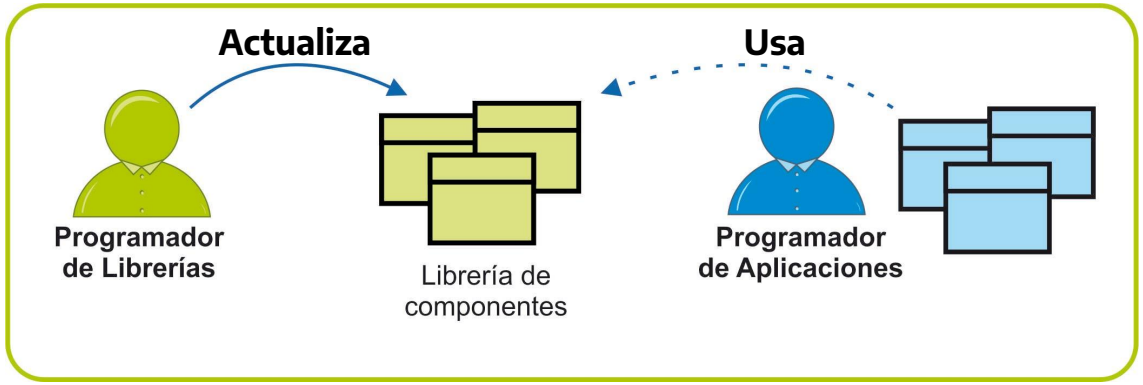
Taller de Lenguajes II

Librerías de componentes

- Creación de librerías: paquete
- Nombres únicos de clases
- Archivos JAR (Java ARchive)

¿Qué podría pasar si se modifica una librería de clases que está siendo usada por otros programadores?



El código podría romperse !!

El **programador de la librería** debe sentirse libre para **mejorar el código** y el **programador de aplicaciones** debería poder **aplicar las mejores** sin necesidad de re-escribir su código.

¿Cómo se asegura esto?

(1) **Garantizando compatibilidad con versiones previas**: no quitar métodos existentes en la versión previa.

(2) **Usando especificadores de acceso** para indicarle al programador de aplicaciones qué está disponible y qué no.

Antes de entrar en especificadores de acceso, falta responder una pregunta útil en este contexto: **¿cómo se crea una librería de clases en java?**

Librería de componentes
Paquetes JAVA

- En Java **una librería de componentes** (clases e interfaces) es un agrupamiento de archivos `.class`, también llamado **paquete.**
- Para **agrupar componentes** en un paquete, debemos anteponer la palabra clave **`package`** junto con el nombre del paquete al comienzo del archivo fuente de cada una de las clases o interfaces.

```
package graficos;
public class Rectangulo {
 //código JAVA
}
```

```
package graficos;
public interface Centrable {
 //código JAVA
}
```

La clase **Rectangulo** y la interface **Centrable** pertenecen al paquete **graficos**

- Las clases e interfaces que se crean **sin usar la sentencia `package`** se ubican en un paquete sin nombre, llamado ***default package***.

```
public class HolaMundo {
      //código JAVA
}
```

La clase **HolaMundo** pertenece al **paquete por defecto**

Usar paquetes propios o el *default package* ¿Qué opinan? ¿por qué?

Librería de componentes
Paquetes JAVA

El **nombre completo de la clase o nombre canónico** contiene el nombre del paquete.

```
graficos.Rectangulo  → Nombre completo de la clase Rectangle
java.util.Arrays     → Nombre completo de la clase Arrays
```

Para usar la clase **Rectangulo** se debe usar la palabra clave **`import`** o **especificar el nombre completo** de la clase:

```
package ar.edu.unlp.taller2;
import graficos.Rectangulo;
// import graficos.*;

class Figuras {
 Rectangulo r = new Rectangulo();
}
```

Importación por demanda: se tienen disponibles todos los nombres de clases e interfaces del paquete

```
package ar.edu.unlp.taller2;
class Figuras {
 graficos.Rectangle r;
 r = new graficos.Rectangle();
}
```

La sentencia **`import`** permite usar el **nombre corto de la clase** en todo el código fuente. Si no se usa el **`import`** se debe especificar el **nombre completo** de la clase.

Librería de componentes

Paquetes JAVA

¿Qué sucede si se crean 2 clases con el mismo nombre?

Supongamos que 2 programadores escriben una clase de nombre **Vector** en el paquete *default,* **se plantea un conflicto de nombres**.

Es necesario **crear nombres únicos: usamos paquetes.**

```
package util;
public class Vector {
   //código JAVA
}
```

```
package taller2.estructuras;
public class Vector {
    //código JAVA
}
```

¿Qué sucede si se importan dos librerías que incluyen el mismo nombre de clase?

```
import taller2.estructuras.*;
import util.*;

Vector vec1 = new Vector();
```

Ambos paquetes contienen la clase Vector

Colisión! ¿A qué clase hace referencia?: el compilador no puede determinarlo

Librería de componentes

Paquetes JAVA

```
package util;
public class Vector {
   //código JAVA
}
```

```
package taller2.estructuras;
public class Vector {
    //código JAVA
}
```

¿Qué sucede si se importan dos librerías que incluyen el mismo nombre de clase?

```
import taller2.estructuras.*;
import util.*;

Vector vec1 = new Vector();
```

Ambos paquetes contienen la clase Vector

Colisión! ¿A qué clase hace referencia?: el compilador no puede determinarlo

Una posible solución:

```
import util.*;

Vector vec1 = new Vector();
taller2.estructuras.Vector vec2 = new taller2.estructuras.Vector();
```

Combinamos **nombre corto** y **nombre completo**

Librería de componentes
Paquetes JAVA

- Las clases e interfaces que son parte de la distribución estándar de JAVA están agrupadas en paquetes de acuerdo a su funcionalidad. Algunos paquetes son:

`java.lang`	**clases básicas para crear aplicaciones.**
`java.util`	**librería de utilitarios, colecciones.**
`java.io`	**manejo de entrada/salida.**
`java.awt /javax.swing`	**manejo de GUI (Graphic User Interface)**.

- Los únicos paquetes que se importan automáticamente es decir no requieren usar la sentencia `import` son el paquete `java.lang` y el **paquete actual** (paquete en el que estamos trabajando).

```
String s="hola";
System.out.print("hola");
```

String y **System** son clase de java.lang, se **pueden usar directamente**

```
package taller2.estructuras;
public class Vector {
  //código JAVA
}
```

En **Vector** se pueden usar todas las clases del paquete **taller2.estructuras** sin importar

Recomendación: usar como primera parte del **nombre del paquete el nombre invertido del dominio de Internet** y así evitar conflicto de nombres. Usar **minúscula para nombres de paquetes** e **inicial mayúscula para nombres de clases**. **Ejemplo:** `ar.edu.unlp.graficos`

Librería de componentes

Paquetes JAVA

Un paquete normalmente está formado por varios archivos **.class**.
Java se beneficia de la **estructura jerárquica de directorios del sistema operativo** y ubica todos los **.class** de un mismo paquete en un mismo directorio. De esta manera, se resuelve:

- el nombre único del paquete
- la búsqueda de los **.class** (que de otra forma estarían diseminados en el disco)

```
package ar.edu.unlp.utiles;
public class Vector {
    //código JAVA
}
```

\ar\edu\unlp\utiles\Vector.class

Cuando el "intérprete" JAVA ejecuta un programa y necesita localizar dinámicamente un archivo **.class**, por ej. cuando se crea un objeto o se accede a un miembro **static**, procede de la siguiente manera:

- Busca en los **directorios estándares del JRE**
- Busca en el directorio actual (paquete de la clase que se está ejecutando)
- Recupera la variable de entorno **CLASSPATH**, que contiene la lista de directorios usados como raíces para buscar los archivos **.class**. Comenzando en la raíz, el intérprete toma el nombre del paquete (de las sentencias **import**) y reemplaza cada **"."** por una barra **"\"** o **"/"** (según el SO) para generar un camino donde encontrar las clases a partir de las entradas del **CLASSPATH**.

Librería de componentes

Paquetes JAVA

Consideremos el dominio **unlp.edu.ar** invertido y obtenemos un nombre de dominio único y global: **ar.edu.unlp**. Si creamos una librería **utiles** con las clases **Vector** y **List**, tendríamos:

```
package ar.edu.unlp.utiles;
public class Vector {
   //código JAVA
}
```

```
package ar.edu.unlp.utiles;
public class List {
   //código JAVA
}
```

Supongamos que a ambos archivos los guardamos en el directorio **c:\tallerjava**.

```
C:\tallerjava\ar\edu\unlp\utiles\Vector.class
C:\tallerjava\ar\edu\unlp\utiles\List.class
```

El **"intérprete" JAVA** comienza a **buscar el paquete ar.edu.unlp a** partir de alguna de las **entradas** indicadas en la variable de entorno **CLASSPATH**:

```
CLASSPATH=.;c:\tallerjava;c:\java\librerias
```

Esta variable puede contener muchas entradas separadas por ";"

Paquetes en JAVA
Formato JAR

Es posible agrupar archivos **.class** pertenecientes a uno o más paquetes en un único archivo con extensión **jar (Java ARchive)**. El formato **JAR** usa el formato **zip**. Los archivos JAR son multi-plataforma, es estándar. Es posible incluir además de archivos .class, archivos de imágenes y audio, recursos en general, etc.

El JSE o JDK tiene una herramienta para crear archivos JAR, desde la línea de comando, es el utilitario **jar**.

Por ejemplo: si se ejecuta el comando **jar** desde el directorio donde están los archivos **.class** podríamos ponerlo así:

```
c:\tallerjava\ar\edu\unlp\utiles\jar cf utiles.jar *.class
```

- En este caso, en el **CLASSPATH** se especifica el nombre del archivo jar:

```
CLASSPATH=.; c:\utiles.jar ;c:\java\librerias
```

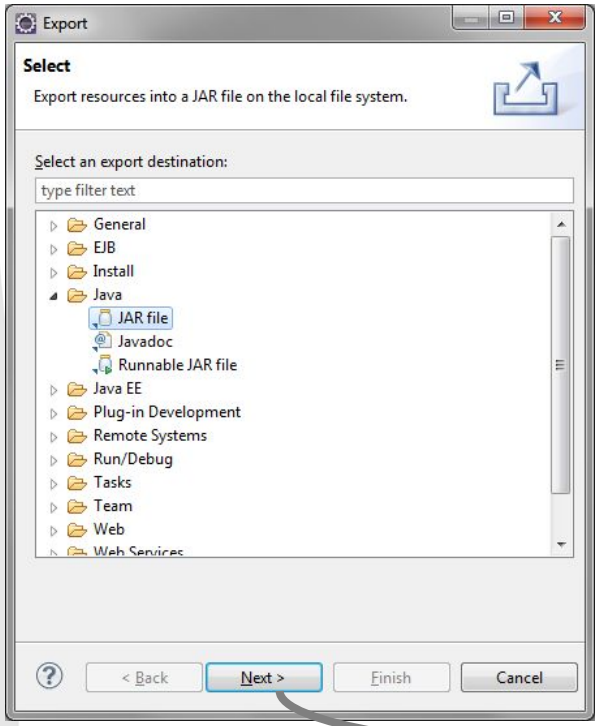
Los archivos jar pueden ubicarse en cualquier lugar del disco

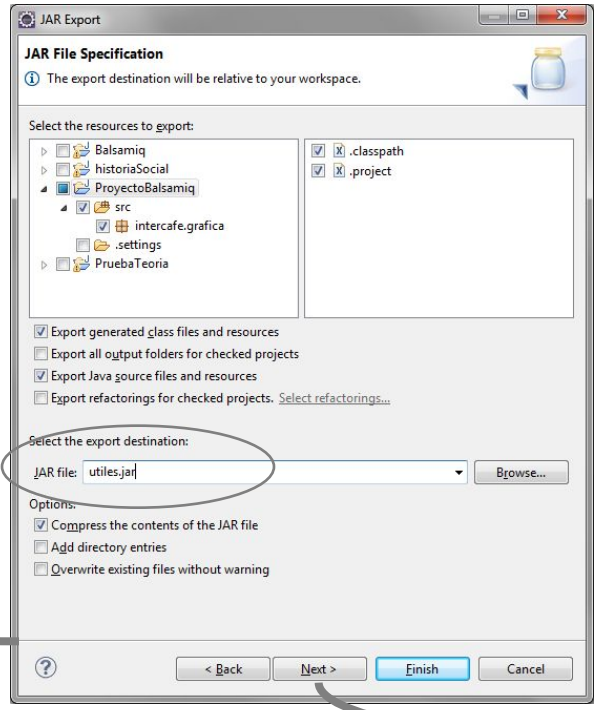
- El "intérprete" JAVA se encarga de buscar, descomprimir, cargar e interpretar estos archivos.

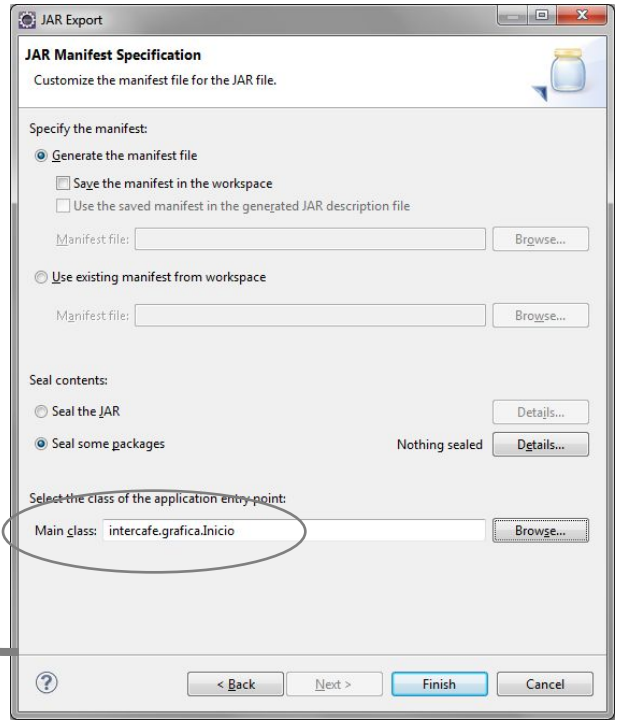
Paquetes en JAVA

Formato JAR

El archivo JAR también puede construirse desde un proyecto Eclipse, con al opción **`export`**.







Paquetes en JAVA
El formato JAR

Los archivos JAR contienen todos los paquetes con sus archivos .class, los recursos de la aplicación y un archivo **MANIFEST.MF** ubicado en el camino META-INF/MANIFEST.MF, cuyo **propósito es indicar cómo se usa el archivo JAR**.

Las **aplicaciones de escritorio** a diferencia de las librerías de componentes o utilitarias, **requieren que el archivo MANIFEST.MF** contenga una **entrada con el nombre de la clase** que actuará como punto de entrada de la aplicación (la clase que contiene método main).

Para especificar la **clase "principal"**, el archivo MANIFEST.MF debe contener la **entrada Main-Class**.

```
Manifest-Version: 1.0
Created-By: 1.6.0_12 (Sun Microsystems Inc.)
Main-Class: capitulo4.paquetes.TestOut.class
```