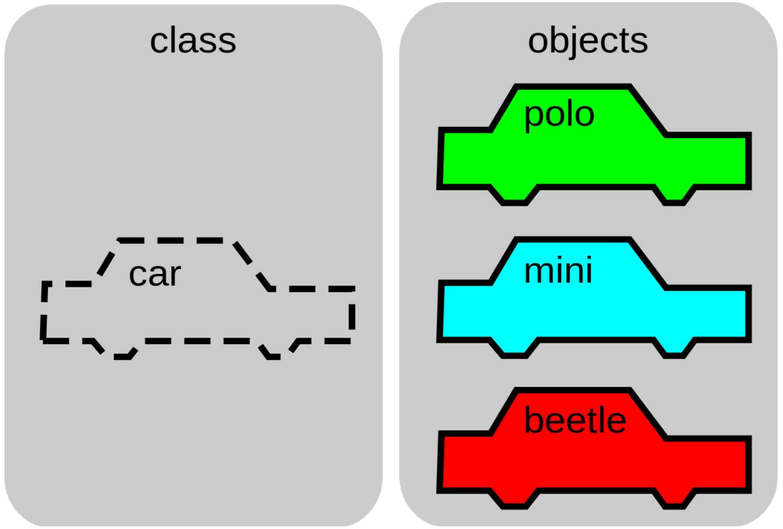
- Simply, a class is a blueprint for creating objects.
- The user-defined objects are created using the “class” keyword. The class is a

  An instance is a specific object created from a particular class. Classes are used to create and manage new objects and support inheritance—a key ingredient in object-oriented programming and a mechanism of reusing code.

E.g. class Car

Instances of the Car class would be:

Polo, Mini, Beetle...



Variables

- Objects have attributes.
  - E.g. Human has a height, a weight, an eye colour....
- These attributes are stored in the form of instance variables.
- Static variables are variables that are the same for all classes. In this case, a static variable for the average Human is the number of legs.

E.g.

```
class Human{
    private int height; //instance variable
    static int numLegs; //static variable
```

Methods

- Objects perform actions.
  - E.g. cars can accelerate.
- These actions by the object are defined in methods.

```
class Car{
  private int speed; //variables here
  public void accelerate(){
    speed++:
  }
```

Constructors

- A constructor is a special type of method whereby it makes the object by giving the instance variables of the object values.

E.g.

```
class Car{
    private int speed;
    public Car(){
      speed = 0;
    }
```

Parameters & Return

- Parameters are variables that go within the () of the method declaration that is passed when the method is called.
- It follows into the method as a local variable.

```
public void accelerate(int amt){
    speed+= amt;}
```

- Return methods are variables that quit the execution of the method and return a value(or no value).
- The return type is the variable type the method returns.

```
public int getSpeed(){
    return speed;}
```

Overloaded Methods

- Overloaded methods are when the methods have the same identifier(name) but different parameters.
- The purpose is to make the different methods serve more specific purposes.

```
public void accelerate(){
   speed++:
}
public void accelerate(int amt){
     speed+=amt;
}
```

Inheritance, Abstraction, Encapsulation, Polymorphism

- These are all key aspects of OOP that I will expand on in OOP2.
- For now:
  - Inheritance - describes how the attributes and public methods are inherited by the subclass from its superclass/parent class.
  - Abstraction - selects data from a larger pool to define, specifically, for the the object.
  - Encapsulation - when attributes and methods are stored in a container(a class).
  - Polymorphism - when the subclass is defined as the superclass.

Personal Reflection

- This unit introduced me to the crux of how Java works, for true Java was intended to be Object-Oriented. It opened my eyes in that in ICS 11, we were coding Java but it wasn’t Object-Oriented and consisted of merely loose classes and methods.
- In terms of the assignments, I did not find them to challenging after I managed to wrap my head around the concept of OOP, which took some time because it is not intuitive. The challenge was linking real life objects to code.