

# Clases interesantes

Luis Fernando Llana Díaz

Departamento de Sistemas Informáticos y Computación  
Universidad Complutense de Madrid

24 de abril de 2007

# Clases interesantes I

## Paquete `java.util`

**Collections** Clase con métodos estáticos para manejar colecciones.

**Date** Para manejar instantes en el tiempo (UTC).

**GregorianCalendar** Para manejar fechas.

**Locale** Para ayudar a la *localización* de las aplicaciones.

**Random** Generación de números aleatorios.

# Clases interesantes II

Paquete `java.lang`

Clases envoltorio `Integer`, `Double`, etc Clases de objetos *inmutables* para los tipos básicos.

`StringBuffer` Cadenas de caracteres de objetos *no inmutables*.

# Expresiones regulares I

## Comprobación de cadenas de caracteres

- Un número de factura es correcto si es de la forma 2345/07.
- Las facturas no se borran, si se borra una la marcamos *poniendo una R* delante.
- A veces interesa construir una factura a partir de unas existentes. Si tengo los números 0967/06, 0124/07, 0345/07, construimos la auxiliar AUX-0967/06-0124/07-0345/07.

Hacer una función que compruebe si una factura es correcta

# Expresiones regulares II

## Sustituciones genéricas

Tenemos un fichero de texto en el que hay muchas fechas con formato mm/dd/aaaa. Las queremos cambiar al formato aaaa/mm/dd.

# Programa de prueba

```
private static void prueba(String patron,
                           String s) {
    Pattern p = Pattern.compile(patron);
    Matcher m = p.matcher(s);

    System.out.format("\nPatron:%s\nCadena:%s\n", patron, s);

    boolean enc = false;
    while (m.find()) {
        enc=true;
        System.out.format("Encontrado %s desde la posición %d "+
                           "hasta la posición %d\n",
                           m.group(),m.start(),m.end());
        for (int i=0; i<=m.groupCount(); i++) {
            System.out.format("Grupo %d:%s: desde la posición %d "+
                               "hasta la posición %d\n",
                               i,m.group(i),m.start(i),m.end(i));
        }
    }
    if (!enc) {
        System.out.println("No encontrado");
    }
}
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23

# Cadena de caracteres

## Una cadena de caracteres casa consigo misma

```
Patron:foo:
Cadena:foofoofoo:
Encontrado foo desde la posición 0 hasta la posición 3
Grupo 0:foo: desde la posición 0 hasta la posición 3
Encontrado foo desde la posición 3 hasta la posición 6
Grupo 0:foo: desde la posición 3 hasta la posición 6
Encontrado foo desde la posición 6 hasta la posición 9
Grupo 0:foo: desde la posición 6 hasta la posición 9
```

1  
2  
3  
4  
5  
6  
7  
8

# Caracter simple

El carácter `.` casa con cualquier caracter. Es un *meta-carácter* (no significan ellos mismos)

```
Patron:gato.:
Cadena:gatos:
Encontrado gatos desde la posición 0 hasta la posición 5
Grupo 0:gatos: desde la posición 0 hasta la posición 5
```

1  
2  
3  
4

Los meta-caracteres son:  $([\{\backslash\sim\$\}])?*\+$ . Si queremos hacer casar un meta-carácter hay que precederlo con `\`.

```
prueba("gato\\.","gato.");

Patron:gato\\.:
Cadena:gato.:
Encontrado gato. desde la posición 0 hasta la posición 5
Grupo 0:gato.: desde la posición 0 hasta la posición 5
```

1  
2  
3  
4  
5  
6

# Caracteres

<code>[abc]</code>	casa con a, b y c.
<code>[a-m]</code>	casa con las letras desde la a hasta la m
<code>[a-zA-M]</code>	casa con las letras desde la a hasta la m o desde la A hasta la M
<code>[^abc]</code>	Casa con cualquier letra excepto con a, b y c.
<code>[a-z&amp;&amp;[^be]]</code>	casa con las letras desde la a hasta la z excepto con b y e
<code>[a-z&amp;&amp;[^b-e]]</code>	casa con las letras desde la a hasta la z excepto desde la b hasta la e

# Caracteres

```
Patron:[aeg]: 1
Cadena:abcdefghijklmnopqrstuvxyz: 2
Encontrado a desde la posición 0 hasta la posición 1 3
Grupo 0:a: desde la posición 0 hasta la posición 1 4
Encontrado e desde la posición 4 hasta la posición 5 5
Grupo 0:e: desde la posición 4 hasta la posición 5 6
Encontrado g desde la posición 6 hasta la posición 7 7
Grupo 0:g: desde la posición 6 hasta la posición 7 8
```

# Caracteres

```
Patron:[^c-v]: 1
Cadena:abcdefghijklmnopqrstuvxyz: 2
Encontrado a desde la posición 0 hasta la posición 1 3
Grupo 0:a: desde la posición 0 hasta la posición 1 4
Encontrado b desde la posición 1 hasta la posición 2 5
Grupo 0:b: desde la posición 1 hasta la posición 2 6
Encontrado x desde la posición 22 hasta la posición 23 7
Grupo 0:x: desde la posición 22 hasta la posición 23 8
Encontrado y desde la posición 23 hasta la posición 24 9
Grupo 0:y: desde la posición 23 hasta la posición 24 10
Encontrado z desde la posición 24 hasta la posición 25 11
Grupo 0:z: desde la posición 24 hasta la posición 25 12
```

# Caracteres

```
Patron:[a-z&&[^b-u]]: 1
Cadena:abcdefghijklmnopqrstuvxyz: 2
Encontrado a desde la posición 0 hasta la posición 1 3
Grupo 0:a: desde la posición 0 hasta la posición 1 4
Encontrado v desde la posición 21 hasta la posición 22 5
Grupo 0:v: desde la posición 21 hasta la posición 22 6
Encontrado x desde la posición 22 hasta la posición 23 7
Grupo 0:x: desde la posición 22 hasta la posición 23 8
Encontrado y desde la posición 23 hasta la posición 24 9
Grupo 0:y: desde la posición 23 hasta la posición 24 10
Encontrado z desde la posición 24 hasta la posición 25 11
Grupo 0:z: desde la posición 24 hasta la posición 25 12
```

# Caracteres predefinidos

- . casa con cualquier carácter, puede o no casar con final línea.

`\d` [0-9]

`\D` [^0-9]

`\s` espacio en blanco [ \t\n\x0B\f\r]

`\S` [^ \t\n\x0B\f\r]

`\w` [a-zA-Z\_0-9]

`\W` [^a-zA-Z\_0-9]

# Caracteres predefinidos

```
prueba("\\d", "aaa bbb 1 df 3 f");
```

```
Patron:\d:
```

```
Cadena:aaa bbb 1 df 3 f:
```

```
Encontrado 1 desde la posición 8 hasta la posición 9
```

```
Grupo 0:1: desde la posición 8 hasta la posición 9
```

```
Encontrado 3 desde la posición 13 hasta la posición 14
```

```
Grupo 0:3: desde la posición 13 hasta la posición 14
```

1  
2  
3  
4  
5  
6  
7  
8

# Repeticiones

Voraz	No voraz	Posesivo	
X?	X??	X?+	0 ó 1 aparición
X*	X*?	X*+	0, 1 ó más
X+	X+?	X++	1 ó más
X{n}	X{n}?	X{n}+	n apariciones
X{n,}	X{n,}?	X{n,}+	al menos n
X{n,m}	X{n,m}?	X{n,m}+	entre n y m

```
Patron:.*foo:
Cadena:xfooxxxxxfoo:
Encontrado xfooxxxxxfoo desde la posición 0 hasta la posición 13

Patron:.*?foo:
Cadena:xfooxxxxxfoo:
Encontrado xfoo desde la posición 0 hasta la posición 4
Encontrado xxxxxxfoo desde la posición 4 hasta la posición 13

Patron:.*+foo:
Cadena:xfooxxxxxfoo:
No encontrado
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12

# Agrupaciones

- (X) Agrupación con captura
- \n *n*-ésimo de captura
- (?:X) Agrupación sin capturar

```
Patron:(?:foo){3}:  
Cadena:foofoofoofoofoofoofoo:  
Encontrado foofoofoo desde la posición 0 hasta la posición 9  
Grupo 0:foofoofoo: desde la posición 0 hasta la posición 9  
Encontrado foofoofoo desde la posición 9 hasta la posición 18  
Grupo 0:foofoofoo: desde la posición 9 hasta la posición 18
```

```
Patron:(foo){3}\1:  
Cadena:foofoofoofoofoofoofoo:  
Encontrado foofoofoofoo desde la posición 0 hasta la posición 12  
Grupo 0:foofoofoofoo: desde la posición 0 hasta la posición 12  
Grupo 1:foo: desde la posición 6 hasta la posición 9
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12

# Agrupaciones

```
prueba("(\\d{1,2})/(\\d{1,2})/(\\d{4})", "6/30/1950");
```

```
Patron:(\\d{1,2})/(\\d{1,2})/(\\d{4}):
```

```
Cadena:6/30/1950:
```

```
Encontrado 6/30/1950 desde la posición 0 hasta la posición 9
```

```
Grupo 0:6/30/1950: desde la posición 0 hasta la posición 9
```

```
Grupo 1:6: desde la posición 0 hasta la posición 1
```

```
Grupo 2:30: desde la posición 2 hasta la posición 4
```

```
Grupo 3:1950: desde la posición 5 hasta la posición 9
```

1

2

3

4

5

6

7

8

9

# Facturas

```
public class PrFactura {
    public static boolean esNumeroCorrecto(String numFactura) {
        Pattern p = Pattern.compile("(AUX-\\d{4}/\\d{2}-)*?R?\\d{4}/\\d{2}");
        Matcher m = p.matcher(numFactura);
        return m.matches();
    }
    public static final void main(final String[] args) {
        String num="0001/04";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
        num="00001/04";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
        num="00001-04";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
        num="0001/2004";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
        num="0001/04";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
        num="R0001/04";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
        num="AUX-0001/04-0001/04";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
        num="AUX-0001/04-0001/04-0001/04-0001/04-0001/04";
        System.out.println(": "+num+": "+esNumeroCorrecto(num));
    }
}
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25

# Fechas

```
public class PrFecha {  
    private static String convierteFecha(String fecha) {  
        String s1 = fecha.replaceAll("(\\d{1,2})/(\\d{1,2})/(\\d{4})",  
                                     "$3/$1/$2");  
        return s1;  
    }  
    public static void main(String [] args) {  
        String fecha = "6/30/1950";  
        System.out.println(convierteFecha(fecha));  
    }  
}
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11