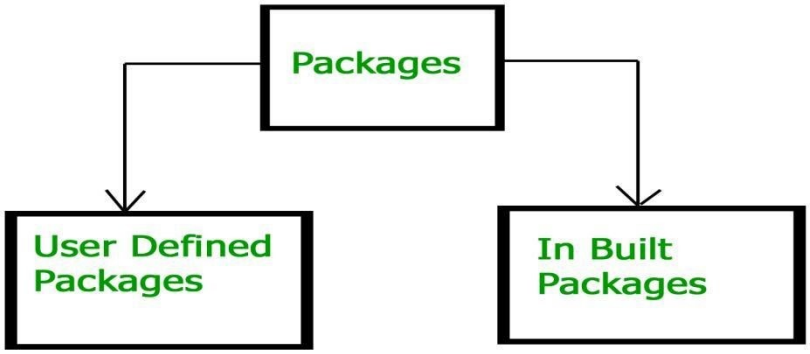
PACKAGES

A package in Java is used to group related classes. Think of it as a folder in a file directory. We use packages to avoid name conflicts, and to write a better maintainable code.

Packages are divided into two categories:

- Built-in Packages (packages from the Java API)
- User-defined Packages (create your own packages)



Built-in Packages

These packages consist of a large number of classes which are a part of Java API.Some of the commonly used built-in packages are:

1. java.lang: Contains language support classes(e.g classes which defines primitive data types, math operations). This package is automatically imported.
2. java.io: Contains classes for supporting input / output operations.
3. java.util: Contains utility classes which implement data structures like Linked List, Dictionary and support ; for Date / Time operations.
4. java.applet: Contains classes for creating Applets.
5. java.awt: Contain classes for implementing the components for graphical user interfaces (like button , ;menus etc). 6)
6. java.net: Contain classes for supporting networking operations.

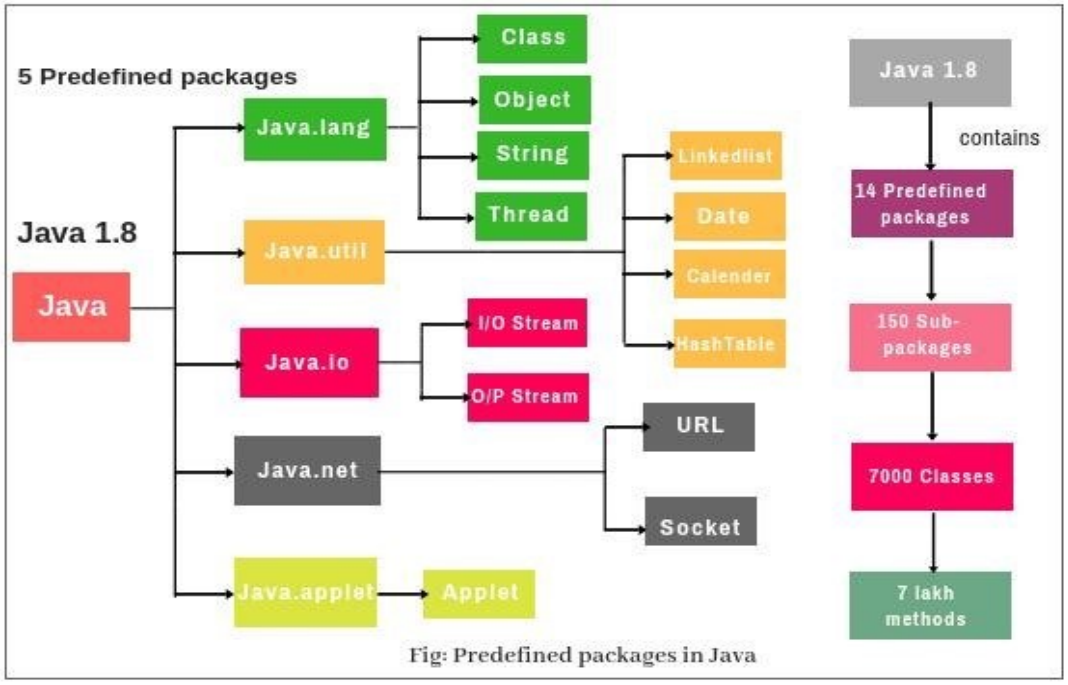


Fig: Predefined packages in Java

Java program of demonstrating use of java.lang package

```
class JavaLangExample {
  public static void main(String args []) {
    int a = 20, b =30; int sum =
    Math.addExact(a,b); int max
    = Math.max(a,b); double pi =
    Math.PI;
    System.out.printf("Sum = "+sum+", Max = "+max+", PI = "+pi);
  }
}
```

Output:

Sum = 50, Max = 30, PI = 3.141592653589793

Java program of demonstrating use of java.io package

```
import java.io.Console;
class JavaIOExample {
  public static void main(String args []) {
    Console cs = System.console();
    System.out.println("Enter your name : ");
    String name = cs.readLine();
    System.out.println("Welcome : "+name);
  }
}
```

Output:

Enter your name : teja

Welcome : teja

Java program of demonstrating use of java.util package

```
import java.util.Arrays;
 class JavaUtilExample {
   public static void main(String args []) {
      int[] intArray = {10,30,20,50,40};
      Arrays.sort(intArray);
      System.out.printf("Sorted array : %s", Arrays.toString(intArray));
    }
 }
```

Output:

Sorted array : [10, 20, 30, 40, 50]

User-defined packages

User-defined packages are those packages that are designed or created by the developer to categorize classes and packages. They are much similar to the built-in that java offers. It can be imported into other classes and used the same as we use built-in packages. But If we omit the package statement, the class names are put into the default package, which has no name.

```
//write a java program using user defined packages.
import java.io.*;
class Greeting {
    public void sayHello() {
        System.out.println("Hello from mypackage!"); }
} class Farewell
{
    public void sayGoodbye() {
        System.out.println("Goodbye from mypackage!"); }
} class ThankYou
{
    public void sayThanks() {
        System.out.println("Thank you from mypackage!"); }
} public class Main
{
    public static void main(String[] args) { Greeting
        greet = new Greeting(); greet.sayHello();

        Farewell farewell = new Farewell(); farewell.sayGoodbye();

        ThankYou thankYou = new ThankYou();
        thankYou.sayThanks();
    }
}
```

Output:

Hello from mypackage!
Goodbye from mypackage!
Thank you from mypackage!

Java File Path

java.io.File contains three methods for determining the file path, we will explore them in this tutorial.

1. getPath(): This file path method returns the abstract pathname as String. If String pathname is used to create File object, it simply returns the pathname argument. If URI is used as argument then it removes the protocol and returns the file name.
2. getAbsolutePath(): This file path method returns the absolute path of the file. If File is created with absolute pathname, it simply returns the pathname. If the file object is created using a relative path, the absolute pathname is resolved in a system-dependent way. On UNIX systems, a relative pathname is made absolute by resolving it against the current user directory. On Microsoft Windows systems, a relative pathname is made absolute by resolving it against the current directory of the drive named by the pathname, if any; if not, it is resolved against the current user directory.
3. [getCanonicalPath](https://docs.oracle.com/javase/7/docs/api/java/io/File.html#getCanonicalPath())(): This path method returns the canonical pathname that is both absolute and unique. This method first converts this pathname to absolute form if necessary, as if by invoking the getAbsolutePath method, and then maps it to its unique form in a system-dependent way. This typically involves removing redundant names such as "." and "..." from the pathname, resolving symbolic links (on UNIX platforms), and converting drive letters to a standard case (on Microsoft Windows platforms).

Java File Path Example

Let's see different cases of the file path in java with a simple program.

```
package com.journaldev.files; import
java.io.File; import
java.io.IOException; import
java.net.URI; import
java.net.URISyntaxException; public
class JavaFilePath {
    public static void main(String[] args) {
        try {
            // Absolute path
```

```
            File file = new File("/Users/pankaj/test.txt"); printPaths(file);
            // Relative path file = new
            File("test.xsd");
            printPaths(file);
            // Complex relative path
            file = new File("/Users/pankaj/../pankaj/test.txt");
            printPaths(file);
            // URI path
            file = new File(new URI("file:///Users/pankaj/test.txt"));
            printPaths(file);
        } catch (IOException | URISyntaxException e) {
            e.printStackTrace();
        }
    }
    private static void printPaths(File file) throws IOException {
        System.out.println("Path: " + file.getPath());
        System.out.println("Absolute Path: " + file.getAbsolutePath());
        System.out.println("Canonical Path: " + file.getCanonicalPath());
        System.out.println("--------------------------"); }
}
```

Output:

```
Path: /Users/pankaj/test.txt
Absolute Path: /Users/pankaj/test.txt
Canonical Path: /Users/pankaj/test.txt
--------------------------Path:
test.xsd
Absolute Path: /Users/pankaj/test.xsd
Canonical Path: /Users/pankaj/test.xsd
--------------------------
Path: /Users/pankaj/../pankaj/test.txt
Absolute Path: /Users/pankaj/../pankaj/test.txt
Canonical Path: /Users/pankaj/test.txt
--------------------------
Path: /Users/pankaj/test.txt
Absolute Path: /Users/pankaj/test.txt
Canonical Path: /Users/pankaj/test.txt
--------------------------
```