

# Lenguaje de programación I



# Caso



- ✓ La empresa Ciberfarma, desarrolló la siguiente aplicación para el mantenimiento de sus productos, ¿Cómo validarías que los datos ingresados son correctos?

A screenshot of a web application window titled 'Mantenimiento de Productos'. The window has a light gray background and a standard Windows-style title bar with minimize, maximize, and close buttons. The form contains several input fields and buttons. The 'Código' field contains 'P0001'. The 'Producto' field contains 'Panadol'. The 'Tipo' field is a dropdown menu with 'Jarabe' selected. The 'Cantidad' field contains '15' and the 'Precio' field contains '1,50'. There are three buttons on the right: 'Nuevo', 'Guardar', and 'Editar'. The 'Guardar' button is highlighted with a blue border.

Mantenimiento de Productos

Código:

Producto:

Tipo:

Cantidad:  Precio:

# Contenido

- Gestión de Excepciones
- ✓ Expresiones Regulares
- Ejercicios de Aplicación

## Logros de la Unidad

- Al término de la unidad, el alumno crea aplicaciones que validan el ingreso de datos, mediante la gestión de diferentes tipos de excepciones, así como el manejo de expresiones regulares.

# Excepciones

- ✓ Se define Excepción a un **ocurrencia o situación anómala** que **altera el flujo normal de un programa**.
  - Ej. Convertir un valor numérico no válido, dividir entre 0, acceder a un índice inválido, abrir un archivo que no existe, interrupción en la comunicación, guardar en una ubicación no válida.

Cantidad:  Precio:

```
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "1,50"  
    at sun.misc.FloatingDecimal.readJavaFormatString(Unknown Source)  
  
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: " 5"  
    at java.lang.NumberFormatException.forInputString(Unknown Source)
```

- ✓ En memoria, **se crea un objeto** que almacena información acerca del tipo de excepción que se ha producido y el lugar donde ha ocurrido.

# Excepciones

✓ Existe una gran variedad de Excepciones (consulta el manual). Ej.

Excepción	Descripción lanzada
Exception	Excepción general
NumberFormatException	Presente al convertir un valor a número.
ArithmeticException	Excepciones en operaciones aritméticas
IOException	Al presentarse excepciones al abrir, guardar, grabar o leer archivos
NullPointerException	Cuando intentamos acceder a un objeto con una variable de referencia cuyo valor actual es null.
ArrayIndexOutOfBoundsException	Al acceder a un arreglo con un valor de índice inválido (negativo o superior a la longitud del arreglo).
IllegalArgumentException	Lanzada cuando un método recibe un argumento formateado de manera diferente a lo que el método esperaba.

✓ Un mecanismo para solucionarlo es la **gestión de excepciones**

- Utiliza los bloques **try – catch - finally**:

- ✓ El bloque **try** o **prueba** es usado para agrupar el **código susceptible de una operación riesgosa**.

- ✓ La gestión se realiza mediante uno o más bloques **catch** (más genérica al final), donde **se realizan las tareas de control o solución de la excepción**.

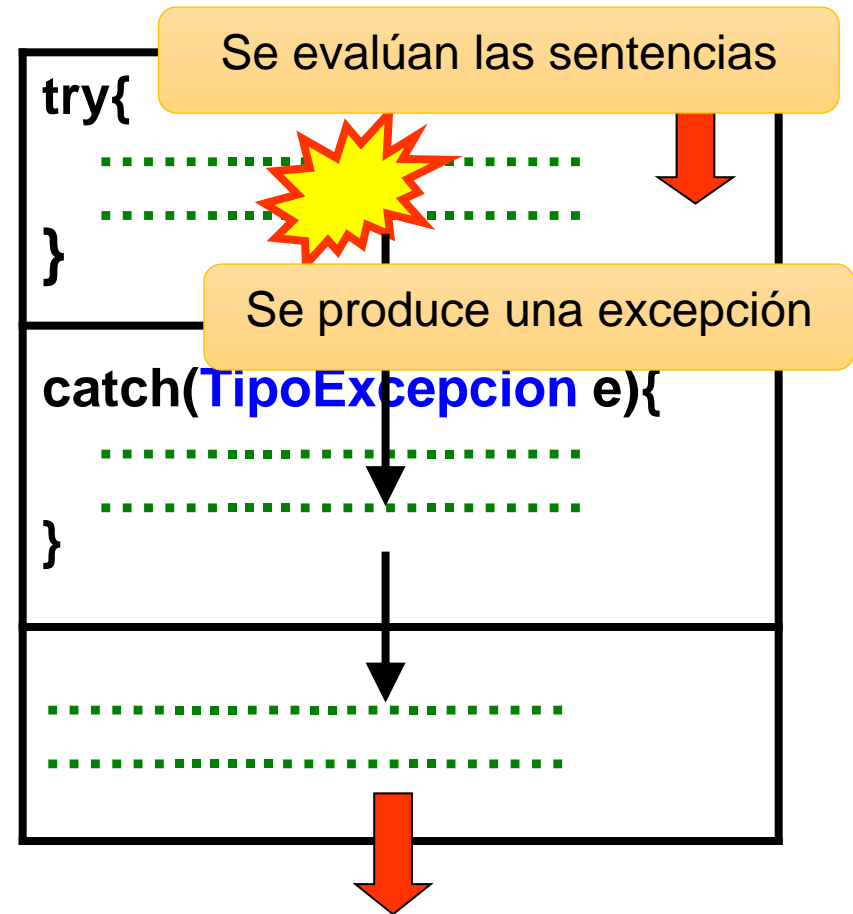
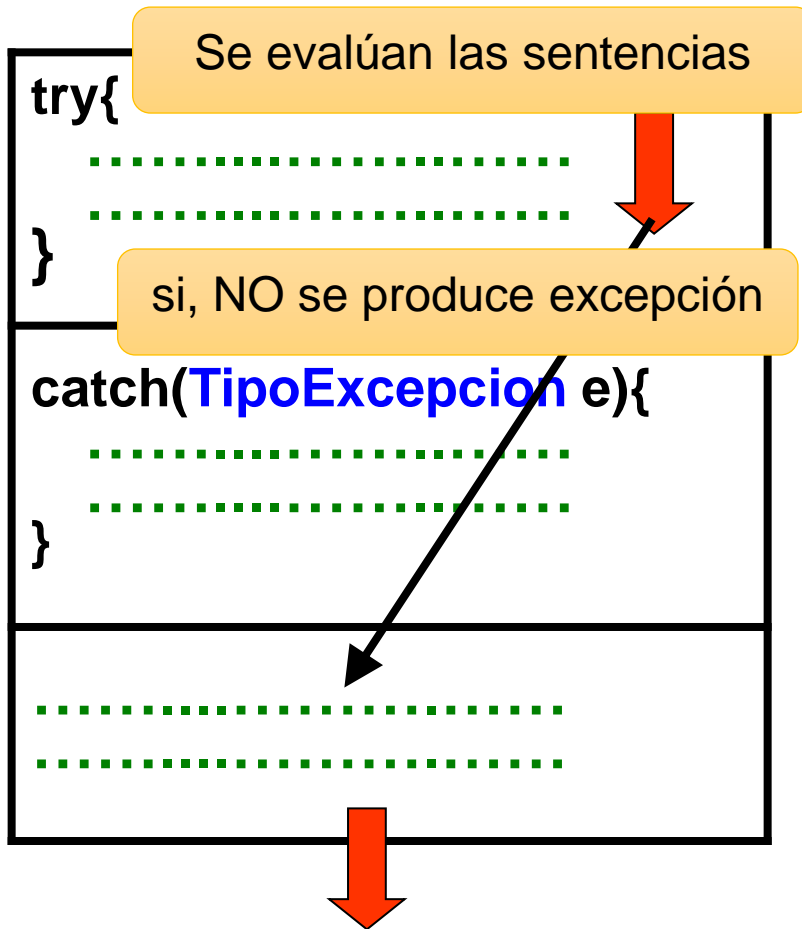
//Bloque no riesgoso

```
try{  
    // código a evaluar  
} catch(Excepcion1 variable){  
    // código de control de la excepción  
} catch(Excepcion2 variable){  
    // código de control de la excepción  
} finally {  
    // código que se realiza siempre  
}
```

//Bloque no riesgoso

- El bloque (opcional) **finally**, define instrucciones a realizar independientemente de que **ocurra o no una excepción** dentro del bloque try.
- Sirve generalmente para liberar recursos afectados en un proceso.
  - Ej. Cerrar bases de datos, archivos de texto

# Explicación



# Explicación

```
try{  
    //sentencias  
}  
  
finally {  
    //Clean up  
}
```

```
try{  
    //sentencias  
}  
  
catch(Exception e){  
    //Gestión de la excepción  
}  
  
finally {  
    //Clean up  
}
```

*Apropiado para asignar valores a objetos que deben ser inicializados ocurra o no una excepción.*





# Ejercicio de seguimiento

Analiza el resultado de las siguientes instrucciones

```
System.out.println(1);  
try {  
    System.out.println(2);  
    int x = 5 / 0;  
    System.out.println(x);  
    System.out.println(3);  
} catch (Exception e) {  
    System.out.println(4);  
}  
System.out.println(5);
```

```
System.out.println(1);  
try {  
    System.out.println(2);  
    int x = 5 / 2;  
    System.out.println(x);  
    System.out.println(3);  
} catch (Exception e) {  
    System.out.println(4);  
} finally {  
    System.out.println(5);  
}  
System.out.println(6);
```



# Aplicación

- ✓ Realiza las validaciones necesarias empleando bloques try..catch, mostrando los mensajes respectivos en caso de error

Mantenimiento de Productos

Código: P0001

Producto: Panadol

Tipo: Jarabe

Cantidad: 15

Precio: 1,50

Nuevo

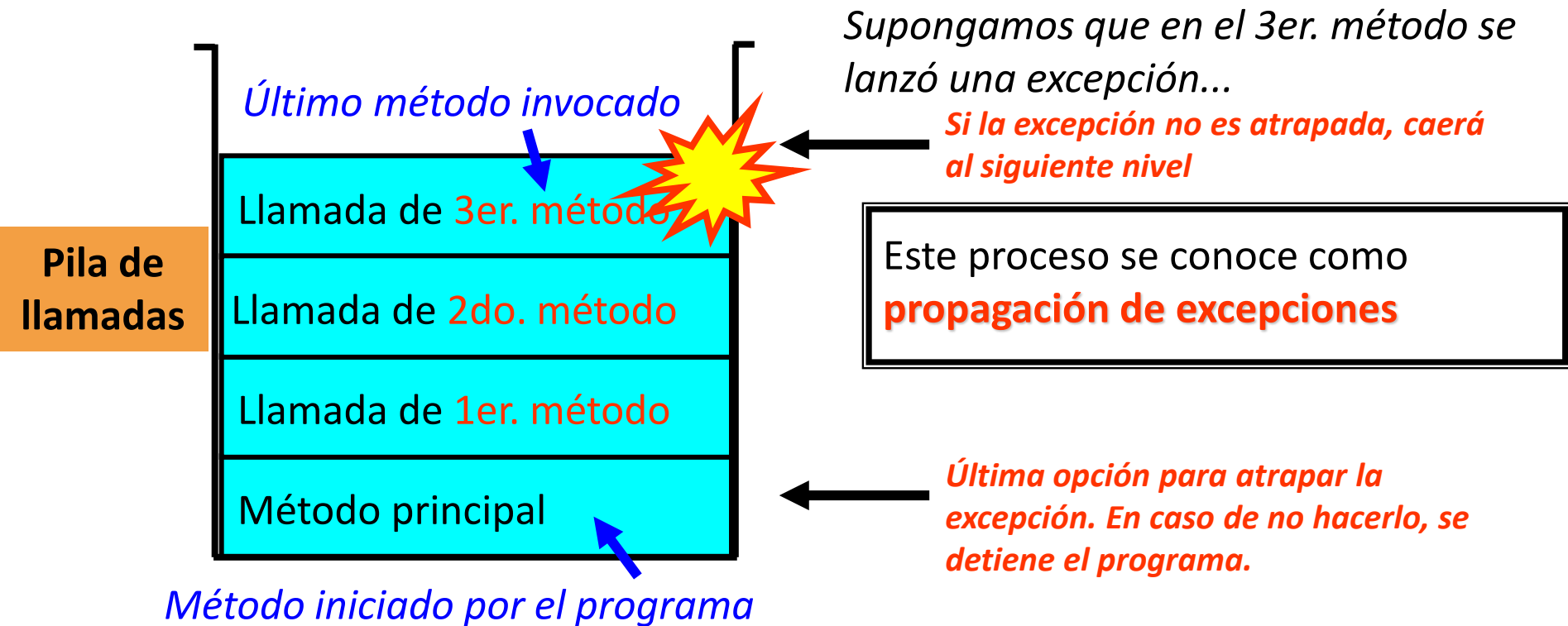
Guardar

Editar

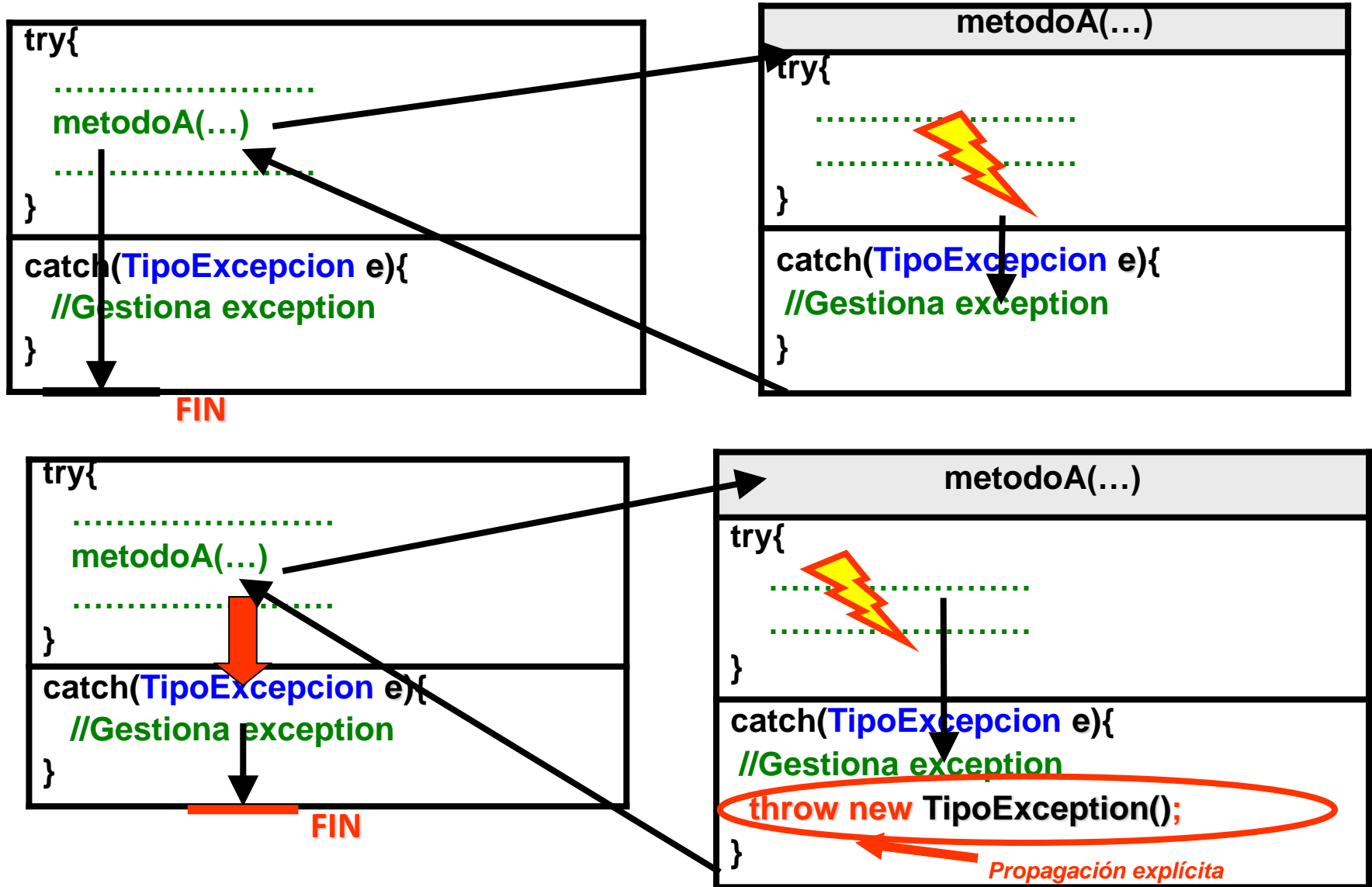
- ✓ Analiza los resultados de los siguientes códigos...

# Propagación de excepciones

- ✓ Se puede **propagar** (lanzar) una excepción sin necesidad de capturarla, dejando que sean otras partes del programa las encargadas de definir acciones.



# Propagación de excepciones



- ✓ Para declarar el tipo de Excepción que se lanzará en caso de "error" se emplea:

```
nombremétodo() throws tipoExcepcion {  
  
}
```

- ✓ Java, también permite "lanzar" explícitamente excepciones, a fin de poder capturarlos, para ello se emplea la sentencia:

**throw** new tipoExcepcion();



# Ejercicio de seguimiento

Analiza el resultado de las siguientes instrucciones

```
package vista;

public class Ejemplo01 {


    public static void main(String[] args) {
        try {
            new Ejemplo01().ingreso();
        } catch (Exception e) {
            System.out.println("Capturado en Principal : " + e);
        }
    }

    void ingreso() {
        String cod = "", nom;
        nom = obtener(cod);
        System.out.println(nom);
    }

    String obtener(String key) {
        if (key == null) {
            throw new NullPointerException("Argumento null");
        }
        if (key.trim().length() == 0) {
            throw new IllegalArgumentException("Argumento Vacío");
        }
        return "José";
    }
}
```

# Caso

- ✓ La empresa Ciberfarma, necesita registrar los datos de sus clientes, validando los datos ingresados



A screenshot of a web application window titled "Mantenimiento de Clientes". The window contains a form with the following fields and labels: "Código:", "DNI:", "Nombre:", "Apellido:", "Teléfono:", and "Correo:". Each label is followed by a text input field. To the right of the form, there are two buttons: "Nuevo" (top) and "Guardar" (bottom). The window has a standard title bar with a minimize button, a maximize button, and a close button.

- ✓ ¿Cómo validar el ingreso de datos correctos?

# Expresiones Regulares

- ✓ Son una **secuencia de caracteres y símbolos** que definen un conjunto de cadenas.

**`([0-9]{3})?[0-9]{3}-[0-9]{4}`**

- ✓ Son útiles para validar la entrada y asegurar que los datos estén en un formato específico. Por ejemplo un código postal debe coincidir de 5 dígitos, un apellido debe contener letras, un número telefónico debe tener solo números, etc.



[ ]	Elige de un conjunto: [abc]	{inicio, fin } {veces}	Especifica la cantidad de veces que se repite
-	Especifica rangos: [a-f]	\d	Un dígito: [0-9]
*	Comprueba 0 a más	\D	No dígito: [^0-9]
.	Cualquier caracter	\s	Carácter espacio
+	1 o más veces	\S	Menos carácter espacio: [^\s]
^	dentro de los corchetes es un NOT, fuera indica el 1º caracter	\w	Un carácter alfanumérico: [a-zA-Z_0-9]
\	Representa caracteres especiales	\W	No alfanumérico: [^\w]

<http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html>



# Utilizando Regular Expressions

- Método 1
- Usando las clases abstractas: **Pattern** y **Matcher** (`javax.util.regex.*`)
  - **Pattern**, Define la expresión (patrones) de búsqueda, a través del método **compile**.
  - **Matcher**, Define la **fuentes**, a partir de la cual se realiza la búsqueda, además proporciona métodos para buscar y devolver el resultado. usa el método **matcher** referenciado del objeto Pattern para especificar la fuente.

```
Pattern p = Pattern.compile("[0-9]");  
Matcher m = p.matcher("132");  
System.out.println(m.matches());
```

- Método 2
- Usando el método **matches** de la cadena

```
System.out.println("132".matches("[0-9]"));
```



# Actividad

- ✓ Ingresar un texto y validar si corresponde a un DNI
- ✓ El apellido con letras en mayúsculas y como máximo 30 caracteres
- ✓ El precio con sólo dos decimales .
- ✓ Valida el ingreso de un código que tiene la siguiente forma:  
i12345678
- ✓ Valida el ingreso de un teléfono fijo (Ej. 555-5555)
- ✓ ¿Validar un correo electrónico?

# Aplicación



✓ Validan los datos usando expresiones regulares

A screenshot of a software application window titled 'Mantenimiento de Clientes'. The window has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. Inside the window, on the left side, there are six text input fields with labels: 'Código:', 'DNI:', 'Nombre:', 'Apellido:', 'Teléfono:', and 'Correo:'. To the right of these fields are two buttons: 'Nuevo' and 'Guardar'. The 'Nuevo' button is positioned above the 'Guardar' button. The background of the window is light gray.

# Referencias

✓ <http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html>

# GRACIAS



## **SEDE MIRAFLORES**

Calle Díez Canseco Cdra 2 / Pasaje Tello  
Miraflores – Lima  
Teléfono: 633-5555

## **SEDE INDEPENDENCIA**

Av. Carlos Izaguirre 233  
Independencia – Lima  
Teléfono: 633-5555

## **SEDE BREÑA**

Av. Brasil 714 – 792  
(CC La Rambla – Piso 3)  
Breña – Lima  
Teléfono: 633-5555

## **SEDE TRUJILLO**

Calle Borgoño 361  
Trujillo  
Teléfono: (044) 60-2000

## **SEDE SAN JUAN DE LURIGANCHO**

Av. Próceres de la Independencia 3023-3043  
San Juan de Lurigancho – Lima  
Teléfono: 633-5555

## **SEDE SAN MIGUEL**

Av. Federico Gallese 847  
San Miguel – Lima  
Teléfono: 632-4900

## **SEDE BELLAVISTA**

Av. Mariscal Oscar R. Benvides 3866 – 4070  
(CC Mall Aventura Plaza)  
Bellavista – Callao  
Teléfono: 633-5555

## **SEDE AREQUIPA**

Av. Porongoche 500  
(CC Mall Aventura Plaza)  
Paucarpata - Arequipa  
Teléfono: (054) 60-3535