Classes and Objects

At the heart of OOP lie two fundamental concepts: classes and objects. They form the foundation of Object-Oriented Programming (OOP).

1. What is a Class?
A class is a blueprint or template. It defines the properties (attributes) and behaviors (methods) that the objects created from it will have.

Think of a class as a recipe. It tells you what ingredients (fields) and instructions (methods) are needed, but it's not the actual dish. You use the recipe to bake a cake (i.e., create an object).

Key Characteristics
It encapsulates data and behavior.
Defines attributes as variables.
Defines methods (functions inside a class) to operate on that data.
Example: Class Blueprint

Python

```
class Car:
    # Constructor
    def __init__(self, brand, model):
        # Attributes (private by convention with underscore)
        self._brand = brand
        self._model = model
        self._speed = 0

    # Method to accelerate
    def accelerate(self, increment):
        self._speed += increment

    # Method to display info
    def display_status(self):
        print(f"{self._brand} is running at {self._speed} km/h.")
```

Java

```
public class Car {
    // Attributes
    private String brand;
    private String model;
    private int speed;
```

```
    // Constructor
    public Car(String brand, String model) {
        this.brand = brand;
        this.model = model;
        this.speed = 0;
    }

    // Method to accelerate
    public void accelerate(int increment) {
        speed += increment;
    }

    // Method to display info
    public void displayStatus() {
        System.out.println(brand + " is running at " + speed + " km/h.");
    }
}
```

This Car class defines what every car object should look like and what it can do.

2. What is an Object?
An object is an instance of a class. When you create an object, you are bringing the blueprint of the class into reality. It consists of state and behavior defined by the class, with each object holding its own copy of the data.

Creating Objects

Python

```
if __name__ == "__main__":
    # Creating objects of the Car class
    corolla = Car("Toyota", "Corolla")
    mustang = Car("Ford", "Mustang")

    corolla.accelerate(20)
    mustang.accelerate(40)

    # Displaying status of each car
    corolla.display_status()
    print("-----------------")
    mustang.display_status()
```

Java

```
public class Main {
    public static void main(String[] args) {
        // Creating objects of the Car class
        Car corolla = new Car("Toyota", "Corolla");
        Car mustang = new Car("Ford", "Mustang");

        corolla.accelerate(20);
        mustang.accelerate(40);

        // Displaying status of each car
        corolla.displayStatus();
        System.out.println("-----------------");
        mustang.displayStatus();
    }
}
```

Here, corolla and mustang are objects of the Car class. They have their own brand , model , and speed fields and can use methods defined in the class.