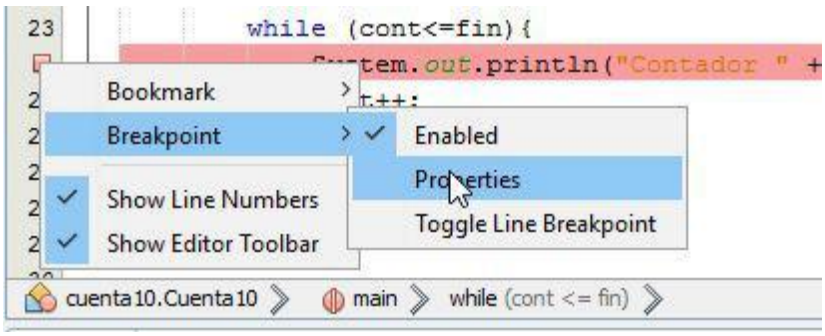
Netbeans: Depurando 2

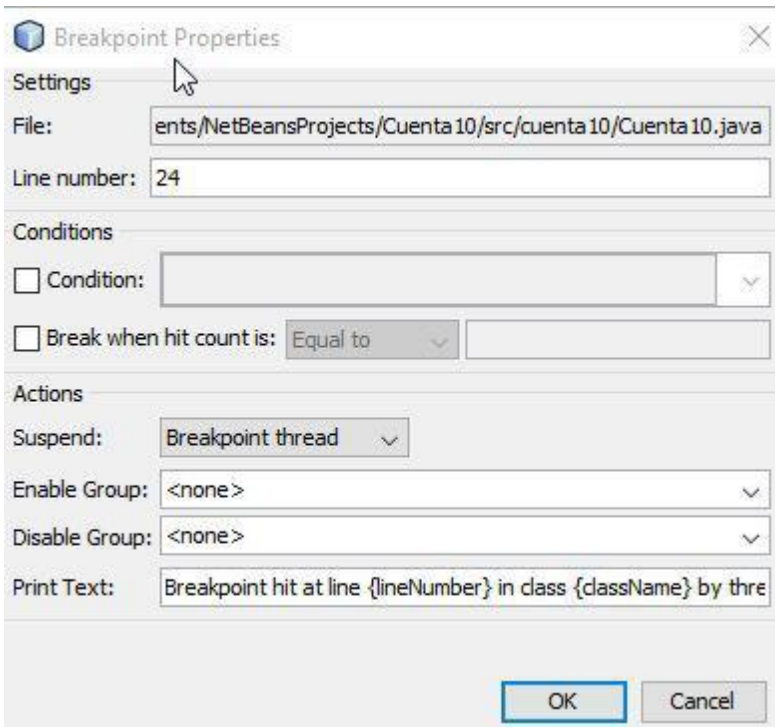
Estableciendo condiciones

En la práctica anterior hemos establecido breakpoints (puntos de ruptura) que nos permiten detener la ejecución del programa y examinar el valor de las variables en ese punto.

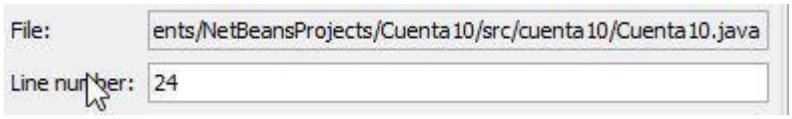
Si queremos añadir mayor control a un simple breakpoint, podemos editar las propiedades del breakpoint pulsando el botón derecho del ratón sobre el breakpoint y seleccionando en el menú contextual Propiedades.



Esto nos mostrará el diálogo de las propiedades del breakpoint.



En esta ventana, podemos ver el número de línea y el archivo al que pertenece el breakpoint.



El número de línea se puede modificar si lo hemos insertado en el número de línea equivocado. Sin embargo lo más interesante de las propiedades son las condiciones que podemos aplicar al breakpoint.

Mediante el punto de entrada de condiciones podemos establecer una condición que se deberá cumplir si queremos que se detenga la ejecución. Ejemplo, si quiero que el código se detenga cuando la variable cont es igual a 20 tendré que añadir la condición cont ==20.

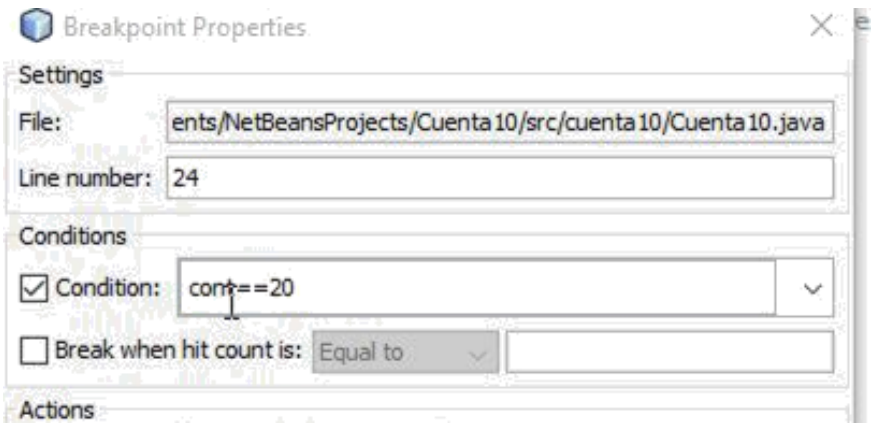
Ejemplo
Tengo el siguiente código

```
19      public static void main(String[] args) {
20          int cont=1;
21          int fin=100;
22          System.out.println("Inicio cuenta ");
23          while (cont<=fin){
24              System.out.println("Contador " + cont );
25              cont++;
26              }
27      }
```

Quiero detener la ejecución en la línea 24 cuando cont valga 20.

Pasos

1. Establezco punto de ruptura en la línea 24 (o línea donde tengas la escritura por pantalla)
2. En propiedades del breakpoint establezco condición



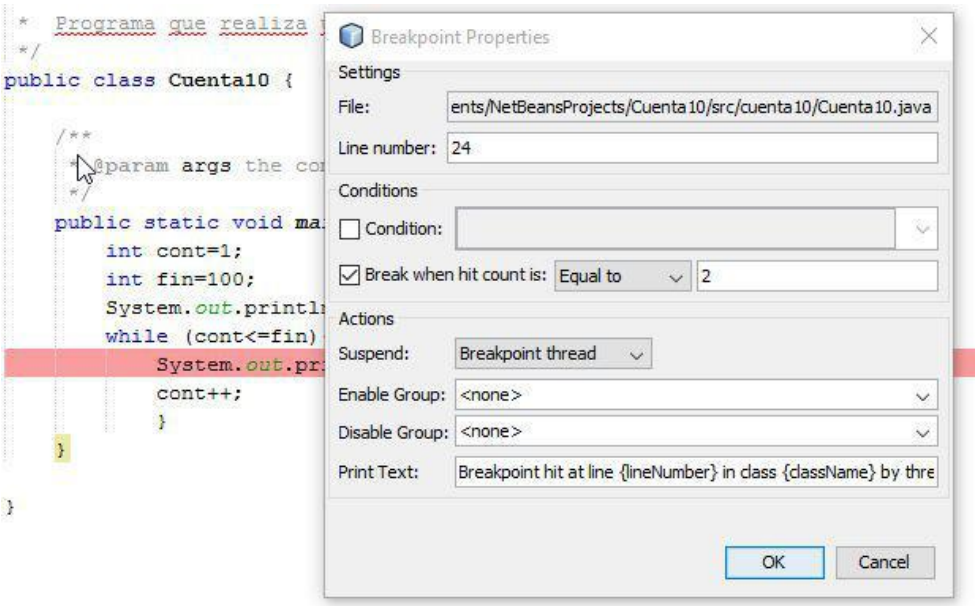
3. Inicio depuración
4. Efectivamente se detiene. Ojo! justo antes de ejecutar el println. Observa que cont vale 20 pero en el output no se muestra el Contador 20


```
Inicio cuenta
Contador 1
Contador 2
Contador 3
Contador 4
Contador 5
Contador 6
Contador 7
Contador 8
Contador 9
Contador 10
Contador 11
Contador 12
Contador 13
Contador 14
Contador 15
Contador 16
Contador 17
Contador 18
Contador 19
```

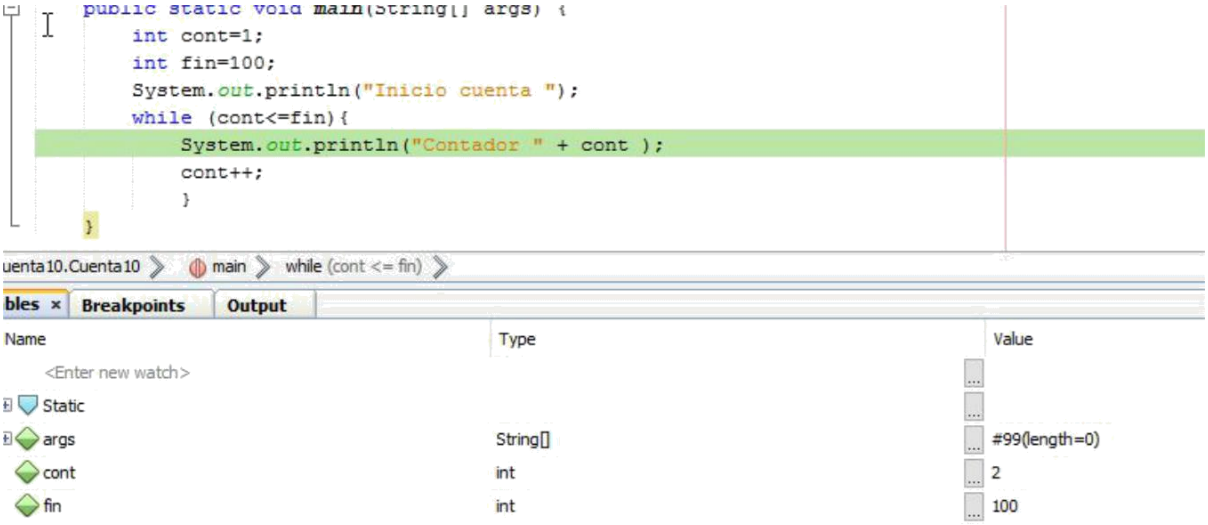
También podemos provocar que la ejecución de la aplicación se detenga cuando el breakpoint se ejecute un número determinado de veces. La detención de la detención se podrá establecer cuando el número de ejecuciones del breakpoint sea

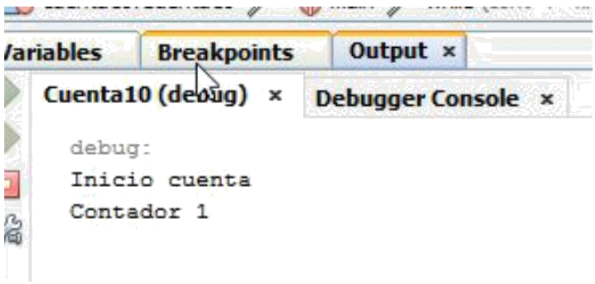
- Equal to → Igual a
- Greater than → Mayor que
- Multiple of → Multiple de

Vamos a realizar pruebas. Volviendo al ejemplo anterior, establezco un breakpoint en la línea 24 y configuramos las propiedades para que se detenga cuando el número de veces que ejecuto el breakpoint sea 2



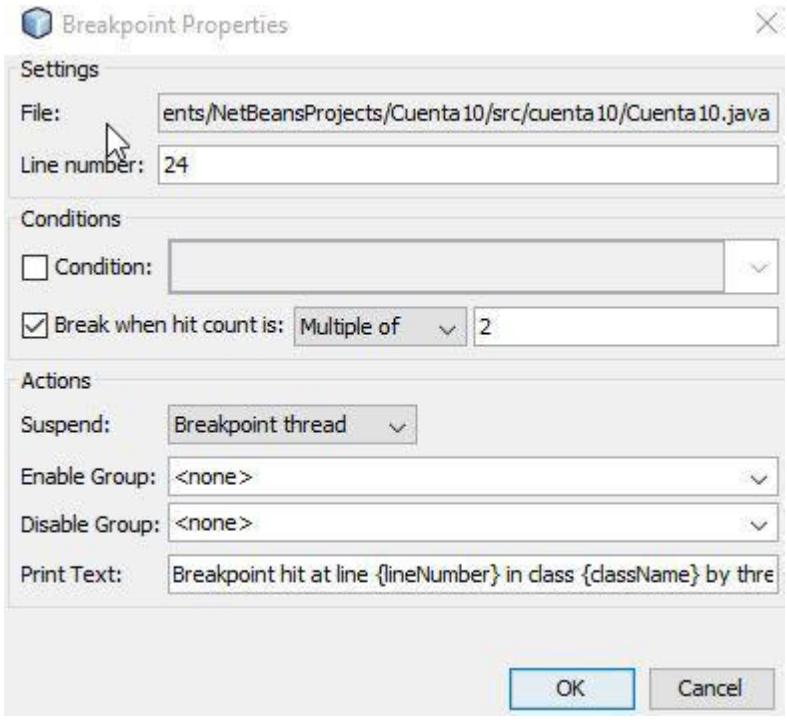
Depuramos y observamos el valor de las variables



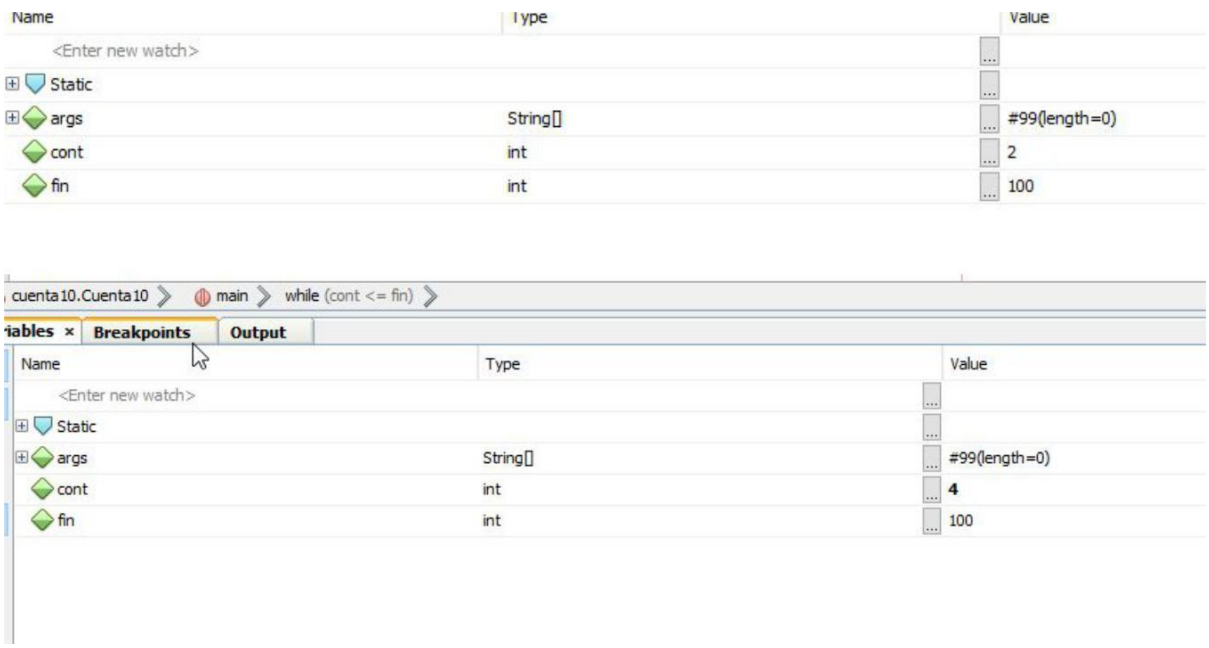


Efectivamente se detiene cuando se va a ejecutar la segunda vez el breakpoint.

Establezco ahora la siguiente condición. Detén la ejecución cuando el número de ejecuciones sea múltiplo de 2.



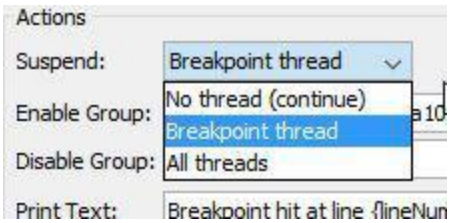
Si depuramos observamos que se detiene en 2, 4, 6, 8…



Finalmente, podemos especificar qué acciones ocurren cuando se produce un breakpoint. No entramos en ello porque no estamos programando con hilos.

Las Actions nos permiten definir que hilos se suspenderán cuando el breakpoint se ejecute. Netbeans permite suspender

- All threads → Todos los hilos
- Breakpoint thread → El hilo del breakpoint
- No threads → No suspende ninguno. Continua ejecución



Practicando

Captura pantallas mostrando la configuración de los breakpoints y la ejecución

1. Realiza un programa llamado Cuenta100 que mediante un bucle while muestre una cuenta de 1 a 100.
   - Establece puntos de ruptura para que la ejecución se detenga cuando el contador tome los siguientes valores : 3, 64, 78.
   - Establece punto de ruptura para que la ejecución se detenga cuando el contador tome el valor 70 y además el breakpoint se haya ejecutado más de 60 veces.
   - Establece punto de ruptura para que la ejecución se detenga cuando el breakpoint se haya ejecutado 57 veces

2. Realiza un programa llamado Pidonumero que pida un número por pantalla entre 0 y 10. Si el usuario introduce un número mayor que 0 y menor 10 el programa escribe “El número introducido es X” , si introduce un número equivocado sigue pidiendo número.

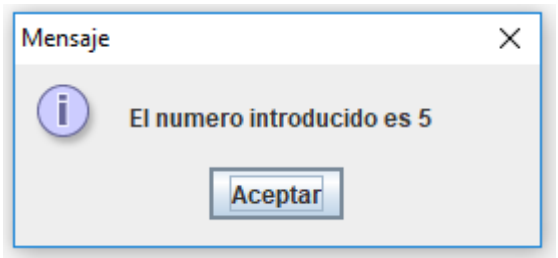
   Para pedir un número utilizaremos la siguiente instrucción que muestra un panel con el mensaje texto mensaje y guardan texto introducido en la variable texto

   ```
   String texto=JOptionPane.showInputDialog("texto mensaje");
   ```

   Y la siguiente instrucción par amostrar el resultado en una ventana de diálogo:

   ```
   JOptionPane.showMessageDialog(null, "El numero introducido es "
   + texto);
   ```

   Una vez realizado el programa establece un punto de ruptura para que el programa se detenga antes de mostrar el mensaje “El número introducido es X” siempre que el valor introducido sea 9