1. Access Modifier Packages

Packages are **containers** for classes, interfaces, and sub-packages.
When you use packages, you also use **access modifiers** (`public`, `protected`, `default`, `private`) to control accessibility of classes across packages:

- **public** → Accessible everywhere (across packages).
- **protected** → Accessible within the package and through inheritance outside the package.
- **default** (no modifier) → Accessible **only within the same package**.
- **private** → Not accessible outside the class (even in the same package).

👉 Packages + access modifiers decide **which classes/members are visible across boundaries**.

2. Built-in Packages (Predefined Packages in Java)

These are **already provided by Java**. Examples:

- `java.lang` → Contains fundamental classes (String, Math, Object, Wrapper classes).
- `java.util` → Utility classes (Collections, Date, Scanner, etc.).
- `java.io` → Input/Output (File, BufferedReader, PrintWriter).
- `java.sql` → Database access (Connection, ResultSet).
- `javax.swing` → GUI components (JButton, JFrame).
- `java.time` → Date and Time API.

Example:

```
import java.util.Scanner;

class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter name: ");
        String name = sc.nextLine();
```

```
        System.out.println("Hello " + name);
    }
}
```

3. User-defined Packages

Created by developers to organize their own code.

Steps:

1. Create a package

```
package mypack;   // Declaring package

public class MyClass {
    public void display() {
        System.out.println("Hello from user-defined package!");
    }
}
```

2. Save file as → `MyClass.java` inside a folder `mypack`.
3. Compile → `javac -d . MyClass.java` (creates package folder automatically).
4. Use in another program:

```
import mypack.MyClass;

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

Modifier	Within Class	Within Package	Subclass (different package)	Outside Package	Example Use

				(non-subclass)	
private	✅ Yes	❌ No	❌ No	❌ No	Encapsulation of sensitive data (e.g., passwords).
default *(no keyword)*	✅ Yes	✅ Yes	❌ No	❌ No	Package-level helper classes.
protected	✅ Yes	✅ Yes	✅ Yes (via inheritance)	❌ No	When you want subclasses to access but not general public.
public	✅ Yes	✅ Yes	✅ Yes	✅ Yes	API methods accessible everywhere.

Quick memory trick:

- `private` → *only me* (class).
- `default` → *my package friends*.
- `protected` → *package + children*.
- `public` → *worldwide access*.