# Day 11

# Java OOP Concepts – Class & Object

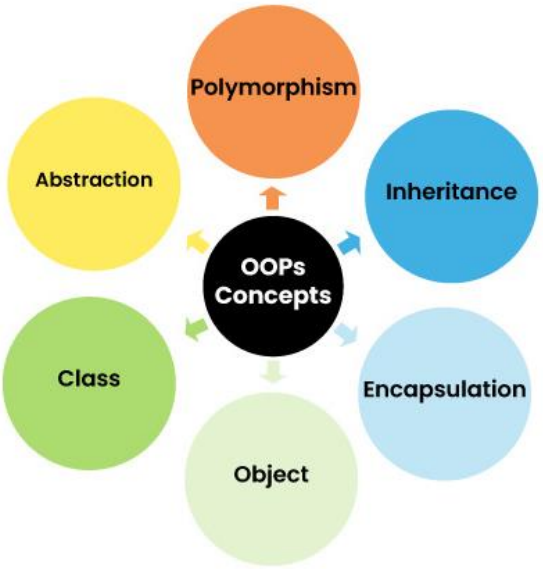
## OOPs (Object-Oriented Programming System)

**Object** means a real-world entity such as a mobile, book, table, computer, watch, etc.

**Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.

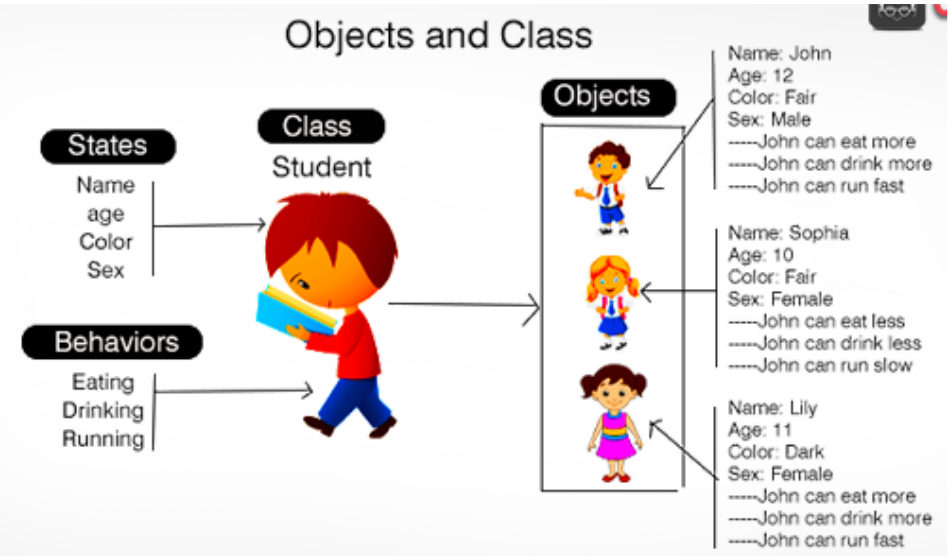
**Object-Oriented Programming Concepts:**

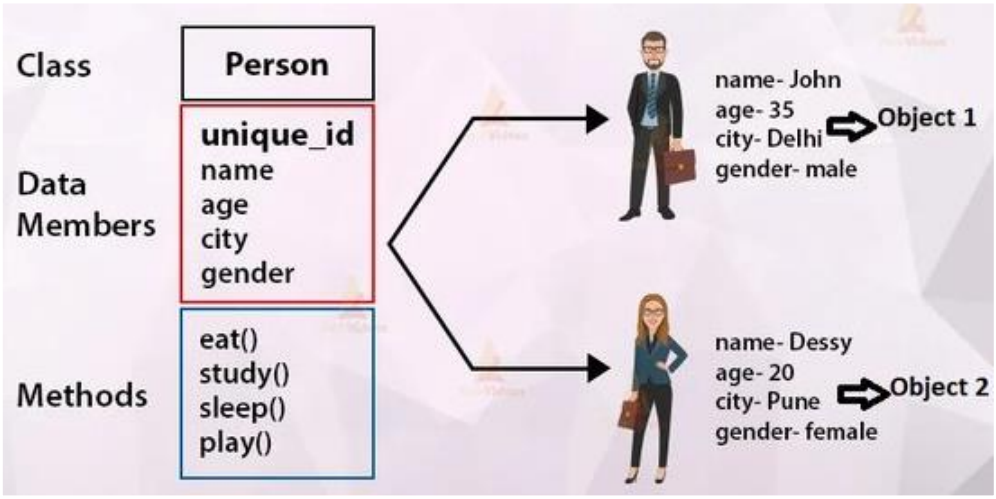
1. Class
2. Object
3. Polymorphism
4. Encapsulation
5. Inheritance
6. Data Abstraction

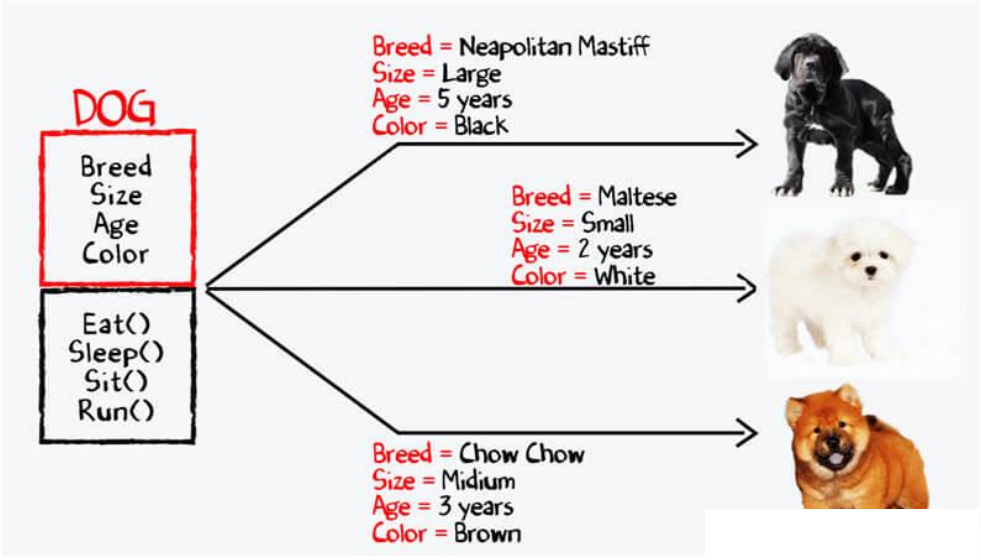



## Real world Classes and Objects:

- **Animal:** Represents a general category (class) for specific animals like Dog, Elephant, Horse, etc.
- **Student:** Represents a general category (class) for specific students like Kim, David, Scott, etc.
- **Employee:** Represents a general category (class) for specific employees like John, Smith, Mary, etc.


Objects and Class
States
Name
age
Color
Sex
Class
Student
Behaviors
Eating
Drinking
Running
Objects
Name: John
Age: 12
Color: Fair
Sex: Male
-----John can eat more
-----John can drink more
-----John can run fast
Name: Sophia
Age: 10
Color: Fair
Sex: Female
-----John can eat less
-----John can drink less
-----John can run slow
Name: Lily
Age: 11
Color: Dark
Sex: Female
-----John can eat more
-----John can drink more
-----John can run fast


Class
Person
Data Members
unique_id
name
age
city
gender
Methods
eat()
study()
sleep()
play()
name- John
age- 35
city- Delhi
gender- male
Object 1
name- Dessy
age- 20
city- Pune
gender- female
Object 2


DOG
Breed
Size
Age
Color
Eat()
Sleep()
Sit()
Run()
Breed = Neapolitan Mastiff
Size = Large
Age = 5 years
Color = Black
Breed = Maltese
Size = Small
Age = 2 years
Color = White
Breed = Chow Chow
Size = Midium
Age = 3 years
Color = Brown

## Class:

- A class is a blueprint or template that defines the structure and behaviour of objects.
- It is a logical entity. So, it does not occupy space in memory.
- A class is a collection of variables (attributes) and methods (behaviours).
- Variables store data and Methods operate data of an object of the class.

**Class creation Syntax:**

```
class ClassName {

    // Fields (Attributes or Variables)
    dataType variableName1;
    dataType variableName2;

    // Methods (Behaviors)
    returnType methodName(parameters) {
        // Method body
    }
}
```

**Example:**

```
class Employee {
    // Variables (Attributes)
    String name;
    int id;
    double salary;

    // Methods (Behaviors)
    void work() {
        System.out.println(name + " is working.");
    }

    void getSalary() {
        System.out.println(name + "'s salary is " + salary);
    }
}
```

## Object:

- An object is an instance of a class. It is a physical entity that occupies space in memory.
- An object is created using a class. It is the actual representation of the class.
- You can create multiple objects from a single class.

**Object Creation Syntax:**

```
ClassName objectName = new ClassName();
```

**Example:**

```
Employee emp1 = new Employee(); // Object creation
emp1.name = "John";
emp1.id = 101;
emp1.salary = 50000;
emp1.work();
emp1.getSalary();

Employee emp2 = new Employee(); // Another object creation
emp2.name = "Mary";
emp2.id = 102;
emp2.salary = 60000;
emp2.work();
emp2.getSalary();
```

## Methods:

- Methods are functions defined inside a class that represent the behaviours of an object. They are used to perform actions or operations on the object's data.
- **Key Points:**
    1. Methods are used to manipulate the data and execute actions within the class.
    2. Methods can have parameters and return types.

## Initializing Data into Object variables:

There are 3 different ways we can assign the data into object variables.

1. **Reference Variable:** Directly assigns values using the reference to the object.
2. **Method:** Uses a method to assign values.
3. **Constructor:** Initializes values at the time of object creation using a constructor.

**1.By Reference Variable**

You can directly assign values to the instance variables using a reference variable.

```
public class Student {
    // Instance variables
    int rollNo;
    String name;

    // Method to display student details
    void display() {
        System.out.println("Roll No: " + rollNo + ", Name: " + name);
    }

    public static void main(String[] args) {
        // Initialize using Reference Variable
        Student student1 = new Student();
        student1.rollNo = 101;
        student1.name = "John Doe";
        System.out.println("Initialized by Reference Variable:");
        student1.display();
    }
}
```

**Output:**

```
Initialized by Reference Variable:
Roll No: 101, Name: John Doe
```

**2. By Method**

You can create a method within the Student class to set the values of instance variables.

```
public class Student {
    int rollNo;
    String name;

    // Method to set values
    void setDetails(int r, String n) {
        rollNo = r;
        name = n;
    }

    void display() {
        System.out.println("Roll No: " + rollNo + ", Name: " + name);
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a Student object
        Student student2 = new Student();

        // Initialize object variables using a method
        student2.setDetails(102, "Jane Smith");

        // Display student details
        student2.display();
    }
}
```

**Output:**

```
Roll No: 102, Name: Jane Smith
```

**3. By Constructor**

You can initialize the instance variables directly when the object is created using a constructor.

```
public class Student {
    int rollNo;
    String name;

    // Constructor to initialize variables
    Student(int r, String n) {
        rollNo = r;
        name = n;
    }

    void display() {
        System.out.println("Roll No: " + rollNo + ", Name: " + name);
    }
}

public class Main {
    public static void main(String[] args) {
        // Initialize object variables using a constructor
        Student student3 = new Student(103, "Alice Johnson");

        // Display student details
        student3.display();
    }
}
```

**Output:**

```
Roll No: 103, Name: Alice Johnson
```

# Lab Assignments

**Assignment 1: Basic Class and Object Creation**

- **Objective:** Create a simple class with attributes and methods.
- **Task:**
    - Define a class named Car with attributes: make, model, and year.
    - Create a method displayInfo() that prints the details of the car.
    - In the main method, create an object of the Car class, set values for the attributes, and call the displayInfo() method to display the car's details.

**Assignment 2: Constructor Implementation**

- **Objective:** Learn how to use constructors to initialize objects.
- **Task:**
    - Modify the Car class to include a constructor that initializes the attributes make, model, and year.
    - Create an object using the constructor and display the car's details using the displayInfo() method.

**Assignment 3: Multiple Objects**

- **Objective:** Practice creating and managing multiple objects.
- **Task:**
    - Create a class Student with attributes: name, rollNumber, and grade.
    - Implement a method displayStudentInfo() to print student details.
    - In the main method, create three Student objects with different details and call the displayStudentInfo() method for each object.

**Assignment 4: Creating a Simple Bank Account Class**

- **Objective:** Implement a class with more complex attributes and methods.
- **Task:**
    - Create a BankAccount class with attributes: accountNumber, accountHolderName, and balance.
    - Implement methods:
        - deposit(double amount) to add money to the account.
        - withdraw(double amount) to subtract money from the account, checking for sufficient balance.
        - checkBalance() to display the current balance.
    - In the main method, create a BankAccount object, perform deposit and withdrawal operations, and display the balance.