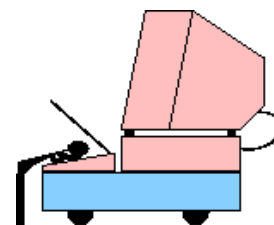




Procesamiento de Formatos en Aplicaciones Telemáticas



Práctica 0: Entorno de trabajo y classpath

No es necesario entregar nada para este enunciado, pero se debe leer el enunciado completo y realizar todas las tareas indicadas en él antes de la primera sesión de laboratorio.

Objeto de la sesión

La realización de la práctica de esta asignatura requiere por una parte de programación en Java y por otra del uso de 2 herramientas: un generador de analizadores léxicos y un generador de analizadores sintácticos. La complejidad de la práctica hace aconsejable organizar las clases Java en paquetes y por otra parte las propias herramientas que se van a manejar requieren el uso de paquetes.

El objetivo de esta sesión de prácticas es que los alumnos instalen y aprendan a manejar el generador de analizadores léxicos (JLex) y el generador de analizadores sintácticos (CUP), así como que conozcan lo necesario para compilar y ejecutar programas en Java que constan de clases organizadas en paquetes, en concreto, lo relativo al `classpath`.



Organización de clases Java en paquetes. El classpath

Al ejecutar `java` o `javac` para ejecutar o compilar clases Java, estas aplicaciones buscan las clases utilizadas por la clase que estamos ejecutando o compilando en ciertos directorios. Estos directorios se definen por medio de lo que se llama el `classpath`. Hay 2 formas de definir el `classpath`: por medio de una variable de entorno y mediante la opción `-classpath` de `java` y `javac`. Una variable de entorno es una variable que el sistema operativo le pasa a un programa al ser ejecutado; no confundir con una variable de un programa que es creada dentro del propio programa. Veamos ahora cómo se define el `classpath`.

En las aulas informáticas cuando se usa Linux, trabajáis con la shell `bash`. En este caso, para definir el `classpath` con una variable de entorno debéis llamar al comando

```
export CLASSPATH=./home2/.../dir1:/home2/.../dir2
```

Con esto lo que hacéis es crear una variable de entorno (`CLASSPATH`), que apunta a un conjunto de directorios separados por ":". En estos directorios es en donde se pueden encontrar las clases Java. El primer directorio que aparece es ".", que hace referencia al directorio en el que se está en el momento de ejecutar Java. Esto permite usar las clases definidas en dicho directorio, sin tener que incluirlo en el `classpath`.

El ejecutar la sentencia `export` anterior en un terminal define la variable de entorno sólo en este terminal. Con el fin de evitar tener que ejecutar este comando cada vez que se abre un terminal, lo mejor es incluirlo en el fichero `.bashrc` o en `.bash_profile` de vuestra cuenta.

La manera de comprobar que el `classpath` está definido es mediante

```
echo $CLASSPATH
```

Esto muestra por pantalla el contenido de la variable de entorno `CLASSPATH`. Si se desea añadir un directorio adicional a los ya incluidos en el `classpath`, se puede hacer

```
export CLASSPATH=$CLASSPATH:/home2/.../nuevoDirectorio
```

Por último, dado que casi todos trabajáis en Windows, para definir o modificar el `classpath` por medio de una variable de entorno debéis acudir a Panel de Control -> Sistema -> Opciones Avanzadas (o Configuración Avanzada del Sistema) -> Variables de entorno. Debeis tener en cuenta que en Windows los directorios que forman parte del `classpath` se separan con el carácter ";". Tened en cuenta también que la variable de entorno para el `classpath` se debe llamar `CLASSPATH`, igual que en Linux.

Donde se deben colocar las clases y donde las buscan java y javac

Cuando la clase que se está buscando pertenece a un paquete, se supone que los paquetes son subdirectorios de alguno de los directorios indicados en el classpath.

Consideremos por ejemplo las siguientes clases Java (las puede obtener [aquí](#)):

```
import Ejem1.EjemPack1;
import Ejem1.Ejem2.EjemPack2;

public class Ejemplo {

    public static void main(String args[]) {
        EjemPack1 obj1= new EjemPack1();
        EjemPack2 obj2= new EjemPack2();

        obj1.setVar1(2);
        obj2.setVar2(3);

        System.out.println(obj1.getVar1() + " + " + obj2.getVar2() + " = " +
            (obj1.getVar1() + obj2.getVar2()));
    }
}
```

```
package Ejem1;

public class EjemPack1 {

    private int var1;

    public EjemPack1() {
        var1=0;
    }

    public void setVar1(int x) {
        var1=x;
    }

    public int getVar1() {
        return var1;
    }
}
```

```
package Ejem1.Ejem2;

public class EjemPack2 {

    private int var2;

    public EjemPack2() {
        var2=0;
    }

    public void setVar2(int x) {
        var2=x;
    }

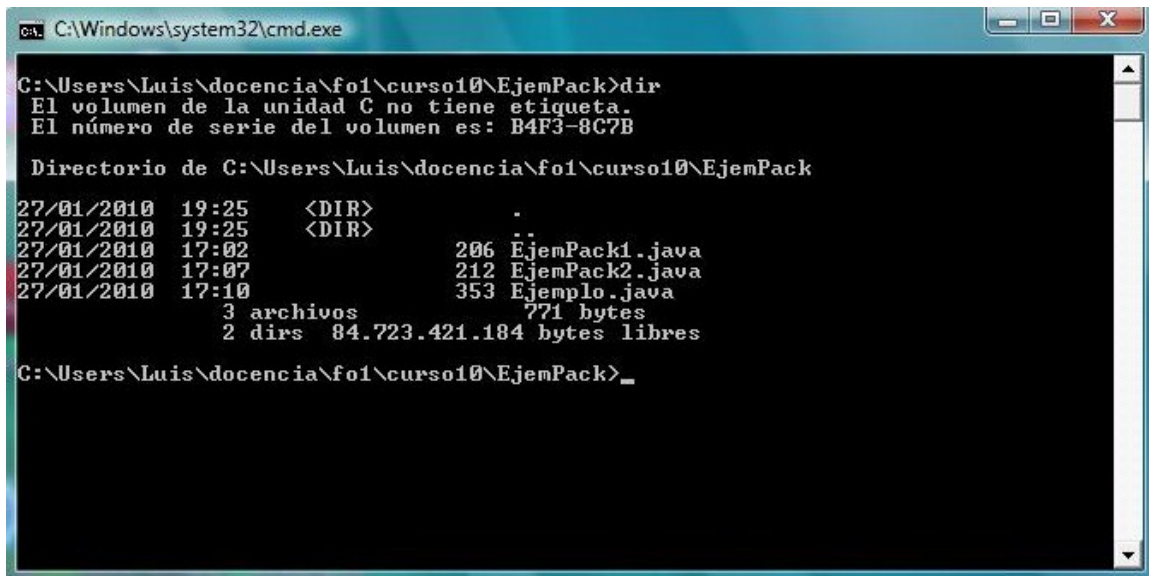
    public int getVar2() {
        return var2;
    }
}
```

Cuando queramos compilar o ejecutar la clase Ejemplo, java o javac necesitan localizar el paquete Ejem1 (y dentro de este la clase EjemPack1) y el paquete Ejem1.Ejem2 (y dentro de este la clase EjemPack2). java y javac para encontrar los paquetes que necesitan buscan en los directorios indicados en el classpath subdirectorios cuyo nombre coincida con el del paquete que están buscando.

Por ejemplo, supongamos que el classpath define los directorios donde buscar las clases /usr/java/ y /home/fo1/class/. Para localizar el paquete Ejem1, java o javac buscarán un directorio que se llame Ejem1 dentro de los directorios definidos en el classpath, es decir, buscarán si existe el directorio /usr/java/Ejem1/ o el directorio /home/fo1/class/Ejem1/. Si dicho directorio no existe, inmediatamente se terminará, emitiendo un mensaje de error. En caso de que alguno de los dos directorios exista (en el caso de que se encontrara el primer directorio se tomaría este como la ubicación del paquete y no se continuaría buscando; por eso no sólo es importante que directorios colocar en el classpath sino también puede ser importante en algún caso el orden en que se colocan), se buscará dentro de el la clase EjemPack1. Supongamos ahora que el directorio /usr/java/Ejem1/ existe. Se pasa a continuación a buscar si dentro de dicho directorio hay un fichero EjemPack1.class. En caso contrario, de nuevo se abortaría y se produciría un mensaje de error.

Análogamente, para buscar el paquete Ejem1.Ejem2, primero se buscará si existe el directorio /usr/java/Ejem1/Ejem2/ o el directorio /home/fo1/class/Ejem1/Ejem2/ y si se encuentra se comprobará si contiene un fichero de nombre EjemPack2.class.

Podemos a continuación comprobar todo esto por nosotros mismos. Empecemos descargando y descomprimiendo el fichero [EjemPack.zip](#) que contiene las 3 clases Ejemplo, EjemPack1 y EjemPack2. Al descomprimir EjemPack.zip se creará un directorio de nombre EjemPack que contiene 3 ficheros: EjemPack1.java, EjemPack2.java y Ejemplo.java. Utilizando una terminal ubiquémonos en el directorio EjemPack. Podemos comprobar su contenido:



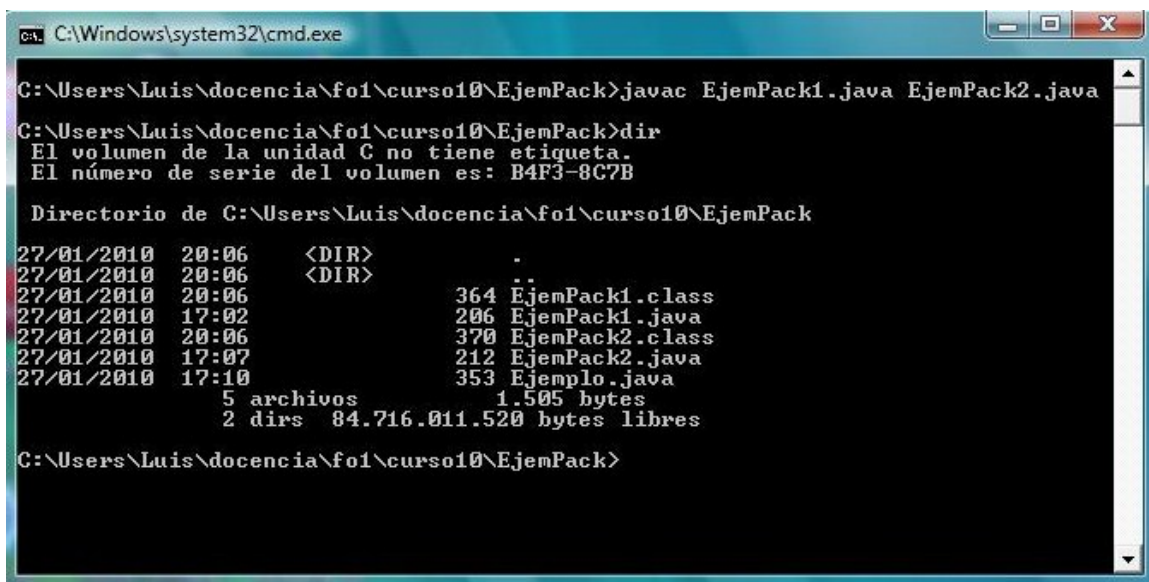
```
C:\Windows\system32\cmd.exe

C:\Users\Luis\docencia\fo1\curso10\EjemPack>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack
27/01/2010  19:25    <DIR>          .
27/01/2010  19:25    <DIR>          ..
27/01/2010  17:02             206 EjemPack1.java
27/01/2010  17:07             212 EjemPack2.java
27/01/2010  17:10             353 Ejemplo.java
                3 archivos             771 bytes
                2 dirs 84.723.421.184 bytes libres

C:\Users\Luis\docencia\fo1\curso10\EjemPack>_
```

Empezamos compilando las clases EjemPack1 y EjemPack2:



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

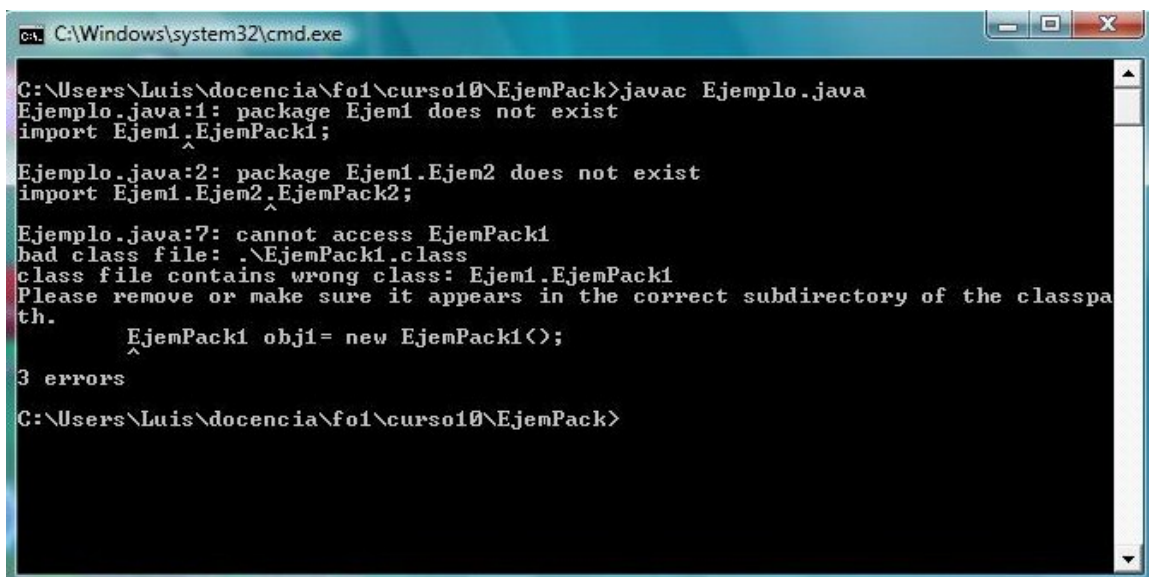
C:\Users\Luis\docencia\fo1\curso10\EjemPack>javac EjemPack1.java EjemPack2.java
C:\Users\Luis\docencia\fo1\curso10\EjemPack>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack
27/01/2010  20:06    <DIR>          .
27/01/2010  20:06    <DIR>          ..
27/01/2010  20:06             364 EjemPack1.class
27/01/2010  17:02             206 EjemPack1.java
27/01/2010  20:06             370 EjemPack2.class
27/01/2010  17:07             212 EjemPack2.java
27/01/2010  17:10             353 Ejemplo.java
                5 archivos             1.505 bytes
                2 dirs 84.716.011.520 bytes libres

C:\Users\Luis\docencia\fo1\curso10\EjemPack>
```

Como podemos observar, la compilación se produce sin problemas y se generan los correspondientes ficheros en formato class, EjemPack1.class y EjemPack2.class. Esto sucede porque ninguna de las 2 clases tiene dependencias (es decir, no requiere para ser compilada ninguna otra clase).

El problema aparece al intentar compilar Ejemplo.java:



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

C:\Users\Luis\docencia\fo1\curso10\EjemPack>javac Ejemplo.java
Ejemplo.java:1: package Ejem1 does not exist
import Ejem1.EjemPack1;
^
Ejemplo.java:2: package Ejem1.Ejem2 does not exist
import Ejem1.Ejem2.EjemPack2;
^
Ejemplo.java:7: cannot access EjemPack1
bad class file: .\EjemPack1.class
class file contains wrong class: Ejem1.EjemPack1
Please remove or make sure it appears in the correct subdirectory of the classpa
th.
    EjemPack1 obj1= new EjemPack1();
^
3 errors

C:\Users\Luis\docencia\fo1\curso10\EjemPack>
```

Los mensajes de error nos explican porque da error la compilación. Los 2 primeros ("package Ejem1 does not exist") nos dicen

que no se ha encontrado el paquete Ejem1 al no haber ningún directorio con nombre Ejem1. El tercero ("bad class file...") nos dice que el fichero EjemPack1.class contiene la clase Ejem1.EjemPack1, lo cual no es correcto porque no está dentro de un directorio que se llame Ejem1 y que se pueda acceder desde el classpath.

Para intentar arreglar esto probemos a colocar las clases EjemPack1.class dentro de un directorio Ejem1 y EjemPack2.class dentro de un directorio Ejem1/Ejem2. El directorio Ejem1 lo ubicaremos dentro de otro directorio que llamaremos paquetes.

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

C:\Users\Luis\docencia\fo1\curso10\EjemPack>mkdir paquetes
C:\Users\Luis\docencia\fo1\curso10\EjemPack>cd paquetes
C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes>mkdir Ejem1
C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes>cd Ejem1
C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes\Ejem1>mkdir Ejem2
C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes\Ejem1>cd ../../
C:\Users\Luis\docencia\fo1\curso10\EjemPack>move EjemPack1.class paquetes\Ejem1
Se han movido      1 archivos.
C:\Users\Luis\docencia\fo1\curso10\EjemPack>move EjemPack2.class paquetes\Ejem1\Ejem2
Se han movido      1 archivos.
```

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

C:\Users\Luis\docencia\fo1\curso10\EjemPack>dir /s
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack
27/01/2010  21:23    <DIR>        .
27/01/2010  21:23    <DIR>        ..
27/01/2010  17:02             206 EjemPack1.java
27/01/2010  17:07             212 EjemPack2.java
27/01/2010  17:10             353 Ejemplo.java
27/01/2010  21:23    <DIR>        paquetes
                        3 archivos          771 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes
27/01/2010  21:23    <DIR>        .
27/01/2010  21:23    <DIR>        ..
27/01/2010  21:23    <DIR>        Ejem1
                        0 archivos          0 bytes

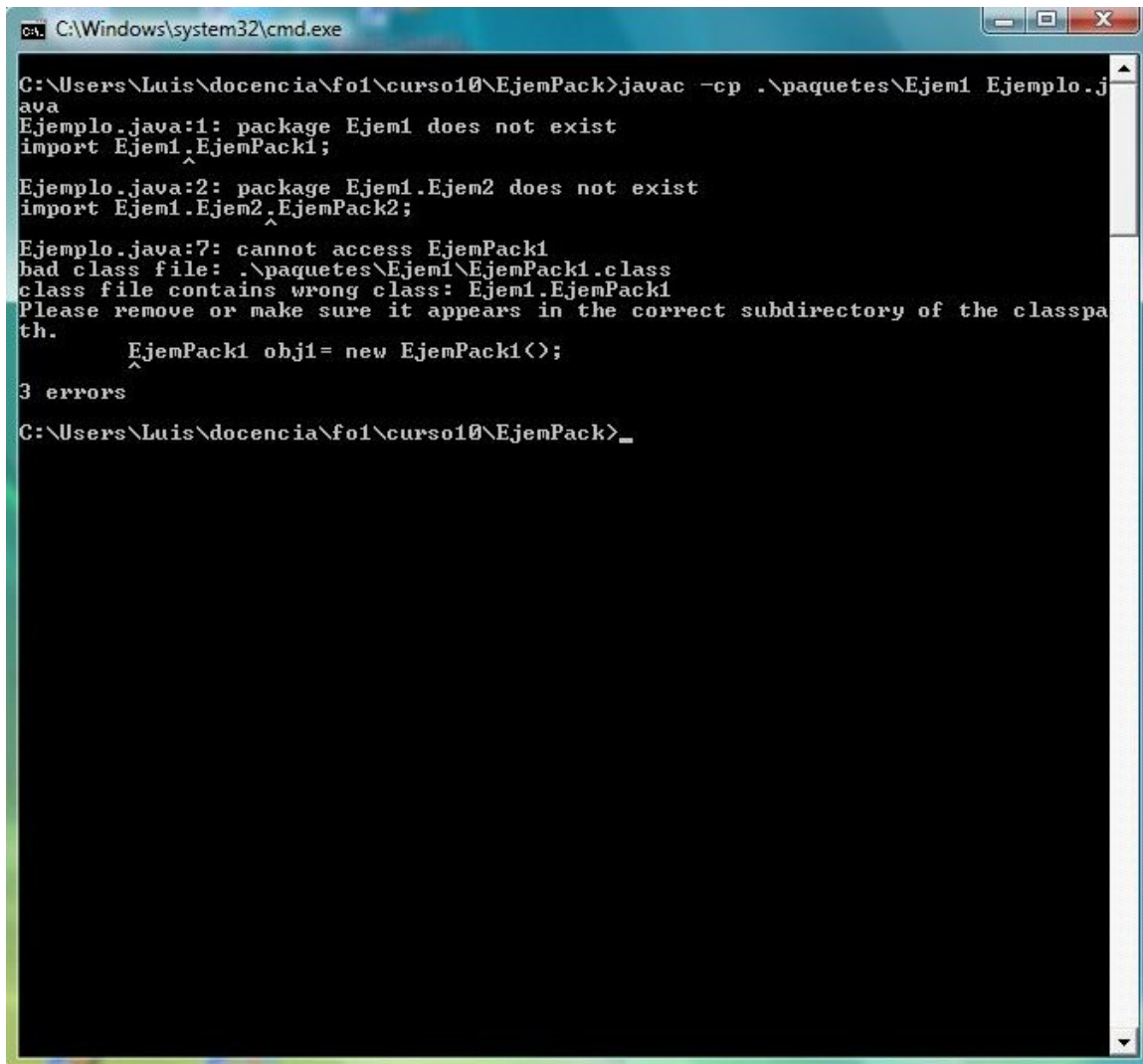
Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes\Ejem1
27/01/2010  21:23    <DIR>        .
27/01/2010  21:23    <DIR>        ..
27/01/2010  21:23    <DIR>        Ejem2
27/01/2010  21:22             364 EjemPack1.class
                        1 archivos          364 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes\Ejem1\Ejem2
27/01/2010  21:23    <DIR>        .
27/01/2010  21:23    <DIR>        ..
27/01/2010  20:06             370 EjemPack2.class
                        1 archivos          370 bytes

Total de archivos en la lista:
      5 archivos          1.505 bytes
     11 dirs 84.717.211.648 bytes libres

C:\Users\Luis\docencia\fo1\curso10\EjemPack>_
```

Volvemos a intentar compilar Ejemplo.java. Esta vez vamos a definir explícitamente el classpath como el directorio Ejem1 que acabamos de crear, utilizando la opción -cp para ello. Como podemos observar la compilación vuelve a fallar:



```

C:\Windows\system32\cmd.exe

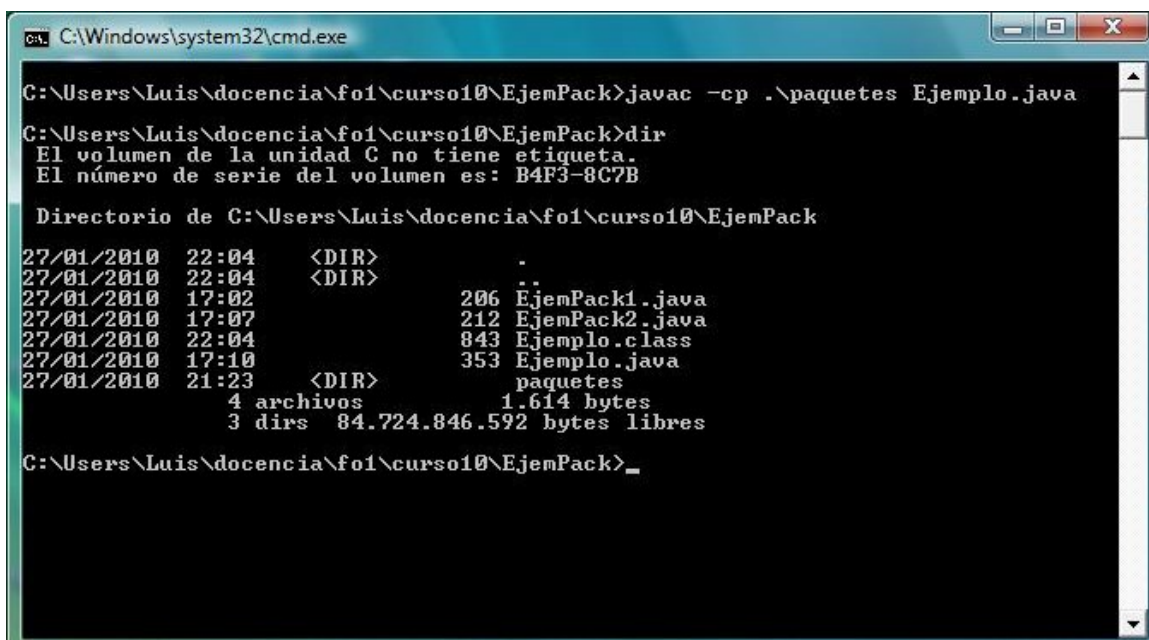
C:\Users\Luis\docencia\fo1\curso10\EjemPack>javac -cp .\paquetes\Ejem1 Ejemplo.java
Ejemplo.java:1: package Ejem1 does not exist
import Ejem1.EjemPack1;
^
Ejemplo.java:2: package Ejem1.Ejem2 does not exist
import Ejem1.Ejem2.EjemPack2;
^
Ejemplo.java:7: cannot access EjemPack1
bad class file: .\paquetes\Ejem1\EjemPack1.class
class file contains wrong class: Ejem1.EjemPack1
Please remove or make sure it appears in the correct subdirectory of the classpath.
    EjemPack1 obj1= new EjemPack1();
    ^
3 errors
C:\Users\Luis\docencia\fo1\curso10\EjemPack>_

```

Podemos observar que los errores son exactamente los mismos que en el caso anterior. En particular, el primer error ("package Ejem1 does not exist") se debe a que javac no encuentra un directorio que se llame Ejem1 dentro de alguno de los directorios del classpath, es decir, no existe el directorio .\paquetes\Ejem1\Ejem1\

Puede usted a estas alturas intentar razonar porque los otros 2 errores se vuelven a producir.

Por último, volvemos a intentar compilar la clase Ejemplo.java, pero esta vez vamos a definir como classpath el directorio paquetes, que es el que contiene el directorio Ejem1:



```

C:\Windows\system32\cmd.exe

C:\Users\Luis\docencia\fo1\curso10\EjemPack>javac -cp .\paquetes Ejemplo.java
C:\Users\Luis\docencia\fo1\curso10\EjemPack>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

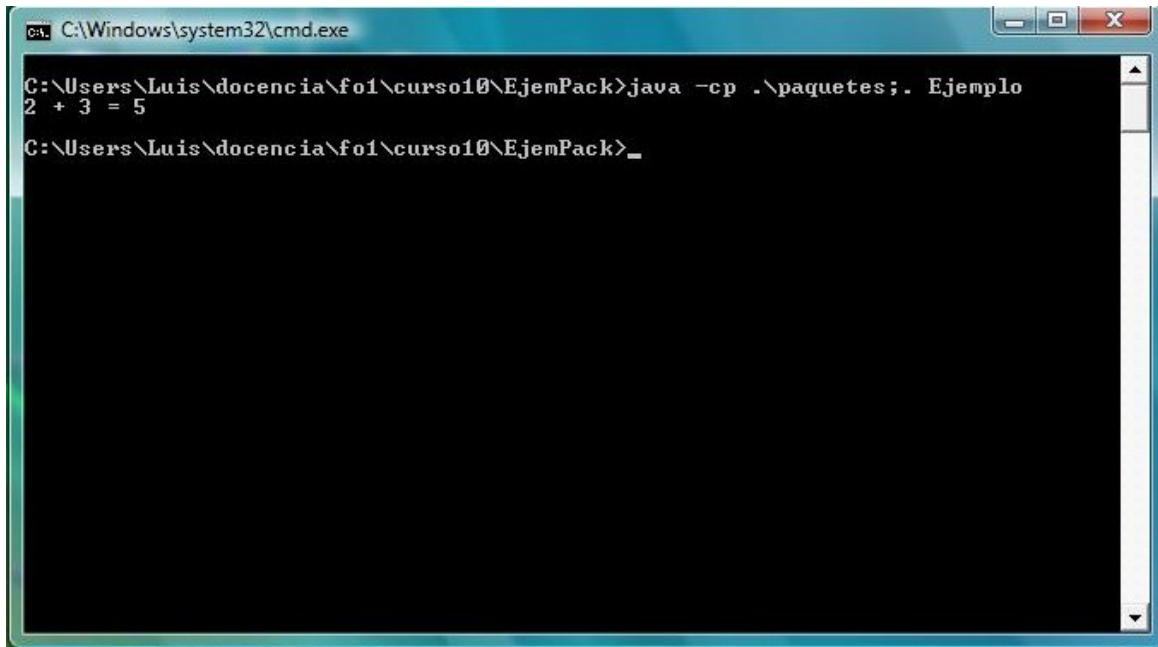
Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack
27/01/2010  22:04    <DIR>        .
27/01/2010  22:04    <DIR>        ..
27/01/2010  17:02           206 EjemPack1.java
27/01/2010  17:07           212 EjemPack2.java
27/01/2010  22:04           843 Ejemplo.class
27/01/2010  17:10           353 Ejemplo.java
27/01/2010  21:23    <DIR>        paquetes
                4 archivos             1.614 bytes
                3 dirs  84.724.846.592 bytes libres
C:\Users\Luis\docencia\fo1\curso10\EjemPack>_

```

Podemos observar que en este caso la compilación se realiza sin errores y se genera el fichero Ejemplo.class. javac busca un directorio Ejem1 dentro de los directorios definidos en el classpath (es decir, en el directorio paquetes, lo encuentra y lo asigna al

paquete Ejem1. Análogamente, javac busca un directorio Ejem1\Ejem2 en el directorio paquetes, lo encuentra y lo asigna al paquete Ejem1.Ejem2. A continuación, en dichos directorios encuentra las clases que necesita de los paquetes Ejem1 y Ejem1.Ejem2.

Ahora ya podemos ejecutar Ejemplo:



```
C:\Windows\system32\cmd.exe
C:\Users\Luis\docencia\fo1\curso10\EjemPack>java -cp .\paquetes;. Ejemplo
2 + 3 = 5
C:\Users\Luis\docencia\fo1\curso10\EjemPack>_
```

Como podemos observar, en este caso el classpath contiene 2 directorios: el directorio "." (el directorio actual), donde java va a encontrar la clase Ejemplo y el directorio paquetes donde java va a encontrar las clases EjemPack1 y EjemPack2.

A estas alturas deberíamos haber entendido cual es la estructura de directorios que esperan java y javac para los paquetes y cual es el classpath que debo definir para que java o javac encuentren determinadas clases y paquetes.

Como se habrá podido observar, el proceso de ubicar los ficheros .class en los directorios adecuados resulta un tanto engorroso. Afortunadamente, javac puede hacer esta tarea por nosotros utilizando la opción -d para indicar donde queremos dejar nuestros ficheros .class.

Procedamos a borrar todo el contenido del directorio paquetes (incluyendo los subdirectorios) y a continuación volvamos a compilar las clases EjemPack1 y EjemPack2, indicándole a javac por medio de la opción -d que deseamos dejar los ficheros .class en el directorio paquetes:

```

C:\Windows\system32\cmd.exe

C:\Users\Luis\docencia\fo1\curso10\EjemPack>dir paquetes
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes
27/01/2010  22:53    <DIR>          -
27/01/2010  22:53    <DIR>          ..
                0 archivos                0 bytes
                2 dirs 84.714.737.664 bytes libres

C:\Users\Luis\docencia\fo1\curso10\EjemPack>javac -d .\paquetes EjemPack1.java EjemPack2.java

C:\Users\Luis\docencia\fo1\curso10\EjemPack>dir /s paquetes
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes
27/01/2010  22:54    <DIR>          -
27/01/2010  22:54    <DIR>          ..
27/01/2010  22:54    <DIR>          Ejem1
                0 archivos                0 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes\Ejem1
27/01/2010  22:54    <DIR>          -
27/01/2010  22:54    <DIR>          ..
27/01/2010  22:54    <DIR>          Ejem2
27/01/2010  22:54    EjemPack1.class  364
                1 archivos                364 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\paquetes\Ejem1\Ejem2
27/01/2010  22:54    <DIR>          -
27/01/2010  22:54    <DIR>          ..
27/01/2010  22:54    EjemPack2.class  370
                1 archivos                370 bytes

Total de archivos en la lista:
                2 archivos                734 bytes
                8 dirs 84.714.659.840 bytes libres

C:\Users\Luis\docencia\fo1\curso10\EjemPack>_

```

Como podemos observar, javac se ha ocupado tanto de crear los directorios Ejem1 y Ejem2 como de dejar los ficheros .class en el lugar adecuado.

En el desarrollo de programas Java complejos, como el que se debe realizar en esta práctica se **recomienda** que se estructure el proyecto de programación en un directorio que contenga dos subdirectorios java y class que contengan los ficheros Java y los ficheros en formato .class respectivamente.

El directorio java, a su vez contendría subdirectorios que reflejasen la estructura de paquetes utilizada. Así, si en el ejemplo anterior hubiésemos organizado los ficheros según esta filosofía, debería existir un directorio java, que contendría un directorio que se llame Ejem1 y el fichero Ejemplo.java. A su vez, el directorio Ejem1 contendría un directorio que se llame Ejem2 y el fichero EjemPack1.java. Finalmente, el directorio Ejem2 contendría exclusivamente el fichero EjemPack2.java.

Las siguientes capturas de pantalla presentan el proceso de compilación y ejecución en este caso.

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

C:\Users\Luis\docencia\fo1\curso10\EjemPack>dir /s
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack
28/01/2010  19:21    <DIR>        .
28/01/2010  19:21    <DIR>        ..
28/01/2010  19:20    <DIR>        class
28/01/2010  19:21    <DIR>        java
                0 archivos                0 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\class
28/01/2010  19:20    <DIR>        .
28/01/2010  19:20    <DIR>        ..
                0 archivos                0 bytes

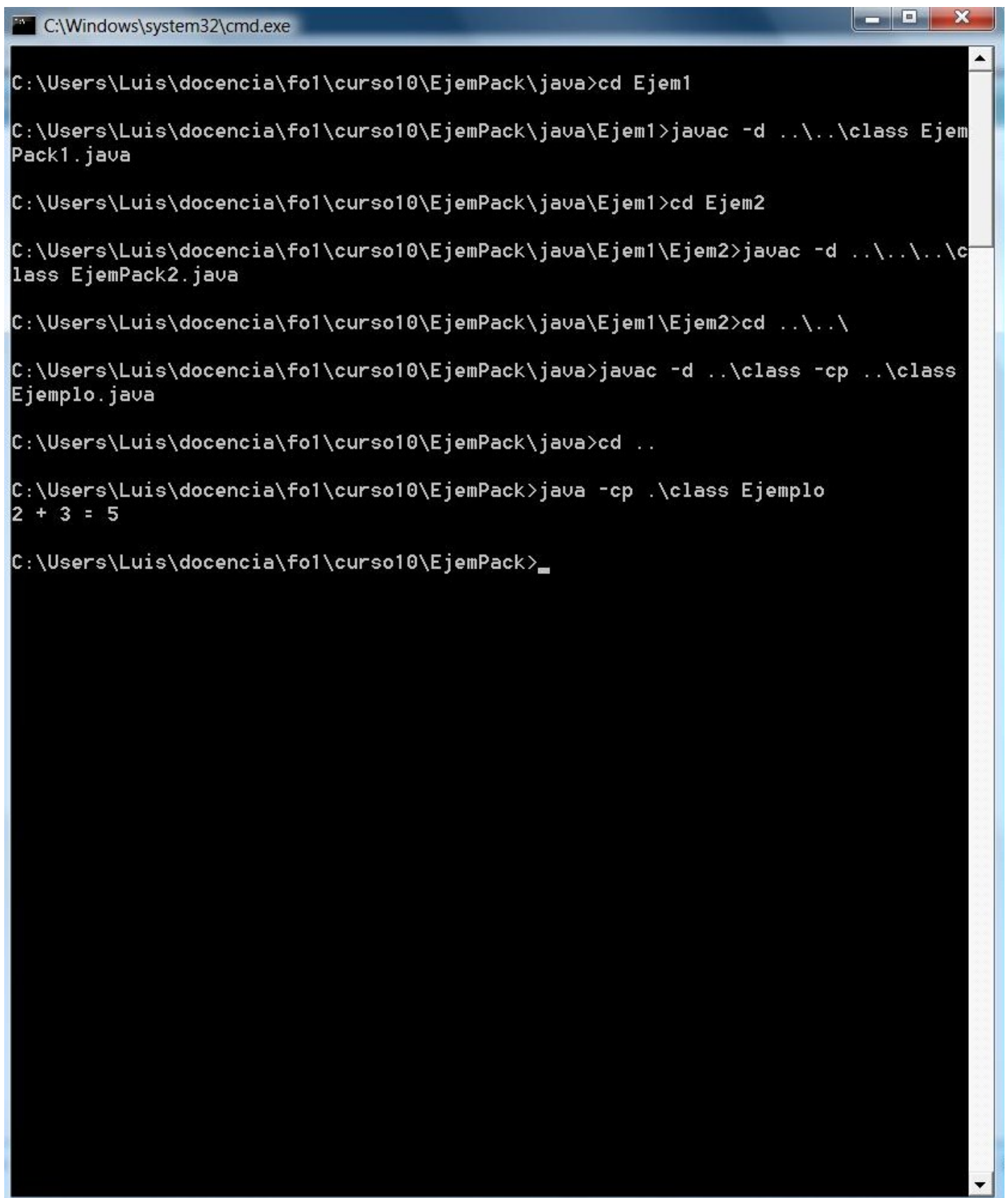
Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\java
28/01/2010  19:21    <DIR>        .
28/01/2010  19:21    <DIR>        ..
28/01/2010  19:20    <DIR>        Ejem1
27/01/2010  17:10                353 Ejemplo.java
                1 archivos                353 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\java\Ejem1
28/01/2010  19:20    <DIR>        .
28/01/2010  19:20    <DIR>        ..
28/01/2010  19:21    <DIR>        Ejem2
27/01/2010  17:02                206 EjemPack1.java
                1 archivos                206 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\java\Ejem1\Ejem2
28/01/2010  19:21    <DIR>        .
28/01/2010  19:21    <DIR>        ..
27/01/2010  17:07                212 EjemPack2.java
                1 archivos                212 bytes

Total de archivos en la lista:
                3 archivos                771 bytes
                14 dirs 84.601.040.896 bytes libres

C:\Users\Luis\docencia\fo1\curso10\EjemPack>_
```

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

C:\Users\Luis\docencia\fo1\curso10\EjemPack\java>cd Ejem1

C:\Users\Luis\docencia\fo1\curso10\EjemPack\java\Ejem1>javac -d ..\..\class EjemPack1.java

C:\Users\Luis\docencia\fo1\curso10\EjemPack\java\Ejem1>cd Ejem2

C:\Users\Luis\docencia\fo1\curso10\EjemPack\java\Ejem1\Ejem2>javac -d ..\..\..\class EjemPack2.java

C:\Users\Luis\docencia\fo1\curso10\EjemPack\java\Ejem1\Ejem2>cd ..\..\

C:\Users\Luis\docencia\fo1\curso10\EjemPack\java>javac -d ..\class -cp ..\class Ejemplo.java

C:\Users\Luis\docencia\fo1\curso10\EjemPack\java>cd ..

C:\Users\Luis\docencia\fo1\curso10\EjemPack>java -cp .\class Ejemplo
2 + 3 = 5

C:\Users\Luis\docencia\fo1\curso10\EjemPack>_
```

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

C:\Users\Luis\docencia\fo1\curso10\EjemPack>cd class

C:\Users\Luis\docencia\fo1\curso10\EjemPack\class>dir /s
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: B4F3-8C7B

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\class

28/01/2010  19:35    <DIR>          .
28/01/2010  19:35    <DIR>          ..
28/01/2010  19:34    <DIR>          Ejem1
28/01/2010  19:44                843 Ejemplo.class
                1 archivos                843 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\class\Ejem1

28/01/2010  19:34    <DIR>          .
28/01/2010  19:34    <DIR>          ..
28/01/2010  19:34    <DIR>          Ejem2
28/01/2010  19:44                364 EjemPack1.class
                1 archivos                364 bytes

Directorio de C:\Users\Luis\docencia\fo1\curso10\EjemPack\class\Ejem1\Ejem2

28/01/2010  19:34    <DIR>          .
28/01/2010  19:34    <DIR>          ..
28/01/2010  19:44                370 EjemPack2.class
                1 archivos                370 bytes

Total de archivos en la lista:
        3 archivos                1.577 bytes
        8 dirs  84.600.750.080 bytes libres

C:\Users\Luis\docencia\fo1\curso10\EjemPack\class>
```

De este ejemplo cabe resaltar lo siguiente:

- Es posible combinar simultáneamente las opciones `-d` y `-cp` como se hace al compilar `Ejemplo.java`, indicando con `-cp` donde encontrar las clases `EjemPack1` y `EjemPack2` y con `-d` donde queremos dejar la clase compilada.
- El fichero `Ejemplo.class` se deja directamente en el directorio `class` (y no en uno de sus subdirectorios) al no haberse declarado un paquete para él.

Como se estudiará en la asignatura, 2 de los componentes básicos de un compilador son el analizador léxico y el analizador sintáctico. El analizador léxico se encarga de leer el programa fuente (el programa que hay que traducir) y reconoce tokens: palabras clave, identificadores, signos de puntuación, etc. Por su parte, el analizador sintáctico lee los tokens producidos por el analizador léxico, comprueba si hay errores de sintaxis, y en caso contrario, produce una representación en forma de árbol del programa fuente.

Para desarrollar los analizadores léxico y sintáctico de un compilador hay 2 alternativas: programarlos directamente o utilizar generadores de analizadores. Un generador de analizadores es una aplicación software que recibe una especificación del analizador que se desea (el como es esa especificación lo dejamos de momento) y genera el código necesario para el analizador.

Nosotros vamos a utilizar como generador de analizadores léxicos a JLex y como generador de analizadores sintácticos a CUP. Ambos tienen en común que en realidad son programas en Java y los analizadores que producen están formados por una o varias clases Java. Por lo tanto, tanto para poder utilizar JLex como CUP es necesario:

- Conocer que paquetes son necesarios para ejecutar el generador de analizadores y donde se ubican (para poder definir correctamente el classpath al ejecutar el generador de analizadores).
- Conocer que paquetes son utilizados por las clases generadas por el analizador y donde se ubican (para poder definir correctamente el classpath al compilar y ejecutar el analizador generado).

Esto lo vamos a ver al describir las herramientas.

Instalación de CUP y JLex

La instalación de CUP y JLex son muy sencillas. En el caso de CUP, lo único que hay que hacer es bajárselo (la versión recomendada es la 0.10k) del [sitio Web de CUP](#) o de esta [copia](#).

A continuación crearemos un directorio donde queramos tener instalado CUP. Descomprimiremos la distribución de CUP que nos hemos descargado en ese directorio. Situados en el directorio donde hemos descomprimido la distribución de CUP ejecutaremos el siguiente comando:

```
javac java_cup/*.java java_cup/runtime/*.java
```

Si la compilación se realiza sin errores tendremos CUP correctamente instalado.

La instalación de JLex es todavía más sencilla. Lo único que tenemos que hacer es bajarnos el único fichero Java que lo compone (la versión recomendada es la 1.2.6), bien del [sitio Web de JLex](#) o bien de esta [copia](#).

A continuación crearemos un directorio donde queramos tener instalado JLex (este directorio tiene que llamarse JLex ya que la clase Main que vamos a compilar pertenece al paquete JLex) y copiaremos en él el fichero Main.java que nos hemos bajado. Compilaremos el fichero Java (javac Main.java). Si la compilación no produce errores hemos terminado con la instalación de JLex.

Ejemplo JLex

Nota: para probar este ejemplo es necesario haber instalado previamente CUP, ya que el analizador léxico utiliza clases Java que forman parte de la distribución de CUP.

Nota: Cada vez que haya que ejecutar o compilar un programa en Java en este ejemplo será necesario definir el classpath (y el valor de la opción -d cuando se compile), teniendo en cuenta los paquetes que se vayan a utilizar. Los comandos que se proporcionan no incluyen la opción -d ni la opción -cp, pero usted las debe incluir.

En su cuenta cree un directorio EjemplosFOI. Dentro de EjemplosFOI cree un directorio java y otro class. Dentro de EjemplosFOI/java cree un directorio Lexer y otro Parser.

Copie al directorio EjemplosFOI/java/Lexer este pequeño [ejemplo de fichero JLex \(fichero Yylex\)](#) y esta [clase java \(fichero LexerMain.java\)](#) que contiene un método main con el que probar el analizador léxico que va a obtener.

El fichero JLex que se pasa reconoce los siguientes tokens: ";", "+", "*", "(", ")" y enteros no negativos. El generador que se va a generar imprime los tokens reconocidos por pantalla, lo que nos permitirá comprobar su comportamiento.

Para ejecutar el el generador de analizadores JLex es necesario ejecutar la clase Main del paquete JLex, pasándole como argumento el nombre del fichero que contiene la especificación del analizador léxico. No se necesita acceso a ningún otro paquete. Por ejemplo:

```
java JLex.Main Yylex
```

Como hemos visto anteriormente, el paquete JLex está asociado al directorio JLex que habremos creado.

Se obtendrá el analizador léxico que vamos a probar. JLex produce un único fichero con el mismo nombre que el fichero

que le hemos pasado añadiéndole la extensión `.java`. Para compilar y ejecutar un generador de analizadores léxicos obtenido con JLex (en nuestro caso, el fichero `Yylex.java`) es necesario tener acceso al paquete `java_cup.runtime`, que, como nos podemos imaginar, se encuentra en el directorio `java_cup/runtime` que hemos creado cuando descomprimos e instalamos el CUP.

Además, tenemos que saber que el fichero `Yylex` que usamos en este ejemplo indica que la clase Java a generar debe pertenecer a un paquete de nombre `Lexer` (podemos comprobarlo abriendo el fichero `Yylex.java`).

Ahora debemos compilar las clases Java `Yylex.java` y `LexerMain.java` siguiendo las indicaciones que se han explicado en el apartado anterior. Teniendo en cuenta todo lo que acabamos de explicar piense como debe definir el classpath.

Para probar el analizador léxico ejecutaremos, definiendo el classpath adecuado:

```
java LexerMain
```

La clase `LexerMain` no pertenece a ningún paquete.

A través de teclado podemos introducir texto (solamente se reconocen los dígitos y los símbolos `"(", ")", ";", "+"` y `"*"`). Cada vez que pulsemos el retorno de carro se imprimirá en pantalla la cadena de tokens equivalente la texto introducido por teclado. Para terminar pulsaremos simultáneamente las teclas `"Ctrl"` y `"D"`.

Ejemplo CUP (con JLex)

Nota: Cada vez que haya que ejecutar o compilar un programa en Java en este ejemplo será necesario definir el classpath (y el valor de la opción `-d` cuando se compile), teniendo en cuenta los paquetes que se vayan a utilizar. Los comandos que se proporcionan no incluyen la opción `-d` ni la opción `-cp`, pero usted las debe incluir.

Ahora vamos a probar un ejemplo que utiliza un fichero en formato CUP junto con el ejemplo JLex anterior para hacer una sencilla calculadora.

Copie al directorio `EjemplosFOI/java/Parser` este pequeño [ejemplo de fichero CUP \(fichero: `minimal.cup`\)](#).

Para obtener el analizador sintáctico es necesario ejecutar la clase `Main` del paquete `java_cup`, que, como podemos adivinar, se corresponde con el directorio `java_cup` que obtuvimos al descomprimir e instalar el CUP. No se necesita acceder a otros paquetes. Se pasa como argumento el fichero que contiene la especificación del analizador sintáctico a generar. En nuestro caso, ejecutaremos (definiendo el classpath como corresponda):

```
java java_cup.Main minimal.cup
```

Se obtendrá el analizador sintáctico que vamos a probar. El resultado de ejecutar el CUP, si el fichero de entrada no contiene errores, siempre consiste en 2 ficheros: `parser.java` y `sym.java`. Para poder compilar y ejecutar estos 2 ficheros es necesario acceder al paquete `java_cup.runtime`.

Ahora debemos compilar las clases Java `parser.java` y `sym.java`, obtenidas al generar el analizador sintáctico. Tenemos que tener en cuenta que el fichero `minimal.cup` que usamos en este ejemplo indica que las clases Java a generar deben pertenecer a un paquete de nombre `Parser` (podemos comprobarlo abriendo el fichero `parser.java`) y que el fichero `parser.java` generado debe contener un método `main` que a su vez utiliza la clase `Yylex` del paquete `Lexer`.

Para probar el analizador sintáctico ejecutaremos, definiendo el classpath adecuado:

```
java Parser.parser
```

Al igual que en el caso anterior, a través del teclado podemos introducir texto. Pruebe, por ejemplo, a introducir el siguiente texto:

```
(3 + 4) * 5;;
```

Para terminar pulsaremos simultáneamente las teclas `"Ctrl"` y `"D"`.



[Localización](#) | [Personal](#) | [Docencia](#) | [Investigación](#) | [Novedades](#) | [Intranet](#)
[inicio](#) | [mapa del web](#) | [contacta](#)

Last Revision: 02/20/2014 16:58:11