

# Memoria practicas 3 y 4

## Marco teórico

**Consideraciones semánticas:** ejemplo: un profesor puede impartir una o muchas asignaturas el profesor al impartir muchas asignaturas se pone un asterisco y ahí pararía, pero con una nota se apunta también que la asignatura puede ser impartida por uno o muchos profesores.

El rombo tiene que estar compuesto de y es una agregación oscura es una agregación y reacciona total rombo clarito, Ej: la clase automóvil se le agrega otra clase llamada rueda.

**Asociación ternaria:** debe de ser evitable y es una relación 3 clases es un mismo punto, obliga a que se tengan que dar los 3 objetos a la vez.

Responde a las siguientes cuestiones: ¿Qué es **UML**? ¿Qué son los **diagramas de estructura**? ¿Y los de **comportamiento**? ¿Y los de **iteración**?

Busca un ejemplo en internet de diagrama de clases, diagrama de casos de uso y diagrama de secuencia. Adjúntalo.

Que es una clase y que tipos de diagramas existen ¿Que es UML?

El **Lenguaje Unificado de Modelado (UML)** es un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. **UML** tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación.

**UML** no es un lenguaje de programación es un lenguaje de modelado, es para crear diseños lógicos, **UML** es el lenguaje actual más usado para el modelado de software, se construye de manera lógica no física.

¿Que son los **diagramas de casos de uso**?

Los **diagramas de casos de uso** sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los **actores** y los **casos de uso** en un sistema.

¿Que son los **diagramas de comportamiento**?

Los **diagramas de comportamiento**: se encargan de especificar o de detallar la estructura de los elementos del sistema, se llaman diagramas estructurales. No indican el **movimiento de datos** y se les llama diagramas de **representación estática**.

Como se va comportando el sistema, indican el movimiento de los datos por el sistema llamados diagramas de comportamiento y se les llama también diagramas dinámicos

**Diagramas de iteración**: cómo interactúan los diagramas entre ellos.

Diagrama de clase

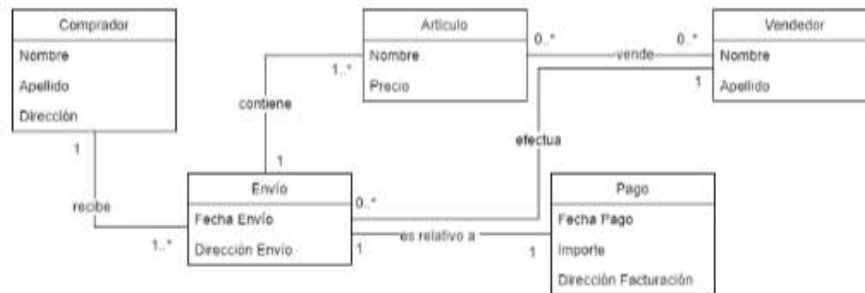
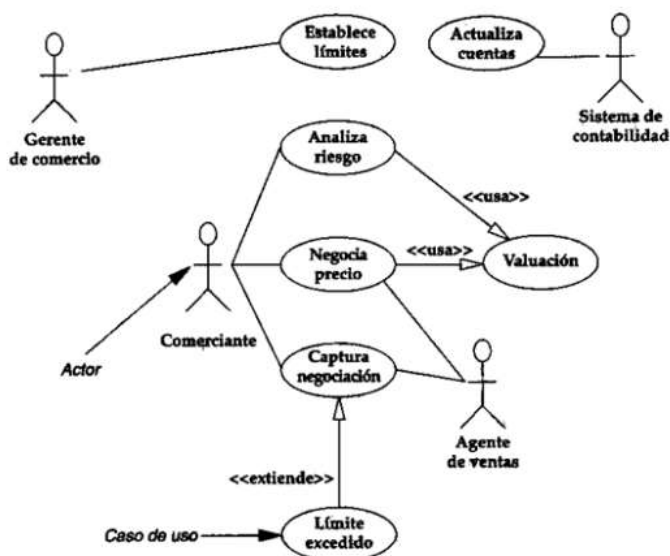


Diagrama de Casos de uso



La asociación entre dos clases: cardinalidades (muchas) y se escribe **1:\***, con un 1 en cuanto se instancia es obligatorio decir que se está instanciando y 0 que se instancia en el tiempo

**Cardinalidad mínima: 1**

Navegabilidad: **sentido unidireccional(1:\*) o bidireccional.**

### ¿Qué son los diagramas de clase?

Es un tipo de **diagrama estático de estructuras** que describe la estructura del sistema mostrando sus clases y las asociaciones entre ellas. Está compuesto por:

**Clases:** definición y representación. Los atributos y su visibilidad. Los métodos y su visibilidad.

Una clase es una unidad básica que encapsula toda la información de un objeto (instancia de una clase). Un atributo representa alguna propiedad de la clase que se encuentra en todas las instancias de la clase. Se puede representar con su tipo e incluso con su valor por defecto.

Pueden ser **públicos (+)**, **privados (-)**, **protegidos (#)** o **empaquetados ()**.

**Representar la clase de producto:** Tiene un código, una descripción, un precio de venta, una cantidad en el almacén y una cantidad mínima de pedido. Se hace necesario calcular el stock, el precio venta público, conocer el código y la descripción del producto.

**Relaciones, sus asociaciones y su cardinalidad. Navegabilidad.**



## Práctica 31. Ejercicios de depuración

### Ejercicio 1




Para encontrar los errores de este ejercicio con el Debugger, he puesto una serie de breakpoints que me ayudaran a saber donde se encontraban los errores.

El primer breakpoint lo he puesto en el while del bucle do while y ahí se encuentra el primer error:



Al iniciar el programa y meter un numero veo como la variable num funciona y como la expresión también funciona

Name	Value
 "num >= 2"	true
 Add new expression	



  

Variables	
Name	Value
 args	String[0] (id=16)
>  sn	Scanner (id=18)
 num	2

Pero cuando avanzo en el programa veo que en la expresión sale error y como la variable sum desaparece.

Expressions	
Name	Value
>  "num >= 2"	<error(s)_during_the_evaluation>
 Add new expression	

Variables	
Name	Value
 args	String[0] (id=16)
>  sn	Scanner (id=17)

En el while esta expresión, `!(num<=1)` significa que cuando esto se cumpla el valor es igual a false por lo tanto da fallo. La solución a este fallo esta en cambiar esta condición del while por la siguiente, `!(num>1)`:

```

- }while(!(num<=1)); //condición para salir
+ }while(!(num>1)); //condición para salir

```

Al corregir este he podido ver como con el debugger sigue avanzando con el programa y como entra en el bucle for y va sumando hasta conseguir la suma:

Name	Value
$x+y$ "num >= 2"	true
Add new expression	

Variables	
Name	Value
args	String[0] (id=16)
> sn	Scanner (id=18)
num	3
suma	3

El segundo error lo he encontrado con el debugger asignando un breakpoint en el bucle for, la condición que tiene dentro está mal ya que antes de que termine de contar se va a salir ya que el contador es menor que num.

Variables	
Name	Value
args	String[0] (id=16)
> sn	Scanner (id=18)
num	2
suma	1

Para solucionar este problema he cambiado la condición de la siguiente manera:

```

//Realizamos la suma
- for(int contador=1;contador<num;contador++){
+ for(int contador=1;contador<=num;contador++){

```

Así cuando la suma del contador sea menor o igual que num podrá salir correctamente y mostrar bien la suma.

Name	Value
args	String[0] (id=16)
> sn	Scanner (id=18)
num	2
suma	3

```
Inserta un numero entero mayor que 1
```

```
2
```

```
La suma es: 3
```

## Ejercicio 2

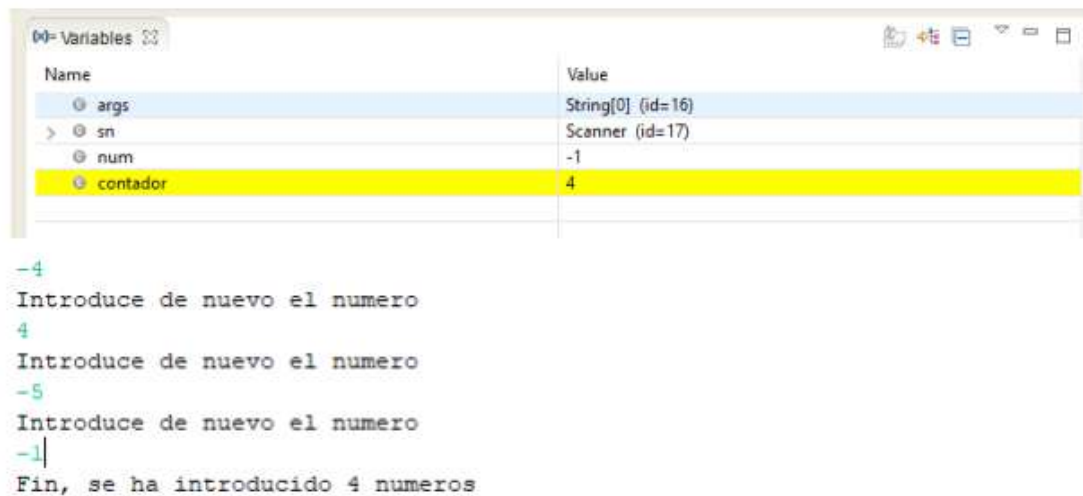
Para encontrar el fallo en este ejercicio he puesto un breakpoint en bucle while y mientras se va ejecutando me he dado cuenta de que contador está declarado, pero al ejecutar el bucle while no está contando las veces que lo hace:

Name	Value
args	String[0] (id=16)
> sn	Scanner (id=18)
num	-1
contador	0

Para solucionar este problema dentro del bucle while he introducido la siguiente línea:

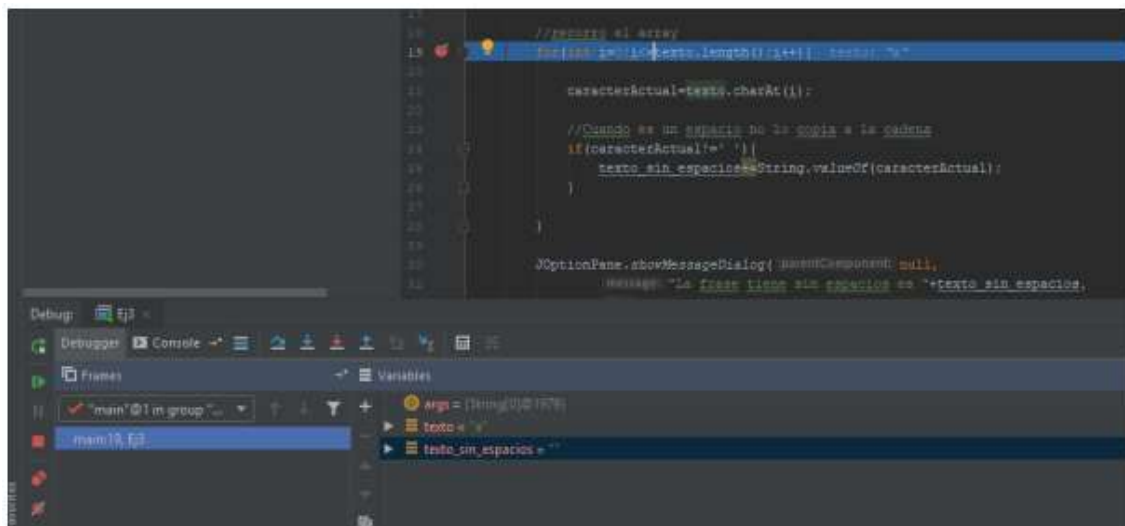
**contador ++;**

Entonces cada vez que se ejecute el bucle irá contando las veces que se ha ejecutado y cuando salga mostrará el total:



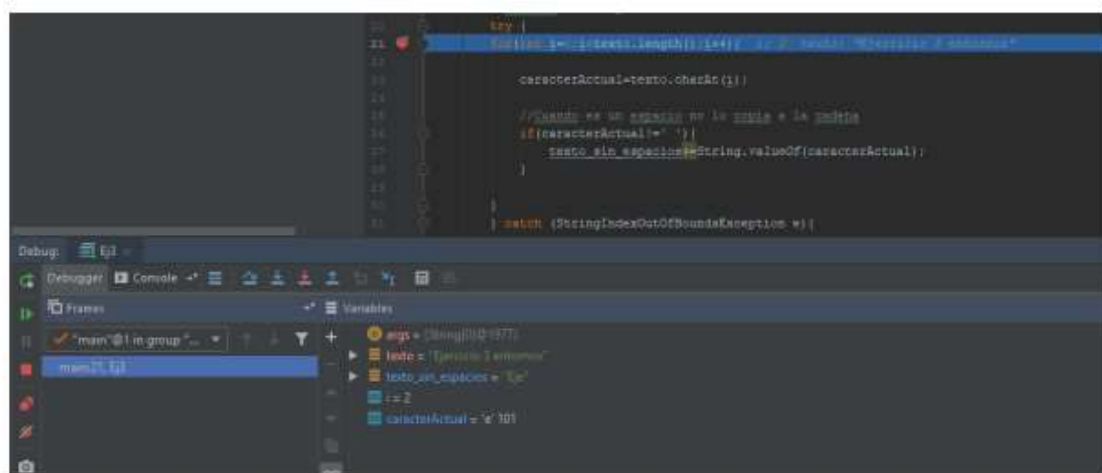
### Ejercicio 3

Para encontrar los errores en este ejercicio he puesto un breakpoint en el bucle for para ver qué pasaba cuando llegaban el valor de las variables:



Al introducir un carácter en la ventana automáticamente salta error porque la condición dice que `i=0` y que `si i<=texto`, aquí `texto` siempre va ser mayor que uno por lo tanto siempre salta error.

La solución está en cambiar el `i<=texto` por `i<texto`, así el `texto` introducido entrará y en el bucle y se completará el programa



## Ejercicio 4

Al depurar este ejercicio me he dado cuenta de que cadenaResultante en vez de estar concatenando está igualándose con texto:



Dando como resultado solo la última cadena introducida:





Para solucionarlo en donde se concatena el texto he puesto un + y un = para que aparte de igualarlo se añada a la cadena:



Ahora conforme voy añadiendo cadenas de texto se van añadiendo una detrás de otra dando como resultado el siguiente:



## Ejercicio 5

El principal error detectado en este ejercicio es que hay dos String con palabra1 y más abajo en el if hay una comparación entre palabra1 y palabra2, por lo que al segundo palabra1 le he cambiado el nombre a palabra2:

Código Original	Código Modificado
<pre>System.out.println("Escribe la palabra 1"); String palabra1=sn.next(); System.out.println("Escribe la palabra2"); String palabra2=sn.next();</pre>	<pre>System.out.println("Escribe la palabra 1"); String palabra1=sn.next(); System.out.println("Escribe la palabra2"); <b>String palabra2=sn.next();</b></pre>

Después de cambiar eso el programa se ejecuta y las variables palabra1 y palabra2 cogen sus datos y se comparan en el if dando un resultado:

Name	Value
args	String[0] (id=16)
sn	Scanner (id=19)
palabra1	"ejercicio5" (id=24)
palabra2	"Ejercicio5" (id=28)

```

Escribe la palabra 1
ejercicio5
Escribe la palabra 2
Ejercicio5
Las palabras son iguales
  
```

## Ejercicio 6

El primer error que he encontrado en este ejercicio con el debugger es en el bucle for en la condición y es que i llega hasta 12 cuando solo debería de llegar a 11.

Name	Value
args	String[0] (id=16)
sn	Scanner (id=18)
meses	Mes[12] (id=24)
dias	28
i	12

Lo que he hecho ha sido cambiar la condición a `i < meses.length` para que solo llegue hasta once:

Name	Value
args	String[0] (id=16)
sn	Scanner (id=18)
meses	Mes[12] (id=24)
dias	28
i	11

```

for (int i = 0; i < meses.length; i++) {
    if (meses[i].getNumDias() == dias + 1) {
        System.out.println(meses[i].toString());
    }
}
  
```

El segundo fallo que he encontrado es que en la condición que está dentro del bucle, días se le está sumando + 1 entonces cuando intento poner un numero por lo que si introduzco estaría igualando numDias con 29, entonces lo que he hecho ha quitar ese +1 de la condición if:

Variables	
Name	Value
args	String[0] (id=16)
sn	Scanner (id=17)
meses	Mes[12] (id=24)
días	28
i	11

```
Escribe un numero de días
28
El mes febrero es el mes 2 y tiene 29 días
```

Por último, en la clase Mes en el método getNumDias he quitado los ++ del return para que cuando coja en numDias no le sume uno y no muestre un día de más:

Código Original	Código Modificado
for(int i=0;i<=meses.length;i++)	for(int i=0;i<meses.length;i++)
if(meses[i].getNumDias()==dias+1)	if(meses[i].getNumDias()==dias)

## Ejercicio 7

Algunos de los errores encontrados en este ejercicio han sido solucionados antes de usar el debugger ya no puedes ni arrancar el programa si no los solucionas:

El primer error encontrado es que el scanner no está declarado, aparte de eso la línea que llama a ese scanner también está mal y lo he corregido de la siguiente manera:

Código Original	Código Modificado
<pre>public class Ej7 {     public static void main(String[] args) {         int nota;          System.out.print("Introduzca una nota: ");         nota=Entrada.entero();     } }</pre>	<pre>public class Ej7 {     public static void main(String[] args) {         int nota;         Scanner Entrada = new Scanner(System.in);         System.out.print("Introduzca una nota: ");         nota= Entrada.nextInt();     } }</pre>

El segundo error encontrado es que el case final está mal declarado por lo tanto no funciona lo he cambiado por un if y ha quedado de la siguiente manera:

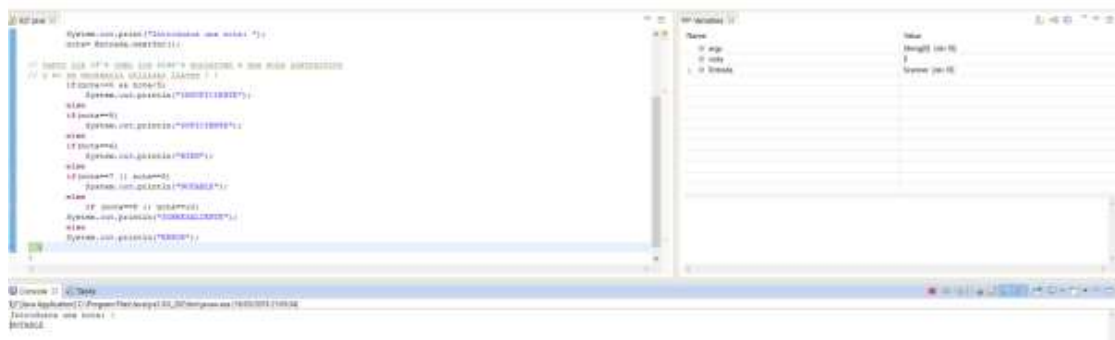
### Código Original

```
else
case 10:
System.out.println("SOBRESALIENTE");
break;
default:
System.out.println("ERROR");
break;
}
```

### Código Modificado

```
else
if(nota==7 || nota==8)
System.out.println("NOTABLE");
else
if (nota==9 || nota==10)
System.out.println("SOBRESALIENTE");
else
System.out.println("ERROR");
}
}
```

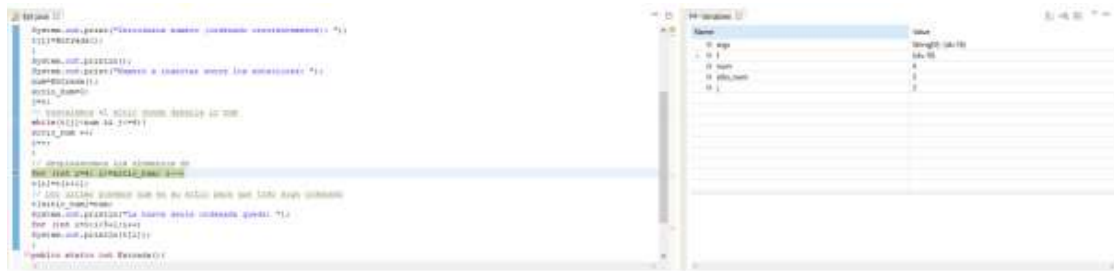
### Resultado de las variables y final del programa



## Ejercicio 8

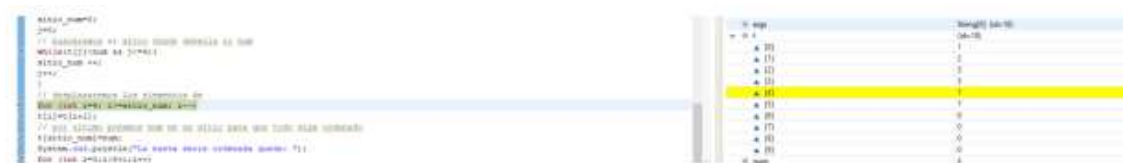
El primer error solucionado en el ejercicio es la declaración del scanner para que coja las entradas por teclado.

El segundo error solucionado en el ejercicio es que  $j=1$  es erróneo ya que si queremos poner el 4 entre 1-6 se pondría en el lugar del 5.



Igualando la variable  $j$  a 0 he conseguido que el número se coloque correctamente en su posición.

El siguiente fallo que he encontrado ha sido que en el bucle for que va justo después del while la igualación de  $t[i]=t[i+1]$  está mal. Por ejemplo, si quiero que entre 1-7 meta el 1 me lo va a meter en la posición 4 el número 7, por lo tanto, la salida final va a ser errónea.



Poniendo al revés, es decir, de la siguiente manera  $t[i+1]=t[i]$  he conseguido que, si quiero meter un 4, la primera posición que se cambia 5 y le asigna un 6:

4	6
5	6
6	0
7	0

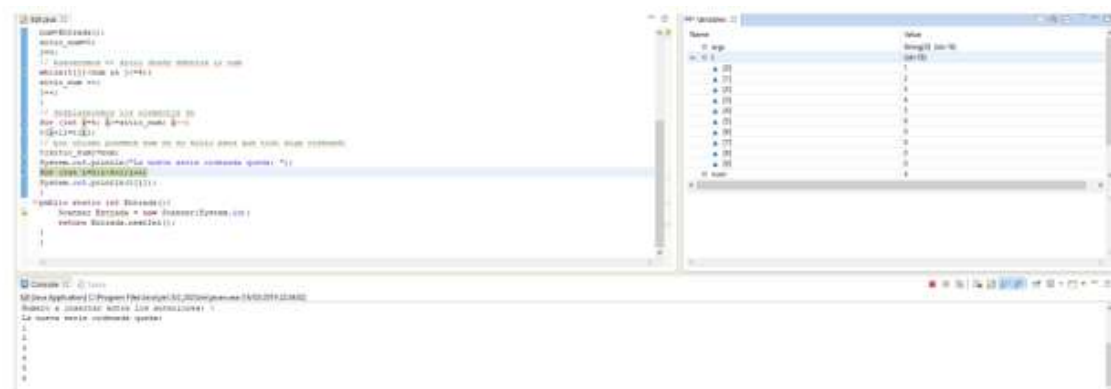
Después de asignarle a la posición 5 el 6 vuelve a la 4 para asignarle el 5:

args	String[0] (id=16)
t	(id=18)
[0]	1
[1]	2
[2]	3
[3]	5
[4]	5
[5]	6

Y la última posición que modifica es la de sitio\_num que esta igualado a num es decir que en la posición 3 va a asignarle el valor de num que es 4:

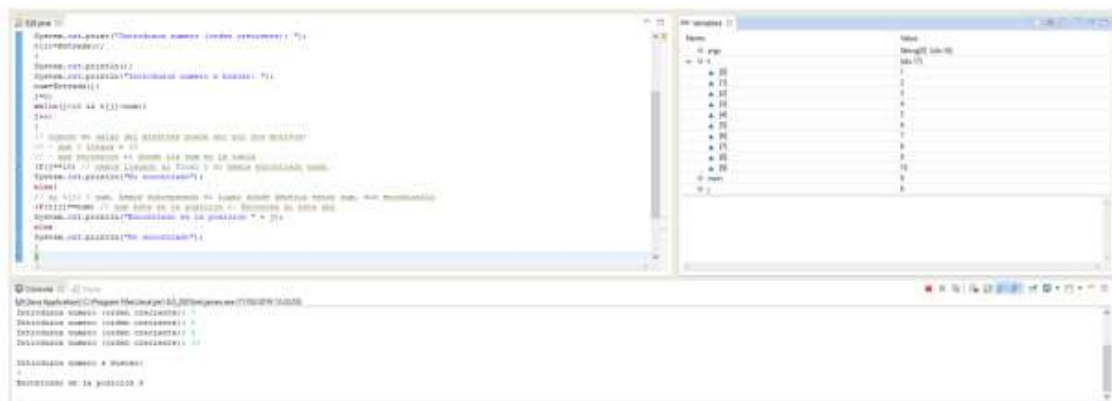
args	String[0] (id=16)
t	(id=18)
[0]	1
[1]	2
[2]	3
[3]	4
[4]	5
[5]	6

Y concluye el programa con una serie ordenada:



## Ejercicio 9

No se ha encontrado ningún error con el debugger ya que al arreglar los errores del scanner automáticamente el programa funciona perfectamente y las variables toman los valores correctos sin ningún tipo de problemas:

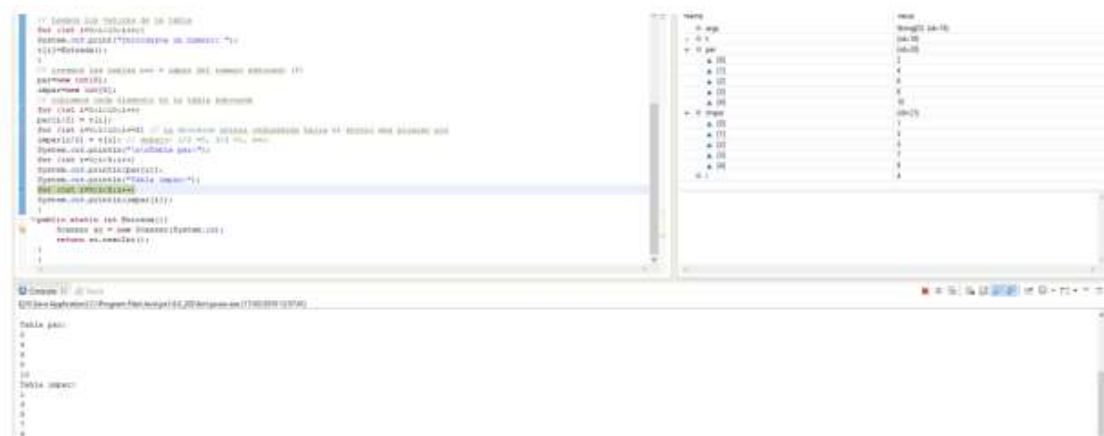


## Ejercicio 10

El primer fallo que me encuentro al debuggar el programa de este ejercicio es que el valor que cogen las variables de par e impar son incorrectas, es decir, las variables de impar están cogiendo los valores de la par y viceversa:



Entonces he igualado la i a 0 para que los números impares se metan en el bucle for impar y en el bucle for par el incremento de  $i+=2$  lo he sustituido por un  $i++$  para que el incremento sea de 1 y no de 2 así podrán salir los números pares:



# Práctica 33. Calculadora

## Clase Main

```
public class Main {

    public static void main(String[] args) {
        Calculadora calculadora = new Calculadora();
        calculadora.setVisible(true);
    }
}
```

## Clase Calculadora

```
public class Calculadora extends JFrame {

    /** Muestra el numero tecleado tecleado */
    JTextField pantalla;

    /** Guarda el resultado de la operacion
    anterior o el numero tecleado*/
    double resultado;

    /** Variable encargada de guardar la operacion
    */
    String operacion;

    /** Paneles donde se colocaran los botones */
    JPanel panelNumeros, panelOperaciones;

    /** Indica si estamos iniciando o no una
    operación */
    boolean nuevaOperacion = true;

    /**
     * Constructor. Crea los botones y componentes
     de la calculadora
     */
    public Calculadora() {
        super();
        setSize(250, 300);
        setTitle("Calculadora Simple");

        setDefaultCloseOperation(WindowConstants.EXIT_ON_C
        LOSE);

        setResizable(false);

        // Vamos a dibujar sobre el panel
        JPanel panel = (JPanel)
        this.getContentPane();
        panel.setLayout(new BorderLayout());

        pantalla = new JTextField("0", 20);
        pantalla.setBorder(new EmptyBorder(4, 4,
        4, 4));
        pantalla.setFont(new Font("Arial",
        Font.BOLD, 25));

        pantalla.setHorizontalAlignment(JTextField.RIGHT);
        pantalla.setEditable(false);
        pantalla.setBackground(Color.WHITE);
        panel.add("North", pantalla);

        panelNumeros = new JPanel();
        panelNumeros.setLayout(new GridLayout(4,
        3));
        panelNumeros.setBorder(new EmptyBorder(4,
        4, 4, 4));
        //Matriz de botones numéricos
        for (int n = 9; n >= 0; n--) {
            nuevoBotonNumerico("" + n);
        }

        nuevoBotonNumerico(".");

        //Se añaden los botones al panel central
        panel.add("Center", panelNumeros);

        panelOperaciones = new JPanel();
        panelOperaciones.setLayout(new
        GridLayout(6, 1));
        panelOperaciones.setBorder(new
        EmptyBorder(4, 4, 4, 4));

        nuevoBotonOperacion("+");
        nuevoBotonOperacion("-");
        nuevoBotonOperacion("*");
        nuevoBotonOperacion("/");
        nuevoBotonOperacion("=");
        nuevoBotonOperacion("CE");

        panel.add("East", panelOperaciones);

        validate();
    }

    /**
     * Crea los botones numericos y enlaza sus
     eventos con el listener que le corresponde
     boton a crear
     */
    private void nuevoBotonNumerico(String digito)
    {
        JButton btn = new JButton();
        btn.setText(digito);
        btn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseReleased(MouseEvent
            evt) {
                JButton btn = (JButton)
                evt.getSource();
                numeroPulsado(btn.getText());
            }
        });

        panelNumeros.add(btn);
    }

    /**
     * Crea los botones de operacion y los enlaza
     con sus eventos
     */
    private void nuevoBotonOperacion(String
    operacion) {
        JButton btn = new JButton(operacion);
        btn.setForeground(Color.RED);

        btn.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseReleased(MouseEvent
            evt) {
                JButton btn = (JButton)
                evt.getSource();
            }
        });
    }
}
```



```

        teclaOperacion(btn.getText());
    }
});
panelOperaciones.add(btn);
}

/**
 * Gestiona las pulsaciones de teclas
 * numéricas
 */
*
* Tecla pulsada
*/
private void numeroPulsado(String digito) {
    if (pantalla.getText().equals("0") ||
nuevaOperacion) {
        pantalla.setText(digito);
    } else {
        pantalla.setText(pantalla.getText() +
digito);
    }
    nuevaOperacion = false;
}

/**
 * Gestiona el gestiona las pulsaciones de
 * teclas de operación
 */
private void teclaOperacion(String tecla) {
    if (tecla.equals("=")) {
        Operar();
    } else if (tecla.equals("CE")) {
        resultado = 0;
        pantalla.setText("");
        nuevaOperacion = true;
    } else {
}
}

```

```

        operacion = tecla;
        if ((resultado > 0) &&
!nuevaOperacion) {
            Operar();
        } else {
            resultado = new
Double(pantalla.getText());
        }
    }

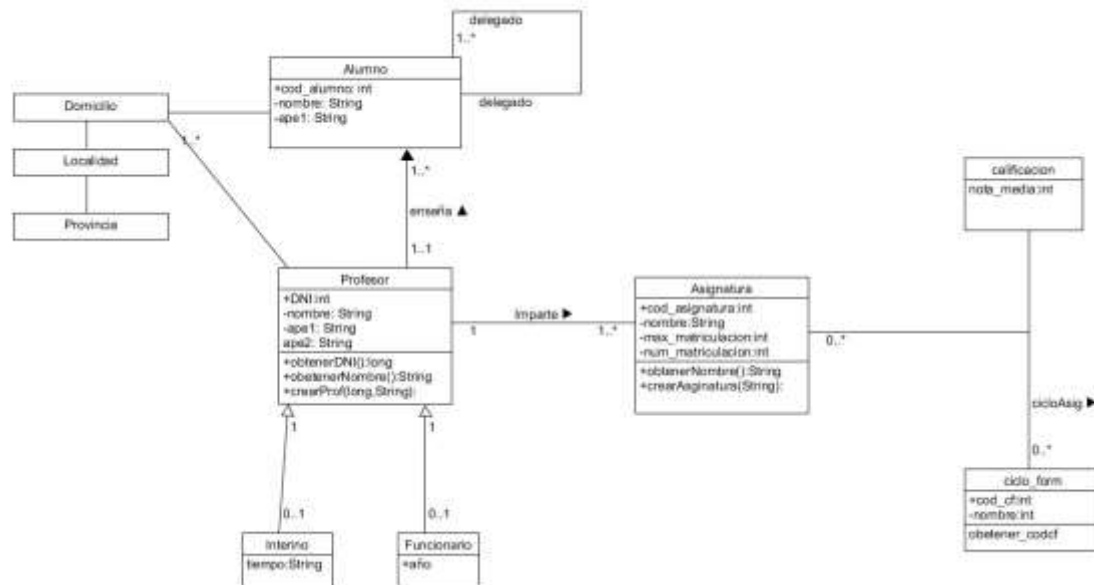
    nuevaOperacion = true;
}

/**
 * Calcula el resultado y lo muestra por
 * pantalla
 */
private void Operar() {
    if (operacion.equals("+")) {
        resultado += new
Double(pantalla.getText());
    } else if (operacion.equals("-")) {
        resultado -= new
Double(pantalla.getText());
    } else if (operacion.equals("/")) {
        resultado /= new
Double(pantalla.getText());
    } else if (operacion.equals("*")) {
        resultado *= new
Double(pantalla.getText());
    }

    pantalla.setText("" + resultado);
    operacion = "";
}
}

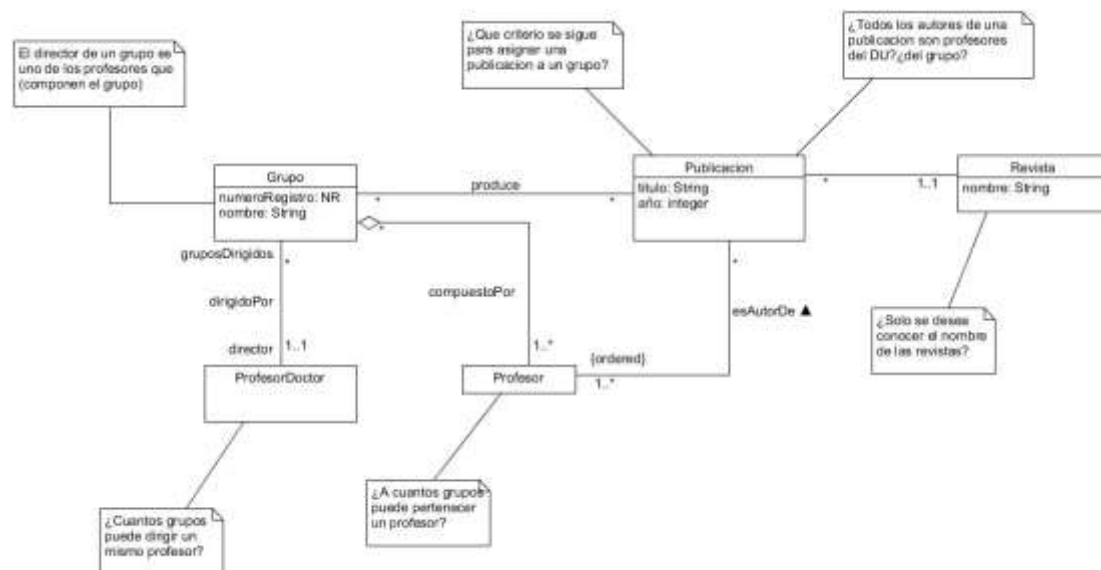
```

## Práctica 4. Diagrama de clases Centro Formación

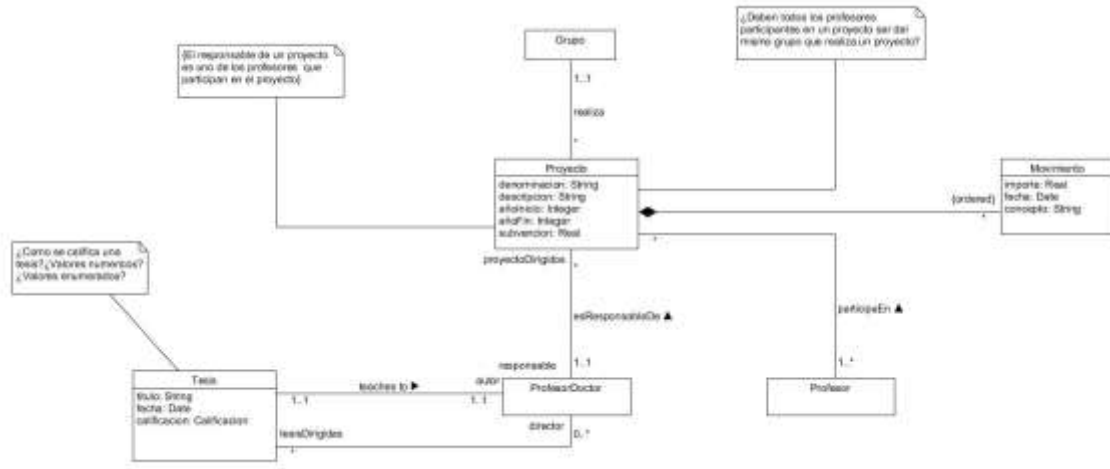


## Práctica 4.1 Diagrama de clases Departamento Universitario

RI1, RI2, RI3 y RN1



RI1, RI2, RI6, RI7 y RN1



RI1, RI4, RI5, RN1, RN2, RN3

