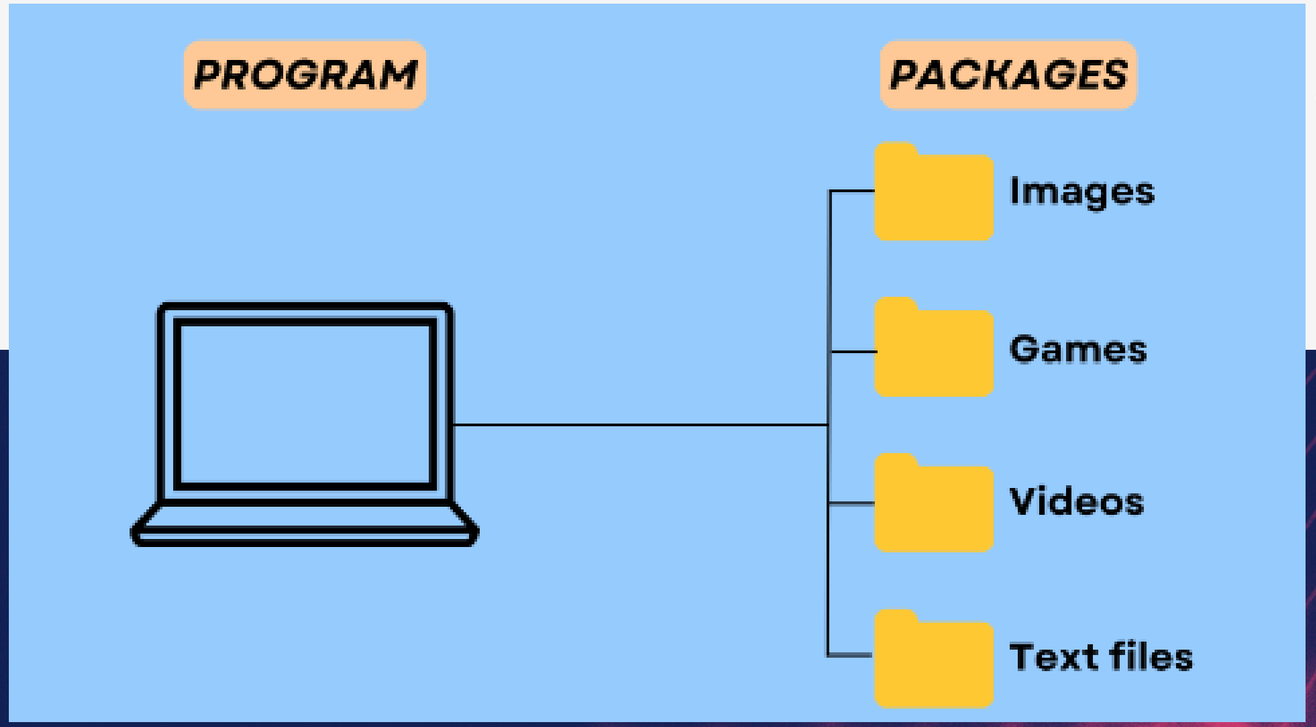
Java
PACKAGES

Team Members:

02504092023 : Ishita Pathak
02604092023 : Kajal
02704092023 : Kanishka Kharbanda
02804092023 : Kareena Negi

Table of CONTENTS

What are Packages?



- A **java package** is a group of similar types of classes, interfaces and sub-packages.
- Package in java can be categorized in two form, built-in package and user-defined package.
- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Advantages
OF PACKAGES

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

- Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier
- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
- Packages can be considered as data encapsulation (or data-hiding).

DEFINING A PACKAGE

```java
//save as Simple.java
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Welcome to package");
   }
}
```

```
Output:Welcome to package
```

The **package** keyword is used to create a package in java.

How to Compile and Run Java Package?

The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot).

To Compile: javac -d . Simple.java

To Run: java mypack.Simple

CLASS PATH

Type the following command in your Command Prompt and press enter.

```
set CLASSPATH=%CLASSPATH%;C:\Program Files\Java\jre1.8\rt.jar;
```

CLASSPATH is an environment variable which is used by Application ClassLoader to locate and load the .class files. The CLASSPATH defines the path, to find third-party and user-defined classes that are not extensions or part of Java platform. Include all the directories which contain .class files and JAR files when setting the CLASSPATH.

In the above command, The set is an internal DOS command that allows the user to change the variable value. CLASSPATH is a variable name. The variable enclosed in percentage sign (%) is an existing environment variable. The semicolon is a separator, and after the (;) there is the PATH of rt.jar file.

Access Modifiers

Access modifiers are keywords that can be used to control the visibility or accessibility of fields, methods, and constructors in a class. It determines which parts of a program can access or modify a particular member.

01 Default (No keyword required)

02 Private

03 Protected

04 Public

Types Of Access Modifiers

Public

Members declared as public are accessible from any other class, regardless of the package.

Default

If no access modifier is specified, it defaults to package-private. Members with package-private access are accessible within the same package but not outside of it.

Protected

Members declared as protected are accessible within the same package and also by a subclass even if it is in a different package.

Private

Members declared as private are only accessible within the class where they are declared.

ACCESS MODIFIERS IN SAME CLASS

Access Modifiers In Same Class

```
public class AccessModifiers {
    public int publicField = 10;
    private String privateField = "Private Data";
    protected double protectedField = 3.14;
    int defaultField = 42;
    public AccessModifiers() {
        System.out.println("Constructor: Setting up an instance of class");
    }
    public void publicMethod() {
        System.out.println("Public Method: " + publicField);
    }
    private void privateMethod() {
        System.out.println("Private Method: " + privateField);
    }
    protected void protectedMethod() {
        System.out.println("Protected Method: " + protectedField);
    }
    void defaultMethod() {
        System.out.println("Default Method: " + defaultField);
    }
 public static void main(String[] args) {
        AccessModifiers example = new AccessModifiers();
        System.out.println("Accessing fields and methods within the same
class:");
        System.out.println("Public Field: " + example.publicField);
        System.out.println("Private Field: " + example.privateField);
        System.out.println("Protected Field: " + example.protectedField);
        System.out.println("Default Field: " + example.defaultField);

        example.publicMethod();
        example.privateMethod();
        example.protectedMethod();
        example.defaultMethod();
    }
```

ACCESS MODIFIERS IN SAME PACKAGE

myPackage > Parent.java > Parent > packagePrivateField

```java
package myPackage;

// Superclass
public class Parent {
    public String publicField = "Accessible everywhere";
    protected String protectedField = "Accessible within package and subclasses";
    String packagePrivateField = "Accessible within the package only";
    private String privateField = "Accessible only within this class";

    public void publicMethod() {
        System.out.println(x:"Public method");
    }
    protected void protectedMethod() {
        System.out.println(x:"Protected method");
    }
    void packagePrivateMethod() {
        System.out.println(x:"Package-private method");
    }
    private void privateMethod() {
        System.out.println(x:"Private method");
    }
}
```

SAME PACKAGE SUBCLASS

myPackage > Child.java > Child

```java
package myPackage;

// Subclass
public class Child extends Parent {
    void accessParent() {
        // Accessing all types of fields and methods of superclass
        System.out.println(publicField);
        System.out.println(protectedField);
        System.out.println(packagePrivateField);
        // Private fields and methods are not accessible in subclasses
        System.out.println(privateField); // Error: privateField has private access in Parent
        publicMethod();
        protectedMethod();
        packagePrivateMethod();
        privateMethod(); // Error: privateMethod() has private access in Parent
    }
}
```

SAME PACKAGE NON-SUBCLASS

```java
// Non-subclass
class OtherClass {
    void accessParent() {
        Parent parent = new Parent();
        System.out.println(parent.publicField);
        System.out.println(parent.protectedField);
        System.out.println(parent.packagePrivateField);
        // Private fields and methods are not accessible in non-subclasses
        System.out.println(parent.privateField); // Error: privateField has private access in Parent
        parent.publicMethod();
        parent.protectedMethod();
        parent.packagePrivateMethod();
        parent.privateMethod(); // Error: privateMethod() has private access in Parent
    }
}
```

Within the same package in Java, subclasses and non-subclasses have access to public, protected, and default members of other classes. However, private members are only accessible within the declaring class, not by any other class, even within the same package.

Access Specifiers in Different Packages

Package-1

variables > VariableClass.java > VariableClass

```java
package variables;

public class VariableClass {
    public int publicVar = 10;
    protected int protectedVar = 20;
    int defaultVar = 30;
    private int privateVar = 40;
}
```

Package-2

methods > MethodClass.java > {} methods

```java
package methods;
public class MethodClass {
    public void publicMethod() {
        System.out.println(x:"This is a public method");
    }
    protected void protectedMethod() {
        System.out.println(x:"This is a protected method");
    }
    void defaultMethod() {
        System.out.println(x:"This is a default method");
    }
    private void privateMethod() {
        System.out.println(x:"This is a private method");
    }
}

```

Different Package Subclass

subclass > Main.java > Subclass2 > accessMethods()

```java
package subclass;
import variables.VariableClass;
import methods.MethodClass;

class Subclass extends VariableClass {
    public void accessVariables() {
        System.out.println("Public variable from VariableClass: " + publicVar);
        System.out.println("Protected variable from VariableClass: " + protectedVar);
        System.out.println("Default variable from VariableClass: " + defaultVar); // Error: Cannot access defaultVar
        System.out.println("Private variable from VariableClass: " + privateVar); // Error: Cannot access privateVar
    }
}
class Subclass2 extends MethodClass {
    public void accessMethods() {
        publicMethod();
        protectedMethod();
        defaultMethod(); // Error: Cannot access defaultMethod
        privateMethod(); // Error: Cannot access privateMethod
    }
}
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Subclass subclass = new Subclass();
        subclass.accessVariables();

        Subclass2 subclass2 = new Subclass2();
        subclass2.accessMethods();
    }
}
```

Different Package Non-Subclass

bclass > NonSubclass.java > NonSubclass > accessMembers()

```java
package nonsubclass;

import variables.VariableClass;
import methods.MethodClass;

public class NonSubclass {
    public void accessMembers() {
        VariableClass variableClass = new VariableClass();
        System.out.println("Public variable from VariableClass: " + variableClass.publicVar);
        System.out.println("Protected variable from VariableClass: " + variableClass.protectedVar); // Error: Cannot access protec
        System.out.println("Default variable from VariableClass: " + variableClass.defaultVar); // Error: Cannot access defaultVar
        System.out.println("Private variable from VariableClass: " + variableClass.privateVar); // Error: Cannot access privateVar

        MethodClass methodClass = new MethodClass();
        methodClass.publicMethod();
        methodClass.protectedMethod(); // Error: Cannot access protectedMethod
        methodClass.defaultMethod(); // Error: Cannot access defaultMethod
        methodClass.privateMethod(); // Error: Cannot access privateMethod
    }
}
```

Summary

	Private	No modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Table 9-1. *Class Member Access*

Point to Remember

Class has only two possible access levels :
default and public
When a class is declared as public, it is accessible by any
other code.
If a class has default access, then it can only be accessed by other code
within its same package.

Thank You