

Eclipse. Depurando 2

Al igual que en NetBeans en Eclipse también podremos establecer condiciones de depuración en un código para buscar errores lógicos.

Detener la ejecución en función de una condición

Mediante Eclipse podremos conseguir que la ejecución del código se pare en el breakpoint mediante dos condiciones:

☐ Suspend when 'true' ☒ Suspend when value changes

- Cuando la condición sea cierta. (al igual que hacíamos con Netbeans)
- Cuando la condición cambie. (detendrá la ejecución cuando por ejemplo cambie el valor de una variable).

Vamos a ver el funcionamiento de ambas condiciones.

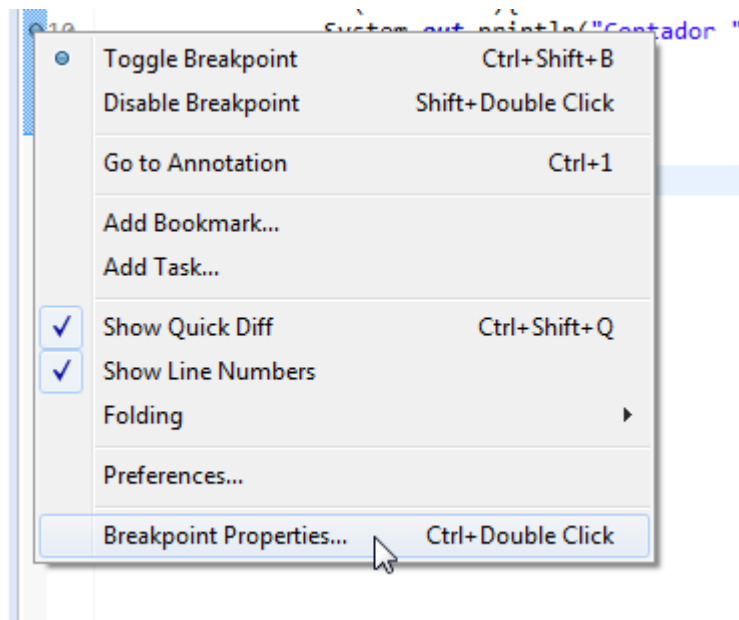
Introducimos el siguiente código de ejemplo

```
3 public class Cuenta {
4
5     public static void main(String[] args) {
6         int cont=1;
7         int fin=100;
8         System.out.println("Inicio cuenta ");
9         while (cont<=fin){
10            System.out.println("Contador " + cont );
11            cont++;
12        }
13    }
14 }
15 }
```

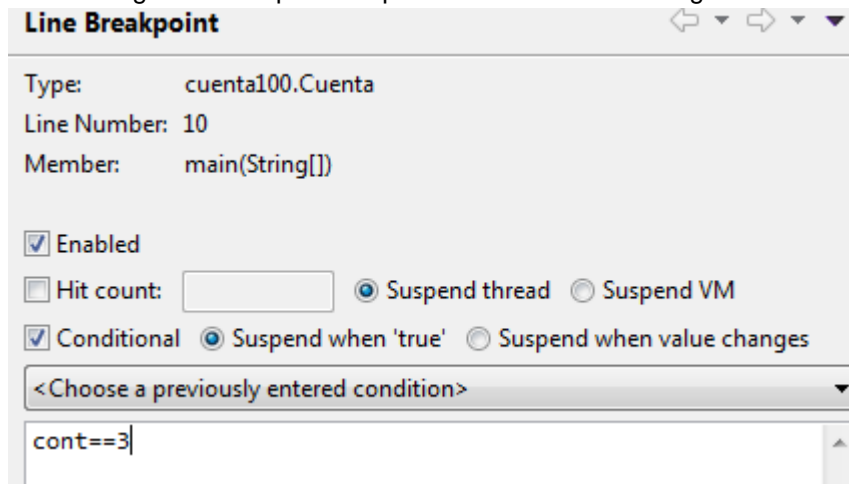
Breakpoint se detiene cuando condición es cierta

Introducimos un breakpoint en la línea 10. Lo configuramos para que se detenga siempre que la condición sea cierta. Para ello:

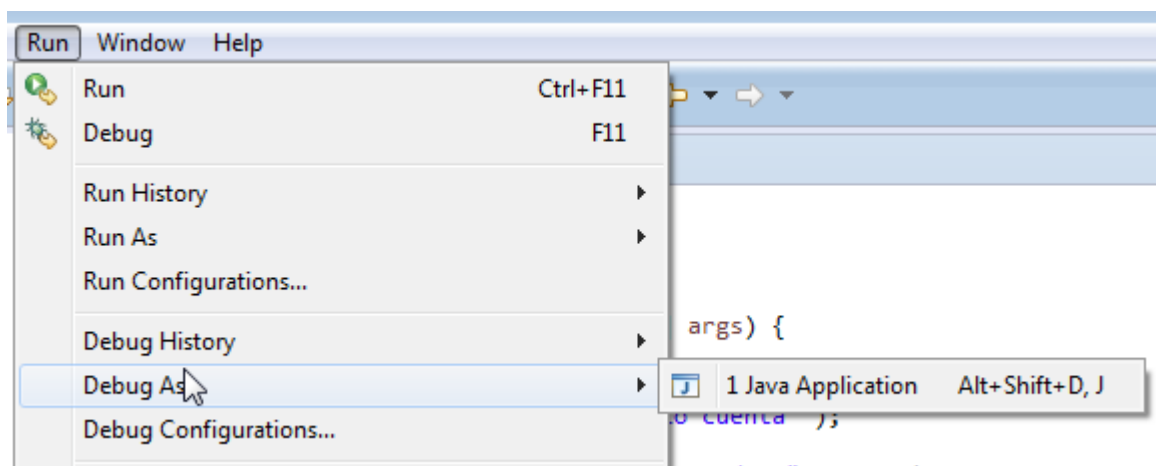
1. Introduzco breakpoint.
2. Botón derecho sobre el breakpoint. En el menú contextual selecciono **breakpoint Properties**



3. En el diálogo de Breakpoint Properties introducimos las siguientes condiciones



4. Pulsamos OK.
5. Seleccionamos Run>Debug as Java Application



Detiene el breakpoint

Nos abre la ventana de depuración donde observamos los siguientes detalles:

Variables

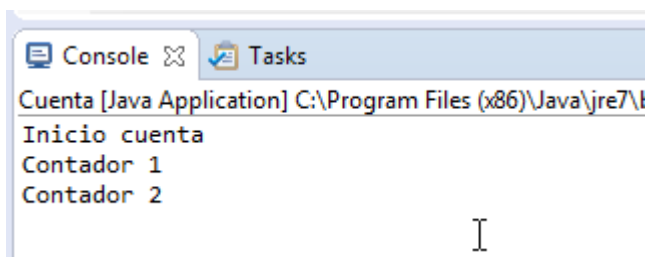
Name	Value
args	String[0] (id=16)
cont	3
fin	100

Fijate que cont vale 3.

Código. Vemos punto detención

```
4  
5 public static void main(String[] args) {  
6     int cont=1;  
7     int fin=100;  
8     System.out.println("Inicio cuenta ");  
9     while (cont<=fin){  
10        System.out.println("Contador " + cont );  
11        cont++;  
12    }  
13 }  
14 }  
15 }
```

Output



```
Cuenta [Java Application] C:\Program Files (x86)\Java\jre7\l  
Inicio cuenta  
Contador 1  
Contador 2
```

Es decir ha detenido cuando cont vale 3, pero justo antes que se ejecute la instrucción del breakpoint ya que no muestra en el Output contador 3.

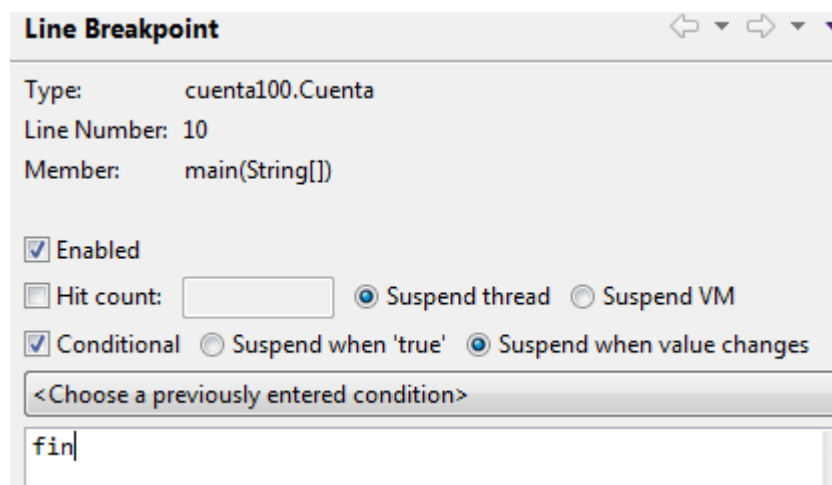
Breakpoint se detiene cuando condición varía.

Volvemos al punto inicial con nuestro programa.

```
3 public class Cuenta {
4
5     public static void main(String[] args) {
6         int cont=1;
7         int fin=100;
8         System.out.println("Inicio cuenta ");
9         while (cont<=fin){
10            System.out.println("Contador " + cont );
11            cont++;
12        }
13    }
14 }
```

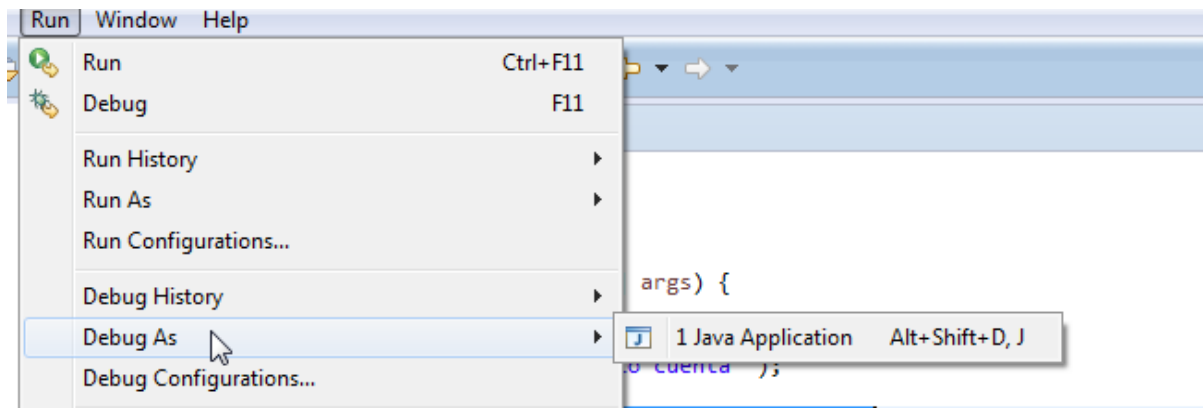
Establecemos breakpoint en la línea 10.

En propiedades del breakpoint ahora configuramos de la siguiente forma. Fíjate que hemos seleccionado *Suspend when value changes* y introducimos la variable fin (toma valor inicial 100 y luego ya no varía)



Pulsamos OK

Ahora Run>Debug as java application



Se detiene en la primera llegada al breakpoint. (fin toma valor 100)
Observamos **variables**

Name	Value
args	String[0] (id=16)
cont	1
fin	100

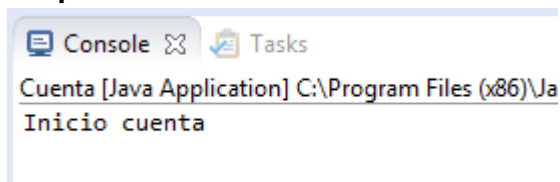
Código

```

3 public class Cuenta {
4
5     public static void main(String[] args) {
6         int cont=1;
7         int fin=100;
8         System.out.println("Inicio cuenta ");
9         while (cont<=fin){
10            System.out.println("Contador " + cont );
11            cont++;
12        }
13    }
14 }
15

```

Output

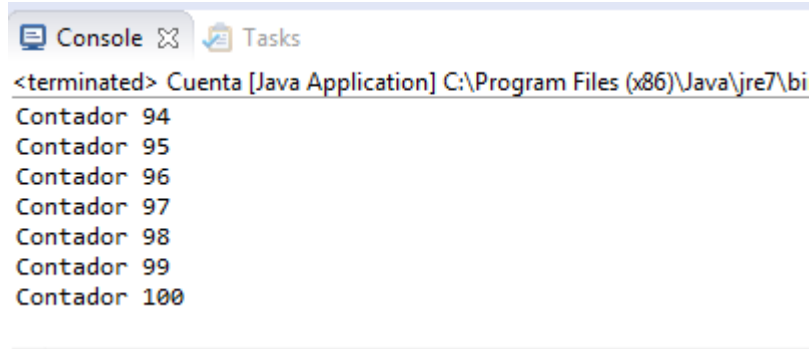


Se detiene justo antes de ejecutar el breakpoint.



Seguimos depuración pulsando

Ya no se detiene, ya que el valor de la variable no varía

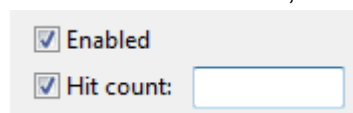


The screenshot shows the Eclipse IDE's console window. At the top, there are tabs for 'Console' and 'Tasks'. The console title is '<terminated> Cuenta [Java Application] C:\Program Files (x86)\Java\jre7\bi'. The output text in the console is as follows:

```
Contador 94  
Contador 95  
Contador 96  
Contador 97  
Contador 98  
Contador 99  
Contador 100
```

Detener la ejecución tras un determinado número de ejecuciones

Si queremos detener la ejecución en un breakpoint tras la ejecución de un determinado número de ejecuciones de esa línea, seleccionaremos Hit Count e introducimos un número.



Esto se utilizará para depurar bucles si por ejemplo estamos interesados en la tercera ejecución.

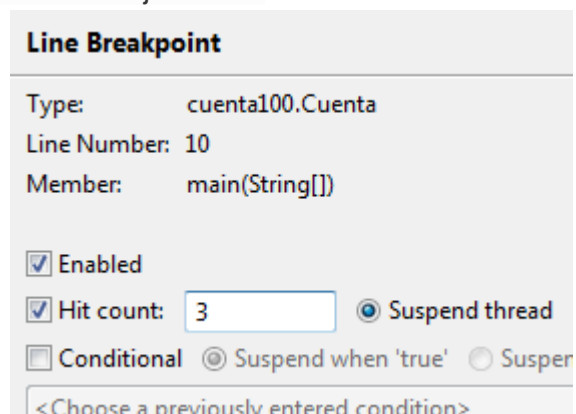
Volvemos al ejemplo inicial el breakpoint en la línea 10.

```

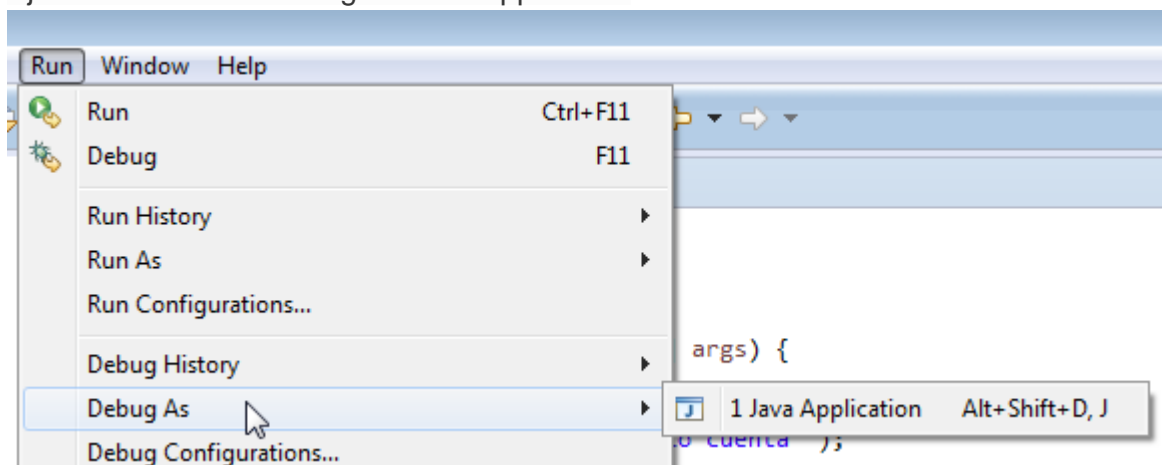
3 public class Cuenta {
4
5     public static void main(String[] args) {
6         int cont=1;
7         int fin=100;
8         System.out.println("Inicio cuenta ");
9         while (cont<=fin){
10            System.out.println("Contador " + cont );
11            cont++;
12        }
13    }
14 }

```

En breakpoint properties establezco la siguiente configuración. Hit count: 3, es decir se detendrá en la tercera ejecución.



Ejecutamos Run > Debug as Java Application



Vemos que se detiene en la tercera ejecución justo antes de ejecutar la línea del breakpoint.

Variables


(x)= Variables  Breakpoints

Name	Value
args	String[0] (id=16)
cont	3
fin	100

Código

```
2
3 public class Cuenta {
4
5     public static void main(String[] args) {
6         int cont=1;
7         int fin=100;
8         System.out.println("Inicio cuenta ");
9         while (cont<=fin){
10             System.out.println("Contador " + cont );
11             cont++;
12         }
13     }
14 }
15
```

Output

Console  Tasks

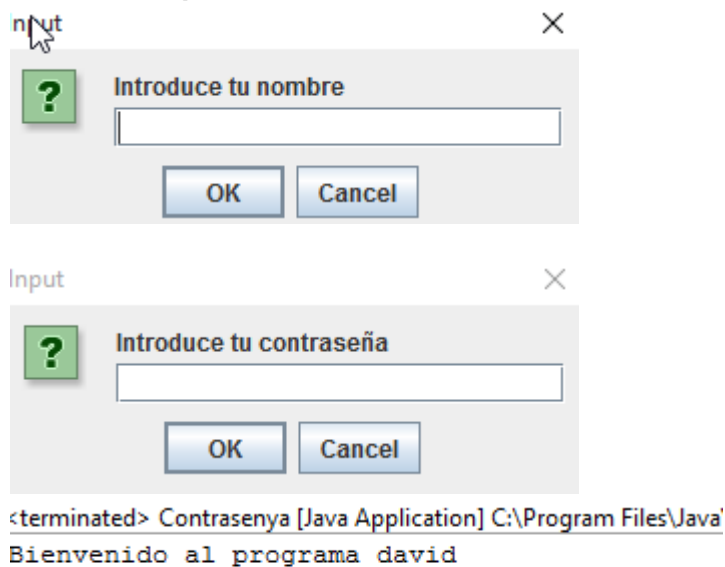
Cuenta [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw
Inicio cuenta
Contador 1
Contador 2

Practicando

Ejercicio 1

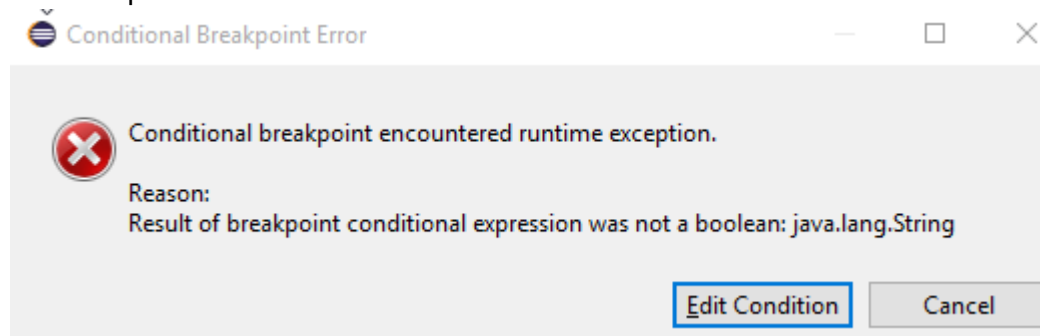
Realiza un programa mediante Eclipse que solicite el nombre de usuario y su contraseña. El nombre de usuario será tu nombre y la contraseña "abreme". Seguirá pidiendo nombre de usuario mientras que no se introduzca la contraseña correcta:"abreme". Una vez se introduce la contraseña correcta el programa muestra por pantalla Bienvenido al programa tu_nombre.

Funcionamiento esperado

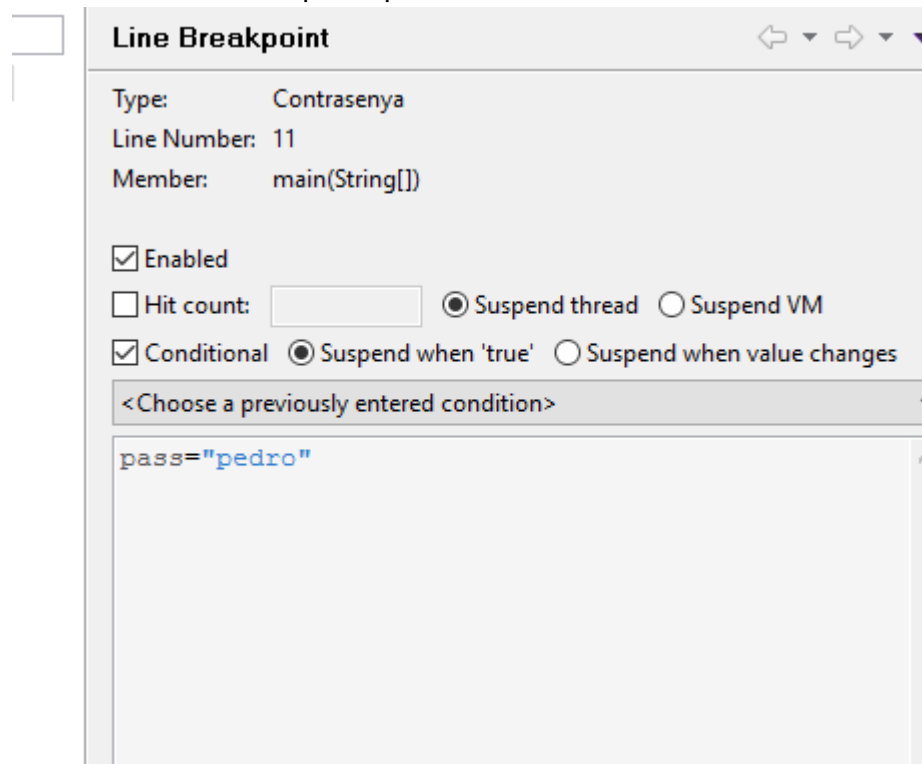


Nuestro compañero Pedro está realizando la depuración del programa, ha introducido un breakpoint que quiere que se detenga justo antes de la instrucción que escribe Bienvenido al programa _____, cuando el nombre del usuario introducido es "pedro". Tiene el problema que cuando introduce la condición del breakpoint le aparece un error. Observa las condiciones del breakpoint que ha introducido Pedro y modificalas para solucionar el error y que el programa se detenga donde quiere pedro y cuando el introduce el usuario pedro.

error le aparece a Pedro



Condiciones del breakpoint que tiene Pedro



Para que funcione la condición debe ser



Ejercicio 2

Implementa un programa que mediante un bucle while muestre un contador de 1 a 10.

```
<terminated> Iguales [Java Applicat  
el contador vale 1  
el contador vale 2  
el contador vale 3  
el contador vale 4  
el contador vale 5  
el contador vale 6  
el contador vale 7  
el contador vale 8  
el contador vale 9  
el contador vale 10
```

Detén el bucle cuando se ejecute la segunda ejecución y visualiza el valor de las variables.

