**Entornos de desarrollo – Alberto Ruiz**

**Ejercicio de depuración con Java y Eclipse**

Crear un nuevo proyecto con estas clases

**public class** Contador {

**private int** resultado = 0;

**public int** getResultado()

{

**return** resultado;

}

**public void** contar()

{

**for** (**int** i = 0; i < 100; i++)

{

resultado += i + 1;

}

}

}

**public class** Test

{

**public static void** main(String[] args) {

Contador contador = **new** Contador ();

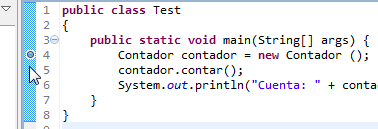
contador.contar();

System.*out*.println("Cuenta: " + contador.getResultado());

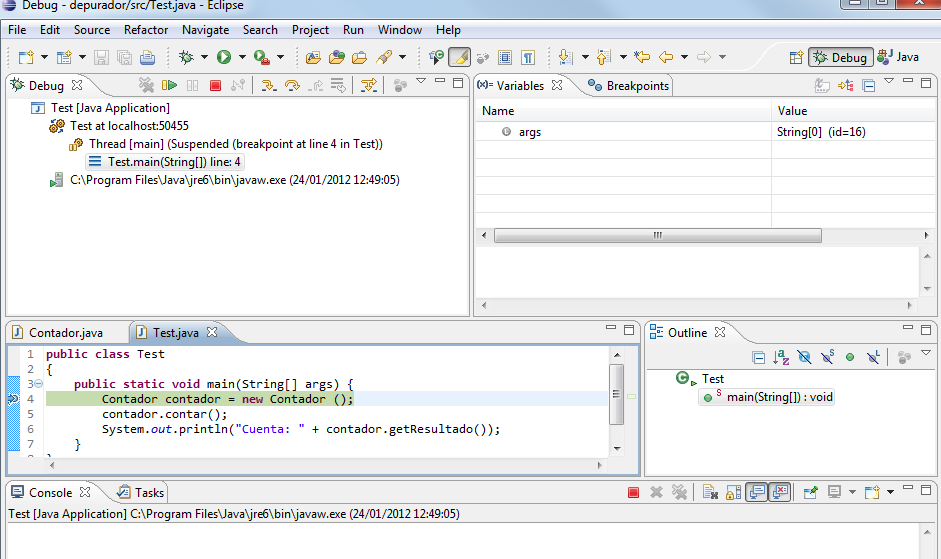
}

}

Poner un punto de parada haciendo doble clic



Comenzar la depuración (la ejecución normal no se parará en los puntos de parada). La depuración no hará nada especial si no hemos puesto puntos de parada. Se inicia con el botón del insecto (bug), y cambiamos a la vista de depuración:

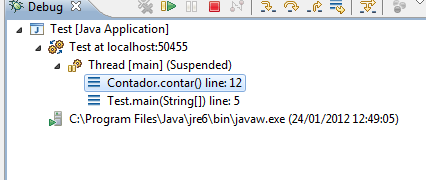


Botones de Continue, Terminate, Step into y Step (con sus atajos de teclado)

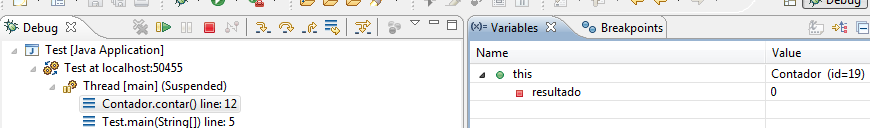


El botón de F7 lo que hace es saltar al código que llamó a la función en la que nos encontramos. Es útil si nos metimos con Step Into a examinar el código de una llamada a función, pero queremos salirnos y continuar después de la llamada

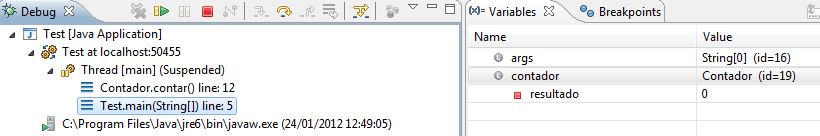
La ventana superior izquierda muestra la pila de llamadas



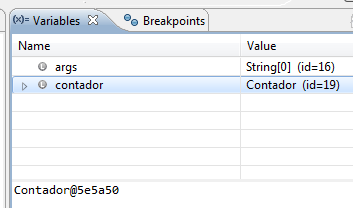
La ventana de variables muestra las variables locales visibles desde la llamda actual. Si estoy en Contador:



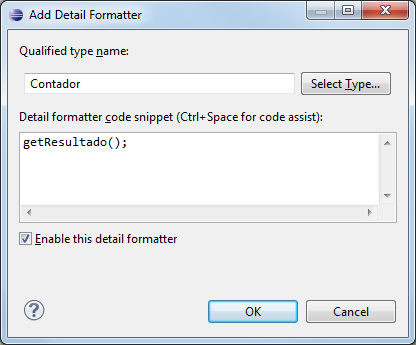
Pero si estoy en Test (puedes hacer clic en cada uno, sólo que aparecerán en gris si no están activas en este momento)



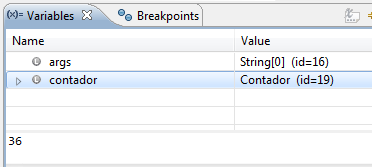
Las variables se muestran con el método predeterminado toString, que aplicado a un objeto simplemente escribe su dirección de memoria

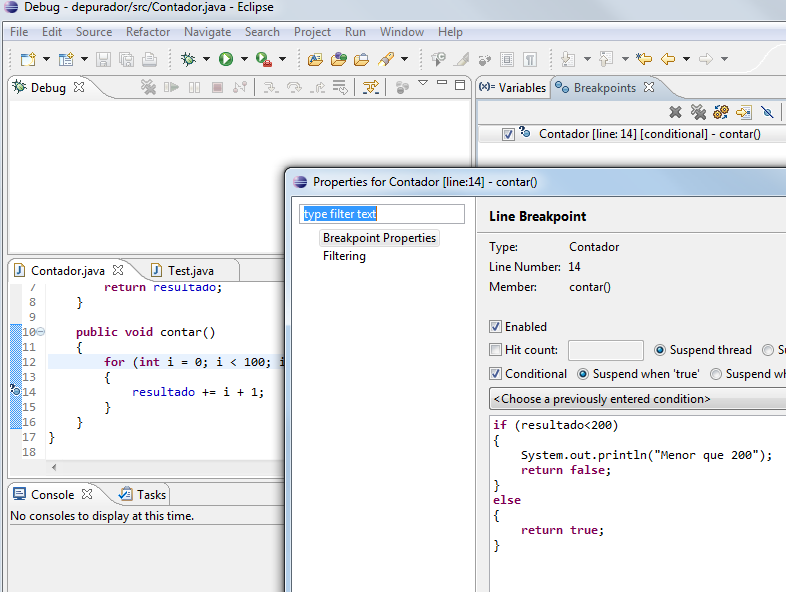


Podemos personalizar cómo quiero yo mostrar el objeto “contador”, para ello hago clic con el botón derecho y escogo New Detail Formatter

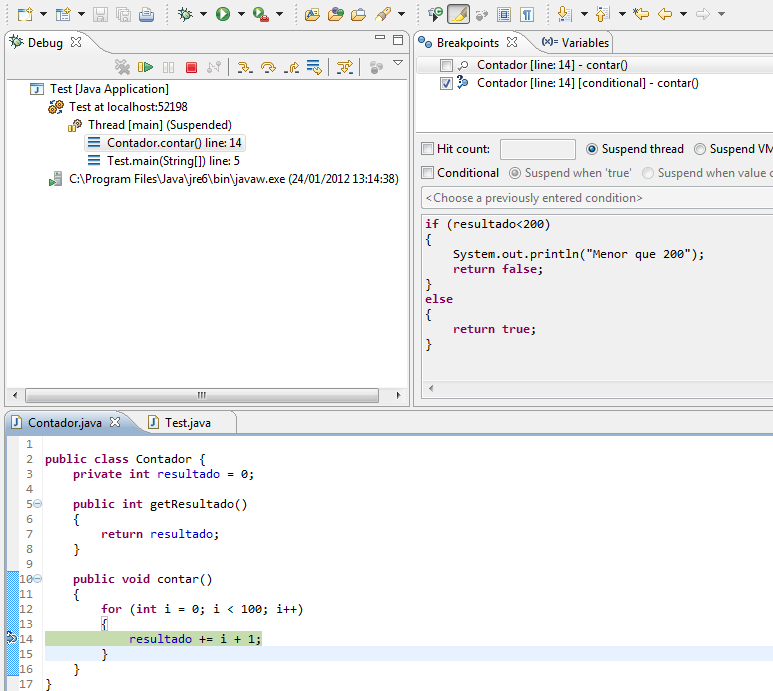


Escribo getResultado, diciendo que lo que quiero mostrar sobre contador es lo que devuelva esa llamada. Ahora veo algo así:

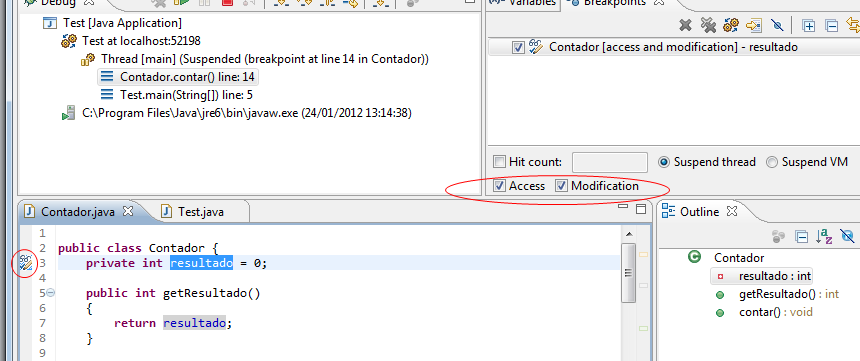


Eclipse permite puntos de parada condicionales, para personalizar cuándo o porqué se para el programa. Pon un punto de ruptura en contar() y, en la ventana breakpoints, haz clic con el botón derecho sobre el mismo. Escoge “propiedades”. Si marcas Hit counter y escribes 10, el programa se parará cuando haya pasado por este punto 10 veces. Si marcas condicional, puedes escribir una condición (que puede hacer referencia a cualquier variable disponible en ese punto del código). Cuando se cumpla (return true) se parará. Y aunque no se cumpla, podemos lanzar mensajes de depuración.

Si prefieres puedes hacerlo desde la misma ventana

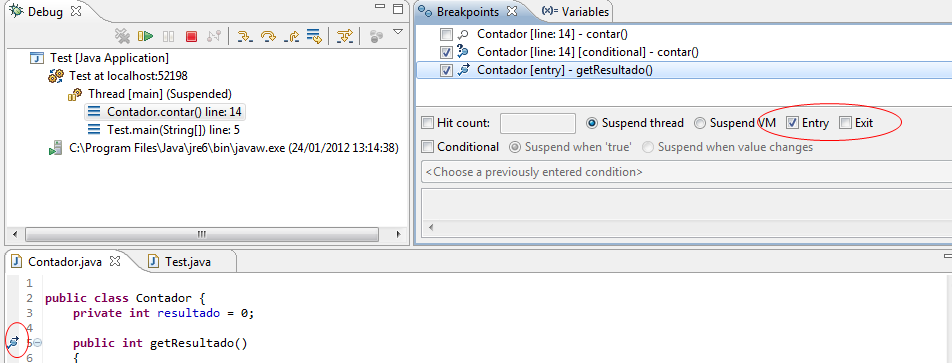


Eclipse añade además otros tipos de breakpoint. Puedes poner uno junto al atributo de una clase: en ese caso el programa se detendrá cuando el atributo sea accedido. Su icono son unas gafas. Observa que puedes marcar si se parará por acceso o por modificación (lectura o escritura).

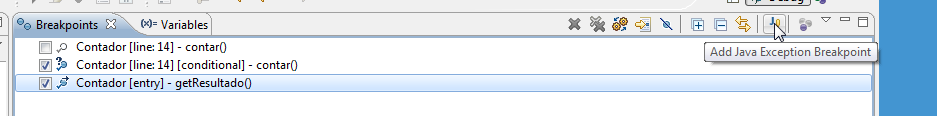


Observa también que la condición de hit counter siempre está disponible sin ni siquiera tener que entrar en las propiedades del punto de parada.

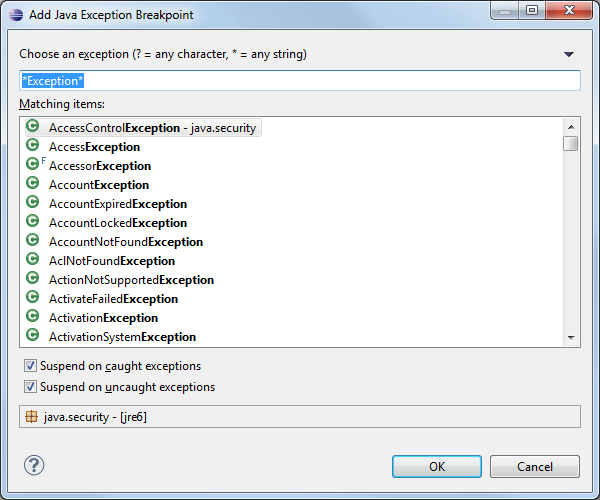
Análogamente existe un punto de parada para métodos. En este caso indicamos si queremos que se pare al entrar o al salir del método.



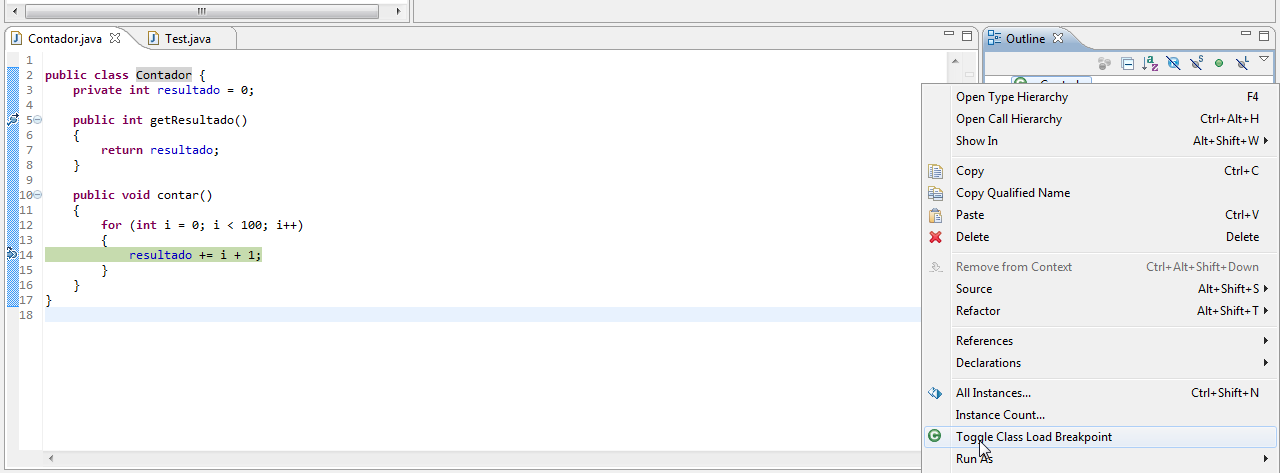
Existe también un punto de parada por excepción



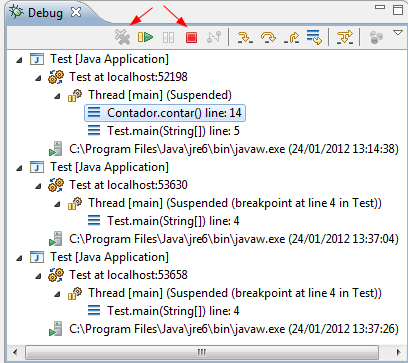
Se puede indicar si queremos que pare por alguna excepción concreta, o si queremos que pare sólo en excepciones capturadas o no capturadas



Aún hay un último tipo de punto de parada: el que para cuando se carga una clase. Haz clic con el botón secundario sobre una clase en la vista Outilne, e indica que quieres un punto de ruptura de tipo “class load”



Por último, observa que la pila de llamadas se puede complicar bastante si ejecutas varias depuraciones y “olvidas” ejecuciones en marcha detenidas en algún punto de parada. Incluso aunque llegues al final, la pila queda ahí para que puedas examinar las variables. Para “limpiar” esta ventana, utiliza los botones “Terminar” y “limpiar todas las terminadas”.



Ejercicios:

Indicar los valores de “resultado” e “i” en el momento de parada de acuerdo a las siguientes condiciones:

1. Ponemos un breakpoint en la línea 13 de Contador de forma que el programa se pare cuando pase 15 veces por el breakpoint
2. Ponemos un breakpoint en la línea 13 de Contador de forma que la ejecución se detenga cuando se cumpla la condición de que “resultado” sea mayor que 1000
3. Ponemos un breakpoint en el atributo “resultado” de forma que el programa se detenga cuando se modifique el atributo por décima vez.
4. Ponemos un breakpoint en el método getResultado() de forma que el programa se detenga al abandonar el método
5. Ponemos un breakpoint en el método contar() de forma que el programa se detenga al abandonar el método