Chapter 15: Java Packages & App Distribution

- In today's era, the major concern for Java programmers is to ensure that their source code does not conflict with the source code of other Java programmers.
- Source code conflict arises when two Java programmers define two classes performing different tasks but have the same name.
- Concept called Package was introduced to solve this problem.
- Package in Java is an encapsulation mechanism that can be used to group related classes, interfaces, enums and subpackages.

Package Names:

- Conventionally, a global naming scheme based on the Internet domain names is used to uniquely identify the packages i.e. take the domain name and add a project name to it.
- Eg:- **com.company.myproject** is a package name.
- The package statement has the following syntax: Package <full qualified package name>;
- The classes or interfaces or enums defined in a package must have their source files in the same directory structure as the name of the package.

Note:

If the package declaration is omitted in a program, the Java byte code for the declarations in the program will belong to an unnamed package (also called default package) which is typically synonymous with the current working directory on the host system.

Package Structure:

```
package packageName;
```

```
import package.*

import package.subpackage.*;
```

```
class className { }

class className { }

interface interfaceName { }

enum enumName { }
```

Note1: class cannot have private or protected modifier.

Note2: default class cannot be accessed from outside the pkg.

- **Create:**
  - javac -d destination_folder source_folder
- **Using Package: (from Java 1.0)**
  - **import statement:**
    - import pkgName .*;
    - import pkgName.className;
- **Using Package : (from Java 5.0)**
  - **static import statement:**
    - import static pkgName.className.*;
    - import static pkgName.className.memName;

Access Modifier:

Access / Location	private	public	protected	default
Inside class	Yes	Yes	Yes	Yes
Other class in the same package	No	Yes	Yes	Yes
Sub class in the same package	No	Yes	Yes	Yes
Other class in other package	No	Yes	No	No
Sub class in other package	No	Yes	Yes	No

JPMS:

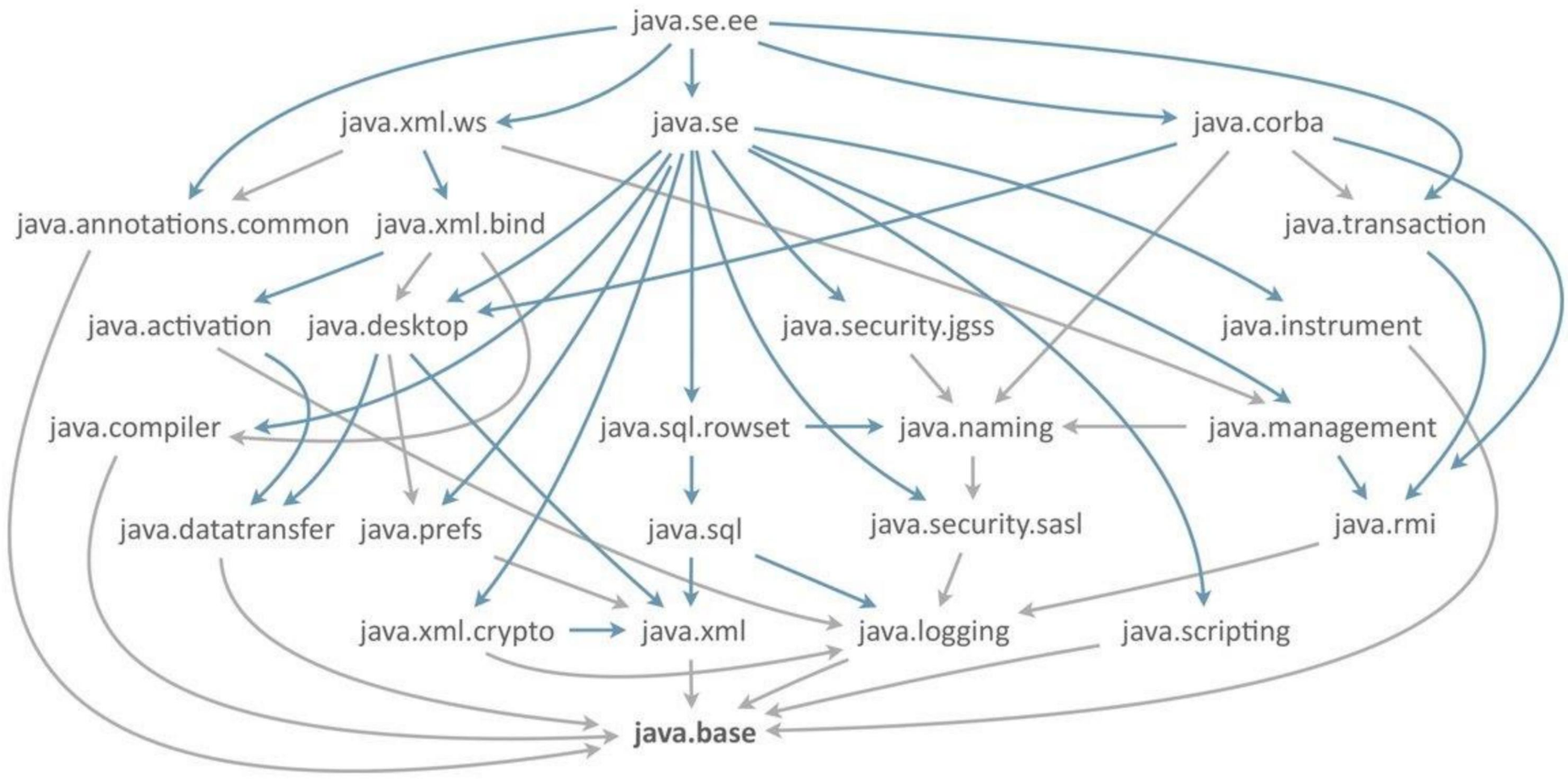
- Project Jigsaw has designed and implemented standard module system for the Java SE 9 platform.
- The modularity features from Java 9 are together referred to by the name Java Platform Module System (JPMS).
- Java introduces a new language construct to create reusable components called modules.
- JPMS makes it easy for developers to create contained units that have clearly established dependencies on other modules.

Module:

- Module is a named, self-describing collection of packages which in turn contain classes and interfaces.

Handling inter module dependencies:

- Breaking the application into modular parts enables these modular parts to potentially be reused in multiple other applications.
- **export:**
  - Every module needs to specifically declare what types in its module are okay for use outside the module.
  - This is done using export followed by the package you want to export.
- **requires:**
  - This is the way to declare a dependency on a module.
  - We write requires followed by the name of the module that is required.
  - Note: we can require other modules, and not packages or classes.



JLink:

- To execute a small “hello world” application we require the following .class files: Main.class, String.class, System.class and Object.class.
- The default JRE provided contains 4300+ predefined Java .class files which occupies 226MB size.
- Drawbacks of default JRE:
  - Waste of memory
  - Performance hit
  - Not suitable of IoT devices.
- JLink is the command line tool through which we can create our own customize JRE.