



Modulo 14

Línea de Comandos

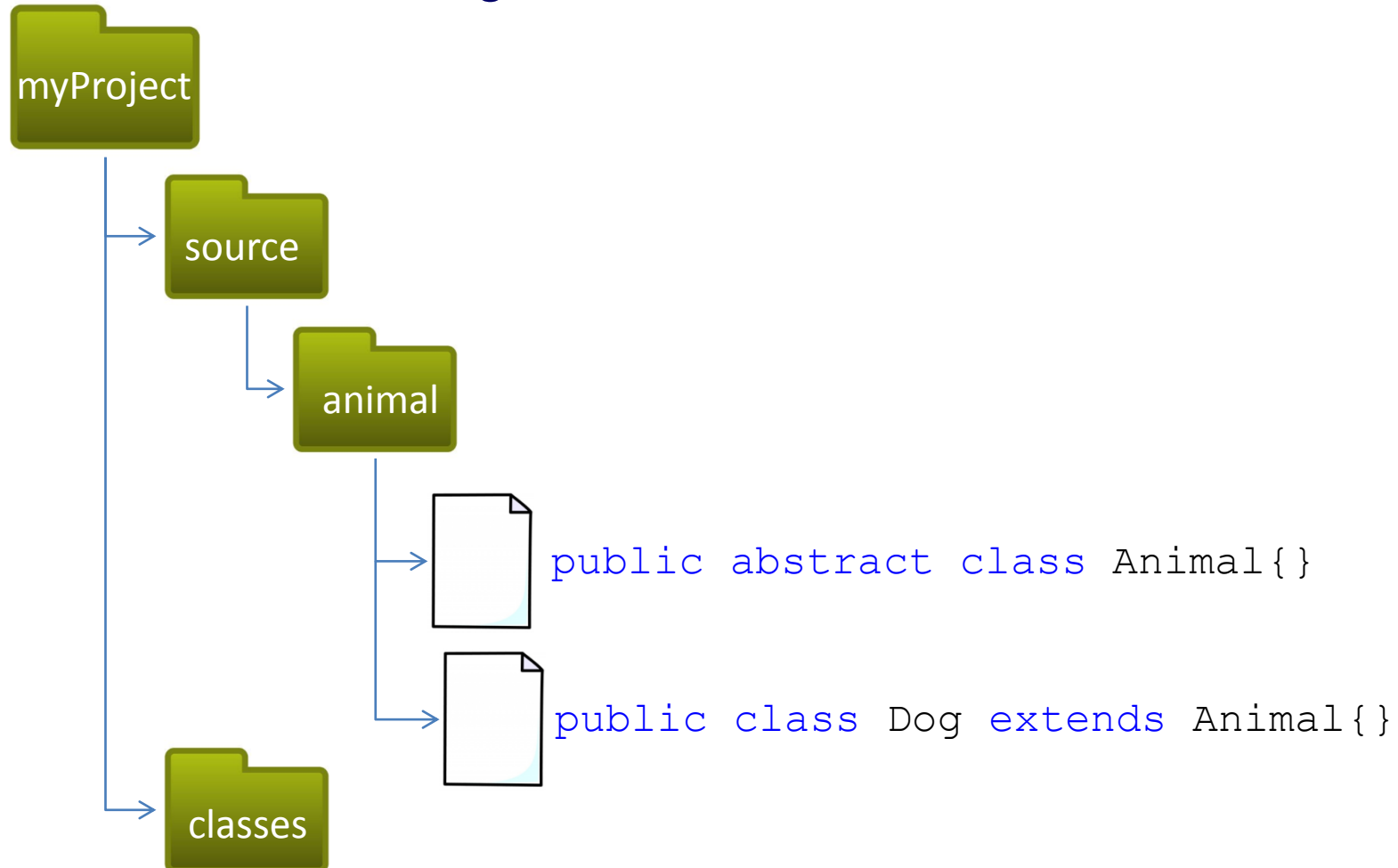
Compilando con javac, opción -d:



- Por default el compilador pone los archivos `.class` en el mismo directorio que los archivos que contienen el código fuente (`.java`).
- Esto funciona con proyectos pequeños, pero cuando son muy extensos, es necesario mantener estos archivos en directorios separados.
- Así con la opción `-d` podemos indicar el directorio en el cual serán colocados los archivos `.class`

Compilando con javac, opción -d:

- Vamos a crear la siguiente estructura de directorios:



Clases Animal y Dog:



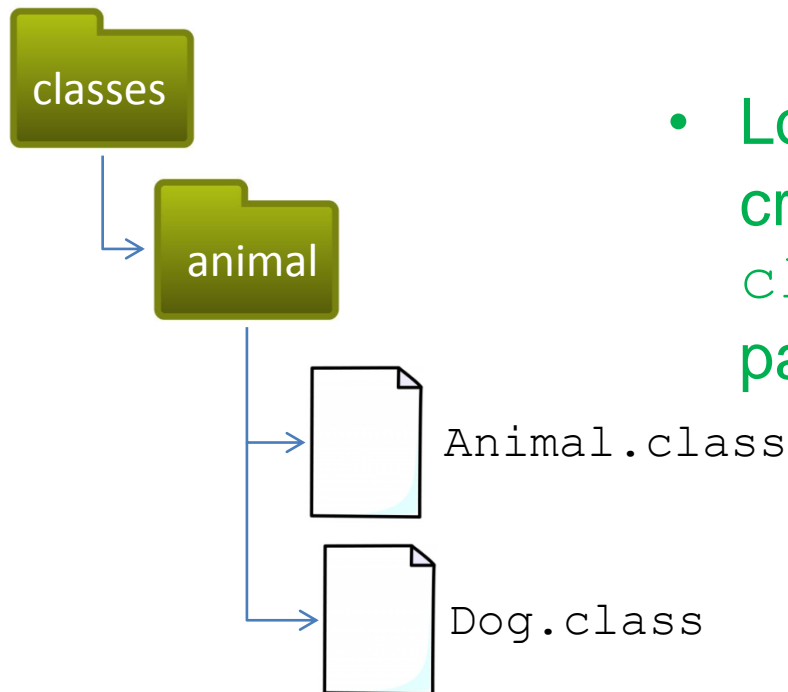
```
public abstract class Animal{  
    public abstract void eat();  
}  
  
public class Dog extends Animal{  
    public void eat(){  
        System.out.println("Dog eating croquettes");  
    }  
}
```

Compilando con javac, opción -d:



- Ejecutamos los siguientes comandos desde consola:

- ✓ `cd myProject`
- ✓ `cd source`
- ✓ `javac -d ../classes animal/Dog.java`



- Los archivos `.class` fueron creados dentro de la carpeta `classes` al igual que sus paquetes.

Compilando con javac, opción -d:



- En este caso el compilador construirá exactamente la jerarquía de directorios en las que se encuentran los archivos `.class`.
- Lo siguiente que necesitas saber para el examen es que si el directorio destino no existe se tendrá un error de compilación, que dirá algo como:

```
java:5: error while writing Dog:  
classes/Dog.class (No such file or directory)
```

Opciones necesarias para el examen:



- Para el examen necesitas conocer las opciones:
 - classpath (-cp)
 - d
- También es importante conocer la estructura de los comando:

```
javac [opciones] [archivo fuente]
java  [opciones] clase [argumentos]
```
- Las partes del comando java [opciones] y [argumentos] son opcionales y ambas pueden tener múltiples valores.

- El compilador necesita tener la misma lista de lugares(directorios), para realizar la búsqueda.
- En el caso de que encuentren en la búsqueda dos archivos con el mismo nombre , el primer archivo encontrado será el que se usara.
- El primer lugar donde el compilador busca es en los directorios que contienen clases que vienen en J2SE.
- En segundo lugar donde buscan es en los directorios definidos en las classpaths.



Classpaths:



- Las classpaths son listas de directorios donde las clases pueden ser encontradas.
- Hay dos lugares donde las classpaths pueden ser declaradas:
 - ✓ En una variable de entorno.
 - ✓ Desde línea de comandos, con la opción `-classpath` (`-cp`) para java y javac.

Declarando y usando classpaths:



- Las classpaths consisten en un conjunto de directorios, separados por un delimitador.
- En los sistemas operativos Unix se utiliza (/), para los directorios y como separador (:)
 - ✓ `-classpath /com/foo/acct:/com/foo`
- Se esta especificando dos directorios en las cuales las clases podrán ser encontradas:
 - ✓ `/com/foo/acct` y
 - ✓ `/com/foo.`
- En ambos casos los directorios son absolutos, es decir puestos desde la raíz del sistema.

Declarando y usando classpaths:



- Cuando estas buscando archivos de clases con el comando java o javac no busca en el directorio actual por default, **TU SE LO TIENES QUE INDICAR!**
- La manera en la que java o javac buscan en el directorio actual es agregando un punto(.) a la classpath.
 - ✓ `-classpath /com/foo/acct:/com/foo:.`
- Las classpaths realizan la búsquedas de izquierda a derecha:
 - ✓ `-classpath /com:/foo:.`
- No es lo mismo que:
 - ✓ `-classpath ./foo:/com`

Usando classpaths:



- Regresamos a nuestro proyecto anterior y seguimos las siguientes instrucciones:
 1. Eliminamos la carpeta animal que se encuentra dentro de classes la cual contiene los archivos .class.
 2. Creamos una carpeta con el nombre de misclases en c:
 3. Ejecutamos los siguientes comandos desde consola:
 - ✓ `cd myProject`
 - ✓ `cd source`
 - ✓ `javac -d ../classes -classpath /misclases animal/Dog.java`

Usando classpaths:



```
Símbolo del sistema
C:\myProject\source>javac -d ../classes -classpath /misc/animal/Dog.java
animal\Dog.java:2: cannot find symbol
symbol: class Animal
public class Dog extends Animal{
                        ^
1 error
C:\myProject\source>
```

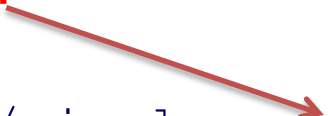
- ¿por que no encuentra la clase Animal si ésta se encuentra en el mismo directorio de Dog?

Usando classpaths:



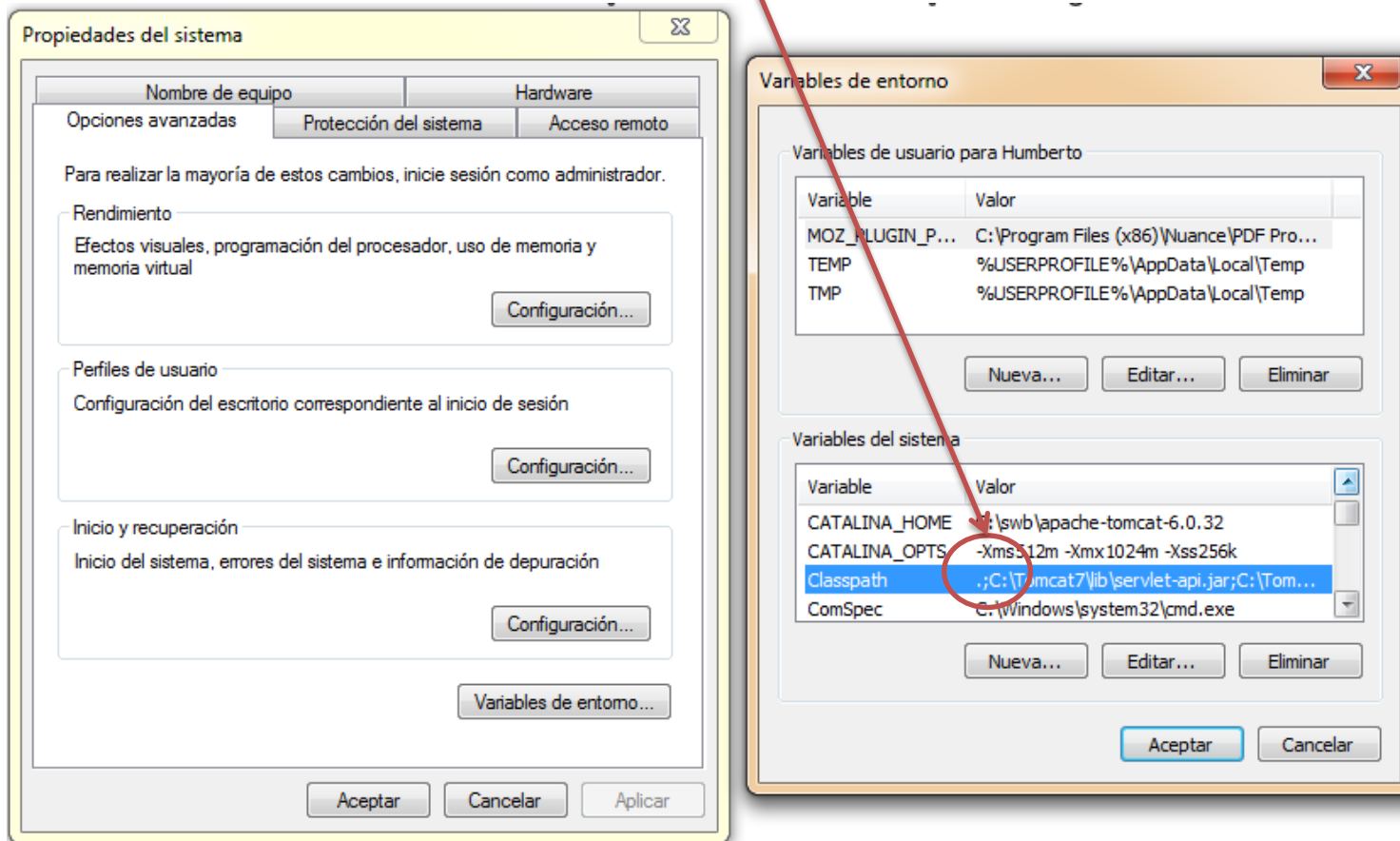
- Cuando se esta buscando archivos de clases con el comando java o javac no busca en el directorio actual por default, **TU SE LO TIENES QUE INDICAR!**

✓ `javac -d ../classes -classpath /misclasses;. animal/Dog.java`

A red arrow originates from the bold text 'TU SE LO TIENES QUE INDICAR!' and points directly to the semicolon in the classpath argument of the code snippet below.

Usando classpaths:

- La otra parte donde puedes declarar tus classpath son en una variable entorno, analicemos lo siguiente:



Usando classpaths:

- En mi variable classpath del sistema tengo definido el directorio actual.
- Entonces, ¿Por qué no busco en el directorio actual cuando no lo definí desde consola?



Usando classpaths:

- Cuando tu defines tus classpaths desde consola, las del sistema son ignoradas.



- Una vez que has creado y probado tu aplicación necesitas una forma para distribuirla a otra gente.
- Ese es el mecanismo que Java provee para realizar dicha acción son los archivos JAR, los cuales comprimen los datos (similar a los archivos ZIP).
- Una vez que has creado tu archivo JAR que contiene todos los archivos de tu aplicación, puede ser movido de un lugar a otro y todas las clases dentro del JAR pueden ser acsesadas vía `classpath` usando `java` y `javac`

- El comando `jar` crea un archivo JAR que contendrá el directorio de tu aplicación, todo el árbol de subdirectorios y archivos.
- Continuando con nuestro proyecto, hagamos los siguiente:
 1. Elimina el archivo `Dog.class` que se encuentra dentro de `classes`.
 2. Es necesario que el directorio actual de tu consola este en `classes`.
 3. Ejecuta el siguiente comando:
✓ `jar -cf MyJar.jar animal`

4. Elimina el archivo Animal.java de la source.
5. Elimina la carpeta animal que se encuentra dentro de classes.
6. Ahora la única copia de la clase animal es la que se encuentra dentro del jar.
7. Mueve tu archivo .jar a la carpeta misclases que creaste en c:
8. Sitúa tu consola en el directorio source.
9. Ejecuta la siguiente linea de comandos:
 - ✓ `javac -d ../classes -classpath /misclases/MyJar.jar animal/Dog.java`

Usando .../jre/lib/ext con archivos JAR:



- Cuando instalas Java, se crea un enorme árbol de directorios, incluyendo archivos JAR que contienen las clases que vienen con J2SE.
- Pero también puedes colocar los tuyos en el directorio:
 - ✓ `$JAVA_HOME/jre/lib/ext/`
- Si pones tus archivos JAR dentro de este árbol de subdirectorios el comando java y javac podrán encontrarlos y usar las clases que se encuentran en ellos.
- **NO necesitas mencionarlos usando la classpath.**



Garbage Collector

Garbage Collector:

- A diferencia del lenguaje C++ Java proporciona un mecanismo de administración de la memoria automático, lo cual implica que la tarea de remover objetos que ya no son usados, no es tarea del programador si no de la propia JVM, este mecanismo es conocido como Garbage Collector.



- El propósito del GC es eliminar los objetos que ya no son alcanzables por alguna referencia desde algún Thread vivo.
- El mecanismo del GC solo lo conoce quien implementa la JVM, y puede variar de una implementación a otra, por lo cual el algoritmo no se garantiza que sea el mismo de una JVM.
- Los objetos deben ser elegibles antes de que ellos sean recolectados por el GC.

- Las íslas de objetos pueden ser recolectadas incluso si los objetos que la forman hacen referencia entre ellos.
- No puedes invocar el mecanismo del GC solo le puedes sugerir a la JVM que tienes objetos que pueden ser elegibles para recolectar con el método `System.gc()` o `Runtime.gc()`.

- La clase `Object` tiene un metodo `finalize()`.
- El método `finalize()` puede llegar a ser ejecutado por el GC antes de eliminar el objeto, pero no hay garantias de que se ejecute siempre, de lo que si hay garantía es que si lo invoca solo lo haga una sola vez.
- La firma del método es:

```
protected void finalize()
```