**Day 18**

Java Package

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.

Here, we will have the detailed learning of creating and using user-defined packages.

Advantage of Java Package

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.



Simple example of java package

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

1. //save as Simple.java
2. **package** mypack;
3. **public** **class** Simple{
4. **public** **static** **void** main(String args[]){
5. System.out.println("Welcome to package");
6. }
7. }

How to compile java package

If you are not using any IDE, you need to follow the **syntax** given below:

1. javac -d directory javafilename

For **example**

1. javac -d . Simple.java

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).

How to run java package program

You need to use fully qualified name e.g. mypack.Simple etc to run the class.

|  |
| --- |
| **To Compile:** javac -d . Simple.java |
| **To Run:** java mypack.Simple |

Output:Welcome to package

|  |
| --- |
| The -d is a switch that tells the compiler where to put the class file i.e. it represents destination. The . represents the current folder. |

How to access package from another package?

There are three ways to access the package from outside the package.

1. import package.\*;
2. import package.classname;
3. fully qualified name.

1) Using packagename.\*

If you use package.\* then all the classes and interfaces of this package will be accessible but not subpackages.

The import keyword is used to make the classes and interface of another package accessible to the current package.

Example of package that import the packagename.\*

1. //save by A.java
2. **package** pack;
3. **public** **class** A{
4. **public** **void** msg(){System.out.println("Hello");}
5. }
6. //save by B.java
7. **package** mypack;
8. **import** pack.\*;
10. **class** B{
11. **public** **static** **void** main(String args[]){
12. A obj = **new** A();
13. obj.msg();
14. }
15. }

Output:Hello

2) Using packagename.classname

If you import package.classname then only declared class of this package will be accessible.

Example of package by import package.classname

1. //save by A.java
3. **package** pack;
4. **public** **class** A{
5. **public** **void** msg(){System.out.println("Hello");}
6. }
7. //save by B.java
8. **package** mypack;
9. **import** pack.A;
11. **class** B{
12. **public** **static** **void** main(String args[]){
13. A obj = **new** A();
14. obj.msg();
15. }
16. }

Output:Hello

3) Using fully qualified name

If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

It is generally used when two packages have same class name e.g. java.util and java.sql packages contain Date class.

Example of package by import fully qualified name

1. //save by A.java
2. **package** pack;
3. **public** **class** A{
4. **public** **void** msg(){System.out.println("Hello");}
5. }
6. //save by B.java
7. **package** mypack;
8. **class** B{
9. **public** **static** **void** main(String args[]){
10. pack.A obj = **new** pack.A();//using fully qualified name
11. obj.msg();
12. }
13. }

Output:Hello

**Access Modifiers**

**Access Modifiers in Java**

* **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
* **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
* **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
* **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

**Questions**

**1. how many types of access modifiers is available in Java?**

There are four type of access modifiers in Java:

* public
* private
* protected
* package-private (default)

**2. Difference between public, protected, and private access modifier?**

public is the least restricted while private is the most restricted. This means any class, method or field with public modifier is access to all the classes inside and outside the package on which those classes, methods or fields are declared but in case of private they are only accessible within the class they are declared.

**3. What is the default access modifier? What happens if you don't specify access modifier?**

Default access modifier in Java is package level access. This is also known as package private. If you don't specify any modifier then its only visible inside the package. For example following class is only visible to other classes within the same package

Calculator{

  // your code

}

**4. Can we make a class private in Java?**

Yes, you can make nested classes private in Java but you cannot make a top level class private in Java. They are not allowed. Same goes for interface, you cannot make an interface private in Java as interface are by default and inherently public.

**5. Can we declare a class as protected?**

No, we cannot. It will create compile time error. This is true for both top level class and nested class in Java.

**6. What is difference between static and non-static class?**

A nested class can be either static or non-static, in that case its known as Inner class. The difference is that you can access static class without creating instance of top level class but you would need an instance of containing class to access a non-static nested class.

**7. What is final modifier, can you use it with classes?**

final keyword is not an access modifier but its very important from immutability and performance perspective. Yes, you can make a class final in Java, in that case you cannot extend it further. Final classes in Java cannot be sub classed.

**8. Why you should make your field private in Java?**

To achieve higher level of encapsulation. If a field or method is private then you can safely change them without affecting others as they are not visible outside the class.

**9. Why getter method is better than public variables in Java?**

It provide higher level of abstraction, you can add log statement or return something else like a normalized or calculated value if you want to which cannot be possible if your client is directly accessing the public variable.

**10. Which variables you should mark public in Java?**

you can mark constants as public variable like public final int daysInWeek = 7;

**11. What is access modifier in Java?**

All the Java keywords which restrict access of an variable, method or class are referred as access modifier in Java. There are 4 access modifier in Java, public, protected, private and package.

**12. How many access modifiers are available in Java?**

There are 4 access modifier in Java, public, private, protected and package. By default all the variables, methods or class have package level access which means if you don't specify any access modifier then they will be accessible only in the package they are declared.

**13. Why its a best practice to by default make a field or method private in Java?**

When a field is private this means you know that no one else outside the class using that variable, method or class which means it can easily be changed without much impact to outside world.

**14. What is difference between public and protected modifier in Java?**

Here is a nice table to show the difference between all access modifier in Java

**15. What is the default modifier in Java?**

Its package private access. For example if you don't provide any access modifier then it will visible only on that package.

**16. What is difference between private and package modifier in Java?**

private is more restrictive than package private because any package private variable is visible inside the package to other classes but private variables and methods are only visible or accessible on the same class they are declared.

**17. Can we make a field both public and private at same time?**

No, because both are mutual exclusive, A variable can not be public and private at the same time.

**18. How does access modifier affects method overriding?**

They prevent overriding by restricting access, for example you cannot override private methods in Java.

**19. Which access modifier can be used with top level class?**

Only public and package private or default (nothing) can be used with top level class.

**20. What is the advantage and disadvantage of using public modifier in Java?**

universal access means you cannot modify them later without impacting other clients which are using it. This is particular true for public classes. For example, removing a constructor of a public class will break existing clients of that class.