JAVA METHODS & OOPS CONCEPTS

INDEX:

- Java Methods
- Recursion
- About Object oriented Programming
- Classes & Objects
- Constructor

Java Methods:

- In java, a method is a block of code that performs a specific task.
- A method has the following,

1. A name (also known as the method identifier)
2. A return type (void if it doesn't return a value)
3. Parameters (input values) in parentheses
4. A body (the code that's executed when the method is called)

Syntax:

```
access-modifier  non-access modifier  return_type  method_name(parameters) {
//method body
 //statements;
 }
```

Example:

```
public int add(int x, int y) {
    return x + y;
}
```

Method Types:

Predefined Methods:

- In Java, predefined methods are also known as built-in methods or library methods.
- These are methods that are already defined and implemented in the Java Class Library (JCL) and can be used directly by programmers.

Example:

println(), nextInt(), next() etc.

User Defined Methods:

- In Java, user-defined methods are methods that are created by the programmer to perform specific tasks.
- These are defined inside a class or interface.

Example:

```
public class Calculator {
    public int add(int x, int y) {
        return x + y;
    }
}
```

Calling Methods:

- Instance Method call: Calling a method on an object instance.

  MyObject obj = new MyObject();

  obj.myMethod();
- Static Method Call: Calling a static method on a class.

  MyClass.myStaticMethod();
- Method Chaining: Calling multiple methods in a single statement.

  myObject.method1().method2().method3();
- Method Overloading: Calling a method with different parameters.

  myObject.myMethod(int);

  myObject.myMethod(String);
- Method Overriding: Calling a subclass method that overrides a superclass method.

  MySubclass obj = new MySubclass();

  obj.myMethod(); // calls the subclass method
- Recursive Method Call: Calling a method within itself.

  public void myMethod() {

  myMethod(); // recursive call

  }

Object Oriented Programming:

- Java Object-Oriented Programming (OOP) is a programming paradigm that revolves around the concept of objects and classes.
- It's a way of designing and organizing code that simulates real-world objects and systems.

The key aspects of Java OOP:

1.Classes

2.Objects

3.Inheritance

4.Polymorphism

5.Encapsulation

6.Abstraction

Benefits of using Oop:

- Modularity
- Reusability
- Improved code organization and structure
- scalability
- Better representation of real-world systems

Class:

- A Class is a blueprint that defines the properties and behavior of an object.

Properties:

1.Variables / Data members

2.Methods / Functions

Example:

```
public class Car {
    // Variables (data members)
    String color;
    String model;

    // Method (action)
    public void startEngine() {
        System.out.println("The " + color + " " + model + " is starting its engine.");
    }
}
```

Object:

- An object is an instance of a class, represents a real-world entity .
- Objects have state (data) and behavior (methods).

New Keyword:

- The new keyword is used to create a new object.
- It allocates memory for the object and initializes its state.

Object Creation:

- Object creation is the process of creating a new object from a class using the new keyword.

Syntax:

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

InstanceOf Keyword:

- The instanceof keyword is useful for checking the type of an object at runtime, especially when working with inheritance and polymorphism.
- Syntax:

```
objectName instanceof ClassName
```

Example:

- Car class

```
public class Car {
    String color;
    String model;
}
```

- Object Creation:

```
Car myCar = new Car(); // creates a new Car object
```

- InstanceOf Keyword:

```
System.out.println(myCar instanceof Car); // Output: true
```

Constructor:

- A constructor is a special method that is used to initialize objects when they are created.
- It must be the name of the class.
- It doesn’t have the return type.

Uses:

- Initializing objects.
- Setting default values
- Ensuring valid state

Example: Person class with a constructor

- public class Person {

```
String name;
int age;
// Constructor
public Person(String name, int age) {
    this.name = name;
    this.age = age;
}
public void displayDetails() {
    System.out.println("Name: " + name + ", Age: " + age);
}
}
```

- Creating an Object:

```
Person person = new Person("John", 30);
person.displayDetails(); // Output: Name: John, Age: 30
```

THANK

YOU