Packages in Java

Definition:

A package is a collection of related classes and interfaces grouped together.

Packages help in organizing code and avoiding name conflicts.

Why Packages are Needed

- Avoid class name conflicts
- Better code organization
- Provides access protection
- Improves reusability

Types of Packages

1. Built-in Packages (java.lang, java.util, java.io)
2. User-defined Packages

Built-in Packages

java.lang : String, System, Math (auto imported)

java.util : Scanner, ArrayList

java.io : File, BufferedReader

Creating a User-defined Package

Step 1: Write package statement as first line

```
package mypack;

public class A {
    public void show() {
        System.out.println("Inside package");
    }
}
```

Compiling the Package Program

Use the command:

javac -d . A.java

This creates a folder named 'mypack' automatically.

Using the Package in Another Class

```
import mypack.A;

public class Test {
    public static void main(String[] args) {
        A obj = new A();
        obj.show();
    }
}
```

Alternative Without Import

```
public class Test {
    public static void main(String[] args) {
        mypack.A obj = new mypack.A();
        obj.show();
    }
}
```

Important Points

- Package statement must be first line
- Only one package statement per file
- Package name = folder name
- Use -d option to create directory structure
- java.lang is imported by default

Access Specifiers and Packages

public : Accessible everywhere

protected : Accessible in same package and subclasses

default : Accessible only in same package

private : Accessible only within the class

Inner Class in Java

Definition:

An inner class is a class defined inside another class.

Purpose:

- Logical grouping of classes
- Improves encapsulation
- Used when one class is strongly related to another

Types of Inner Classes in Java

Java supports four types of inner classes:

1. Instance Inner Class (Member Inner Class)
2. Static Inner Class (Static Nested Class)
3. Local Inner Class
4. Anonymous Inner Class

Instance Inner Class (Member Inner Class)

- Defined inside outer class without static keyword
- Requires outer class object to create inner class object
- Can access all members of outer class

Syntax:

Outer.Inner obj = outerObj.new Inner();

Example: Instance Inner Class

```
class Outer {
    int x = 10;

    class Inner {
        void display() {
            System.out.println("Value of x: " + x);
        }
    }
}
public class Test {
    public static void main(String[] args) {
        Outer o = new Outer();
        Outer.Inner i = o.new Inner();
        i.display();
    }
}
```

Static Inner Class (Static Nested Class)

- Declared using static keyword
- Does not require outer class object
- Can access only static members of outer class

Syntax:

Outer.Inner obj = new Outer.Inner();

Example: Static Inner Class

```
class Outer {
    static int x = 20;

    static class Inner {
        void show() {
            System.out.println("Value of x: " + x);
        }
    }
}

public class Test {
    public static void main(String[] args) {
        Outer.Inner obj = new Outer.Inner();
        obj.show();
    }
}
```

Local Inner Class

- Defined inside a method
- Scope is limited to the method
- Can access effectively final local variables

Example: Local Inner Class

```
class Outer {
   void message() {
      int y = 30;

      class Inner {
         void show() {
            System.out.println(y);
         }
      }

      Inner i = new Inner();
      i.show();
   }
}
```

```
public class Test {
   public static void main(String[] args)
{
      Outer o = new Outer();
      o.message();
   }
}
```

Anonymous Inner Class

- No class name
- Used to override methods of a class or interface instantly
- Used when class is needed only once

Example: Anonymous Inner Class

```
abstract class A {
   abstract void show();
}

public class Test {
   public static void main(String[] args) {
      A obj = new A() {
         void show() {
            System.out.println("Anonymous Inner Class");
         }
      };
      obj.show();
   }
}
```

Comparison of Inner Classes

- Instance Inner: Needs outer object, accesses all members
- Static Inner: No outer object, accesses only static members
- Local Inner: Defined in method, limited scope
- Anonymous Inner: No name, one-time use

Advantages of Inner Classes

- Better logical grouping
- Improves encapsulation
- Cleaner and readable code
- Commonly used in GUI, event handling, threads