

NOTAS – REPASO – EDITS:   
[UT1, UT3 Generación de números aleatorios] status: updated  
Falta: [UT4 Programación Orientada a Objetos, final]

# UT6 Paquetes en Java

**PAQUETES EN JAVA**

Haremos un fichero HolaMundo en java usando **paquetes**.

* **[Qué son paquetes en java](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php" \l "paquetes).**
* **[La clase HolaMundo en un paquete](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php" \l "holamundo).**
* **[Estructura de directorios asociada](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php" \l "directorios).**
* **[Compilar clases de un paquete](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php" \l "compilar).**
* **[Ejecutar clases de un paquete](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php" \l "ejecutar).**
* **[El Classpath cuando hay paquetes](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php" \l "classpath).**
* **[Paquetes anidados](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php" \l "anidados).**

**QUÉ SON PAQUETES EN JAVA**

Cuando hacemos programas más grandes, es normal que el número de clases vaya creciendo. Cada vez tenemos más y más clases. Meterlas todas en el mismo directorio no suele ser buena idea. Es mejor hacer **grupos de clases**, de forma que todas las clases que traten de un determinado tema o estén relacionadas entre sí vayan juntas. Por ejemplo, si hacemos un programa de una agenda de teléfonos que los guarde en una base de datos, podemos meter todas las clases que tratan con la base de datos en un **paquete** (grupo), todas las de ventanas en otro, las de imprimir en otro, etc.

Un **paquete** de clases es un grupo de clases que agrupamos juntas porque consideramos que están relacionadas entre sí o tratan de un tema común.

**LA CLASE HOLA MUNDO EN UN PAQUETE**

**Java** nos ayuda a organizar las clases en **paquetes**. En cada fichero .java que hagamos, al principio, podemos indicar a qué **paquete** pertenece la clase que hagamos en ese fichero.

Por ejemplo, si decidimos agrupar nuestros programas de pruebas, incluido el Hola Mundo, en un paquete "prueba", pondríamos esto en nuestro fichero HolaMundo.java

package prueba;

public class HolaMundo

{

public static void main (String [ ] args)

{

System.out.println ("Hola mundo");

}

}

El código es exactamente igual al de nuestro Hola Mundo original, pero hemos añadido la línea "**package prueba;**" al principio.

Hasta aquí todo correcto. Sin embargo, al hacer esto, para que todo funcione bien, java nos obliga a organizar los directorios, compilar y ejecutar de cierta forma.

**ESTRUCTURA DE DIRECTORIOS ASOCIADA**

Si hacemos que HolaMundo.java  pertenezca al paquete prueba, java nos obliga a crear un subdirectorio "prueba" y meter el HolaMundo.java ahí.

Es decir, debemos tener esto así (unix y windows)

/<directorio\_usuario>/mi\_proyecto\_HolaMundo/prueba/HolaMundo.java

C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo\prueba\HolaMundo.java

Las mayúsculas y minúsculas son importantes. Tal cual lo pongamos en "package", debemos poner el nombre del subdirectorio.

**COMPILAR UNA CLASE QUE ESTÁ EN UN PAQUETE**

Para compilar la clase que está en el paquete debemos situarnos en el directorio padre del paquete y compilar desde ahí

$ cd /<directorio\_usuario>/mi\_proyecto\_HolaMundo

$ javac prueba/HolaMundo.java

C:\> cd C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo

C:\> javac prueba\HolaMundo.java

Si todo va bien, en el directorio "prueba" se nos creará un HolaMundo.class

**EJECUTAR UNA CLASE QUE ESTÁ EN UN PAQUETE**

Una vez generado el HolaMundo.class en el directorio prueba, para ejecutarlo debemos estar situados en el directorio padre de "prueba". El nombre "completo" de la clase es "paquete.clase", es decir "prueba.HolaMundo". Por ello, para ejecutar, debemos hacerlo así

$ cd /<directorio\_usuario>/mi\_proyecto\_HolaMundo

$ java **prueba.HolaMundo**

Hola Mundo

C:\> cd C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo

C:\> java **prueba.HolaMundo**

Hola Mundo

Si todo va como debe, debería salir "Hola Mundo" en la pantalla

**EL CLASSPATH CUANDO LAS CLASES ESTÁN EN PAQUETES**

Cuando las clases están en paquetes, la [variable](http://www.chuidiang.org/java/novatos/hola_mundo_java.php" \l "classpath) **CLASSPATH** debe ponerse de tal forma que encuentre el paquete, NO la clase. Es decir, en nuestro ejemplo:

$ CLASSPATH=/<directorio\_usuario>/mi\_proyecto\_HolaMundo

$ export CLASSPATH

$ java prueba.HolaMundo

Hola Mundo

C:\> set CLASSPATH="C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo"

C:\> java prueba.HolaMundo

Hola Mundo

O bien, si usamos la opción -cp en la línea de comandos

$ java -cp /<directorio\_usuario>/mi\_proyecto\_HolaMundo prueba.HolaMundo

Hola Mundo

C:\> java -cp "C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo" prueba.HolaMundo

Hola Mundo

**PAQUETES ANIDADOS**

Para poder organizar mejor las cosas, java permite hacer **subpaquetes** de los paquetes y subpaquetes de los subpaquetes y así sucesivamente. Por ejemplo, si quiero partir mis clases de prueba en pruebas básicas y pruebas avanzadas, puedo poner más niveles de paquetes separando por puntos.

package prueba.basicas;

package prueba.avanzadas;

A nivel de subdirectorios tendría que crear los subdirectorios "basicas" y "avanzadas" debajo de "prueba" y meter ahí las clases que correspondan. Para compilar, en el directorio del proyecto habría que compilar poniendo todo el path hasta llegar a la clase. Lo mismo para el **CLASSPATH**. Para ejecutar, el nombre de la clase va con todos los paquetes separados por puntos, es decir "prueba.basicas.HolaMundo" y "prueba.avanzadas.HolaMundoRemoto".

Es decir:

/<directorio\_usuario>/mi\_proyecto\_HolaMundo/prueba/basicas/HolaMundo.java

$ cd /<directorio\_usuario>/mi\_proyecto\_HolaMundo

$ javac prueba/basicas/HolaMundo.java

$ java -cp /<directorio\_usuario>/mi\_proyecto\_HolaMundo prueba.basicas.HolaMundo

Hola Mundo

C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo\prueba\basicas\HolaMundo.java

C:\> cd C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo

C:\> javac prueba\basicas\HolaMundo.java

C:\> java -cp C:\<directorio\_usuario>\mi\_proyecto\_HolaMundo prueba.basicas.HolaMundo

Hola Mundo

**QUÉ SON LOS FICHEROS JAR**

Cuando tenemos un programa grande, con varios paquetes y clases, ya sabemos cómo organizarlo, compilarlo y ejecutarlo. Sin embargo, si queremos usarlo en otra máquina o en otro proyecto, tenemos que llevarnos directorios enteros, con los ficheros que hay dentro y demás.

Lo ideal es meter todos estos ficheros y directorios en un único fichero comprimido. Java, con su comando jar, que está en el directorio bin de la carpeta donde tengamos java, nos permite hacer esto. Empaqueta todo lo que le digamos (directorios, clases, ficheros de imagen o lo que queramos) en un único fichero de extensión .jar. Un fichero de extensión .jar es similar a los .zip de Winzip, a los .rar de Winrar o a los ficheros .tar del tar de unix. De hecho, con Winzip o Winrar se puede ver y desempaquetar el contenido de un fichero .jar

Java nos da además otra opción, podemos ejecutar las clases del fichero .jar sin tener que desempaquetarlo. Estará listo para ejecutar.

Vamos a ver cómo hacer todo esto:

* **Cómo crear un fichero jar.**
* **Ver qué hay dentro de un jar.**
* **Modificar el contenido de un fichero jar.**
* **Ejecutar un fichero jar.**
* **El fichero de manifiesto.**

**CÓMO CREAR UN FICHERO JAR**

Para crear un fichero jar, en primer lugar tenemos que tener todo ya perfectamente preparado y compilado, funcionando.

Si las clases de nuestro programa **no pertenecen a paquetes**, simplemente debemos meter las clases en el fichero .jar Para ello, vamos al directorio donde estén los ficheros .class y ejecutamos el siguiente comando:

|  |
| --- |
| $ cd directorio\_con\_los\_class |
| $ jar cf fichero.jar fichero1.class fichero2.class fichero3.class … |

La opción "c" indica que queremos crear un fichero.jar nuevo. Si ya existía, se machacará, así que hay que tener cuidado. La opción "f" sirve para indicar el nombre del fichero, que va inmediatamente detrás. En nuestro caso, fichero.jar. Finalmente se pone una lista de ficheros .class (o de cualquier otro tipo) que queramos meter en nuestro jar. Se pueden usar comodines, estilo \*.class para meter todos los .class de ese directorio.

Si las clases de nuestro programa pertenecen a paquetes, debemos meter en nuestro jar la estructura de directorios equivalente a los paquetes entera. Para ello, nos vamos al directorio padre de donde empiece nuestra estructura de paquetes. En el caso de nuestro [HolaMundo con paquete](http://www.chuidiang.org/java/novatos/hola_mundo_paquetes.php), debemos meter el directorio **prueba** completo. El comando a ejecutar es este:

|  |
| --- |
| $ cd directorio\_padre\_de\_prueba $ jar -cf fichero.jar prueba |
|  |

Las opciones son las mismas, pero al final en vez de las clases, hemos puesto el nombre del directorio. Esto meterá dentro del jar el directorio y todo lo que hay debajo.

Otra opción sería meter los .class, pero indicando el camino relativo para llegar a ellos:

|  |
| --- |
| $ cd directorio\_padre\_de\_prueba $ jar -cf fichero.jar prueba/HolaMundo.class |
|  |

En windows la barra va al revés...

**VER QUÉ HAY DENTRO DE UN JAR**

Para comprobar si nuestro jar está bien hecho, podemos ver su contenido. El comando es este

|  |
| --- |
| $ jar tf fichero.jar |

La opción "t" indica que queremos un listado del fichero.jar. La opción "f" es igual que antes. Esto nos dará un listado de los class (y demás ficheros) que hay dentro, indicando en que directorio están. Deberíamos comprobar en ese listado que están todas las clases que necesitamos y la estructura de directorios concuerda con la de paquetes.

**MODIFICAR UN JAR**

Para cambiar un fichero dentro de un jar o añadirle uno nuevo, la opción del comando jar es "u". Si el fichero existe dentro del jar, lo reemplaza. Si no existe, lo añade.

Por ejemplo, si hacemos un cambio en nuestro HolaMundo.class con paquete y lo recompilamos, podemos reemplazarlo así en el jar

|  |
| --- |
| $ jar uf fichero.jar prueba/HolaMundo.class |

**CÓMO EJECUTAR UN JAR**

Para ejecutar un jar, simplemente debemos poner el fichero jar en el CLASSPATH. Ojo, hay que poner el fichero.jar, **NO** el directorio en el que está el fichero.jar. Este suele ser un error habitual al empezar, pensar que basta con poner el directorio donde está el jar. Para los .class, basta poner el directorio, para los .jar hay que poner el fichero.jar

El path para indicar la ubicación del fichero puede ser absoluto o relativo al directorio en el que ejecutemos el comando java. El comando para ejecutar una clase dentro de un jar, en nuestro caso del HolaMundo con paquete, suponiendo que estamos en el directorio en el que está el fichero.jar, sería este:

|  |
| --- |
| $ java -cp ./fichero.jar prueba.HolaMundo |

Hola Mundo

Simplemente, en la opción -cp del CLASSPATH hemos puesto el fichero.jar con su PATH relativo. Detrás hemos puesto el nombre de la clase, completo, con su paquete delante. Nuevamente, en Windows la barra de directorio va al revés.

**EL FICHERO DE MANIFIESTO**

Ejecutar así tiene una pega. Además de acordarse de poner la opción -cp, hay que saber el nombre de la clase que contiene el método main(). Además, si nuestro programa es muy grande, tendremos varios jar, tanto nuestros como otros que nos bajemos de internet o de donde sea. La opción -cp también puede ser pesada de poner en ocasiones.

Una opción rápida que a todos se nos ocurre es crearse un pequeño fichero de script/comandos en el que se ponga esta orden. Puede ser un fichero .bat de Windows o un script de Unix. Este fichero debe acompañar al fichero.jar y suponiendo que estén en el mismo directorio, su contenido puede ser este

java -cp ./fichero.jar prueba.HolaMundo

Para ejecutarlo, se ejecuta como un fichero normal de comandos/script. Si el fichero se llama ejecuta.sh o ejecuta.bat, según sea Unix o Windows:

$ ./ejecuta.sh

Hola Mundo

C:\> ejecuta.bat

Hola Mundo

Sin embargo, java nos ofrece otra posibilidad de forma que no tengamos que hacer este fichero. Simplemente, en un fichero de texto metemos una línea en la que se ponga cuál es la clase principal. Este fichero se conoce como fichero de manifiesto y su contenido puede ser este:

Main-Class: prueba.HolaMundo

Cuando construimos el jar, debemos incluir este fichero de una forma especial. Por ejemplo, si el fichero lo llamamos manifiesto.txt y lo ponemos en el directorio donde vamos a construir el jar, el comando para hacerlo sería este:

$ jar cmf manifiesto.txt fichero.jar prueba/HolaMundo.class

En Windows, nuevamente, la barra al revés. Al comando de crear jar le hemos añadido la opción "m" para indicar que vamos a añadir un fichero de manifiesto. Hemos añadido además el fichero manifiesto.txt. El orden de las opciones "mf" es importante. El fichero de manifiesto y el fichero.jar se esperan en el mismo orden que pongamos las opciones. En el ejemplo, como hemos puesto primero la opción "m", debemos poner manifiesto.txt delante de fichero.jar. El resto de ficheros son los que queremos empaquetar.

Una vez construido, se ejecuta fácilmente. Basta con poner:

$ java -jar fichero.jar

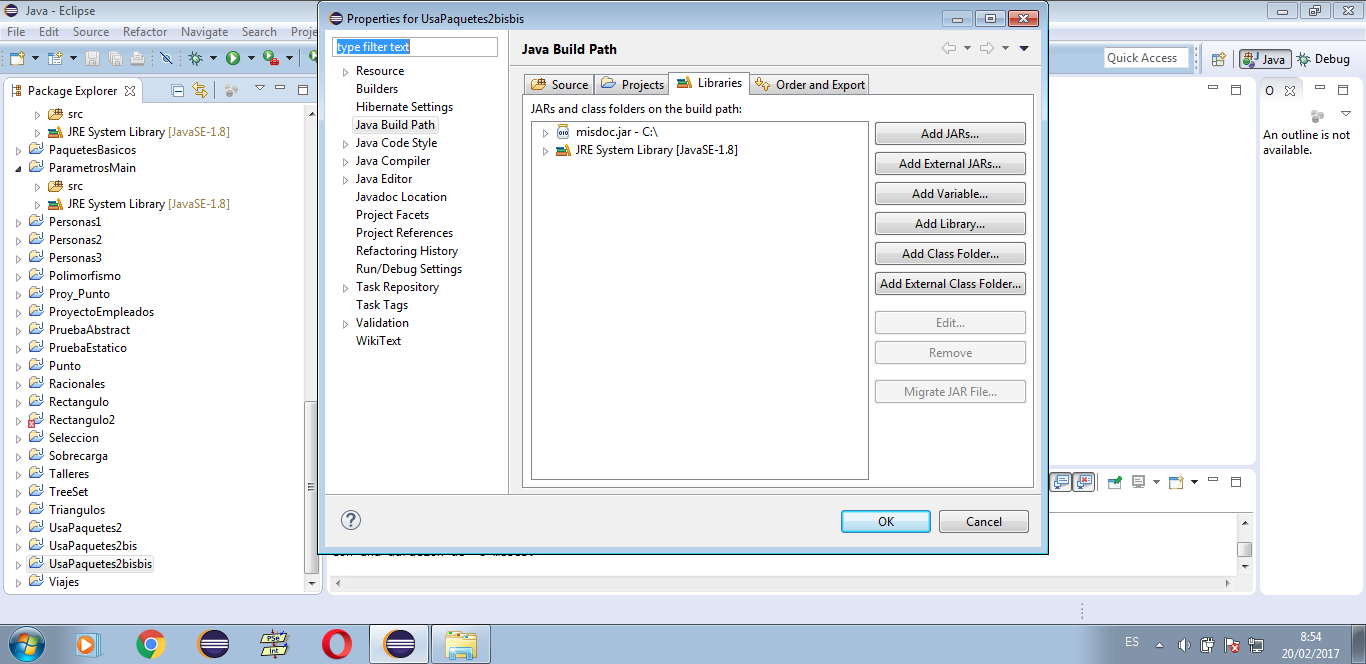
La opción "-jar" indica que se va a ejecutar el fichero.jar que se ponga a continuación haciendo caso de su fichero de manifiesto. Como este fichero de manifiesto dice que la clase principal es prueba.HolaMundo, será esta la que se ejecute.

De esta forma nos basta con entregar el jar y listo. El comando para arrancarlo es sencillo.

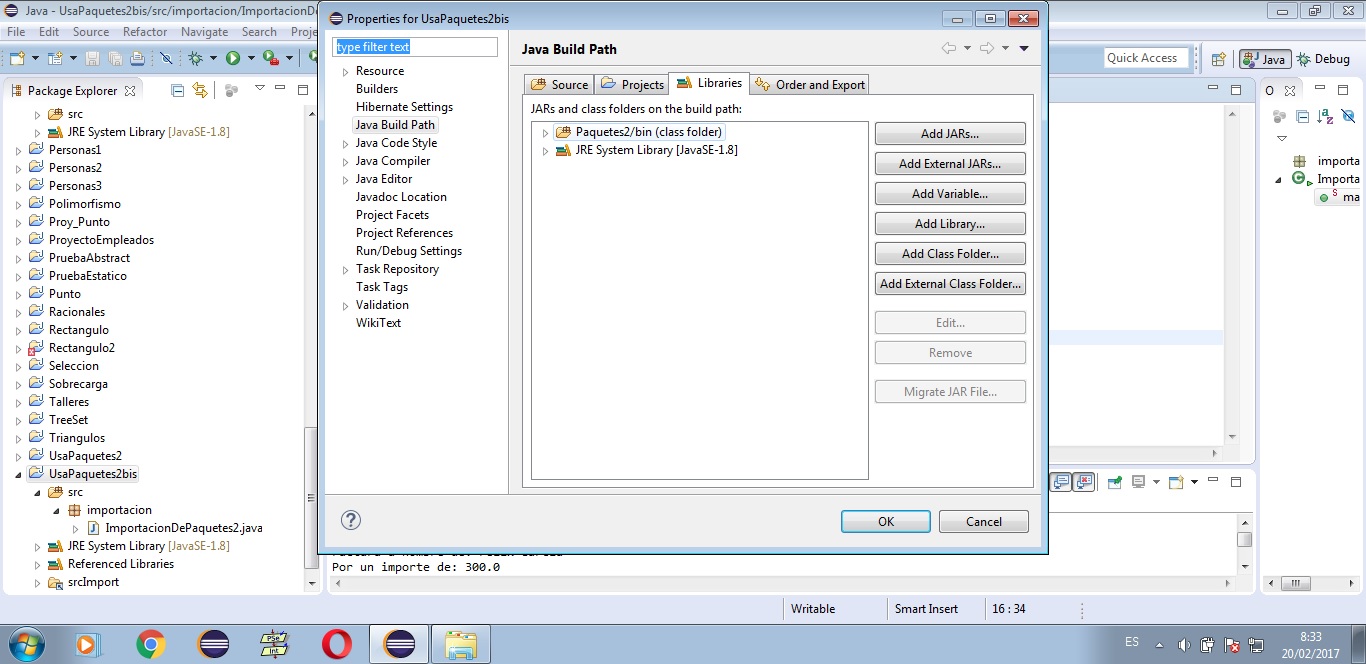
Es más, en Windows, si lo configuramos para que los ficheros jar se abran con java y la opción -jar, bastará con hacer doble click sobre ellos para que se ejecuten.

# UT6 Uso de Paquetes desde Fuera

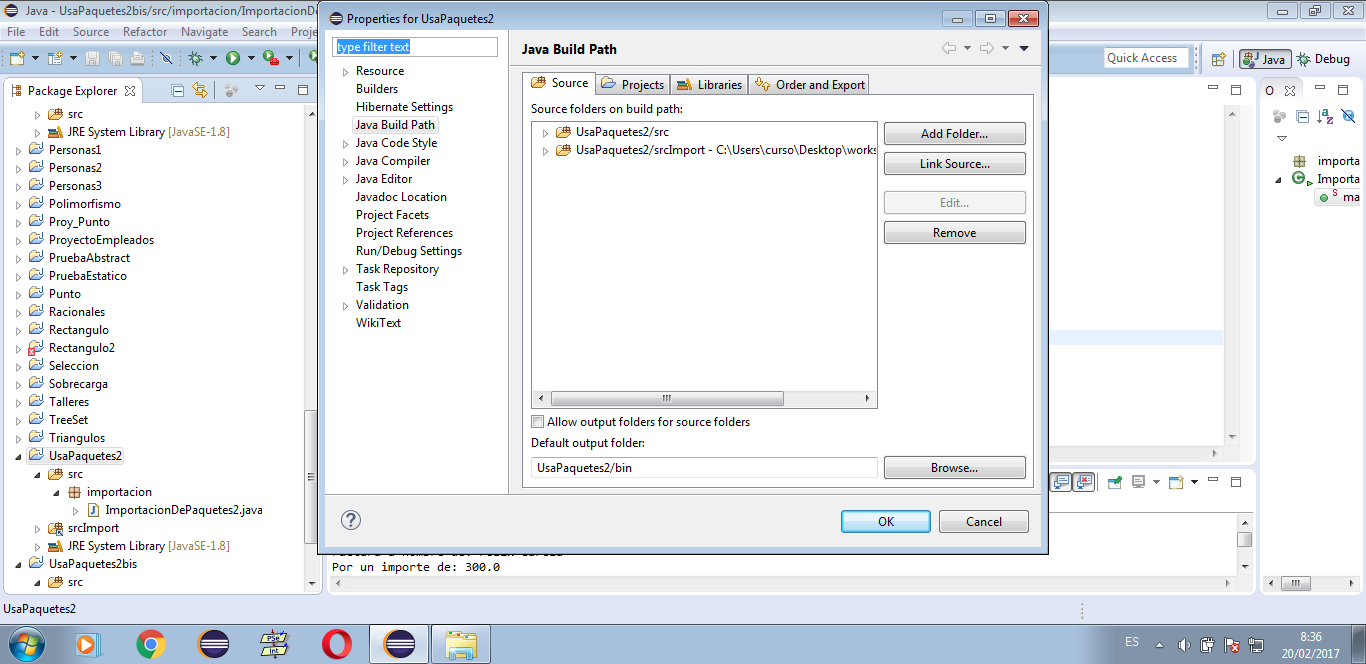
Con Jar Externo:



Incluir Class Folder:



Incluir Fuente de Paquetes:



# UT6 Ejercicio para practicar paquetes

**EJERCICIO DE APLICACIÓN DE PAQUETES**

Crea una aplicación en la que se pidan 2 números, estos se sumarán, restarán, multiplicarán y dividirán, mostrándonos el resultado por consola.  
  
Para ello crea un proyecto llamado **"Operaciones"**, en él crea 2 paquetes, uno llamado **"operaciones"** y el otro **"resultados"**:

En el paquete **"operaciones"** escribe las clases:

* + **Sumar**
  + **Restar**
  + **Multiplicar**
  + **Dividir**

Cada una de ellas tendrá lo siguiente:

* Una sentencia **"package operaciones"** para que sea guardado en dicho **"package"**.
* 2 atributos, uno llamado **"x"** y otro llamado **"y"** de tipo float.
* Un constructor con 2 parámetros para dar valor a los atributos.
* Un método (suma, resta, multiplica y divide, según proceda) en el cual se realiza la operación y muestra por pantalla el resultado.

Posteriormente tendremos un segundo  **paquete**  llamado **"resultados"** el cual tendrá las siguientes clases:

* + **IntroducirNumeros**
  + **Calcular**
* La clase **"IntroducirNumeros",** realiza las siguientes funciones:
  + La agregamos al package **"resultados”.**
  + En su **constructor** crea un objeto **"Scanner"** llamado teclado, y pide 2 números por consola que almacenamos en las variables **"i"** y **"j"** (que serán los atributos de esta clase).
* La clase **"Calcular"** centrará todas las funciones que se realizan en el resto de clases (tendrá el método main()):
  + La agregamos al package **"resultados"**.
  + Posteriormente importamos el package **"operaciones"** para poder utilizar las clases que hay en su interior.
  + Instanciamos en el **main** la clase **"IntroducirNumeros"**, para poder usar sus atributos.
  + Finalmente creamos una instancia para cada clase del package **"operaciones"** y realizamos las operaciones matemáticas. Le pasamos a los constructores de cada una de ellas los valores que hemos introducido por teclado, a través de la clase "**introducirNumeros**” (serán los operadores de nuestras operaciones).
  + *No hacer la declaración de los atributos ni pública ni privada, si no se pone ningún modificador de acceso, la visibilidad será a nivel del paquete*

Tened en cuenta:

* Si no ponemos ningún modificador de acceso la visibilidad será a **nivel de paquete (hazlo así, sin getter ni setter)**, esto quiere decir que aunque importemos un package en otras clases, si alguna de estas clases o alguno de sus atributos o métodos tiene visibilidad de paquete, no podrán ser usados/ejecutados en otros paquetes, **SOLO** en las clases que estén en su mismo paquete.
* Para utilizar la clase Scanner del paquete **java.util** no lo importes, usa el nombre del paquete directamente.

Java.util.Scanner teclado = new Java.util.Scanner (System.in)

Haz como **segunda parte** del ejercicio la implementación de los paquetes en dos proyectos diferentes (llámalos Resultados y Operaciones), accediendo mediante acceso a fichero ,jar a las clases de los paquetes que necesites.

Sobre el paquete operaciones -> click dcho -> Export Java: JAR file

He guardado el op.jar en esta ruta:

G:\DOCUMENTS\DA1D1E\Programación\Eclipse-workspace2\Operaciones\src\resultados

Creas una copia del proyecto

Borras el paquete operaciones de Operaciones2

En Operaciones2:

Project properties -> Java Build Path -> Libraries -> Add external JARs…

G:\DOCUMENTS\DA1D1E\Programación\Eclipse-workspace2\Operaciones\src\resultados