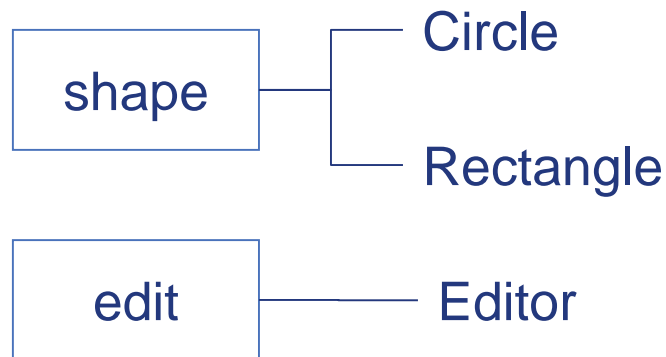


Packages

- ❖ Package definition
- ❖ Accessing classes in a package
- ❖ The keyword “import”
- ❖ Default package
- ❖ public class and non-public class
- ❖ Protected fields/methods
- ❖ Static import
- ❖ Class path

Package

- ❖ Package is a group of classes and interfaces.
- ❖ Package is like directory ! In other words,
 - A package contains multiple classes, even sub-packages.
 - Classes can have the same name as long as they are within different packages.



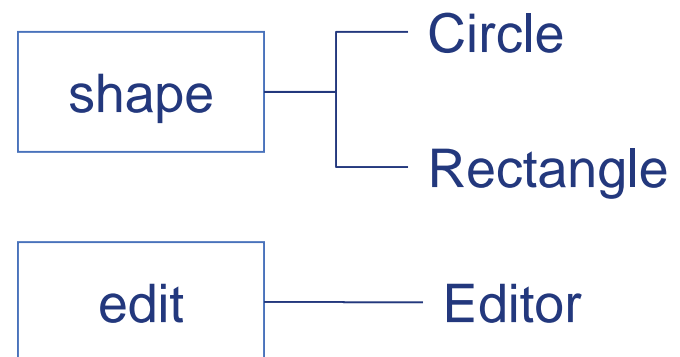
Package: Definition

- ❖ The keyword “package” is used to define the package for all the classes in a file.

```
// Circle.java
package shape;
public class Circle {
    ...
}
```

```
// Rectangle.java
package shape;
public class Rectangle {
    ...
}
```

```
// Editor.java
package edit;
public class Editor {
    ...
}
```



Accessing classes in a package

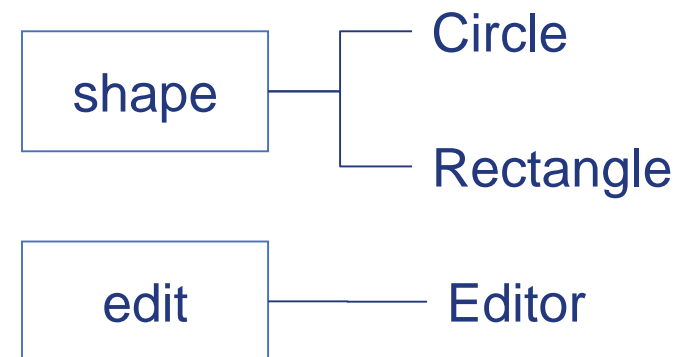
- ❖ A class in a package should be accessed with its package name

```
// Editor.java
package edit;
public class Editor {
    Circle c = new Circle() ;
    Rectangle r = new Rectangle() ;
}
```

**Circle cannot be
resolved to a type**



```
// Editor.java
package edit;
public class Editor {
    shape.Circle c = new shape.Circle() ;
    shape.Rectangle r = new shape.Rectangle() ;
}
```



Accessing classes in the same package

- ❖ You can omit the package name if the class is in the same package.

```
// EditorTest.java
```

```
package edit;
```

```
public class EditorTest {
```

```
    public static void main(String[] args) {
```

```
        Editor ed = new Editor();
```

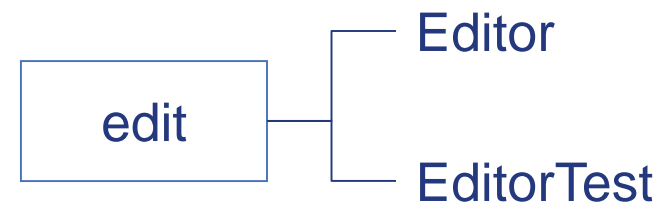
```
        // edit.Editor ed = new edit.Editor();
```

```
        shape.Circle c = new shape.Circle();
```

```
        shape.Rectangle r = new shape.Rectangle();
```

```
    }
```

```
}
```



The package of class Editor is the same with EditorTest

import

- ❖ The keyword “import” provides a convenient method for using classes in other packages
- ❖ You can access all the classes in the imported package without the package name

```
// Editor.java
```

```
package edit;  
public class Editor {  
    shape.Circle c = new shape.Circle() ;  
    shape.Rectangle r = new shape.Rectangle() ;  
}
```

Without import

```
// Editor.java
```

```
package edit;  
import shape.*; // import all the classes in the shape package  
public class Editor {  
    Circle c = new Circle() ;  
    Rectangle r = new Rectangle() ;  
}
```

With import

import

- ❖ Each class can be imported individually.

```
// Editor.java
package edit;
import shape.*; // import all the classes in the shape package
public class Editor {
    Circle c = new Circle() ;
    Rectangle r = new Rectangle() ;
}
```

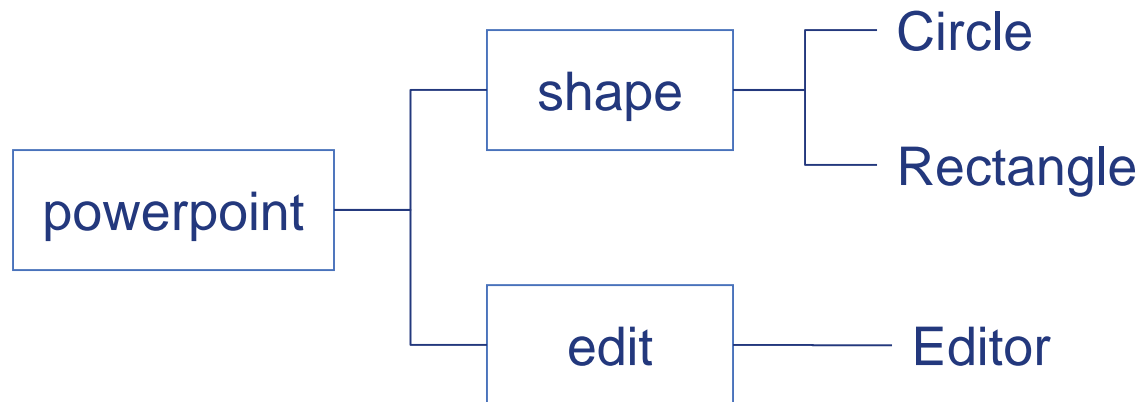
Importing all the
classes in a
package

```
// Editor.java
package edit;
import shape.Circle;
import shape.Rectangle;
public class Editor {
    Circle c = new Circle() ;
    Rectangle r = new Rectangle() ;
}
```

Importing an
individual class

Sub-packages

- ❖ A package can contain another packages (sub-packages) as well as classes.



Sub-packages

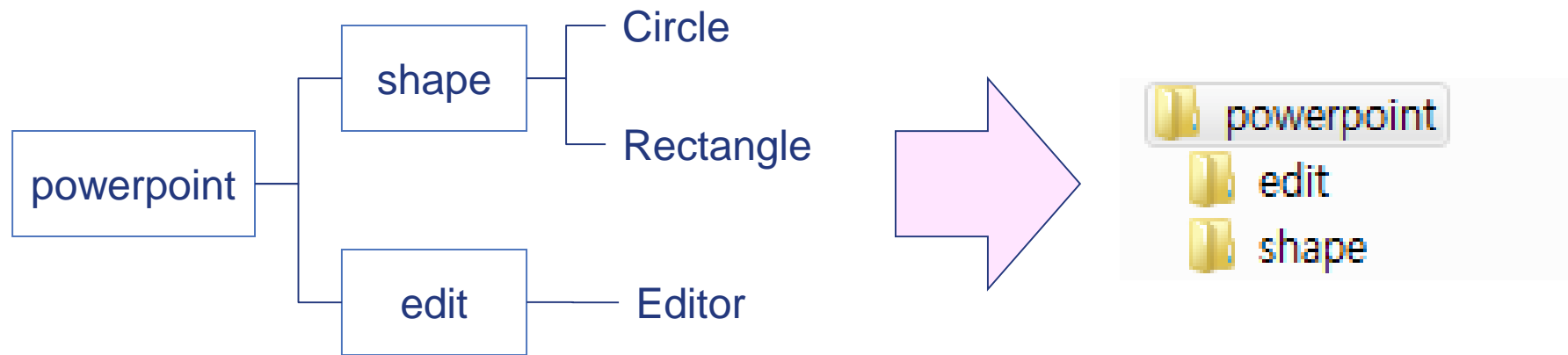
```
// Circle.java
package powerpoint.shape;
public class Circle {
    ...
}
```

```
// Rectangle.java
package powerpoint.shape;
public class Rectangle {
    ...
}
```

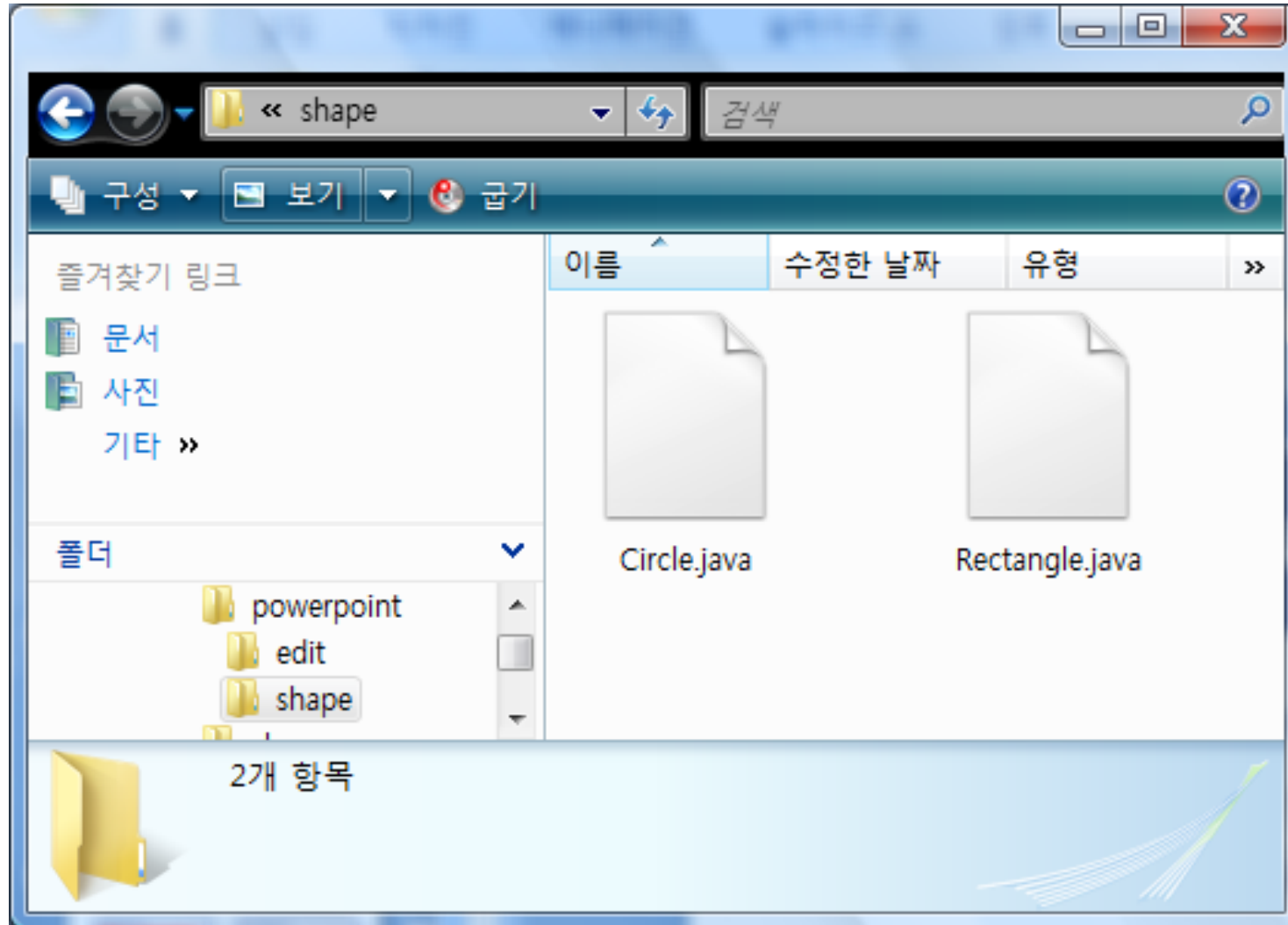
```
// Editor.java
package powerpoint.edit;
import powerpoint.shape.*;
public class Editor {
    Circle c = new Circle() ;
    Rectangle r = new Rectangle() ;
}
```

Package and Directory

- ❖ The packages are implemented with directory structures.
- ❖ In other words, the packages should have the same structures with its directories.

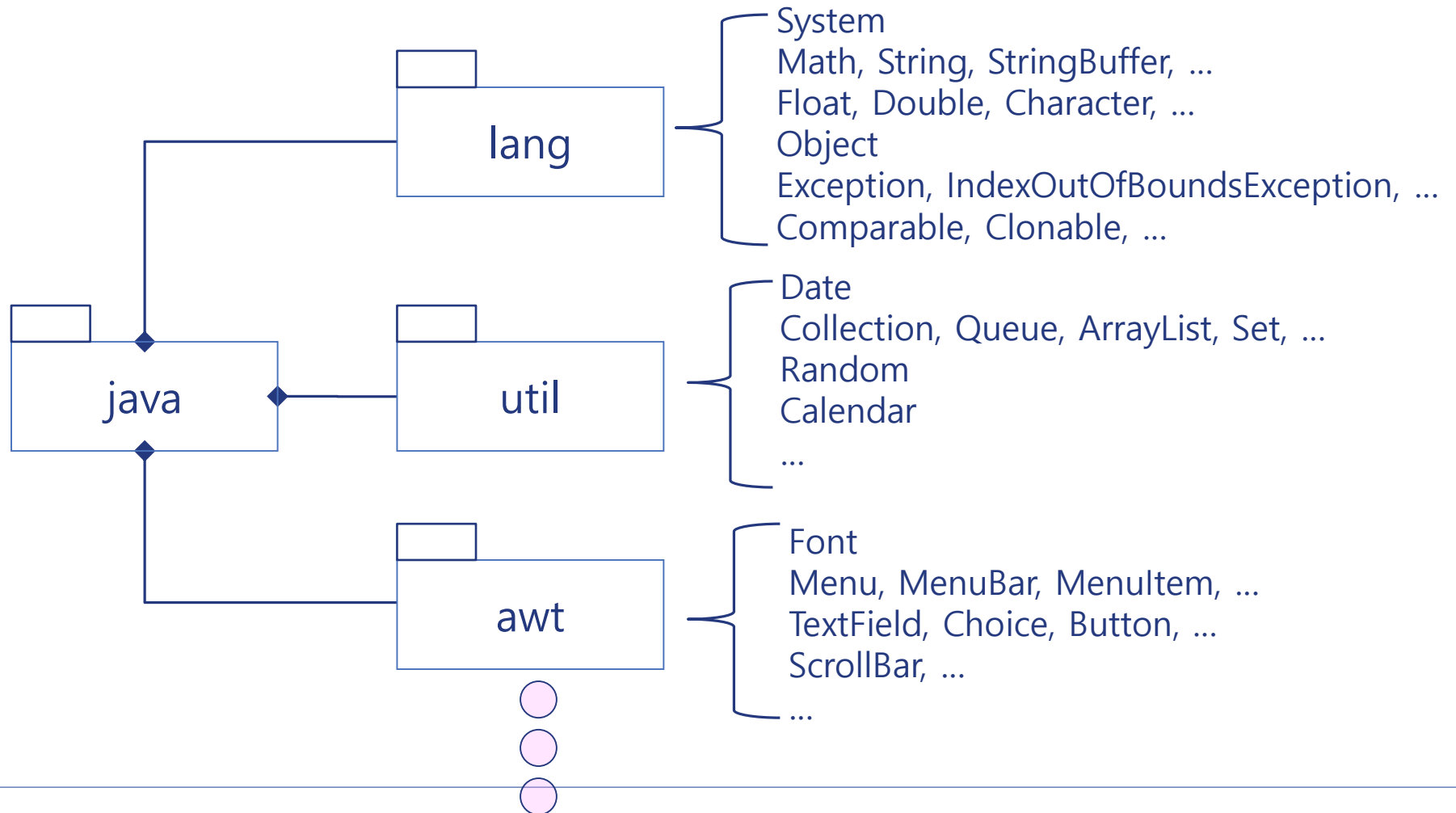


Package and Directory

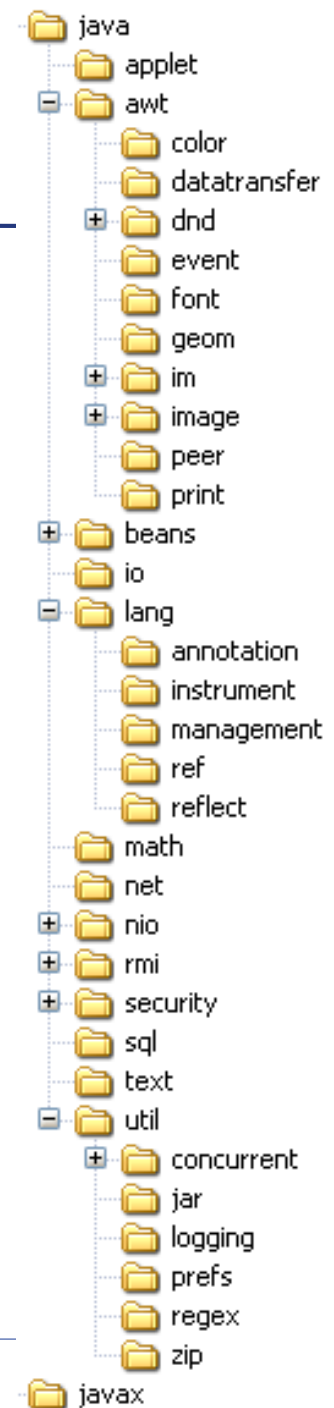


Package Hierarchy

- ❖ The standard Java library is distributed over a number of packages.



More Java Packages



java.lang package

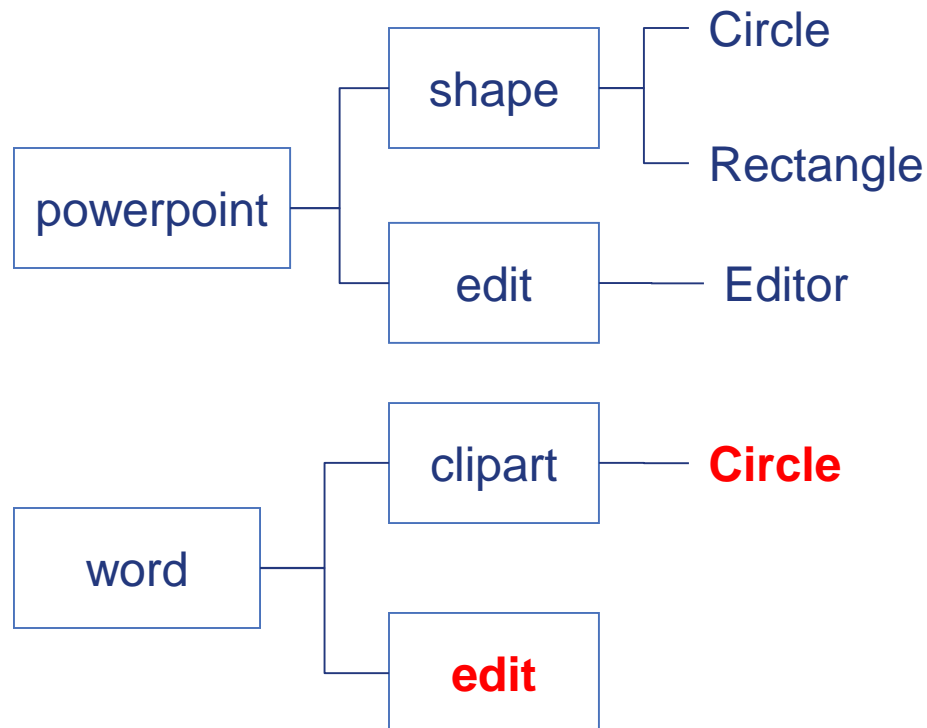
- ❖ Many basic classes and interfaces are in package java.lang
- ❖ They are automatically imported; You don't need to import it.

```
// import java.lang.* ; // defines System, String, ...
import java.util.* ; // defines Scanner, ArrayList

public class JavaLangPackageTest {
    public static void main(String[] args) {
        Scanner scannerObject = new Scanner(System.in) ;
        List<String> strs = new ArrayList<String>() ;
        while ( true ) {
            String word = scannerObject.next() ;
            if ( word.equals("quit") ) break ;
            strs.add(word) ;
        }
        for ( String str: strs ) System.out.println(str) ;
    }
}
```

Package: Uniqueness of name

- ❖ You can define multiple classes/packages with the same name in different packages.



Package: Benefits

- ❖ Packages are usually used to avoid name conflict!
- ❖ Assuming that you develop a `ArrayList<T>` with better performance than in the JDK Library

```
public class MyArrayList<Comparable<T>> {  
    ...  
}
```

Without package, we have to give different name

```
package edu.pnu.PBP2020;  
public class ArrayList<Comparable<T>> {  
    ...  
}
```

You can use the same name `ArrayList` with you own package.

Package: Benefits

- ❖ Program can be evolved by indicating different packages.
- ❖ Assuming that the my ArrayList is better than the JDK version.

```
import java.util.* ;  
public class MyPowerPoint {  
    private ArrayList<Circle> circles ;  
    ...  
}
```

MyPowerPoint used ArrayList in JDK.

```
import edu.pnu.PBP2020.* ;  
public class MyPowerPoint {  
    private ArrayList<Circle> circles ;  
    ...  
}
```

Now, better ArrayList in my package is used.

Default package


- ❖ Each class with no package definition is assumed to belong to the default package.

```
public class MyRectangle {  
    ...  
}
```



```
public interface MyComparable {  
    ...  
}
```

```
public class Circle  
    implements MyComparable {  
    ...  
}
```



All of them
belong to the
default package

Visibility of classes in packages

❖ Public class or non-public(default) class

```
// Circle.java  
package powerpoint.shape;  
public class Circle {  
    ...  
}
```

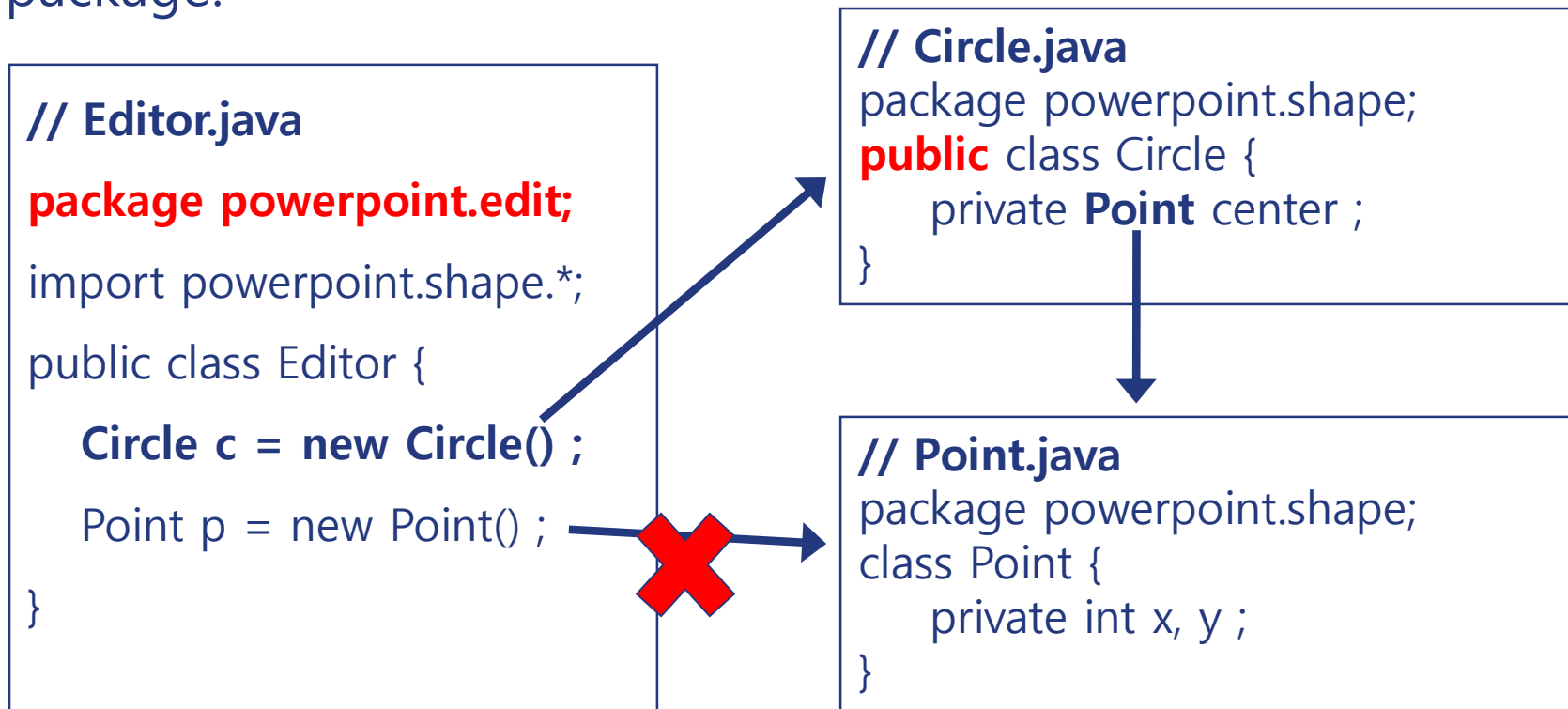
Circle is a public class

```
// Point.java  
package powerpoint.shape;  
class Point {  
    ...  
}
```

Point is NOT a public class

Visibility of classes in packages

- ❖ Only public classes can be accessed from outside of the package
- ❖ Non-public classes can be accessed from the class in the same package.

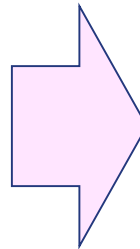



One public class in a file

- ❖ Only one public class can be defined in a source file.
- ❖ You cannot define multiple public classes in a source file.

```
// Shape.java
package powerpoint.shape;
public class Circle {
    private Point center ;
}

public class Rectangle {
    private Point leftTop, rightBottom;
}
```



```
// Circle.java
package powerpoint.shape;
public class Circle {
    private Point center ;
}
```

```
// Rectangle.java
package powerpoint.shape;
public class Rectangle {
    private Point leftTop, rightBottom;
}
```

One public class in a file

- ❖ Each file does not need to define a public class. It can only include non-public classes.

```
// Point.java
```

```
package powerpoint.shape;
```

```
class Point {
```

```
    private int x, y ;
```

```
}
```

No public class in the
Point.java

Package visibility

- ❖ Package(default visibility) fields/methods can be accessed from **EVERY** class in the same package.

```
// Point.java
package powerpoint.shape;
class Point {
    int x, y ;
}
```

allows

```
// Circle.java
package powerpoint.shape;
public class Circle {
    private Point center ;
    void setCenter(int x, int y) {
        center.x = x ;
        center.y = y ;
    }
}
```

- ❖ Change in class Point can cause the change in Circle.
- ❖ Therefore, you should not use package fields/methods.

static import

- ❖ Starting with JDK 5.0, static import has been introduced to permit the importing of static methods and fields, not just classes.

```
import static java.lang.System.*;  
// Note that System is a class, not a package  
public class StaticImportTest {  
    public static void main(String args[]) {  
        out.println("Hello World"); // java.lang.System.out  
        exit(0) ; // java.lang.System.exit()  
    }  
}
```


static import

```
Math.sqrt(Math.pow(x, 2)+ Math.pow(y,2))
```



```
import static java.lang.Math.*;  
sqrt(pow(x, 2)+ pow(y,2))
```

```
if ( d.get(Calendar.DAY_OF_WEEK) == Calendar.MONDAY )
```



```
import static java.util.Calendar.* ;  
if ( d.get(DAY_OF_WEEK) == MONDAY )
```

How the JVM locates classes?

❖ **Class Path:** directories and archive files for locating classes.

- -classpath or -cp option in java command
- CLASSPATH environment variable

Note: there is no default on the current directory.

❖ Let's consider a class path: **C:\java\.;C:\java\lib\archive.jar**

❖ Suppose the JVM searches for the class file of the edu.pnu.shape.Circle

1. Looks in the system class files that are stored in archives in the **jre/lib and jre/lib/ext**
2. **According to the specified CLASSPATH**
 1. **C:\java\edu\pnu\shape\Circle.class**
 2. edu\pnu\shape\Circle.class starting from **the current directory**
 3. edu\pnu\shape\Circle.class **inside c:\java\lib\archive.jar**

More on Class Path

- ❖ -classpath or -cp option in java command; used for the particular program
 - % java -cp C:\jdk1.0 MyRectangleTest
 - % java -cp C:\jdk1.5 MyCircleTest
- ❖ CLASSPATH environment variable; shared by all Java programs
 - SET CLASSPATH=C:\jdk1.0
 - % java MyRectangleTest
 - % java MyCircleTest
- ❖ -cp has priority over CLASSPATH
 - SET CLASSPATH=C:\jdk1.0
 - % java MyRectangleTest
 - % java -cp C:\jdk1.5 MyCircleTest
- ❖ Current directory
 - The current directory is not automatically included in the class path
 - You have to add the current directory in the class path
 - SET CLASSPATH=C:\jdk1.0;.
 - % java -cp C:\jdk1.5;. MyCircleTest

Q&A
