INTRODUCTION TO PROGRAMMING LANGUAGE II(JAVA)

Fariha Zahin

final keyword

final keyword is used to **restrict the user** in Java. It can be applied to:

- **Variables** – to make them constant (immutable)
- **Methods** – to prevent overriding
- **Classes** – to prevent inheritance

final Variable :

final variable is **constant**; its value cannot be reassigned.

- It must be initialized at declaration o constructor (if not static).

```
public class FinalVariableExample {
    public static void main(String[] args) {
        final int MAX = 100;
        System.out.println("MAX: " + MAX);
        // MAX = 200; // ✗ Error
    }
}
```

Declares a constant variable MAX. Attempting to reassign it will cause a compile-time error.

Blank Final Variable

- A **blank final** variable is a final variable declared **without initialization**.
- It must be initialized **only once**, typically in the **constructor**.

speed variable is declared but not initialized immediately. It's initialized once inside the constructor. Useful when the value depends on runtime input.

```
class BlankFinalExample {
    final int speed; // blank final

    BlankFinalExample(int s) {
        speed = s; // must assign value here
    }

    void showSpeed() {
        System.out.println("Speed: " + speed);
    }

    public static void main(String[] args) {
        BlankFinalExample obj = new BlankFinalExample(90);
        obj.showSpeed();  // Output: Speed: 90
    }
```

Static Blank Final Variable

- A **static blank final** is a static final variable declared without a value.
- It must be initialized **once** in a **static block** (not constructor).

A class-level final variable that's assigned in a static block. It's initialized once when the class is loaded, not per object.

```java
class StaticBlankFinalExample {
    static final int MAX;

    static {
        MAX = 500; // must initialize here
    }

    public static void main(String[] args) {
        System.out.println("MAX: " + MAX);  // Output: MAX: 500
    }
}
```

Final Method

A final method **cannot be overridden** by subclasses.

The method show() in the Parent class is marked as final, so the subclass Child cannot override it. This is used to protect method logic from being altered.

```
class Parent {
    final void show() {
        System.out.println("Final method in Parent");
    }
}

class Child extends Parent {
    // void show() { } // ❌ Error: cannot override final method
}

public class FinalMethodExample {
    public static void main(String[] args) {
        new Child().show();  // Output: Final method in Parent
    }
```

Final Class

A final class **cannot be extended/inherited**.

The class Vehicle is declared as final, so no other class can extend it. This is commonly used for utility or security-sensitive classes

```
final class Vehicle {
    void run() {
        System.out.println("Vehicle is running");
    }
}

// class Car extends Vehicle {} // ✗ Error

public class FinalClassExample {
    public static void main(String[] args) {
        Vehicle v = new Vehicle();
        v.run();  // Output: Vehicle is running
    }
```