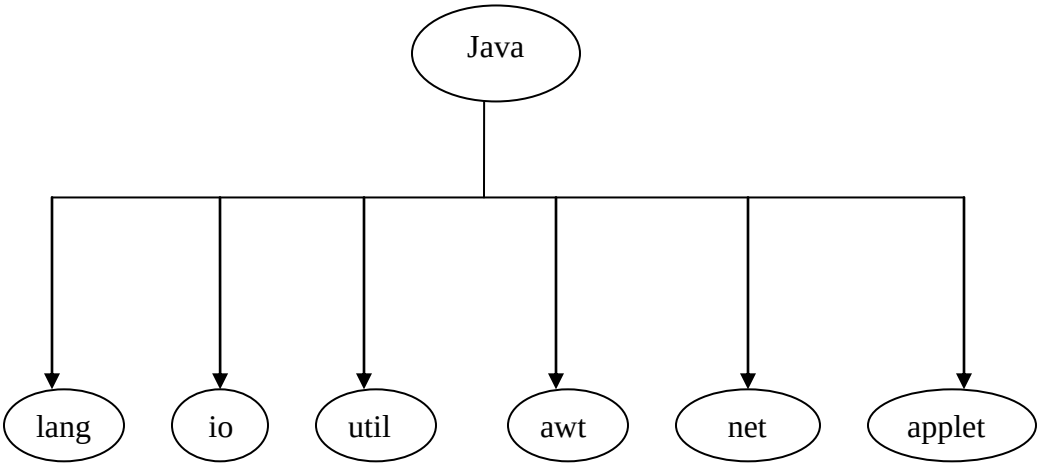
Packages

- Packages are Java's way of grouping a variety of classes and/or interfaces together.
- Packages act as "containers" for classes.
- The grouping is usually done according to functionality.
- The classes contained in the packages of other programs can be easily reused.
- In packages, classes can be unique compared with the classes in other packages, i.e. two classes in two different packages can have the same name.
- Packages provide a way to "hide" classes thus preventing other programs or packages from accessing classes that are meant for internal use only.
- Java packages are classified into two types-Java API packages and user defined packages.

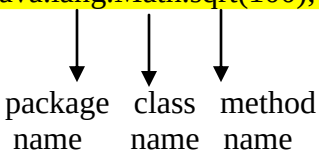
- **Java API packages**



Package name	Contents
java.lang	Language support classes. It includes classes for primitive types, strings, math functions, threads and exceptions.
java.util	Language utility classes such as vectors, hash tables, random numbers, date etc.
java.io	Input/output support classes. They provide facilities for the input/ output of data.
java.awt	Set of classes for implementing graphical user interface. They include classes for windows, buttons, lists, menus and so on.
java.net	Classes for networking. They include classes for communicating with local computers as well as with internet servers.
java.applet	Classes for creating and implementing applets.

- To access a particular class stored in a package, use the fully qualified class name of the class. The syntax is – **import** packagename.classname;
- To access a number of classes stored in a package use the following syntax- **import** packagename.*****;
- The above statement imports every class contained in the specified package.
- The **import** statements must appear at the top of the file before any class declaration.
- Packages can be named using the standard Java naming rules.
- Packages begin with lowercase letters. For example-

```
double y=java.lang.Math.sqrt(100);
```



- Every package name must be unique. Duplicate package name will cause run-time errors.
- Since multiple users work on internet, duplicate package names are unavoidable. Java designers have recognized this problem and therefore suggested a package naming convention that ensures uniqueness. This suggests the use of domain names as prefix to the preferred package names. For example- xyz.abc.mypackage
  Where xyz denotes city name and abc denotes organization name.
- We can create a hierarchy of packages within packages by separating levels with dots.

- **Creating packages**

- To create a package first declare the name of the package using the **package** keyword followed by a package name. This must be the first statement in a Java source file.
  For example consider the following-

```
package mypackage;
Public class Myclass
{
  Body of the class
}
```

- Here the package name is mypackage and the class Myclass is now considered a part of this package.
- This listing would be saved as a file called Myclass.java and located in a directory named mypackage. When the source file is compiled, Java will create a .class file and store it in the same directory.
- The .class files must be located in a directory that has the same name as the package, and this directory should be a subdirectory of the directory where classes that will import the package are located.
- Java also supports the concept of package hierarchy. This is done by specifying multiple names in a package statement separated by dots. For example-

```
package firstPackage.secondPackage ;
```

- This approach allows us to group related classes into a package and then group related packages into a larger package.

- Let us now consider a simple program that will use classes from other packages-

```
package mypackage;
public class Firstclass
{
  public void show()
   {
     System.out.println("Within Firstclass!");
    }
 }
```

- The source file should be named Firstclass.java and stored in the subdirectory mypackage. Now compile this java file. The resultant Firstclass.class will be stored in the same subdirectory.
- Now consider the listing given below-

```
import mypackage.Firstclass;
class Testpackage
{
  public static void main(String args[])
  {
    Firstclass ob1=new Firstclass();
    ob1.show();
   }
 }
```

- During the compilation of Testpackage.java the compiler checks for the file Firstclass.class in the mypackage directory for the information it needs, but it does not actually include the code from Firstclass.class in the file Testpackage.class.
- When the Testpackage program is run, Java looks for the file Testpackage.class and loads it using something called class loader. Now the interpreter knows that it also needs the code in the file Firstclass.class and loads it as well.
- When we import multiple packages it is likely that two or more packages contain classes with identical names. Since the compiler cannot understand which one to use and therefore generates an error. In such instance, we have to be more explicit about which one we intend to use. For example consider the following code fragment-

```
package mypackage1;
public class Student
{……}
public class Teacher
{……}

package mypackage2;
public class Student
{……}
public class Courses
{……}
```

```
import mypackage1;
import mypackage2;
Student ob1; //create a Student object but it is erroneous
```

- To solve this problem use the following-

```
mypackage1.Student ob1;
mypackage2.Student ob2;
```

- It is also possible to subclass a class that has been imported from another package.

- To add a class in a package follow the steps given below-
  1. Define the class and make it public.
  2. Place the package statement before the class definition as follows-

```
package mypackage;
public class Newclass
{……..}
```

  3. Store this as Newclass.java file under the directory mypackage.
  4. Compile Newclass.java file. This will create Newclass.class file and place it in the directory mypackage.

- We can also add a non-public class to a package using the same procedure.
- Since a Java source file can have only one class declared as public, we cannot put two or more public classes together in a .java file. This is because of the restriction that the file name should be same as the name of the public class with .java extension.

- If you want to create a package with multiple public classes in it then follow the steps given below-
  1. Give a name of the package.
  2. Create a subdirectory with this name under the directory where main source files are stored.
  3. Create classes that are to be placed in the package in separate source files and declare the package statement at the top of each source file.
  4. Switch to the subdirectory created earlier and compile each source file. When completed, the package would contain .class files of all the source files.

- When we import packages using asterisk (*), all public classes are imported. However we may prefer to not import certain classes. That is, we may like to **hide** these classes from accessing from outside of the package. Such classes should be declared not public.