



Object-Oriented Programming in JAVA

Lecture 05



Static Class Data

- If a data item in a class is declared as static, only one such item is created for the entire class, no matter how many objects there are.
- A static data item is useful when all objects of the same class must share a **common** item of information.
- A member variable defined as static has characteristics similar to a normal static variable: It is visible only within the class, but its lifetime is the entire program. It continues to exist even if there are no objects of the class.
- **Use of Static Class Data**
 - ❖ Suppose an object needed to know how many other objects of its class were in the program.
 - ❖ In a road-racing game, for example, a race car might want to know how many other cars are still in the race.
 - ❖ In this case a static variable count could be included as a member of the class. All the objects would have access to this variable. It would be the same variable for all of them; they would all see the same count.

Static Class Data in Java

- In Java, a static variable is **shared** among all instances of a class.
- Only one **instance** is created, no matter how many objects exist.
- Use Case: Useful when all objects of the same class need to share common information.

Example of Static Class Data in Java

```
class Car {  
    private static int count = 0;  
    Car() { count++; }  
    public int getCount() {  
        return count;  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Car c1 = new Car();  
        Car c2 = new Car();  
        Car c3 = new Car();  
        System.out.println("Number of cars: " + c1.getCount());  
    }  
}
```

Output: Number of cars: 3

Working of Static Class Data

- ❑ The count variable is declared as static.
- ❑ The constructor increments count every time an object is created.
- ❑ All objects share the same count value.

Static Functions in Java

Static Methods:

- **Belong** to the class, not to individual objects.
- Can be accessed using the **class name**.

Static Functions in Java

```
class Car {  
    private static int count = 10;  
    public static void showCount() {  
        System.out.println("Count: " + count);  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Car.showCount();  
    }  
}
```

Count: 10

Static Functions in Java

- ❑ Static methods can directly access static variables.
- ❑ No need to create an object to call the method.

Final Member Functions (Equivalent to `const` in C++)

□ **Final** Keyword:

- Prevents a method from being **overridden**.
- Java doesn't have **const** member functions like C++, but final ensures immutability.

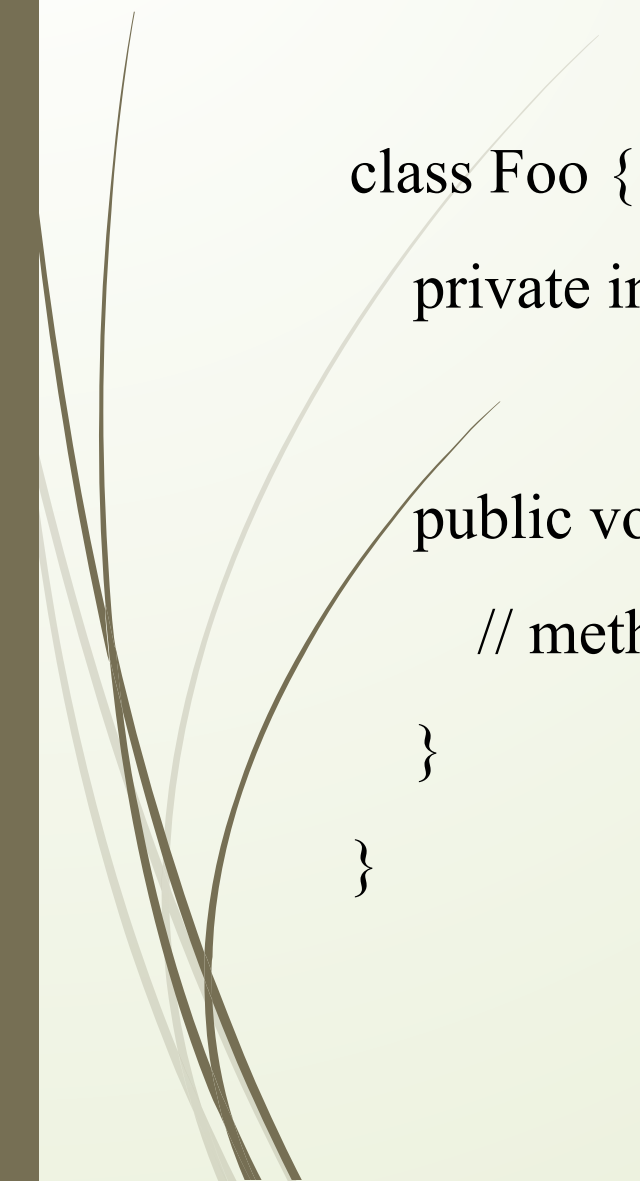

Final Member Functions

```
class Distance {  
    private int feet;  
    private float inches;  
  
    public Distance(int ft, float in) {  
        feet = ft;  
        inches = in;  
    }  
  
    public final void showDistance() {  
        System.out.println(feet + " - " + inches + "\"");  
    }  
}
```

Structures vs. Classes in Java

- ❑ In Java, there are no **structs** like in C++.
- ❑ **Classes** in Java are used to group data and methods.
- ❑ By default, members of a class are private.

Structures vs. Classes in Java



```
class Foo {  
    private int data1;  
  
    public void func() {  
        // method code  
    }  
}
```