

# UT6

## Paquetes y ficheros jar

Módulo - Programación (1º)

Ciclos - Desarrollo de Aplicaciones Multiplataforma | Desarrollo de Aplicaciones Web

CI María Ana Sanz

---

# API de Java

- biblioteca de clases standard de Java
  - incluye cientos de clases organizadas en multitud de paquetes

Paquete	Descripción
<b>java.lang</b>	clases centrales de la plataforma (cadenas, números,...). No es necesario incluir la sentencia import cuando se utilizan clases de este paquete
<b>java.util</b>	utilidades varias: generadores de n°s aleatorios, colecciones, fechas (antes de Java 8), ...
<b>java.io</b>	clases para realizar operaciones de E/S (entrada / salida, uso de ficheros)
<b>java.time</b>	clases para fechas y horas (nuevo en Java 8)
<b>java.sql</b>	clases para acceso a Bases Datos
<b>java.net</b>	clases que permiten implementar aplicaciones distribuidas

# ¿Qué es un paquete?

- Conjunto de clases lógicamente relacionadas
  - colección de ficheros fuente agrupados por funcionalidad
- Todas las clases pertenecen a un paquete
  - Si no se especifica ninguno explícitamente las clases pertenecen al paquete por defecto (**default package**)
  - Cada una de las clases que nosotros hemos creado hasta ahora pertenecen al paquete por defecto.

# Razones para utilizar paquetes

- **localizar fácilmente las clases**
  - las clases con funciones similares se sitúan en el mismo paquete
- **evitar conflictos de nombres de nuestras clases con las de otros programadores**
  - dos clases con el mismo nombre pero en distinto paquete
    - Date – java.sql / Date – java.util

# Razones para utilizar paquetes

- **distribuir el software más fácilmente**
  - habitualmente en ficheros jar
- **proteger las clases**
  - los paquetes proporcionan protección al código
    - por defecto, si no se especifica visibilidad, los miembros de una clase son accesibles únicamente dentro del paquete al que pertenecen, se dice que tienen visibilidad de paquete, **package**

# Sentencia import

```
public class Estudiante
{
    ..... •
    private java.time.LocalDate fechaNacimiento;
    private java.util.ArrayList<Curso> cursos;
    ..... • •
}
```

- El nombre de una clase debe ir precedido por el paquete (ruta de paquetes) al que pertenece
  - **nombre completamente calificado de la clase.**

# Sentencia import

```
import java.time.LocalDate;
import java.util.ArrayList;

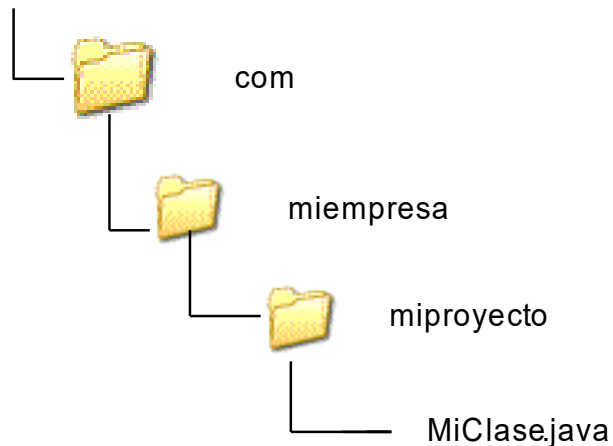
public class Estudiante
{
    ..... •
    private LocalDate fechaNacimiento;
    private ArrayList<Curso> cursos;
    ..... • •

}
```

- Se utiliza la declaración **import** para poder omitir la ruta de paquetes al nombrar una clase

# Requisitos para crear paquetes

- el paquete ha de contener una o más clases / interfaces
  - no vacío
- las clases han de tener sus ficheros fuente en la misma estructura de directorios que el nombre del paquete
  - mapeo del nombre del paquete con directorio



```
package com.miempresa.miproyecto;
import java.util.ArrayList;
public class MiClase
{
    .....
}
```



# Nombrando paquetes

- los nombres de paquetes han de ser únicos
- en grandes organizaciones
  - se toma el nombre de dominio y se invierte
  - *ej* – los paquetes del proyecto <http://ant.apache.org> se denominan `org.apache.ant`
  - `com.micompañia.miproyecto`
  - `informatica.daw.programacion.gestorstock`
  - `pkgconjunto.modelo`
- Por convención, nombres de paquetes en minúsculas



`informatica.daw.programacion.gestorstock`

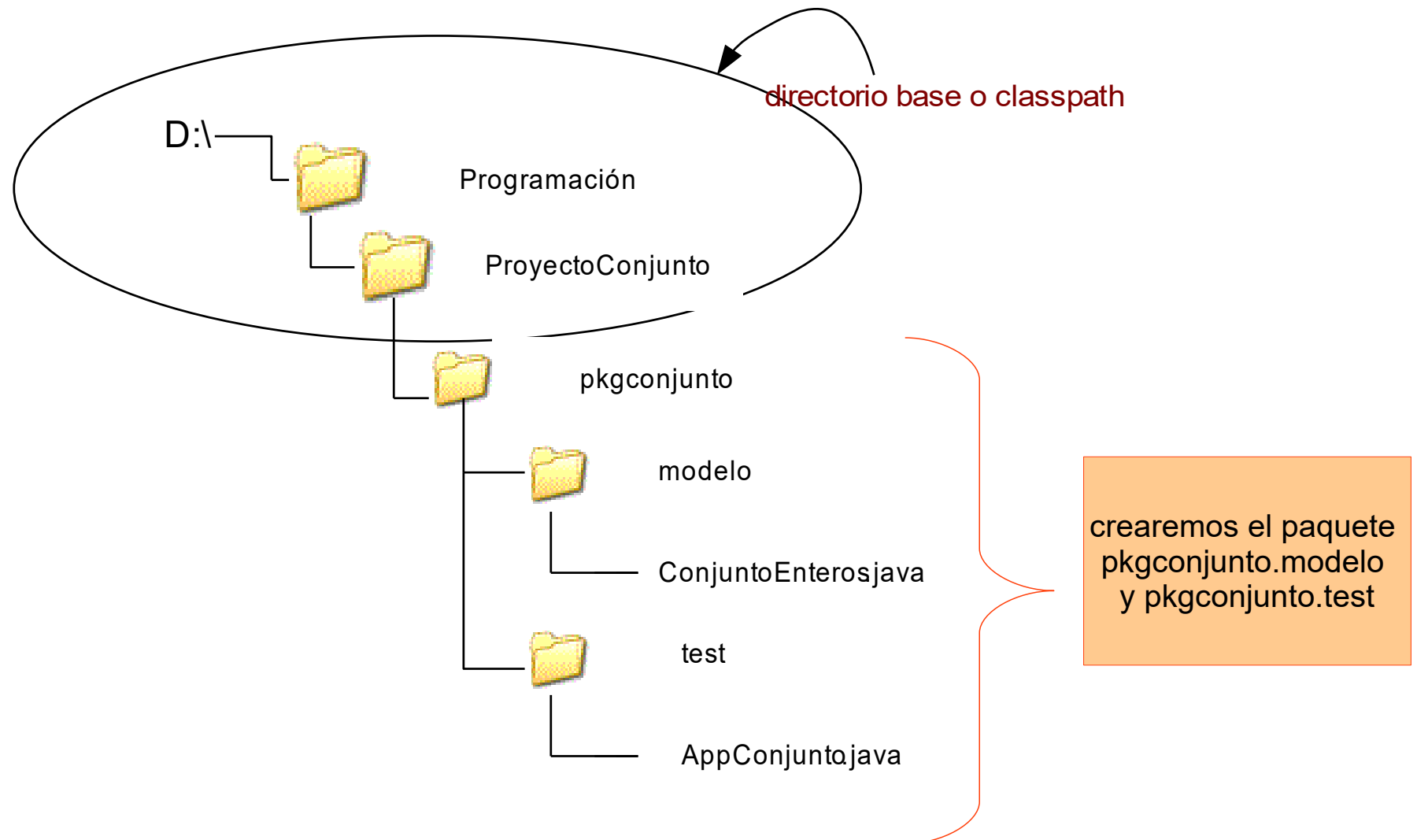
Notación UML para paquetes

# Declarar miembros del paquete

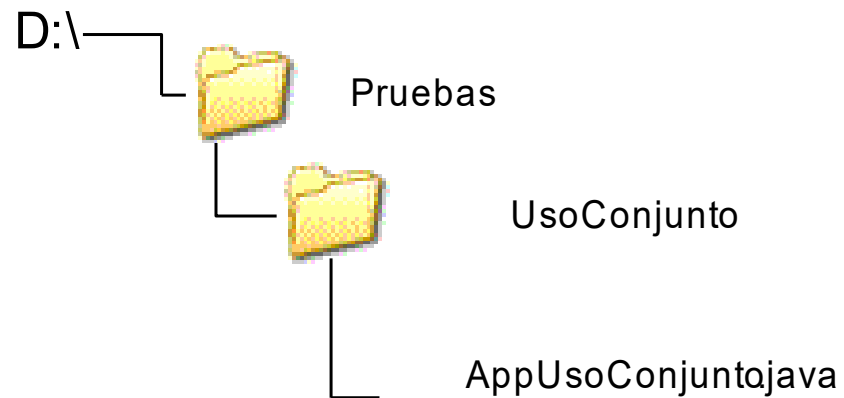
- decidir qué clase pertenecerá al paquete
- a través de la sentencia **package**
  - se indica que la clase pertenecerá al paquete
  - se incluye al principio del código fuente

```
package informatica.daw.programacion.gestorstock;  
package pkgconjunto.modelo;  
package pkgconjunto.test;
```

# Ejemplo ProyectoConjunto - Qué queremos hacer?



# Ejemplo ProyectoConjunto - Qué queremos hacer?



utilizaremos desde otro  
proyecto una clase de  
ProyectoConjunto

# Paquetes y línea de comandos

- Proyecto ConjuntoEnteros
  - editamos con NotePad++ la clase ConjuntoEnteros
  - incluimos como primera sentencia
    - `package pkgconjunto.modelo;`
  - editamos con NotePad++ la clase AppConjunto
  - incluimos como primera sentencia
    - `package pkgconjunto.test;`

# Paquetes y línea de comandos

- Proyecto ConjuntoEnteros
  - creamos desde línea de comandos la estructura de directorios adecuada tomando como **directorio activo**  
D:\Programación\ProyectoConjunto
    - D:\Programación\ProyectoConjunto>mkdir pkgconjunto
    - D:\Programación\ProyectoConjunto>mkdir pkgconjunto\modelo
    - D:\Programación\ProyectoConjunto>mkdir pkgconjunto\test
  - movemos los ficheros fuente a sus directorios correspondientes
    - D:\Programación\ProyectoConjunto>move ConjuntoEnteros.java  
pkgconjunto\modelo
    - D:\Programación\ProyectoConjunto>move AppConjunto.java  
pkgconjunto\test

# Paquetes y línea de comandos

## ■ Proyecto ConjuntoEnteros

### ■ compilamos las clases

– nos situamos en el directorio D:\Programación\ProyectoConjunto

– .....>javac pkgconjunto\modelo\ConjuntoEnteros.java

– .....>javac pkgconjunto\test\AppConjunto.java

– error al compilar, no encuentra la clase ConjuntoEnteros, hay que importarla

AppConjuntos tiene una variable de tipo ConjuntoEnteros.  
por eso da error al compilar

– editamos AppConjunto.java e incluimos

import pkgconjunto.modelo.ConjuntoEnteros;

debemos importar despues de la sentencia package

– .....>javac pkgconjunto\test\AppConjunto.java

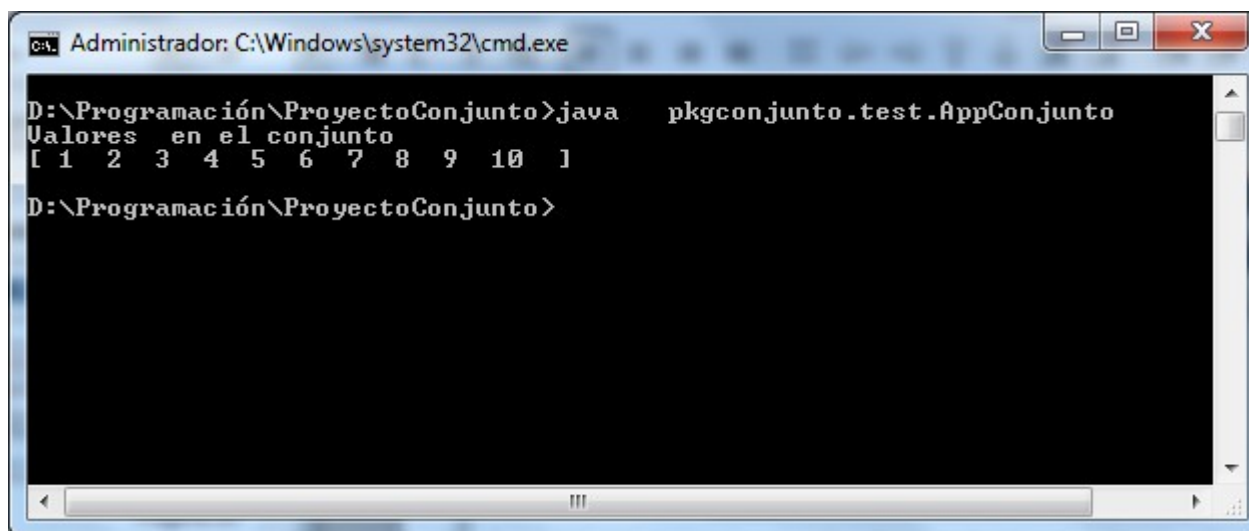
# Paquetes y línea de comandos

- Proyecto ConjuntoEnteros

- ejecutamos

- .....>java pkgconjunto.test.AppConjunto

o: java pkgconjunto\test\AppConjunto.java



```
Administrador: C:\Windows\system32\cmd.exe

D:\Programación\ProyectoConjunto>java pkgconjunto.test.AppConjunto
Valores en el conjunto
[ 1 2 3 4 5 6 7 8 9 10 ]

D:\Programación\ProyectoConjunto>
```



# Paquetes y línea de comandos

- Proyecto UsoConjunto
  - copiamos el proyecto en D:\Pruebas
  - editamos UsoConjunto.java y añadimos
    - `import pkgconjunto.modelo.ConjuntoEnteros;`
  - cambiamos el directorio activo a D:\Pruebas\UsoConjunto>
  - compilamos
    - D:\Pruebas\UsoConjunto>`javac AppUsoConjunto.java`
    - error en compilación - no reconoce ConjuntoEnteros
  - Java busca el paquete `pkgconjunto.modelo` en el **directorio activo** y no está
  - hay que especificar el **classpath** para que localice las clases dentro del paquete

creo que seria la carpeta  
UsoConjunto

# Qué es el classpath?

- **variable de entorno** del sistema
  - define las rutas en las que el compilador y el intérprete java buscan los paquetes, las clases compiladas .class y los .jar (como el path para los ficheros .bat y .exe).
  - El directorio actual (.) normalmente está incluido en el classpath. Ahí es donde la JVM busca los paquetes por defecto y las clases compiladas.
  - Si se importan paquetes que no están en el . hay que indicar al compilador y a la JVM dónde buscar.

# Cómo se establece el classpath?

- desde línea de comandos a través del comando **set** antes de ejecutar el comando para compilar `javac`  
direct raiz: en el que estan los paquetes
  - .....>**set classpath=.;d:\Programación\ProyectoConjunto**  
ruta sin espacios ni comillas
  - D:\Pruebas\UsoConjunto>**javac AppUsoConjunto.java**
  - D:\Pruebas\UsoConjunto>**java AppUsoConjunto**
- El classpath así establecido solo vale para esa sesión de línea de comandos

# Cómo se establece el classpath?

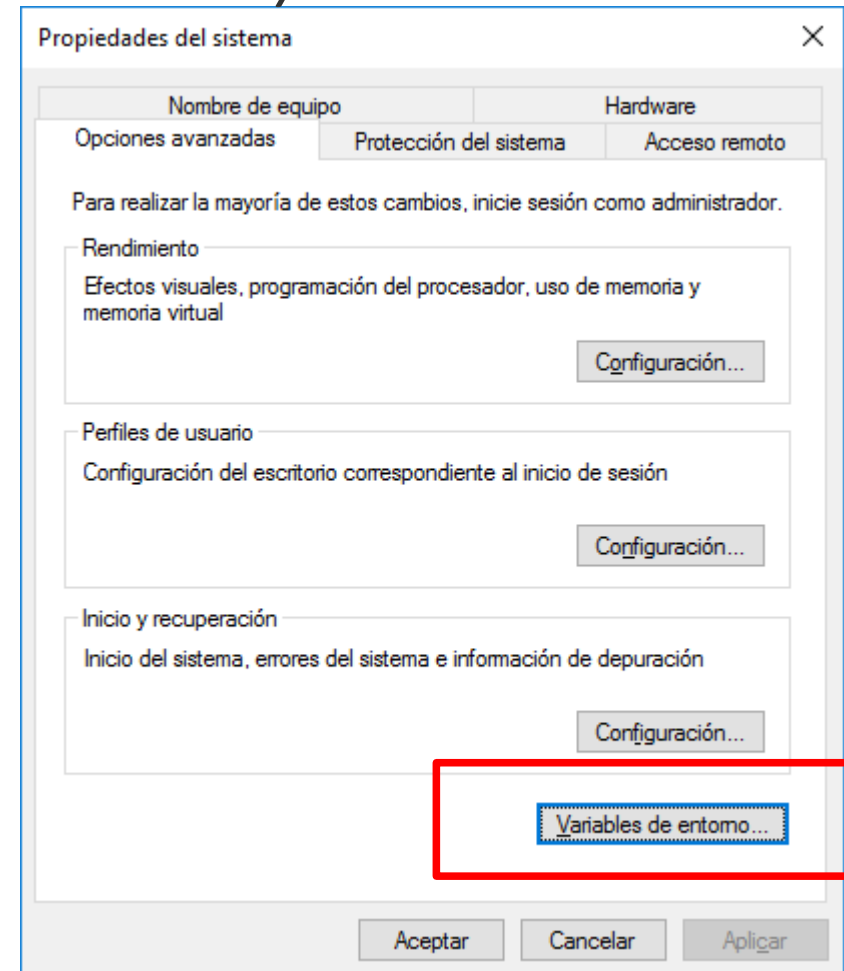
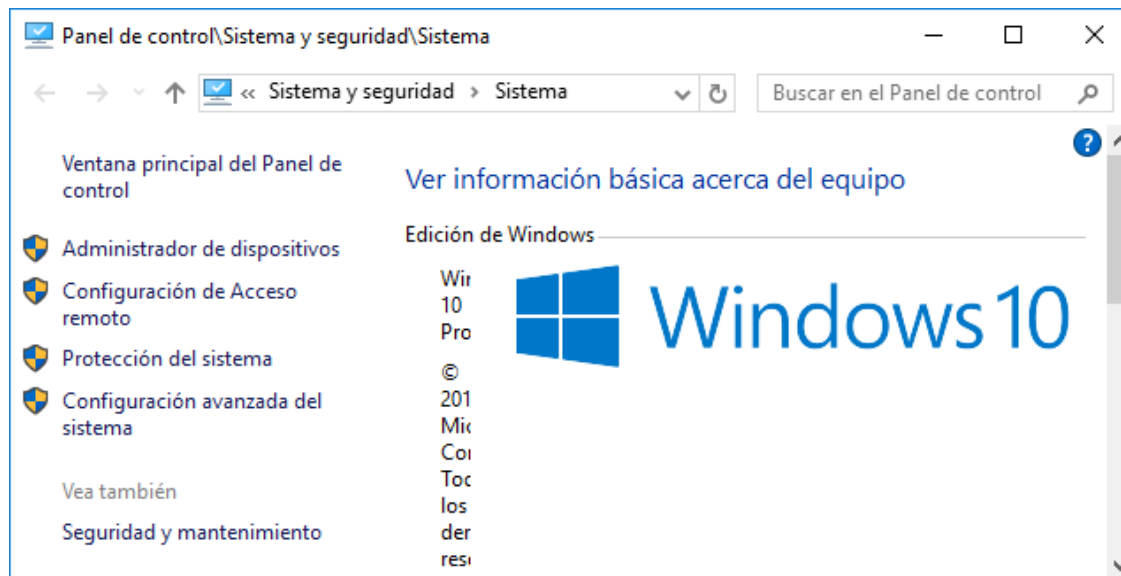
- desde línea de comandos cuando se ejecuta el comando para compilar **y ejecutar**

- D:\Pruebas\UsoConjunto>javac -cp .;d:\Programación\ProyectoConjunto AppUsoConjunto.java
- D:\Pruebas\UsoConjunto>java -cp .;d:\Programación\ProyectoConjunto AppUsoConjunto

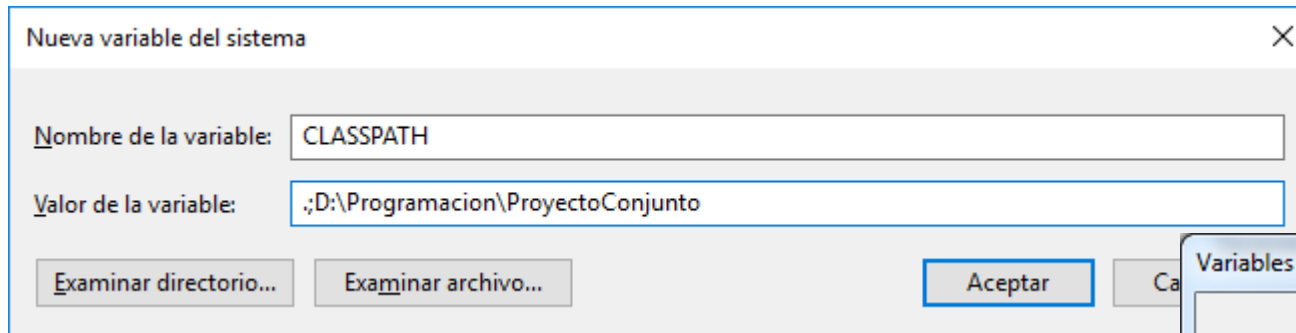
el cambio de classpath se da solo al escribir el comando, por ello se debe especificar siempre al usar esos parametros

# Cómo se establece el classpath?

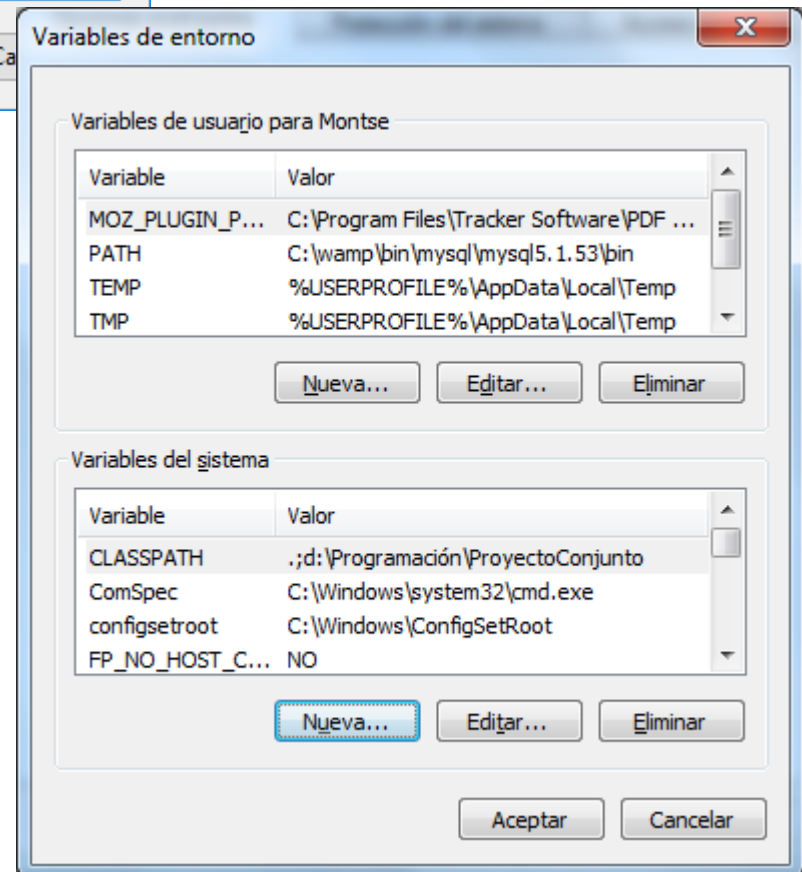
- Windows 10 - desde *Panel de control / Sistema y Seguridad / Sistema / Configuración avanzada del sistema / Variables de entorno*



# Cómo se establece el classpath?



- compilamos
  - D:\Pruebas\UsoConjunto>javac  
AppUsoConjunto.java
- ejecutamos
- D:\Pruebas\UsoConjunto>java  
AppUsoConjunto



# Resumen Proyecto Conjunto

```
package pkgconjunto.modelo;

/**
 * clase ConjuntoEnteros
 */
import java.util.HashSet;
import java.util.Iterator;

public class ConjuntoEnteros
{
    ...
}
```

```
package pkgconjunto.test;
import pkgconjunto.modelo.ConjuntoEnteros;

/**
 * Clase AppConjunto - incluye el método main()
 * Arranca la aplicación
 */
public class AppConjunto
{
    ...
    /**
     *
     */
}
```

```
import pkgconjunto.modelo.ConjuntoEnteros;

/**
 * Clase AppUsoConjunto - incluye el método main()
 * Arranca la aplicación
 */

public class AppUsoConjunto
{
    ...
    /**
     *
     */
}
```

# Ficheros jar

- Fichero **jar** – **Java Archive File**
  - fichero java de archivos empaquetados
  - extensión .jar
  - mecanismo para empaquetar software y distribuirlo fácilmente
  - no es obligatoria la compresión pero si se usa es como en los archivos .zip
  - las clases de la API están almacenadas así
- Ventajas del uso de ficheros .jar
  - seguridad – pueden ser firmados digitalmente
  - decrementan el tiempo de descarga
  - portabilidad (todos los JRE saben cómo manejar los .jar)

<https://docs.oracle.com/javase/tutorial/deployment/jar/index.html>



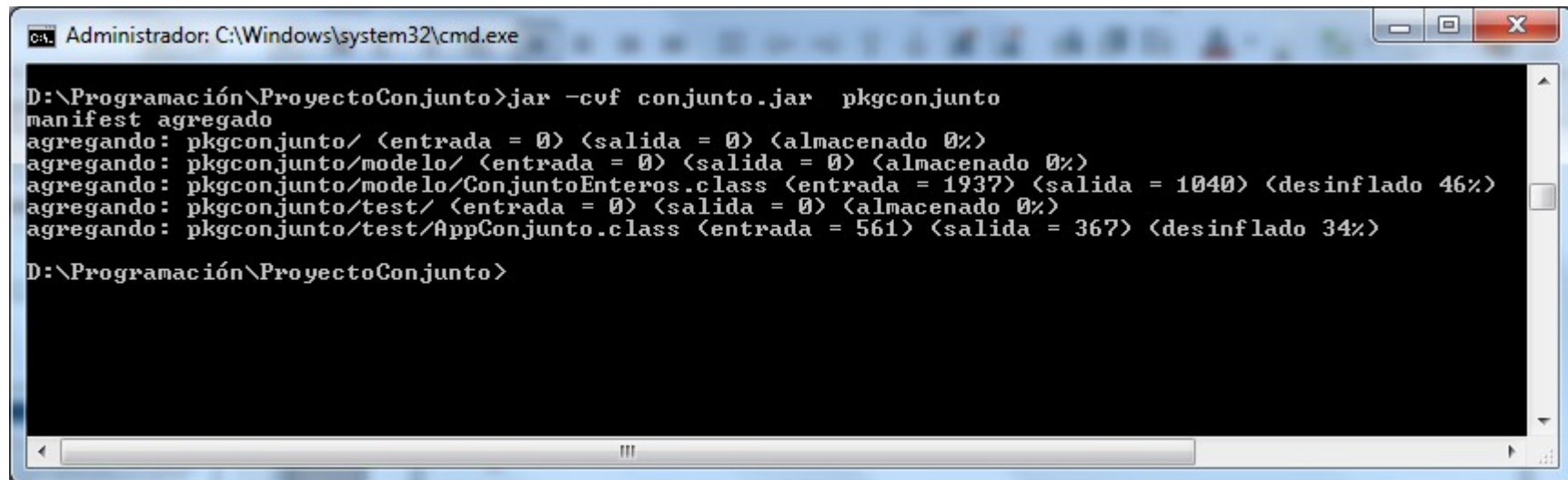
# Cómo crear un fichero .jar – Línea de comandos - jar -cvf

- Herramienta **jar**

- d:\Programación\ProyectoConjunto>jar -cvf conjunto.jar  
pkgconjunto

**c** – crear  
**v** – verbose – información en pantalla  
**f** – salida a un fichero  
**m** – se especifica fichero manifiesto

si las clases **compiladas**  
están en este paquete



```
Administrador: C:\Windows\system32\cmd.exe

D:\Programación\ProyectoConjunto>jar -cvf conjunto.jar pkgconjunto
manifest agregado
agregando: pkgconjunto/ <entrada = 0> <salida = 0> <almacenado 0%>
agregando: pkgconjunto/modelo/ <entrada = 0> <salida = 0> <almacenado 0%>
agregando: pkgconjunto/modelo/ConjuntoEnteros.class <entrada = 1937> <salida = 1040> <desinflado 46%>
agregando: pkgconjunto/test/ <entrada = 0> <salida = 0> <almacenado 0%>
agregando: pkgconjunto/test/AppConjunto.class <entrada = 561> <salida = 367> <desinflado 34%>

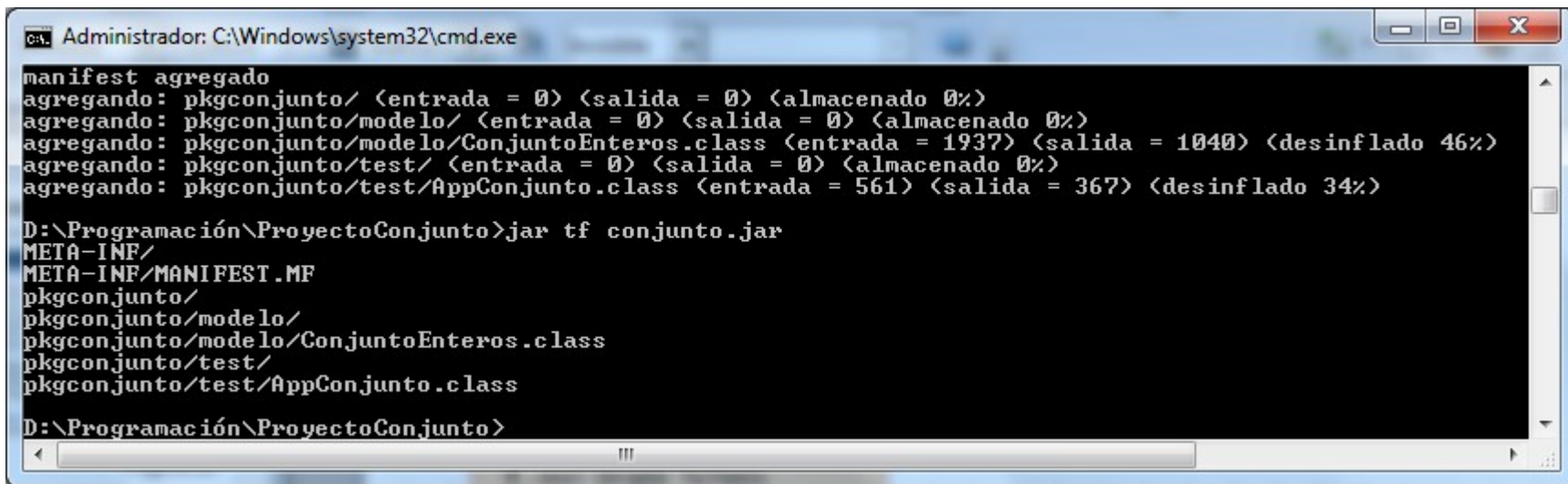
D:\Programación\ProyectoConjunto>
```

# Cómo ver el contenido de un fichero .jar

## Línea de comandos - jar tf

- Herramienta **jar**
  - d:\Programación\ProyectoConjunto>jar tf conjunto.jar

**t** – print tabla de contenidos  
**f** – leer desde fichero



```
C:\Windows\system32\cmd.exe
manifest agregado
agregando: pkgconjunto/ <entrada = 0> <salida = 0> <almacenado 0%>
agregando: pkgconjunto/modelo/ <entrada = 0> <salida = 0> <almacenado 0%>
agregando: pkgconjunto/modelo/ConjuntoEnteros.class <entrada = 1937> <salida = 1040> <desinflado 46%>
agregando: pkgconjunto/test/ <entrada = 0> <salida = 0> <almacenado 0%>
agregando: pkgconjunto/test/AppConjunto.class <entrada = 561> <salida = 367> <desinflado 34%>

D:\Programación\ProyectoConjunto>jar tf conjunto.jar
META-INF/
META-INF/MANIFEST.MF
pkgconjunto/
pkgconjunto/modelo/
pkgconjunto/modelo/ConjuntoEnteros.class
pkgconjunto/test/
pkgconjunto/test/AppConjunto.class

D:\Programación\ProyectoConjunto>
```

# Extraer ficheros de un .jar

## Línea de comandos – jar xf

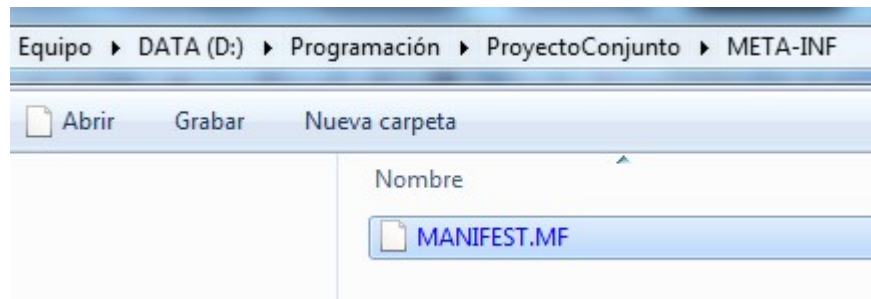
- Herramienta **jar**
  - d:\Programación\ProyectoConjunto>jar xf conjunto.jar

**x** –extraer ficheros  
**f** – leer desde fichero

- extrae el fichero manifiesto y la estructura de directorios del paquete

# El fichero manifiesto


- Contiene información sobre el *jar* y los ficheros que contiene




Manifest-Version: 1.0  
Created-By: 1.6.0\_25 (Sun Microsystems Inc.)

# Haciendo jar ejecutables

Otra manera de crear .jar

- Para hacer un fichero *jar* ejecutable
  - incluir en el fichero manifiesto la clase que contiene el método `main()`
  - se edita un fichero ***manifiesto.txt*** y se incluye  
*Main-Class: pkgconjunto.test.AppConjunto*  
(línea en blanco al final y espacio en blanco después de Main-Class:)
- `d:\Programación\ProyectoConjunto>jar cmf manifiesto.txt  
conjunto.jar pkgconjunto`
- `d:\Programación\ProyectoConjunto>java -jar conjunto.jar`  
 para ejecutar

# El flag -e

- flag **-e** Otra manera de crear .jar
  - introducido a partir de Java 6
  - permite especificar el punto de entrada a la aplicación sin editar o crear el fichero manifiesto
  - `d:\Programación\ProyectoConjunto>jar cvfe conjunto.jar  
pkgconjunto.test.AppConjunto pkgconjunto`
  - `d:\Programación\ProyectoConjunto>java -jar conjunto.jar`  
 para ejecutar

# El flag -e

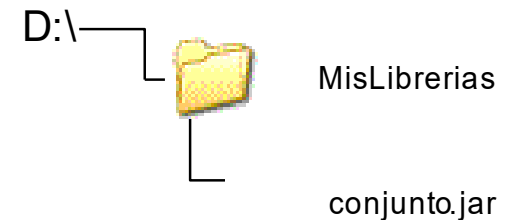
## ■ flag -e

- d:\Programación\ProyectoConjunto>jar cvfe conjunto.jar  
pkgconjunto.test.AppConjunto \*
- d:\Programación\ProyectoConjunto>java -jar conjunto.jar
- d:\Programación\ProyectoConjunto>jar cvfe conjunto.jar  
pkgconjunto.test.AppConjunto .
- d:\Programación\ProyectoConjunto>java -jar conjunto.jar
- d:\Programación\ProyectoConjunto>jar cvfe conjunto.jar  
pkgconjunto.test.AppConjunto  
pkgconjunto\modelo\\*.class  
pkgconjunto\test\\*.class
- d:\Programación\ProyectoConjunto>java -jar conjunto.jar

← incluyendo solo  
ficheros .class

# Ficheros jar como librerías

- Dejo en D:\MisLibrerias el fichero *conjunto.jar* anterior
- Ahí voy a dejar las clases compiladas y ficheros .jar que quiero poner a disposición de otros

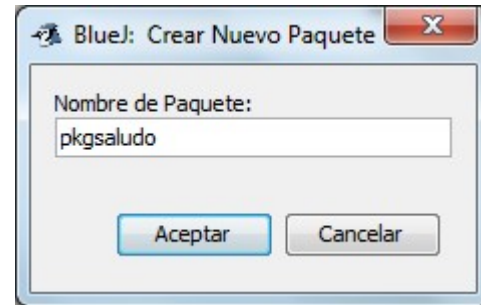


- Elimino todo rastro del classpath anterior para hacer las pruebas
- D:\Pruebas\UsoConjunto>javac -cp  
.;d:\MisLibrerias\conjunto.jar  
AppUsoConjunto.java
- D:\Pruebas\UsoConjunto>java -cp  
.;d:\MisLibrerias\conjunto.jar  
AppUsoConjunto



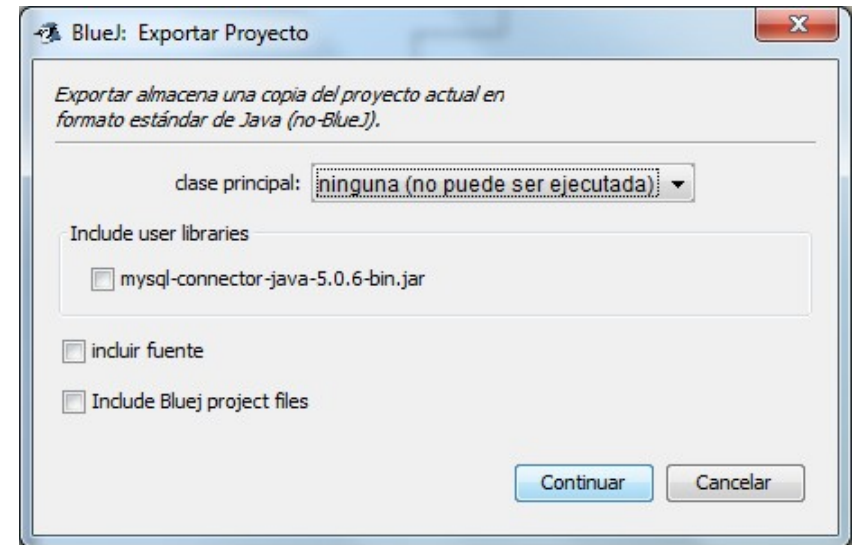
# Paquetes, ficheros jar y BlueJ

- Abrimos el proyecto Saludo
- *Edición / Nuevo paquete*
  - nombre del paquete pkgsaludo
- Añadimos las clases en el paquete incluyendo en cada una de ellas la sentencia
  - `package pkgsaludo;`



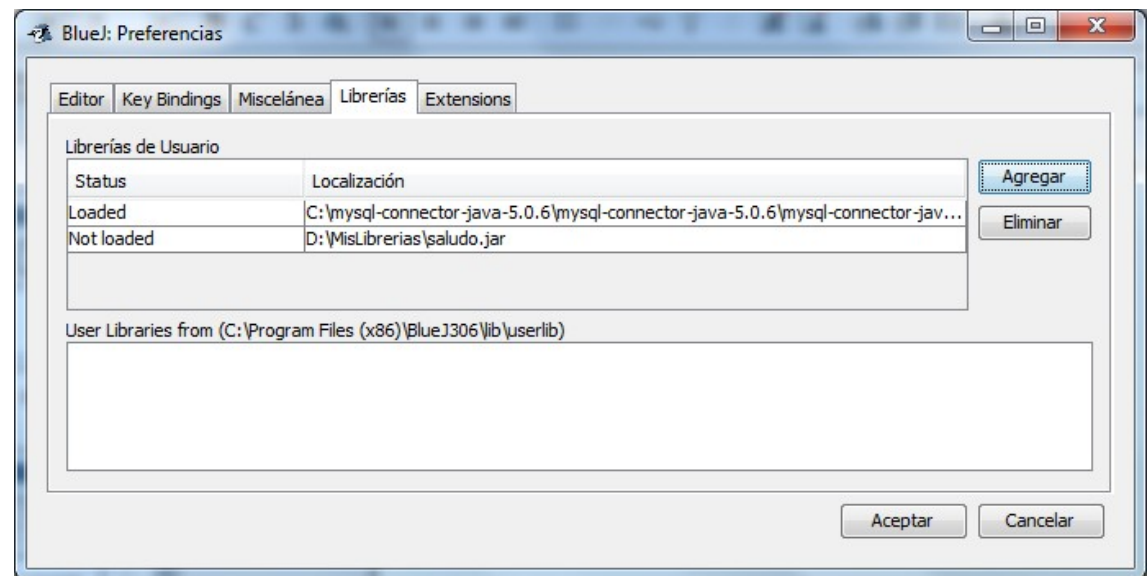
# Paquetes, ficheros jar y BlueJ

- Creamos el fichero *saludo.jar*
  - *Proyecto / Exportar*
- Lo guardamos como *saludo.jar*
- Lo copiamos en D:\MisLibrerias
  - así otros programas pueden hacer uso de las clases compiladas que contiene



# Paquetes, ficheros jar y BlueJ

- Abrimos el proyecto `UsoSaludo` que está en `D:\Pruebas`
- Incluimos en la clase `AppSaludo`
  - `import pkgsaludo.Saludo;`
- Compilamos y error
  - No encuentra la clase `Saludo`



- Pondremos desde BlueJ la librería (*saludo.jar*) a disposición del resto
- Cerramos BlueJ para que surta efecto el cambio y volvemos a abrir.

# Módulos en Java (a partir de versión 9)

- **Módulos**
  - Java 9, entre otras novedades, introdujo los módulos
  - Un **módulo** es
    - una nueva forma de agrupar código, datos y recursos
    - añaden un nivel superior de agrupación por encima de los paquetes
  - Cada módulo describe
    - qué paquetes exhibe y
    - de qué paquetes de otros módulos depende la aplicación
- No los veremos

# Módulos en Java (a partir de versión 9)

---

- <https://www.arquitecturajava.com/java-9-modules/>
- <https://www.oracle.com/corporate/features/understanding-java-9-modules.html>