OBJECT™ TECHNOLOGIES

Packages

www.object.co.in

What will be covered

OBJECT TECHNOLOGY

- Need of Packages
- What is package
- Built in packages
- Creating package
- Compilation
- Using package
- Setting classpath

Need Of Packages

OBJECT

- A package is a namespace that organizes a set of related classes and interfaces.
- Conceptually you can think of packages as being similar to different folders on your computer. You might keep HTML pages in one folder, images in another, and scripts or applications in yet another.
- Because software written in the Java programming language can be composed of hundreds or *thousands* of individual classes, it makes sense to keep things organized by placing related classes and interfaces into packages.

What is a Package

OBJECT

- Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.
- Packages are used for preventing naming conflicts. For example there can be two classes with same name in two packages.
- It helps organize your classes into a folder structure and make it easy to locate and use them.
- More importantly, it helps improve code reusability.
- Package in java can be categorized in two form, built-in package and user-defined package.

What is a Package

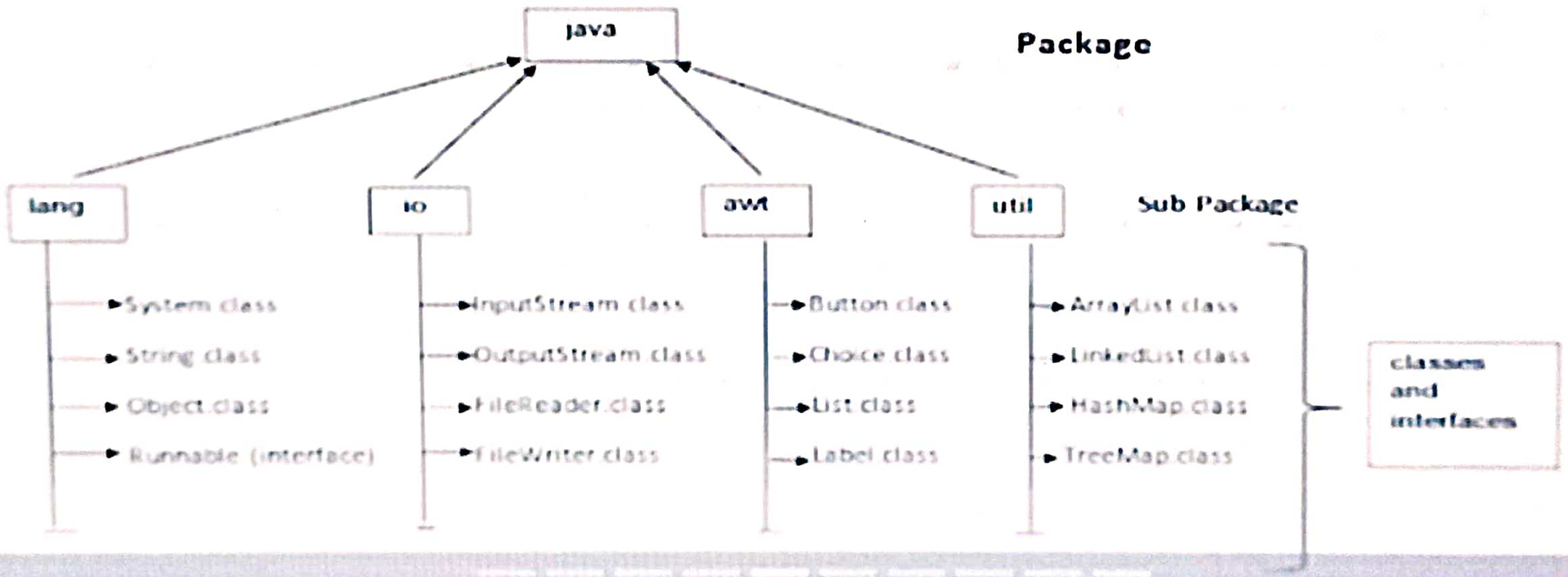
OBJECT

- Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.
- Packages are used for preventing naming conflicts. For example there can be two classes with same name in two packages.
- It helps organize your classes into a folder structure and make it easy to locate and use them.
- More importantly, it helps improve code reusability.
- Package in java can be categorized in two form, built-in package and user-defined package.

Built in packages

OBJECT

- The Java API is a library of prewritten classes, that are free to use, included in the Java Development Environment.
- The library contains components for managing input, database programming, and much much more
- Very frequently need packages are java.lang, java.util, java.sql etc
- Classes in java.lang package are availabel to use without any additional import



Creating Packages

OBJECT

- Creating a package is a simple task as follows :
  - Choose the name of the package
  - Include the package command as the first line of code in your Java Source File.
  - The Source file contains the classes, interfaces, etc you want to include in the package
  - Compile to create the Java packages
- A class can have only one package declaration.
- Package names are written in all lower case to avoid conflict with the names of classes or interfaces.

Creating Packages

OBJECT

```
package mypack;
class MyPackageClass {
    public static void main(String[] args) {
        System.out.println("This is my package!");
    }
}
```

- Creating a class belonging to the package.

```
package mypack.mysubpack;
class MyPackageSubpackageClass {
    public static void main(String[] args) {
        System.out.println("This is my package!");
    }
}
```

- Creating class belonging to the subpackage

Compilation

OBJECT

- Use following command for compiltion :

  javac -d directory javafilename

  example : 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).
- Note : In IDE like eclipse, explicit compilation is not required.
- Class files are created according to package declaration

Using packages

OBJECT

If we need to use the class in the application which belongs to some package :

1. Use fully qualified name of the class

Every time when the class name is referred, instead of using simple name, use fully qualified name. But this affects readability.

example :

```
java.util.List l = new java.util.ArrayList();
```

2. Use import statement

Instead of fully qualified name, use import statement above the class definition to tell the compiler which package should be used to refer the class

```
import java.util.*;
---
List l = new ArrayList();
```

CLASSPATH can be set by any of the following ways:

- CLASSPATH can be set permanently in the environment: In Windows, choose control panel ? System ? Advanced ? Environment Variables ? choose "System Variables" (for all the users) or "User Variables" (only the currently login user) ? choose "Edit" (if CLASSPATH already exists) or "New" ? Enter "CLASSPATH" as the variable name ? Enter the required directories and JAR files (separated by semicolons) as the value (e.g., ".;c:\javaproject\classes;d:\tomcat\lib\servlet-api.jar"). Take note that you need to include the current working directory (denoted by '.') in the CLASSPATH.
- CLASSPATH can be set temporarily for that particular CMD shell session by issuing the following command:

```
> SET CLASSPATH=.;c:\javaproject\classes;d:\tomcat\lib\servlet-api.jar
```