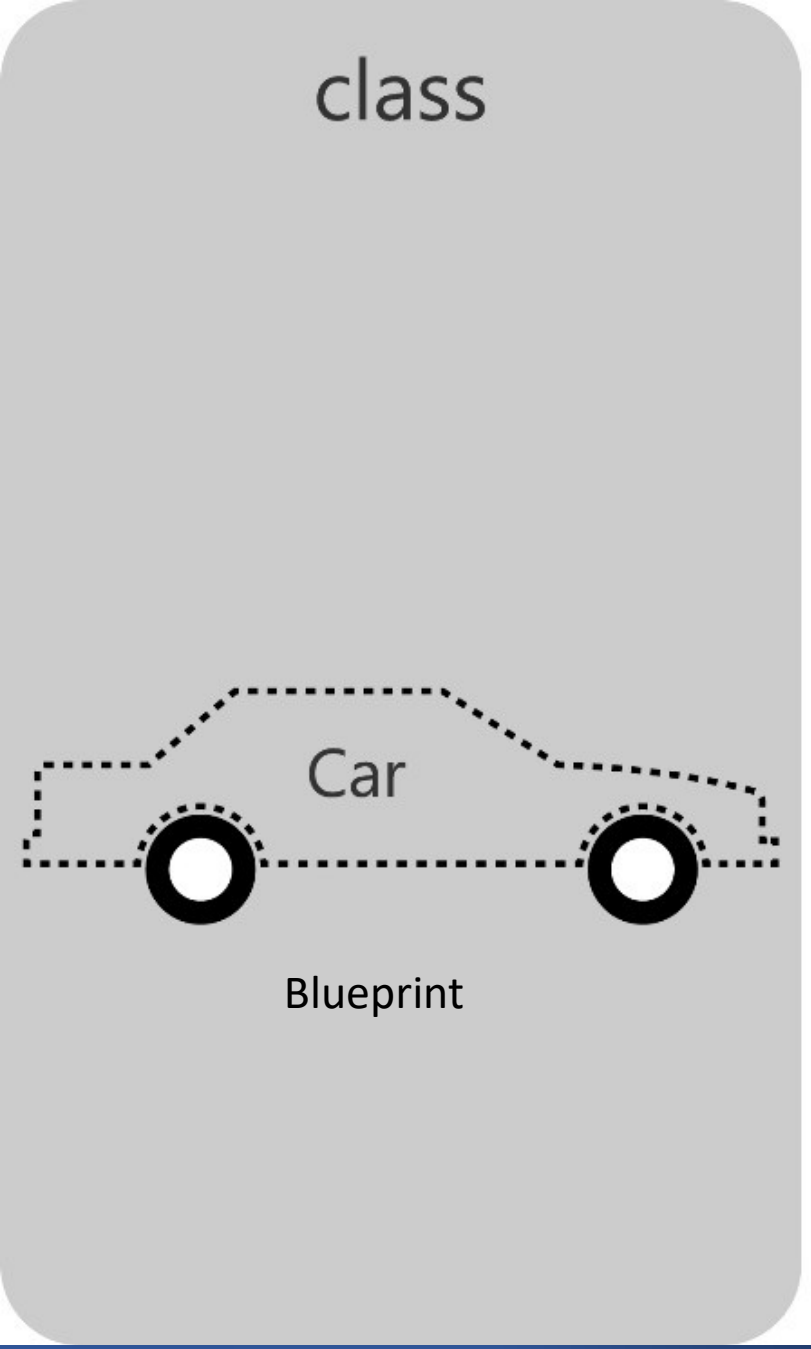
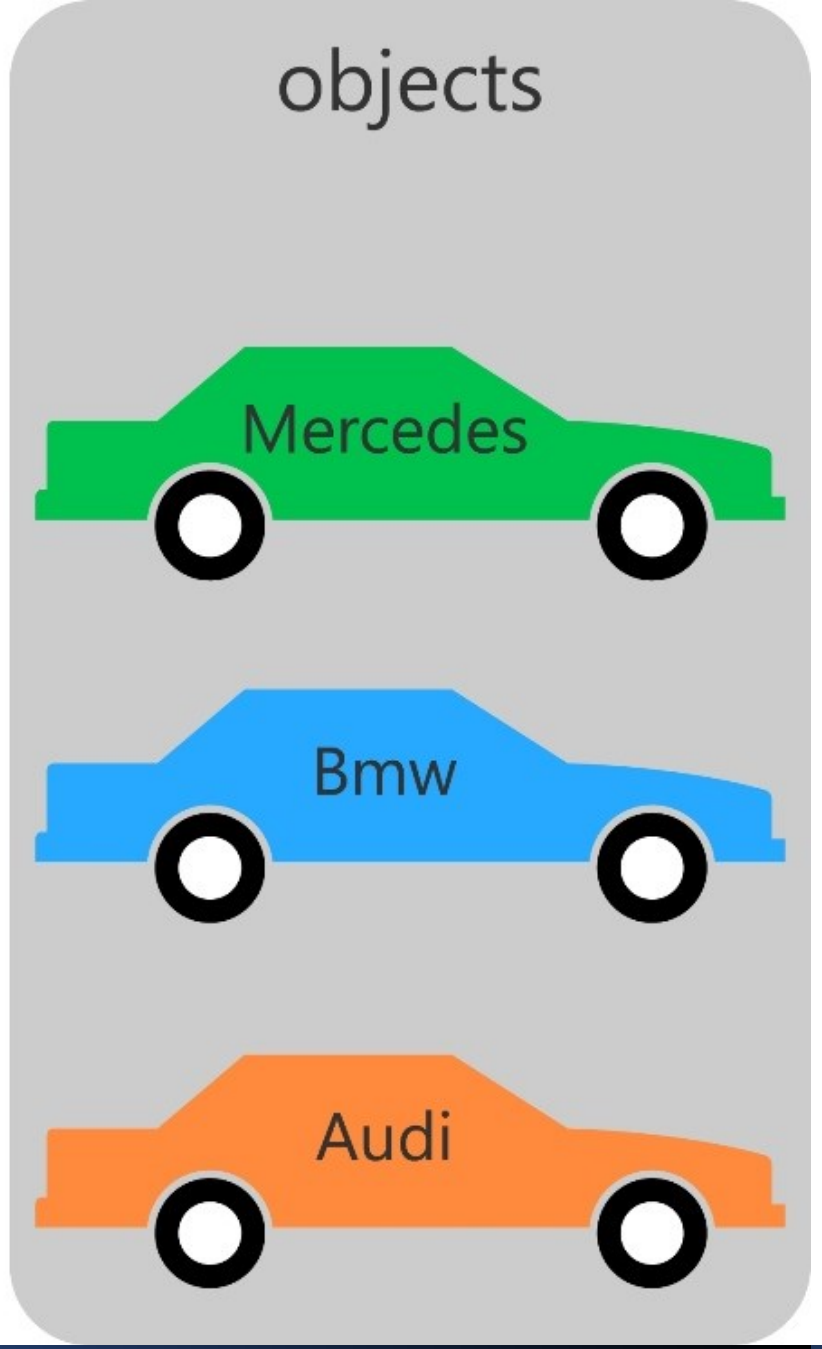
Session 3

Object-Oriented Programming Part 1

- Classes and Objects
- Constructors and Methods
- Access Modifiers
- Static Keyword
- Interview Questions (MCQs)
- Coding Questions





Classes and Objects

- **Classes and Objects - Explanation:**
- - In Java, a class is a blueprint or template for creating objects. It defines the attributes (fields) and behaviors (methods) of objects.
- - An object is an instance of a class. It represents a real-world entity and encapsulates data and functionality.

Difference between Class and Object

OBJECT	CLASS
Object is an instance ofa class.	Class is a blue print from which objects are created
Object is a real world entity such as chair, pen. table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
Object is created through new keyword. Employee ob = new Employee();	Class is declared using class keyword. class Employee{}
There are different ways to create object in java:- New keyword, newinstance() method, clone() method, And deserialization.	There is only one way to define a class, i.e., by using class keyword.

Class:

- - Defines the structure and behavior of objects.
- - Serves as a blueprint for creating multiple objects.

Object:

- - Represents a specific instance of a class.
- - Has its own state (data) and behavior (methods).

```
//Example Code Snippets
//Class Creation:

public class Car {
    String brand;
    int year;

    void start() {
        System.out.println("Car started.");
    }
}
```

Object Creation:

```
public class Main {
    public static void main(String[] args) {
        // Creating an object of class Car
        Car myCar = new Car();
        myCar.brand = "Toyota";
        myCar.year = 2022;

        // Accessing methods
        myCar.start();
    }
}
```

Main Method

In Java, the `main` method is the entry point for a Java application.
It is declared with the following signature:

```java
public static void main(String[] args) {
    // Main code goes here
}
```

The `main` method is where the execution of the Java program begins.
It must be declared as `public`, `static`, and `void`.
The `String[] args` parameter allows command-line arguments to be passed to the program.

//Example**:
//java

```java
public class Main {
    public static void main(String[] args) {
        // Main code goes here
        System.out.println("Hello, world!");
    }
}
```

Constructors and Methods

Constructors

Explanation:
Constructors are special methods used to initialize objects.
They have the same name as the class and no return type.
Constructors are called implicitly when an object is created.

Types of Constructors:
Default Constructor: Constructor with no parameters.
Parameterized Constructor: Constructor with parameters for initializing object properties.

Example Code Snippets**:
Constructor Usage:

```
1.  public class Car {
2.      String brand;
3.      int year;
4.
5.      // Default Constructor
6.      public Car() {
7.          brand = "Unknown";
8.          year = 0;
9.      }
10.
11.     // Parameterized Constructor
12.     public Car(String brand, int year) {
13.         this.brand = brand;
14.         this.year = year;
15.     }
16. }
```

Methods

Explanation:
Methods are functions defined within a class to perform specific tasks.
They encapsulate behavior and define the operations that objects can perform.

```
//Method Usage:

1.  public class Car {
2.      String brand;
3.      int year;
4.
5.      public void start() {
6.          System.out.println("Car started.");
7.      }
8.
9.      public void drive(int distance) {
10.         System.out.println("Driving " + distance + " miles.");
11.     }
12.  }
```

Types of Methods	4 Types of method we can use
Takes nothing Returns Nothing	**Takes Something Returns something**
Takes nothing Returns Something	**Takes Something Return nothing.**

Method with Parameters and Arguments

Explanation:

- Methods in Java can accept parameters, which are variables passed to the method.

- Parameters are defined in the method signature and used within the method body.

- Arguments are actual values passed to the method when it is called.

```
//Example Code Snippets
//java
1.    public class Calculator {
2.        // Method with parameters
3.        public static int add(int num1, int num2) {
4.            return num1 + num2;
5.        }
6.
7.        // Method with multiple parameters
8.        public static double calculateArea(double radius) {
9.            return Math.PI * radius * radius;
10.       }
11.   }
```

```
//Method Call with Arguments**:
//java

1.    public class Main {
2.        public static void main(String[] args) {
3.            // Calling method with arguments
4.            int sum = Calculator.add(5, 3);
5.            System.out.println("Sum: " + sum);
6.
7.            double area = Calculator.calculateArea(2.5);
8.            System.out.println("Area: " + area);
9.        }
10.   }
```

Method with Return Statement

Explanation:

- The `return` statement is used to exit a method and optionally return a value.

- It can be used to return the result of a computation, a variable, or even an object.

```
//Example Code Snippets**:
//java

1.  public class Calculator {
2.      // Method with parameters and return value
3.      public static int add(int num1, int num2) {
4.          return num1 + num2;
5.      }
6.
7.      // Method with parameters and return value
8.      public static double calculateArea(double radius) {
9.          return Math.PI * radius * radius;
10.     }
11. }
```

```
//Method Call and Handling Return Value**:
//java

1.  public class Main {
2.      public static void main(String[] args) {
3.          // Calling method with arguments and handling return
    value
4.          int sum = Calculator.add(5, 3);
5.          System.out.println("Sum: " + sum);
6.
7.          double area = Calculator.calculateArea(2.5);
8.          System.out.println("Area: " + area);
9.      }
10. }
```

Access Modifiers

Overview:

Private: Accessible only within the same class.

Protected: Accessible within the same package and subclasses (even if they are in different packages).

Default (no modifier): Accessible within the same package.

Public: Accessible from anywhere.

Private: Accessible only within the same class.

```
//Example Code Snippets
//Private Access Modifier:

1.    public class Car {
2.        private String model;
3.
4.        public String getModel() {
5.            return model;
6.        }
7.
8.        public void setModel(String model) {
9.            this.model = model;
10.       }
11.   }
```

Protected: Accessible within the same package and subclasses (even if they are in different packages).

```
//Protected Access Modifier:

1.      package vehicles;
2.
3.      public class Vehicle {
4.          protected String brand;
5.
6.          protected void displayBrand()
    {
7.              System.out.println("Brand:
    " + brand);
8.          }
9.      }
```

Default (no modifier): Accessible within the same package.

```
 - Default (No Modifier) Access
Modifier:

1.  package vehicles;
2.
3.      class Vehicle {
4.          String color;
5.
6.          void displayColor() {
7.              System.out.println("Color: " +
    color);
8.          }
9.      }
```

Public: Accessible from anywhere.

```
Public Access Modifier:

1.      public class Circle {
2.          public double radius;
3.
4.          public double
    calculateArea() {
5.              return Math.PI * radius
    * radius;
6.          }
7.      }
```

Static Keyword

Explanation

- The `static` keyword is used in Java to declare members (variables and methods) that belong to the class rather than to any specific instance of the class.
- Static members are shared among all instances of the class and can be accessed directly using the class name.

Static Variables and Methods:

Static Variables:

- Variables declared with the `static` keyword are known as static variables or class variables.
- They are initialized only once at the start of the execution and shared among all instances of the class.

- Static Methods:
  - Methods declared with the `static` keyword are known as static methods.
  - They can be invoked without creating an instance of the class and can only access other static members of the class.

Static Block Initialization:

- Static Block:

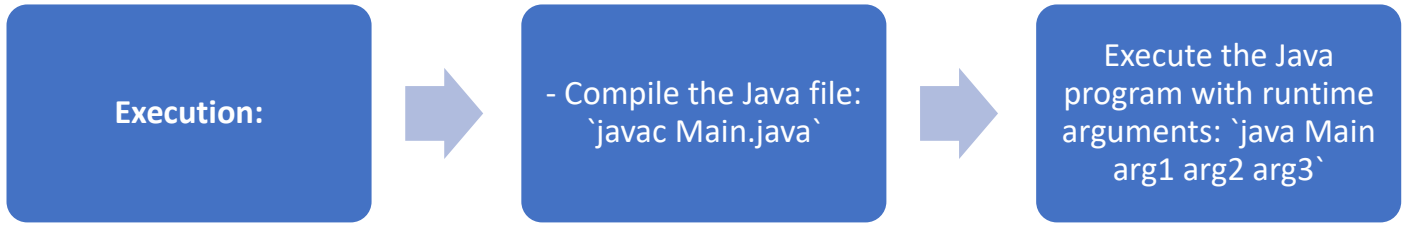
- A static block is a block of code enclosed in curly braces `{}` and preceded by the `static` keyword.
- It is executed only once when the class is loaded into memory.
- It is mainly used for initializing static variables.

```
//Example Code Snippets
//java

1.  public class MyClass {
2.      // Static variable
3.      public static int count = 0;
4.
5.      // Static method
6.      public static void incrementCount() {
7.          count++;
8.      }
9.
10.     // Static block
11.     static {
12.         System.out.println("Static block initialized.");
13.     }
14. }
```

The `static` keyword is used to create members that are associated with the class rather than with any instance of the class, allowing for shared behavior and data across all instances.

Runtime Arguments




```java
//Example Code Snippet
//Java


public class Main {
    public static void main(String[] args) {
        // Accessing runtime arguments
        System.out.println("Number of arguments: " + args.length);
        for (int i = 0; i < args.length; i++) {
            System.out.println("Argument " + (i + 1) + ": " + args[i]);
        }
    }
}
```

Explanation:

- In Java, runtime arguments (also known as command-line arguments) are the values passed to a Java program when it is executed from the command line.
- They are provided after the name of the Java class to be executed and are accessed within the `main` method using the `args` parameter.

Basic Interview Questions (MCQs)

1. What does the 'static' keyword indicate in Java?
   - a) It indicates that the variable or method belongs to an instance of the class.
   - b) It indicates that the variable or method belongs to the superclass.
   - c) It indicates that the variable or method belongs to the class itself.
   - d) It indicates that the variable or method is local to a method.

2. Which of the following is true about static methods?
   - a) They can be invoked using an instance of the class.
   - b) They cannot access other static members of the class.
   - c) They cannot be overridden in subclasses.
   - d) They can access other static members of the class directly.

3. What does a static block in Java allow?
   - a) Dynamic initialization of variables
   - b) Accessing non-static members within a static context
   - c) Initialization of static variables
   - d) Defining new methods within a class

4. How are runtime arguments accessed in a Java program?
   - a) Using the 'arguments' keyword
   - b) Using the 'args' parameter in the main method
   - c) Using the 'runtimeArgs' array
   - d) Using the 'CommandLine' class

5. Which access modifier allows a member to be accessible only within the same class?
   - a) private
   - b) protected
   - c) public
   - d) default

Basic Interview Questions (MCQs)

6. When is the static block executed in Java?
   - a) When an instance of the class is created
   - b) When the class is loaded into memory
   - c) When a static method is called
   - d) When a non-static method is called

7. Which keyword is used to call the superclass constructor from a subclass constructor?
   - a) this
   - b) super
   - c) extends
   - d) parent

8. What is the purpose of encapsulation in Java?
   - a) To define a class
   - b) To restrict access to class members
   - c) To create multiple instances of a class
   - d) To inherit from another class

9. Which keyword is used to refer to the current object within a method or constructor?
   - a) instance
   - b) this
   - c) self
   - d) object

10. Which of the following statements is true regarding static variables in Java?
    - a) They are initialized each time a new object of the class is created.
    - b) They are initialized only once, at the start of the program execution.
    - c) They are not accessible within static methods.
    - d) They cannot be declared with any access modifier.

Coding Questions

Question 1:

Write a Java program to implement a class called `MathUtils` that contains a static method named `factorial` to calculate the factorial of a non-negative integer.

```
1. public class MathUtils {
2.     public static int factorial(int n) {
3.         if (n == 0 || n == 1) {
4.             return 1;
5.         }
6.         int result = 1;
7.         for (int i = 2; i <= n; i++) {
8.             result *= i;
9.         }
10.        return result;
11.    }
12. }
```

Coding Questions

Question 2:

Write a Java program to implement a class called `StringUtils` that contains a static method named `reverseString` to reverse a given string.

```
1. public class StringUtils {
2.     public static String reverseString(String str) {
3.         StringBuilder reversed = new StringBuilder();
4.         for (int i = str.length() - 1; i >= 0; i--) {
5.             reversed.append(str.charAt(i));
6.         }
7.         return reversed.toString();
8.     }
9. }
```

Thank You

By AMA