Day 14 – Packages, Interfaces, and Wrapper Classes in Java

Objective:

To understand how to organize Java programs using packages, implement abstraction using interfaces, and utilize wrapper classes for converting primitive data types into objects.

Content:

Today, I learned about **packages**, **interfaces**, and **wrapper classes**, which are essential features of Java that help in code organization, abstraction, and type conversion.

---

1. Packages in Java

A **package** is a collection of related classes, interfaces, and sub-packages.
It helps in organizing code and avoiding name conflicts.

Types of Packages:

- **Built-in Packages:** Provided by Java (e.g., `java.util`, `java.io`, `java.lang`).
- **User-defined Packages:** Created by the programmer.

Example:

```
package mypackage;
public class Greet {
    public void show() {
        System.out.println("Hello from mypackage!");
}}
```

To use the package:

```
import mypackage.Greet;
class Test {
    public static void main(String[] args) {
        Greet g = new Greet();
        g.show();
    }}
```

---

2. Interfaces

An **interface** defines a contract that classes must follow.
It contains abstract methods that are implemented by classes.

Example:

```
interface Animal {
    void sound();}
class Dog implements Animal {
    public void sound() {
        System.out.println("Dog barks");
    }}
```

Key Points:

- All methods in an interface are **abstract** by default.
- A class can implement **multiple interfaces** (unlike inheritance).
- Interfaces support **polymorphism** and **abstraction**.

---

3. Wrapper Classes

Wrapper classes convert **primitive data types** into **objects**.
They are present in the `java.lang` package and are useful in collections and object-based operations.

Example:

```
int a = 10;
Integer obj = Integer.valueOf(a);  // Autoboxing
int b = obj;                      // Unboxing
System.out.println("Value: " + b);
```

Common Wrapper Classes:

Primitive Type	Wrapper Class
int	Integer
char	Character
float	Float
double	Double
boolean	Boolean

Learning Outcome:

Learned how to create and use packages for better code organization.
Understood how interfaces promote abstraction and flexibility in design.
Explored wrapper classes for converting primitive types to objects and vice versa.