Unit 2: Introducing Classes, Objects and Methods

2.1 Class Fundamentals

Definition

A class is a **blueprint or template** used to create objects.
It contains:

- Data members (variables)
- Member functions (methods)

Features

- Logical representation of an object
- Improves **code organization**
- Supports **encapsulation**

Syntax

```
class ClassName {
    // variables
    // methods
}
```

Program

```
class Student {
    int id;
    String name;

    void show() {
        System.out.println(id + " " + name);
    }
}

public class Main {
    public static void main(String[] args) {
        Student s = new Student();
        s.id = 101;
        s.name = "Ravi";
        s.show();
    }
}
```

Output

```
101 Ravi
```

2.2 Object Creation

Definition

An object is a **real-world entity** created from a class.
It represents **actual memory allocation**.

Features

- Each object has its own data
- Created using new keyword
- Can access class members

Syntax

```
ClassName obj = new ClassName();
```

Program

```
class Box {
    int length = 10;

  void display() {
        System.out.println(length);
    }
}

public class Main {
    public static void main(String[] args) {
        Box b = new Box();
        b.display();
}
}
```

Output

```
10
```

2.3 Methods

Definition

A method is a **block of code** that performs a specific task.
It is used to **reuse code and reduce complexity**.

Types

- Predefined methods
- User-defined methods

Syntax

```
returnType methodName(parameters) {
    // code
}
```

Program

```
class MethodDemo {
    void add(int a, int b) {
        System.out.println(a + b);
    }

    public static void main(String[] args) {
        MethodDemo m = new MethodDemo();
        m.add(10, 20);
    }
}
```

Output

```
30
```

2.4 Command Line Arguments

Definition

Command line arguments are **values passed to the program at runtime**.
They are stored in the `args[]` array of `main()` method.

Features

- No need for user input during execution
- Useful for dynamic data

Syntax

```
public static void main(String[] args)
```

Program

```
class CmdDemo {
    public static void main(String[] args) {
        System.out.println(args[0]);
    }
}
```

Output

(If run as: `java CmdDemo Hello`)

```
Hello
```

2.5 Constructors

Definition

A constructor is a **special method** used to initialize objects.
It has:

- Same name as class
- No return type

Features

- Called automatically
- Used for initialization
- Can be overloaded

Syntax

```
ClassName() {
}
```

Program

```
class Demo {
    Demo() {
        System.out.println("Constructor called");
    }

    public static void main(String[] args) {
        Demo d = new Demo();
    }
}
```

Output

```
Constructor called
```

2.6 Garbage Collection

Definition

Garbage Collection is the process of **automatically removing unused objects from memory**.

Features

- Done by JVM
- Improves memory management
- Programmer does not delete objects

Program

```
class Test {
    public static void main(String[] args) {
        Test t1 = new Test();
        t1 = null;  // eligible for garbage collection
        System.out.println("Object created");
    }
}
```

Output

```
Object created
```

2.7 `this` Keyword

Definition

`this` keyword refers to the **current object**.

Uses

- Differentiate instance variables and local variables
- Call current class methods
- Call current class constructor

Syntax

```
this.variable = variable;
```

Program

```
class Student {
    int id;

    Student(int id) {
        this.id = id;
    }

    void show() {
        System.out.println(id);
    }

    public static void main(String[] args) {
        Student s = new Student(10);
        s.show();
    }
}
```

Output

```
10
```

2.8 Static Fields and Methods

Definition

Static members belong to the **class**, not to objects.

Features

- Memory allocated once
- Accessed using class name
- Shared among all objects

Syntax

```
static int x;
static void method() {}
```

Program

```
class StaticDemo {
    static int count = 0;

    StaticDemo() {
        count++;
    }
    public static void main(String[] args) {
        new StaticDemo();
        new StaticDemo();
        System.out.println(count);
    }
}
```

Output

2

2.9 Nested and Inner Classes

Definition

A class defined **inside another class** is called a nested class.
If it is **non-static**, it is called an inner class.

Features

- Improves code readability
- Used for logical grouping
- Access outer class members

Syntax

```
class Outer {
    class Inner {
    }
}
```

Program

```
class Outer {
    int x = 10;

    class Inner {
        void show() {
            System.out.println(x);
        }
    }
    public static void main(String[] args) {
        Outer o = new Outer();
        Outer.Inner i = o.new Inner();
        i.show();
    }
}
```

Output

```
10
```

2.10 Variable Length Arguments (Var-args)

Definition

Variable length arguments allow a method to **accept any number of arguments**.

Features

- Introduced in Java 5

- Reduces method overloading
- Internally treated as array

Syntax

```
methodName(type... args)
```

Program

```
class VarArgsDemo {
    static void sum(int... a) {
        int total = 0;
        for (int x : a) {
            total += x;
        }
        System.out.println(total);
    }
    public static void main(String[] args) {
        sum(10, 20);
        sum(1, 2, 3);
    }
}
```

Output

```
30
6
```