Attributes/Methods

Attributes.....

- we used the term "variable" for x in the example.
- It is actually an attribute of the class.
- Or you could say that class attributes are variables within a class

In Java Modify…

```java
public class Class_Object {
    int x = 5;

    Run | Debug
    public static void main(String[] args) {
        Class_Object myObj1 = new Class_Object();
        Class_Object myObj2 = new Class_Object();
        myObj2.x = 25;
        System.out.println(myObj1.x);
        System.out.println(myObj2.x);
    }
}
```

In Python Modify…

```
class Main:
    x = 5

myObj1 = Main()
myObj2 = Main()
myObj2.x = 25

print(myObj1.x)
print(myObj2.x)
```

Methods and Functions..

- A **method/function** is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- Methods are used to perform certain actions, and they are also known as **functions**.

Static vs. Public

- we created a static method, which means that it can be accessed without creating an object of the class, unlike public, which can only be accessed by objects.

In Java….

```java
public class Class_Object {
  // Static method
  static void myStaticMethod() {
    System.out.println(x:"Static methods can be called without creating objects");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println(x:"Public methods must be called by creating objects");
  }

  // Main method
  Run | Debug
  public static void main(String[] args) {
    myStaticMethod(); // Call the static method
    // myPublicMethod(); // This would compile an error

    Class_Object myObj = new Class_Object(); // Create an object of Main
    myObj.myPublicMethod(); // Call the public method on the object
  }
}
```

In Python…

- Don't have the concept of static methods in the same way as in Java. Instead, you can use a regular method and then decide how to call it.

Constructors

- A constructor is a **special method** that is used to initialize objects.
- The constructor is called when an object of a class is created.
- It can be used to set initial values for object attributes
- Also note that the constructor is called when the object is created.

In Java….

```java
public class Class_Object {
    // Attribute
    int x;

    // Constructor
    public Class_Object(int initialValue) {
        x = initialValue;
    }

    // Method
    public void display() {
        System.out.println("The value of x is: " + x);
    }

    Run | Debug
    public static void main(String[] args) {
        // Creating an object of MyClass
        Class_Object myObject = new Class_Object(initialValue:10);

        // Accessing the attribute and calling the method
        myObject.display();
    }
}
```

In Python….

```python
class MyClass:
    # Constructor
    def __init__(self, initial_value):
        self.x = initial_value

    # Method
    def display(self):
        print("The value of x is:", self.x)

# Creating an object of MyClass
my_object = MyClass(10)

# Accessing the attribute and calling the method
my_object.display()
```

Task.....

- Task: Bank Account Management
  - Create a BankAccount class with attributes like accountNumber, accountHolder, and balance.
  - Implement a constructor to initialize these attributes.
  - Include methods to deposit, withdraw, and display the account details.
- Task: Online Shopping Cart
  - Design a Product class with attributes like productName, price, and quantity.
  - Implement a constructor to initialize these attributes.
  - Include methods to calculate the total cost of items in the cart and display the product details.
- Task: Car Rental System
  - Create a Car class with attributes such as carModel, rentalRate, and availability.
  - Implement a constructor to initialize these attributes.
  - Include methods to rent a car, return a car, and display the car details.

Task.....

- Task: Student Record System
  - Create a Student class with attributes such as studentID, name, and grades (an array or list).
  - Implement a constructor to initialize these attributes.
  - Include methods to calculate the average grade and display student information.
- Task: Library Catalog
  - Design a Book class with attributes like title, author, and publicationYear.
  - Implement a constructor to initialize these attributes.
  - Include methods to check out a book and return a book, along with displaying book details.

Sample……..

```python
class Car:
    def __init__(self, car_model, rental_rate, availability=True):
        self.car_model = car_model
        self.rental_rate = rental_rate
        self.availability = availability

    def rent_car(self):
        if self.availability:
            self.availability = False
            print(f"{self.car_model} rented successfully.")
        else:
            print(f"{self.car_model} is not available for rent.")

    def return_car(self):
        if not self.availability:
            self.availability = True
            print(f"{self.car_model} returned successfully.")
        else:
            print(f"{self.car_model} was not rented.")

    def display_car_details(self):
        print(f"Car Model: {self.car_model}")
        print(f"Rental Rate: ${self.rental_rate} per day")
        print(f"Availability: {'Available' if self.availability else 'Not Available'}")


# Example usage
car1 = Car("Toyota Camry", 50)
print("\n-----------------------------------------------------------------------")
car1.display_car_details()
print("\n-----------------------------------------------------------------------")
car1.rent_car()
print("\n-----------------------------------------------------------------------")
car1.return_car()
```

```
-----------------------------------------------------------------------
Car Model: Toyota Camry
Rental Rate: $50 per day
Availability: Available

-----------------------------------------------------------------------
Toyota Camry rented successfully.

-----------------------------------------------------------------------
Toyota Camry returned successfully.
```