



Entornos de desarrollo

Act.01_UT04. Diseño y realización de pruebas

Francisco José García Cutillas | 1FPGS_DAM



Índice

Parte 1. Depuración.....3

 Ejercicio 13

 Ejercicio 210

 Ejercicio 315

Parte 2. Pruebas17

 Ejercicio 117

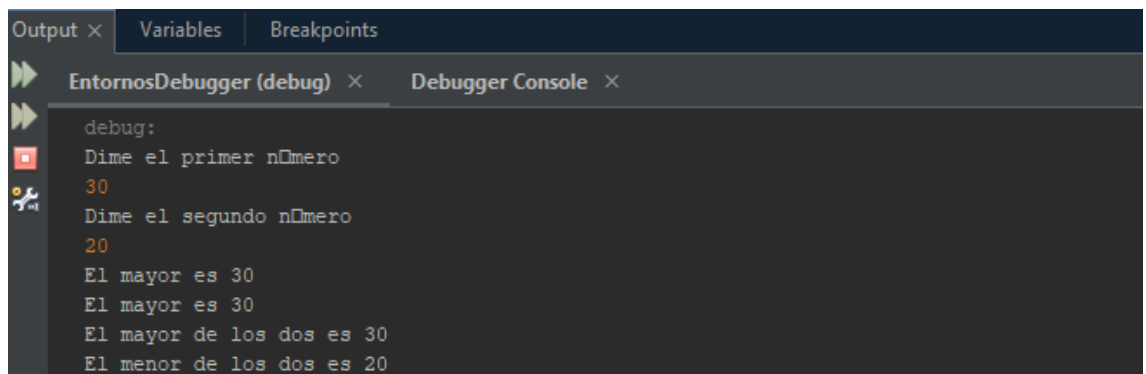
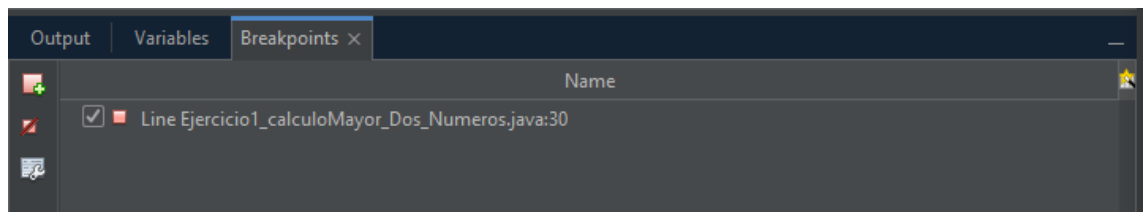
 Ejercicio 220

 Ejercicio 321

Parte 1. Depuración.

Ejercicio 1

1. Asegúrate que la clase por defecto actual es Ejercicio1_calculoMayor_Dos_Numeros. Esta clase pide dos números por consola y debería mostrar por consola por varios mensajes. Sobre esta clase realiza las siguientes cuestiones.
 - Prueba a realizar tres pruebas: Primer número es mayor que el segundo, el segundo es mayor que el primero y luego uno en el que los dos son iguales. ¿Qué conclusiones extraes de ello? ¿Crees que el código es correcto o de su resultado extraes otras conclusiones? En caso de creer que hay errores en el código realiza una depuración línea a línea para resolver el problema.
 - Caso del primer número mayor que el segundo:



En este caso, haciendo la depuración línea a línea poniendo un punto de ruptura en la línea 30 de “Ejercicio1_calculoMayor_Dos_Numeros”, observamos que nos muestra correctamente el número que es mayor y el que es menor, aunque nos muestre tres veces el que es mayor, ya que ésta condición la evalúa en tres sitios diferentes. Cosa que debemos corregir para que sólo se muestre una vez.

- **Caso del segundo número mayor que el primero.**

```

Output x Variables Breakpoints
EntornosDebugger (debug) x Debugger Console x
debug:
Dime el primer número
15
Dime el segundo número
16
El mayor es 16
Son iguales

```

```

43         if (primerNumero>b)
44         {
45             System.out.println("El mayor es "+primerNumero);
46         }
47         else
48         {
49             if (primerNumero>b)
50             {
51                 System.out.println("El mayor es "+b);
52             }
53             else
54             {
55                 System.out.println("Son iguales");
56             }
57         }
58
59
60         //Utilizando el método calcularMayor de la clase Calculo
61         int mayor;
62         mayor=Calculo.calcularMayor(primerNumero, b);
63         System.out.println("El mayor de los dos es "+mayor);

```

En este caso encontramos un error en la línea 55, ya que como no se cumple que “primerNumero” sea mayor que “b” (segundo número), pasa a ejecutar el else, dentro del cual vuelve a ejecutar otro if volviendo a comparar si “primerNumero” es mayor que “b”. El error se encuentra que a continuación hay un else en el que nos muestra el mensaje “Son iguales”. Por lo tanto, iría por este camino ya que “primerNumero” no es mayor que “b”.

Para corregir dicho error hemos modificado la evaluación de cuando los números sean iguales y la hemos dejado de la siguiente forma.

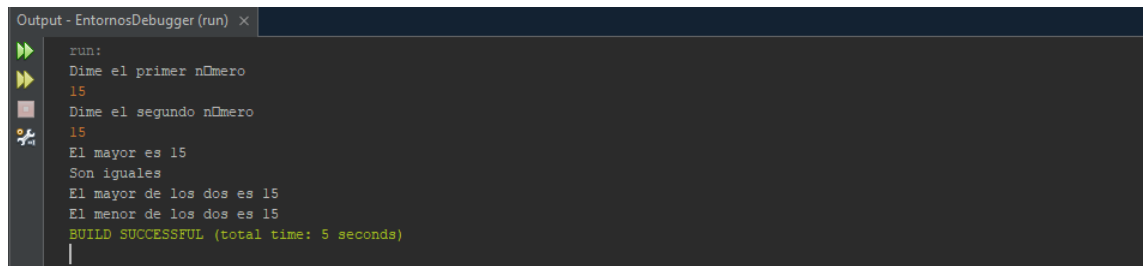
```

//Condición de que sean iguales. En este caso sólo vamos a comprobar
//si los números son iguales, ya que el mayor o menor lo realizamos con
//el método mayor y menor de la clase Calculo

if (primerNumero == b) {
    System.out.println("Son iguales");
}

```

- **Caso en el que los dos números sean iguales.**



```

Output - EntornosDebugger (run) x
run:
Dime el primer número
15
Dime el segundo número
15
El mayor es 15
Son iguales
El mayor de los dos es 15
El menor de los dos es 15
BUILD SUCCESSFUL (total time: 5 seconds)

```

En este caso nos muestra correctamente que los números son iguales. Aunque habría que corregir el código para que en este caso no nos muestre el mayor ni el menor, puesto que son iguales.

Esta ejecución se realiza antes de corregir el código del apartado anterior, el cual ha quedado de la siguiente forma.

```

public static void main(String[] args) {
    // TODO code application logic here
    Scanner teclado=new Scanner(System.in);

    int primerNumero;
    primerNumero=Calculo.introduceNumero(teclado, cadena: "primer");

    int b;
    b=Calculo.introduceNumero(teclado, cadena: "segundo");

    //Condición de que sean iguales. En este caso sólo vamos a comprobar
    //si los números son iguales, ya que el mayor o menor lo realizamos con
    //el método mayor y menor de la clase Calculo

    if (primerNumero == b) {
        System.out.println("Son iguales");
    }

    //Utilizando el método calcularMayor de la clase Calculo
    int mayor;
    mayor=Calculo.calcularMayor(x:primerNumero, y:b);
    System.out.println("El mayor de los dos es "+mayor);

    //Utilizando el método calcularMenor de la clase Calculo
    // Tiene un error
    int menor;
    menor=Calculo.calcularMenor(x:primerNumero, y:b);
    System.out.println("El menor de los dos es "+menor);

}

```

- Pon dos puntos de interrupción en la línea 25 y 40 de la clase Calculo.java y ejecute en modo depuración el programa. En la primera parada del programa

i. Muestra el estado de la pila de llamadas y explica el mismo.

```

14 public static int calcularMayor(int x, int y)
15 {
16     int my;
17     if(x>y)
18     {
19         my=x;
20     }
21     else
22     {
23         my=y;
24     }
25     return my;
26 }
27
28 //Tiene un error
29 public static int calcularMenor(int x, int y)
30 {
31     int mn;
32     if(x>y)
33     {
34         mn=y;
35     }
36     else
37     {
38         mn=x;
39     }
40     return mn;
41 }
42
43 //calcula el factorial del entero positivo n
44 // devuelve un long
45
46 public static long calcularFactorial(int n)
47 {
48     long fact=1;
49

```

Call Stack:

Name
Calculo.calcularMayor:25
Ejercicio1_calculoMayor_Dos_Numeros.main:40

En este caso, podemos observar que el programa en la clase “Ejercicio1_calculoMayor_Dos_Numeros” main en la línea 40 ha hecho una llamada a la clase “Calculo” al método “calcularMayor”, y se ha detenido en la línea 25 de dicha Clase, que es donde hemos puesto el primer punto de ruptura.

ii. Dentro de las variables de la clase Ejercicio1_calculoMayor_Dos_Numeros. ¿qué valor tienen las variables primernumero, b, mayor y menor.

Para esta comprobación vamos a crear un punto de ruptura justo en la línea de la última impresión por pantalla (línea 48), para así poder ver el valor que tienen dichas variables hasta ese momento.

The screenshot shows an IDE with a Java file named `Ejercicio1_calculoMayor_Dos_Numeros`. The code is as follows:

```

43
44 //Utilizando el método calcularMenor de la clase Calculo
45 // Tiene un error
46 int menor;
47 menor=Calculo.calcularMenor(x:primerNumero, y:b);
48 System.out.println("El menor de los dos es "+menor);
49
50
51 }
52
53
54 }
55

```

Below the code editor, the 'Variables' window is open, showing the state of variables at the current execution point. The variables and their values are:

Name	Type	Value
Static		...
args	String[]	... #402(length=0)
teclado	Scanner	... #403
primerNumero	int	... 20
b	int	... 19
mayor	int	... 20
menor	int	... 19

Para esta comprobación se ha metido como primer número el 20 y como segundo el 19.

Podemos observar que justo antes de que imprima el mensaje del menor, última ejecución antes de terminar el programa, los valores de las variables son:

- primerNumero: 20.
- b: 19.
- mayor: 20.
- menor: 19.

iii. Dentro de las variables de la clase Calculo. ¿Qué variables existen en el contexto y qué valor tienen?

Para este caso vamos a volver a poner el punto de interrupción en la línea 25 y en la 40 de la clase "Calculo". Vamos a volver a introducir como número 1 el 20 y como número 2 el 19.

```

23     my=y;
24 }
25 return my;
26 }
27
28 //Tiene un error
29 public static int calcularMenor(int x, int y)
30 {
31     int mn;
32     if(x>y)
33     {
34         mn=y;
35     }
36     else
37     {
38         mn=x;
39     }
40     return mn;
41 }
42
43
44 //calcula el factorial del entero positivo n
45 // devuelve un long
46
47 public static long calcularFactorial(int n)
48 {
49     long fact=1;

```

Variables x | Call Stack | Breakpoints | Output

Name	Type	Value
Static		
x	int	20
y	int	19
my	int	20

En la primera parada de la línea 25, podemos observar que las variables toman los siguientes valores:

- x: 20.
- y: 19.
- my: 20.

Si seguimos con la depuración hasta el siguiente punto de ruptura (línea 40), tenemos los siguientes valores en las variables:

- x: 20.
- y: 19.
- mn: 19.


```

38         mn=x;
39     }
40     return mn;
41 }
42
43 //calcula el factorial del entero positivo n
44 // devuelve un long
45
46 public static long calcularFactorial(int n)
47 {
48     long fact=1;
49
50     Calculo > calcularMenor >

```

Name	Type	Value
Static		
x	int	20
y	int	19
mn	int	19

iv. ¿Por qué líneas de calcularMenor pasa la ejecución del programa?

Para este caso vamos a poner un punto de ruptura en la línea 29 de la Clase “Calculo” (punto de ruptura de método), para hacer que se pare la ejecución cuando se haga la llamada al método “calcularMenor”.

Considerando que hemos introducido como número 1 el 20 y como número 2 el 19, pasamos realmente por las líneas: 30, 31, 32, 33, 34, 35, 40, 41. Aunque el cursor del programa únicamente me marca las líneas 32, 34 y 40.

```

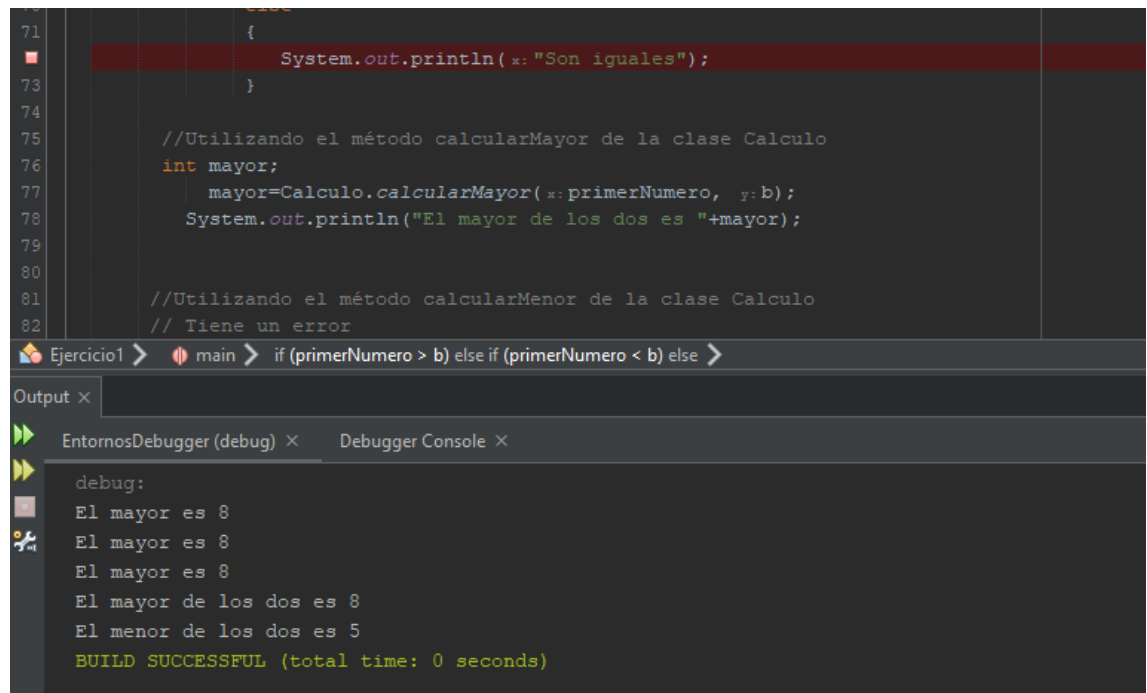
28 //Tiene un error
29 public static int calcularMenor(int x, int y)
30 {
31     int mn;
32     if(x>y)
33     {
34         mn=y;
35     }
36     else
37     {
38         mn=x;
39     }
40     return mn;
41 }
42

```

Ejercicio 2

Cambia la clase por defecto para que sea Ejercicio1.

- Elimina los puntos de interrupción creados en el anterior Ejercicio.**
- Pon un punto de interrupción en la línea 72. Ejecuta en modo de depuración e indica el valor de las variables.**



```

71      {
72          System.out.println("Son iguales");
73      }
74
75      //Utilizando el método calcularMayor de la clase Calculo
76      int mayor;
77      mayor=Calculo.calcularMayor(x:primerNumero, y:b);
78      System.out.println("El mayor de los dos es "+mayor);
79
80
81      //Utilizando el método calcularMenor de la clase Calculo
82      // Tiene un error

```

Debugger Console:

```

debug:
El mayor es 8
El mayor es 8
El mayor es 8
El mayor de los dos es 8
El menor de los dos es 5
BUILD SUCCESSFUL (total time: 0 seconds)

```

No podemos observar el valor de las variables porque el programa nunca pasa por el punto de interrupción para los valores que se le han introducido, por lo que termina el proceso de depuración sin hacer ninguna interrupción.

- c. Elimina el punto de interrupción del apartado anterior e incorpora uno nuevo en la línea 68. ¿Qué valor tienen las variables `primerNumero` y `b`?

The screenshot shows a Java IDE with a code editor and a variable inspection window. The code editor displays the following code:

```

67      {
68          System.out.println("El mayor es "+b);
69      }
70      else
71      {
72          System.out.println(x: "Son iguales");
73      }
74
75      //Utilizando el método calcularMayor de la clase Calculo
76      int mayor;
77      mayor=Calculo.calcularMayor(x: primerNumero, y:b);
78      System.out.println("El mayor de los dos es "+mayor);
79
80
81      //Utilizando el método calcularMenor de la clase Calculo
82      // Tiene un error

```

The variable inspection window is open, showing the following variables and their values:

Name	Type	Value
Static		...
args	String[]	... #393(length=0)
teclado	Scanner	... #394
primerNumero	int	... 5
b	int	... 8

En este caso podemos observar que las variables tienen los siguientes valores:

- `primerNumero`: 5.
- `b`: 8.

- d. Elimina el punto de interrupción anterior y crea uno nuevo en la línea para que pare en la línea 62 solo si b es mayor que primernumero. Cuando pare cambia el valor de primernumero por el valor 11 y sigue la ejecución del programa hasta el final describiendo que pasa.

```

61 //también puede ponerse
62 if (primerNumero>b)
63 {
64     System.out.println("El mayor es "+primerNumero);
65 }
66 else if(primerNumero<b)
67 {
68     System.out.println("El mayor es "+b);
69 }
70 else
71 {
72     System.out.println("Son iguales");
73 }
74
75 //Utilizando el método calcularMayor de la clase Calculo
76 int mayor;
77 mayor=Calculo.calcularMayor(x:primerNumero, y:b);
78 System.out.println("El mayor de los dos es "+mayor);
79
80
81 //Utilizando el método calcularMenor de la clase Calculo
82 // Tiene un error

```

Name	Type	Value
Static		...
args	String[]	... #393(length=0)
teclado	Scanner	... #394
primerNumero	int	... 11
b	int	... 8

Al cambiar el valor de “primerNumero” por un valor mayor que “b”, hemos cambiado el hilo de ejecución del programa, ya que en el caso anterior no entraría en el if de la línea 62, sino que entraría en el else if de la línea 66. Al igual que en los métodos “calcularMayor” y “calcularMenor” de la Clase “Calculo”, los resultados serían diferentes, puesto que el 11 es mayor que 8.

The image shows a Java program in an IDE. The code defines a method to find the maximum and minimum of two numbers, `primerNumero` and `b`. It uses conditional logic to print the result directly or uses methods from a `Calculo` class. The program is run in a debug mode, and the output is shown in the 'Debugger Console'.

```

61 //También puede ponerse
62 if (primerNumero>b)
63 {
64     System.out.println("El mayor es "+primerNumero);
65 }
66 else if(primerNumero<b)
67 {
68     System.out.println("El mayor es "+b);
69 }
70 else
71 {
72     System.out.println("Son iguales");
73 }
74
75 //Utilizando el método calcularMayor de la clase Calculo
76 int mayor;
77 mayor=Calculo.calcularMayor(x:primerNumero, y:b);
78 System.out.println("El mayor de los dos es "+mayor);
79
80 //Utilizando el método calcularMenor de la clase Calculo
81 // Tiene un error
82 int menor;
83 menor=Calculo.calcularMenor(x:primerNumero, y:b);
84 System.out.println("El menor de los dos es "+menor);
85
86 }
87
88 }

```

The 'Variables' window shows the state of the program:

Name	Type	Value
Static		...
args	String[]	... #393(length=0)
teclado	Scanner	... #394
primerNumero	int	... 11
b	int	... 8
mayor	int	... 11
menor	int	... 8

The 'Debugger Console' shows the output of the program:

```

debug:
El mayor es 8
El mayor es 8
El mayor es 11
El mayor de los dos es 11
El menor de los dos es 8

```

Como hemos cambiado el valor de las variables a mitad de programa, también hemos cambiado la salida del mismo.

- e. Elimina el punto de interrupción anterior y crea uno nuevo cuando **b** es menor primernumero. ¿qué valores tienen las variables?

```

43 // y se dirigirá por uno de los dos caminos if o else
44 if (primerNumero>b)
45 {
46     System.out.println("El mayor es "+primerNumero);
47 }
48 else
49 {
50     if (primerNumero<b)
51     {
52         System.out.println("El mayor es "+b);
53     }
54     else
55     {
56         System.out.println("Son iguales");
57     }
58 }
59

```

Ejercicio1 > main >

Variables	Call Stack	Breakpoints	Output
Name	Type	Value	
Static			
args	String[]	#393(length=0)	
teclado	Scanner	#394	
primerNumero	int	5	
b	int	-5	

En este caso vamos a cambiar el valor de la variable “b” por el valor -5, y nos daría como resultado las siguientes variables:

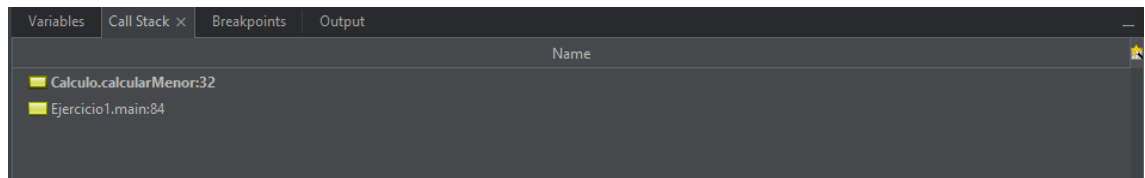
Variables	Call Stack	Breakpoints	Output
Name	Type	Value	
Static			
args	String[]	#393(length=0)	
teclado	Scanner	#394	
primerNumero	int	5	
b	int	-5	
mayor	int	5	
menor	int	-5	

- f. Establece un punto de interrupción para todos los métodos de la clase **Calculo** para así saber cuál se ejecutan o no. ¿Cuáles de ellos son ejecutados y cuál no?

Podemos observar que los métodos de la clase “Calculo” llamados son:

- calcularMayor en la línea 77 del main.
- calcularMenor en la línea 84 del main.

Variables	Call Stack	Breakpoints	Output
Name			
Calculo.calcularMayor:17			
Ejercicio1.main:77			



Sin embargo, no hace llamada a los métodos:

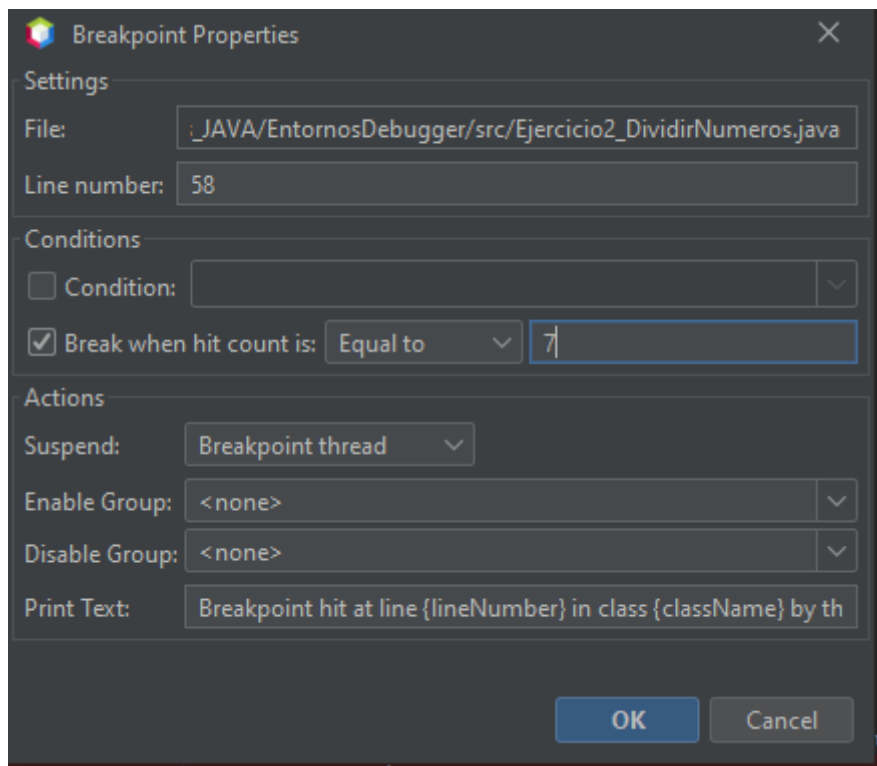
- calcularFactorial.
- sumarDivisores.
- introduceNumero.

Sabemos que no hace llamada porque tienen breakpoint y en el transcurso del programa no hace ninguna interrupción en ninguno de ellos.

Ejercicio 3

Cambia la clase por defecto para que sea Ejercicio2_DividirNumeros.

- a. Elimina todos los puntos de interrupción.
- b. Establece para la variable resultado del método multiplicar un punto de interrupción cuando se haya actualizado 7 veces. ¿Qué muestra la consola cuando se ejecuta en modo depuración?



```

Variables | Call Stack | Breakpoints | Output x
EntornosDebugger (debug) x | Debugger Console x
debug:
El resultado de Dividir 3.0 / 3.0 es 1.0
El resultado de Dividir 5.0 / 2.0 es 2.5
El resultado de Dividir 7.0 / 9.0 es 0.7777777777777778
El resultado de Dividir 4.0 / 1.0 es 4.0
El resultado de Dividir 4.0 / 6.0 es 0.6666666666666666
El resultado de Dividir 3.0 / 9.0 es 0.3333333333333333

```

Al limitarlo a 7 ejecuciones, la consola nos muestra sólo 6 salidas, ya que al llegar a la séptima, lo forzamos a que pare de calcular la variable, hasta que le digamos que siga ejecutando.

- c. Elimina el punto de interrupción anterior y vamos a hacer ahora que la variable resultado sea un elemento observado. Muestra que se haya creado correctamente.

```

Watches x | Breakpoints | Output
Name | Type | Value
[+] resultado | double | 2.5
<Enter new watch>

41 public static void main(String[] args) {
42
43     multiplicar( numero1: num01, numero2: num02);
44     multiplicar( numero1: num03, numero2: num04);
45     multiplicar( numero1: num05, numero2: num06);
46     multiplicar( numero1: num07, numero2: num08);
47     multiplicar( numero1: num09, numero2: num10);
48     multiplicar( numero1: num11, numero2: num12);
49     multiplicar( numero1: num13, numero2: num14);
50     multiplicar( numero1: num15, numero2: num16);
51     multiplicar( numero1: num17, numero2: num18);
52     multiplicar( numero1: num19, numero2: num20);
53     multiplicar( numero1: num21, numero2: num22);
54     multiplicar( numero1: num23, numero2: num24);
55 }
56
57 public static void multiplicar(double numero1, double numero2){
58     resultado = numero1 / numero2;
59     System.out.println("El resultado de Dividir "+numero1+ " / "+numero2+" es "+resultado);
60 }
61
62

```

Hemos puesto el punto de ruptura en la línea 56 para ver que la variable observada se ha creado correctamente y que va cambiando su valor dependiendo de la operación.

Parte 2. Pruebas

Ejercicio 1

Disponemos de un módulo de software de una aplicación bancaria con los siguientes datos de entrada en su tipo

- Código de área. Número de 3 dígitos cuyo primer valor no puede ser ni 0 ni 1.**
- Clave de la operación. 6 caracteres alfanuméricos**
- Órdenes posibles: “Cheque”, “Depósito”, “Pago factura”, “retirada de fondos”**
- Elabora las pruebas que debemos realizar siguiendo el mecanismo de las clases de equivalencia.**

Resultado esperado con las pruebas

- S1. Cheque.
- S2. Depósito.
- S3. Pago factura.
- S4. Retirada de fondos.
- ER1. Código de área incorrecto.
- ER2. Clave de operación incorrecta.
- ER3. Orden no seleccionada.

Condición de entrada	Clases de equivalencia	Clases válidas	COD	Clases no válidas	COD
Código de área	Rango	200 >= Cód. Área <= 999	V1	Cód. Área < 200	NV1
				Cód. Área > 999	NV2
Clave de operación	Valor	Cualquier valor alfanumérico de 6 caracteres	V2	Valor alfanumérico > 6 caracteres	NV3
				Valor alfanumérico < 6 caracteres	NV4
Orden	Miembro de un conjunto	Orden = Cheque	V3	Orden = Elige orden	NV5
		Orden = Depósito	V4		
		Orden = Pago factura	V5		
		Orden = Retirada de fondos	V6		

Caso prueba	Clases equivalencia	Condiciones de Entrada			Resultado esperado
		Código área	Clave de operación	Orden	
CP1	V1, V2, V3	355	12AS23	"Cheque"	S1
CP2	V1, V2, V4	899	SS1548	"Depósito"	S2
CP3	V1, V2, V5	785	SSSSS1	"Pago factura"	S3
CP4	V1, V2, V6	888	1234AA	"Retirada de fondos"	S4
CP5	NV1, V2, V3	155	LL4558	"Cheque"	ER1
CP6	NV1, V2, V4	111	548568	"Depósito"	ER1
CP7	NV1, V2, V5	157	YY21YY	"Pago factura"	ER1
CP8	NV1, V2, V6	99	5485GG	"Retirada de fondos"	ER1
CP9	NV2, V2, V3	1020	89GD65	"Cheque"	ER1
CP10	NV2, V2, V4	1235	FGGG58	"Depósito"	ER1
CP11	NV2, V2, V5	2365	FG4424	"Pago factura"	ER1
CP12	NV2, V2, V6	1111	5256RR	"Retirada de fondos"	ER1
CP13	V1, NV3, V3	203	FD32333	"Cheque"	ER2
CP14	V1, NV3, V4	557	1562RRRR	"Depósito"	ER2
CP15	V1, NV3, V5	485	EEE1252D	"Pago factura"	ER2
CP16	V1, NV3, V6	632	TT12DD12	"Retirada de fondos"	ER2
CP17	V1, NV4, V3	875	45	"Cheque"	ER2
CP18	V1, NV4, V4	695	S4	"Depósito"	ER2
CP19	V1, NV4, V5	333	FF44	"Pago factura"	ER2
CP20	V1, NV4, V6	785	45DR	"Retirada de fondos"	ER2
CP21	V1, V2, NV5	456	45TT66	Elige orden	ER3
CP22	NV1, V2, NV5	152	TT55SA	Elige orden	ER3
CP23	NV1, NV3, V3	185	RRRRRRR	"Cheque"	ER1, ER2
CP24	NV1, NV3, V4	8	RE4548T	"Depósito"	ER1, ER2
CP25	NV1, NV3, V5	11	VB12822	"Pago factura"	ER1, ER2
CP26	NV1, NV3, V6	126	2152SAE	"Retirada de fondos"	ER1, ER2
CP27	NV1, NV4, V3	44	22	"Cheque"	ER1, ER2
CP28	NV1, NV4, V4	18	FD6	"Depósito"	ER1, ER2
CP29	NV1, NV4, V5	66	SS11	"Pago factura"	ER1, ER2
CP30	NV1, NV4, V6	32	125S	"Retirada de fondos"	ER1, ER2
CP31	NV2, NV3, V3	5466	5555555	"Cheque"	ER1, ER2
CP32	NV2, NV3, V4	2222	AS22SE2	"Depósito"	ER1, ER2

CP33	NV2, NV3, V5	1233	22ETF22	"Pago factura"	ER1, ER2
CP34	NV2, NV3, V6	1458	25SE5FSA	"Retirada de fondos"	ER1, ER2
CP35	NV2, NV4, V3	8546	DD	"Cheque"	ER1, ER2
CP36	NV2, NV4, V4	5555	DF25	"Depósito"	ER1, ER2
CP37	NV2, NV4, V5	4789	FF263	"Pago factura"	ER1, ER2
CP38	NV2, NV4, V6	6665	DD4	"Retirada de fondos"	ER1, ER2
CP39	NV1, NV3, NV5	12	DDS1255	Elige orden	ER1, ER2, ER3
CP40	NV1, NV4, NV5	54	122	Elige orden	ER1, ER2, ER3
CP41	NV2, NV3, NV5	4589	DDESA11	Elige orden	ER1, ER2, ER3
CP42	NV2, NV4, NV5	8888	DE	Elige orden	ER1, ER2, ER3

e) Añade las pruebas que consideremos siguiendo la técnica de análisis de valores límite.

Caso prueba	Clases equivalencia	Condiciones de Entrada			Resultado esperado
		Código área	Clave de operación	Orden	
CP43	V1a, V2, V3	200	TR55EE	"Cheque"	S1
CP44	V1b, V2, V4	999	SD6666	"Depósito"	S2
CP45	NV1a, V2, V5	199	12DD22	"Pago factura"	ER1
CP46	NV1b, V2, V6	1000	ADER22	"Retirada de fondos"	ER1

A partir del siguiente código

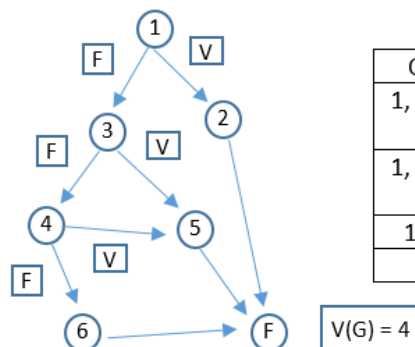
```
// categoria recibe los valores ALTA, MEDIA y BAJA. Edad es un entero positivo
public int devuelveSueldo(String categoria, int edad){
    int sueldoResultado = 0;
    if (categoria.equals("ALTA")){
        sueldoResultado = 2000;
    } else {
        if (edad < 30 || categoria.equals("BAJA")){
            sueldoResultado = 1500;
        } else {
            sueldoResultado = 1700;
        }
    }
    return sueldoResultado;
}
```

Ejercicio 2

Realiza un análisis de caja blanca completo del método devuelveSueldo que puedes ver a continuación

- Puedes hacer el grafo “a mano” en papel, para después escanearlo (foto) e insertarlo en el documento. También puedes hacerlo con una herramienta de dibujo si te resulta más cómodo.
- En cualquier caso, el tamaño y la calidad del dibujo debe permitir leerlo con claridad.
- Debes diseñar las pruebas que consideres adecuadas para tratar de llevar a cabo el conjunto de caminos básicos.

```
// categoria recibe los valores ALTA, MEDIA y BAJA. Edad es un entero positivo
public int devuelveSueldo(String categoria, int edad){
    int sueldoResultado = 0;
    ① if (categoria.equals( anObject: "ALTA")){
        ② sueldoResultado = 2000;
    } else {
        ③ if (edad < 30 || categoria.equals( anObject: "BAJA")){
            ④ sueldoResultado = 1500;
        } else {
            ⑤ sueldoResultado = 1700;
        }
    }
    ⑥ return sueldoResultado;
}
```



Camino	Entrada	Salida
1, 3, 4, 5, F	Categoría = !=ALTA, edad > 30, categoría = BAJA	1500
1, 3, 4, 6, F	Categoría = !=ALTA, edad > 30, categoría = !=BAJA	1700
1, 3, 5, F	Categoría = !=ALTA, edad < 30	1500
1, 2, F	Categoría = ALTA	2000

V(G) = 4

Ejercicio 3

Realiza un análisis de caja negra del método devuelveSueldo

- a) Elabora las pruebas que debemos realizar siguiendo el mecanismo de las clases de equivalencia.
- Entradas y salidas deseadas:
 - Si metemos “ALTA” y un entero <30 o >30 (es indiferente), su salida debería ser: 2000 (S1).
 - Si metemos “BAJA” y un entero <30 o >30 (es indiferente), su salida debería ser: 1500 (S2).
 - Si metemos un String distinto de “ALTA” o “BAJA” y un entero >30, su salida debería ser: 1700 (S3).
 - Si metemos un String distinto de “ALTA” o “BAJA” y un entero <30, su salida debería ser: 1500 (S2).
 - Casos de error:
 - El único caso de error que puede haber en este método es que le introduzcamos una entrada en la parte entera distinta de un número entero. ER1, “No se ha introducido un número entero”. En este caso daría un error de ejecución, aunque en realidad no influye para nada en la posible salida que saldría 1700.

Condición de entrada	Clases de equivalencia	Clases válidas	COD	Clases no válidas	COD
Categoría	Cadena	Cualquier cadena	V1	No hay	-
Edad	Valor	Cualquier valor entero	V2	Cualquier valor no entero	NV1

Caso prueba	Clases equivalencia	Condiciones de entrada		Resultado esperado
		Categoría	Edad	
CP1	V1, V2	“ALTA”	22	S1
CP2	V1, V2	“BAJA”	35	S2
CP3	V1, V2	“ANTONIO”	25	S2
CP4	V1, V2	“MESA”	50	S3
CP5	V1, NV1	“ENTORNOS”	25,3	ER1

- b) Añade las pruebas que consideremos siguiendo la técnica de análisis de valores límite.

Caso prueba	Clases equivalencia	Condiciones de entrada		Resultado esperado
		Categoría	Edad	
CP6	V1, V2a	“BASES DE DATOS”	29	S2
CP7	V1, V2b	“OFICINA”	30	S3