

1. Depuración de código

1.1 Introducción

Na actividade que nos ocupa preténdense os seguintes obxectivos:

- Utilizar o contorno de desenvolvemento libre para depurar código e utilización de puntos de ruptura.
- Examinar e modificar o comportamento dun programa en tempo de execución utilizando o contorno de desenvolvemento libre.

1.2 Actividade

Introdución

A operación de depuración serve para examinar o código da aplicación en tempo de execución e buscar solucións a erros detectados. Permite executar liñas de código ata un momento no que se pode examinar o estado para descubrir problemas.

Os exemplos de depuración desta actividade utilizarán un proxecto Java denominado *estadisticos* que inclúe as clases Estadisticos.java e Main.java. A execución do proxecto permite teclear dous números enteiros positivos (o primeiro maior ou igual co segundo) e visualizar os estatísticos:

- Factorial de cada un dos números: produto de números enteiros dende 2 ata o número.
- Combinacións dos dous números: resultado de dividir o factorial do primeiro número polo produto do factorial do segundo número e o factorial da diferenza de ambos números.
- Variacións sen repetición dos dous números: resultado de multiplicar as combinacións polo factorial do segundo número.
- Variacións con repetición de dous números: resultado de elevar o primeiro número ao segundo.

O código de Main.java é:

```
/*
 * Permite teclear dos valores enteros m y n y visualiza:
 * -El factorial de m.
 * -El factorial de n.
 * -El número de variaciones sin repetición de m elementos tomados de n en n.
 * -El número de combinaciones sin repetición de m elementos tomados de n en n.
 * -El número de variaciones con repetición de m elementos tomados de n en n.
 * Se utiliza la clase Estadisticos
 */
package estadisticos;

import java.util.*;

public class Main {

    public static void main(String[] args) {
        System.out.print("\nCALCULOS ESTADÍSTICOS\n");
        Scanner teclado = new Scanner(System.in);
        boolean error;
        int m, n;
        do {
            try {
                error = false;
                System.out.print("Teclee m (>= 0): ");
                m = teclado.nextInt();
                System.out.print("Teclee n (>= 0 y <= m): ");
                n = teclado.nextInt();
                Estadisticos es = new Estadisticos(m, n);
```

```

        System.out.printf("Permutaciones(%d) = %f\n",n, es.factorial(n));
        System.out.printf("Permutaciones(%d) = %f\n",m, es.factorial(m));
        System.out.printf("Variaciones(%d,%d) = %f\n",m, n, es.variaciones());
        System.out.printf("Combinaciones(%d,%d) = %f\n",m, n, es.combinaciones());
        System.out.printf("Variaciones con repetición(%d,%d)= %f\n",m, n, es.varia-
ciones_repeticion());
    } catch (NumberFormatException e) {
        teclado.nextLine(); //para limpiar INTRO del buffer de teclado
        System.out.println("Error en la conversión");
        error = true;
    } catch (InputMismatchException e) {
        teclado.nextLine(); //para limpiar INTRO del buffer de teclado
        System.out.println("Error. El dato tecleado no es válido");
        error = true;
    } catch (Exception e) {
        teclado.nextLine(); //para limpiar INTRO del buffer de teclado
        System.out.println(e.getMessage()); // Muestra el error
        error = true;
    }
} while (error);
}
}

```

O código de Estadisticos.java é:

```

package estadisticos;

public class Estadisticos {

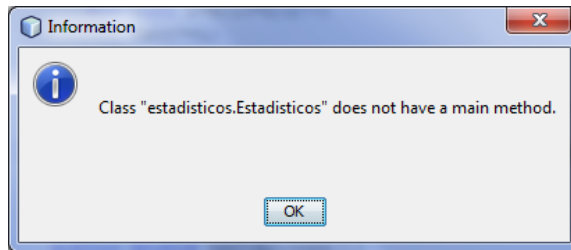
    private int m;
    private int n;

    public Estadisticos(int m, int n) throws Exception {
        if (m < 0 || n < 0 || m < n) {
            throw new Exception("Error. " +
                "Los argumentos tienen que ser >=0 y el primero >= que el segundo");
        }
        this.n = n;
        this.m = m;
    }
    /* Cálculo Factorial o permutaciones de x */
    public double factorial(int x) throws Exception {
        double resultado = 1;
        for (int i = 2; i <= x; i++) {
            resultado *= i;
        }
        return resultado;
    }
    /* Cálculo Combinaciones de m elementos tomados de n en n */
    public double combinaciones() throws Exception {
        double combi = factorial(m)/(factorial(n)*factorial(m-n));
        return combi;
    }
    /* Cálculo Variaciones de m elementos tomados de n en n */
    public double variaciones() throws Exception {
        double vari = combinaciones()*factorial(n);
        return vari;
    }
    /* Cálculo Variaciones con repetición de m elementos tomados de n en n */
    public double variaciones_repeticion() throws Exception {
        double varirepe = Math.pow(m, n);
        return varirepe;
    }
}

```

Sesión de depuración

Para iniciar unha sesión de depuración é indispensable que o arquivo ou proxecto a depurar teña método *main*, é dicir, ten que ser posible executalo, porque se non saíría unha mensaxe de erro do tipo:

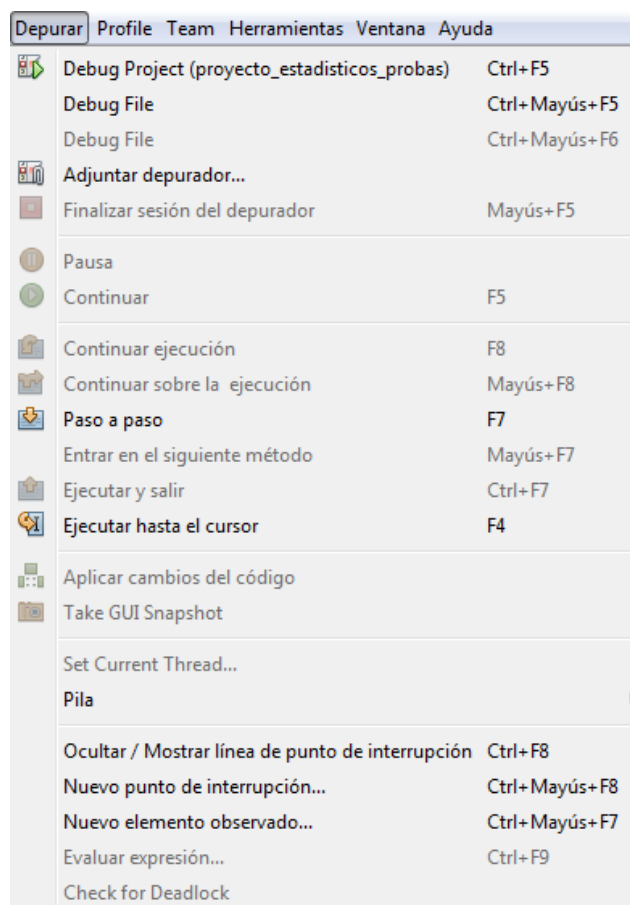


NetBeans permite iniciar sesión de depuración dun proxecto ou dun arquivo e elixir como empezar a depuración de diferentes maneiras. Por exemplo permite depurar:


- Un proxecto establecido como proxecto principal, premendo Ctrl-F5 ou indo ao menú principal e elixindo *Depurar->Debug Project*.
- Un proxecto calquera dende a ventá *Proyectos*, seleccionando o proxecto, facendo clic dereito, e elixindo *Depurar*.
- Un arquivo fonte que se estea editando nese momento, indo ao menú principal e elixindo *Depurar->Debug File*.
- Un arquivo calquera dende a ventá *Proyectos*, seleccionando o arquivo, facendo clic dereito, e elixindo *Debug File*.

En tódolos casos anteriores, a sesión de depuración empeza no arquivo seleccionado ou na clase principal do proxecto seleccionado e sigue a execución ata que finalice normalmente o programa, encontre algún erro ou algún punto de interrupción.


No menú *Depurar*, pódese ver que a sesión de depuración tamén pode iniciarse para que a execución se faga paso a paso ou ata a liña na que estea o cursor.



Execución ata o cursor

A opción *Ejecutar hasta el cursor* permite executar o programa ata a localización do cursor no arquivo que se está editando e pausa o programa ata que se lle indica a seguinte operación a realizar na depuración. O arquivo editado debe ser chamado dende a clase principal do proxecto principal. Para realizar esta depuración débese situar o cursor no código fonte, premer F4 ou seleccionar no menú principal *Debug>Ejecutar hasta el cursor* ou seleccionar  na barra de ferramentas da depuración.

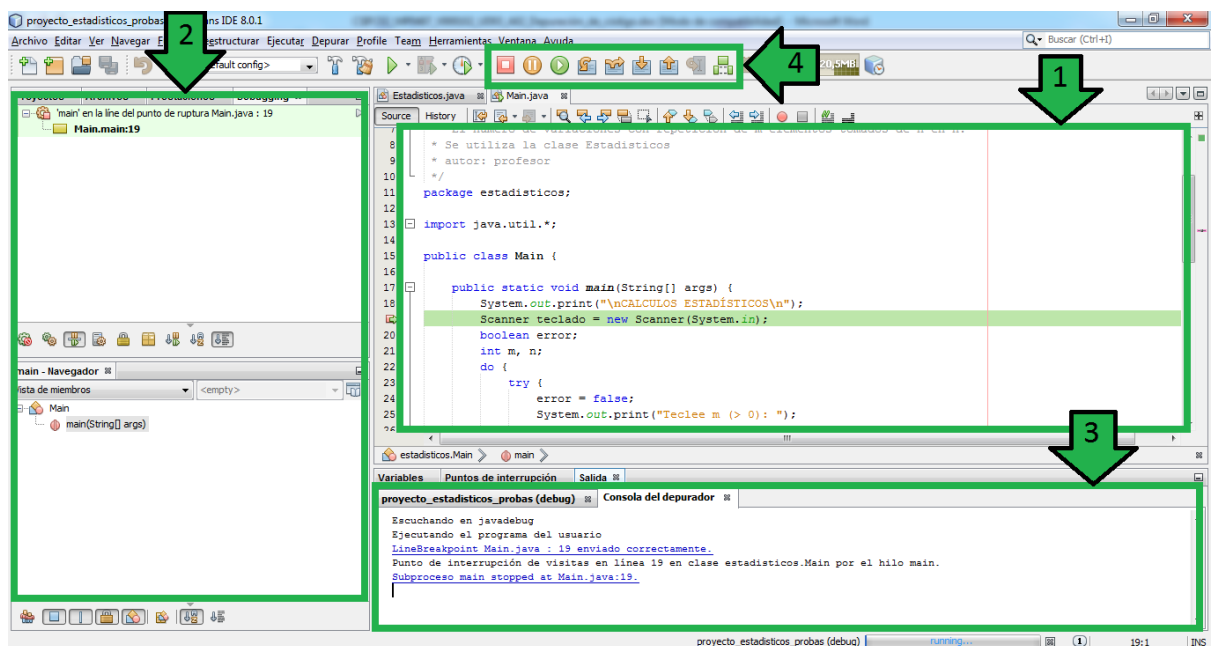
Execución paso a paso

A opción *Paso a paso* permite executar o programa liña a liña e pausa a execución ata que se lle indica a seguinte operación a realizar na depuración. Para iniciar a sesión de depuración paso a paso, elíxese *Depurar-> Paso a paso* no menú principal ou prémese F7 ou  na barra de ferramentas da depuración e a execución deterase na liña coa primeira instrución executable. Cada vez que se queira executar paso a paso a seguinte liña, haberá que volver a pulsar F7. Se a liña de código está composta de varios métodos, aparecerá seleccionado cun borde negro o que se vai a executar paso a paso e poderase utilizar a tecla de tabulación para seleccionar un dos outros métodos. Pódese utilizar a tecla Enter ou F7 para executar paso a paso o método seleccionado.

```
24      /* Cálculo Combinaciones de m elementos tomados de n en n*/  
25      public double combinaciones() throws Exception {  
26          double combi = factorial(m) / (factorial(n) * factorial(m-n));  
27          return combi;  
28      }
```

Pantalla de depuración

A pantalla de depuración aparece despois de iniciada a depuración e ten varias zonas. Por exemplo, despois de iniciada unha sesión de depuración paso a paso aparecen as zonas que se ven na imaxe seguinte.



Zona 1: Zona co código fonte en depuración. A seguinte liña a executar no proceso de depuración aparece marcada con cor de fondo verde e unha frecha verde na marxe esquerda.

Zona 2: **Ventá *Debugging*** con información sobre os procesos que se están executando. Na parte inferior desa ventá aparece un menú de iconas para poder cambiar a vista da información:

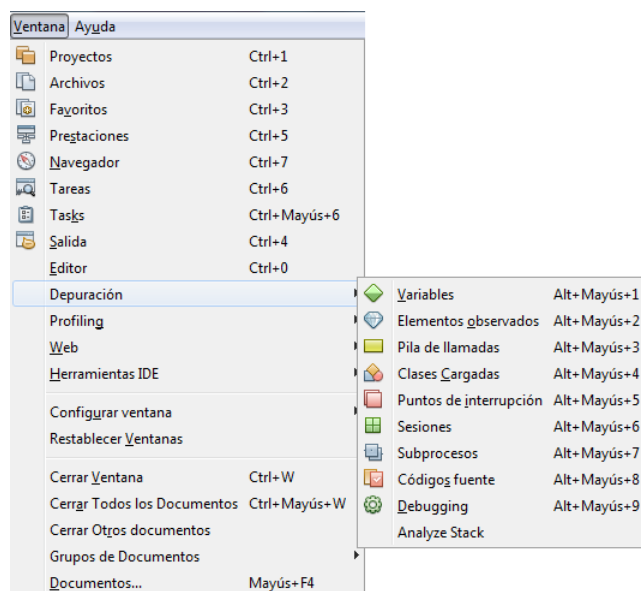


Zona 3: **Zona de ventás:**

- Ventás de saída ou *output* que se subdivide en:
 - *Consola del depurador* con información sobre o proceso de depuración.
 - *estadísticos(debug)*. Neste proxecto en concreto que ten entradas e saídas dende a consola realizaranse aquí as entradas de datos e verase a saída de resultados.
- Ventá *Variables* na que se pode ver e cambiar información sobre as variables locais e expresións.
- Ventá *Puntos de interrupción* ou *breakpoints* na que se pode ver e cambiar información sobre os puntos de interrupción.

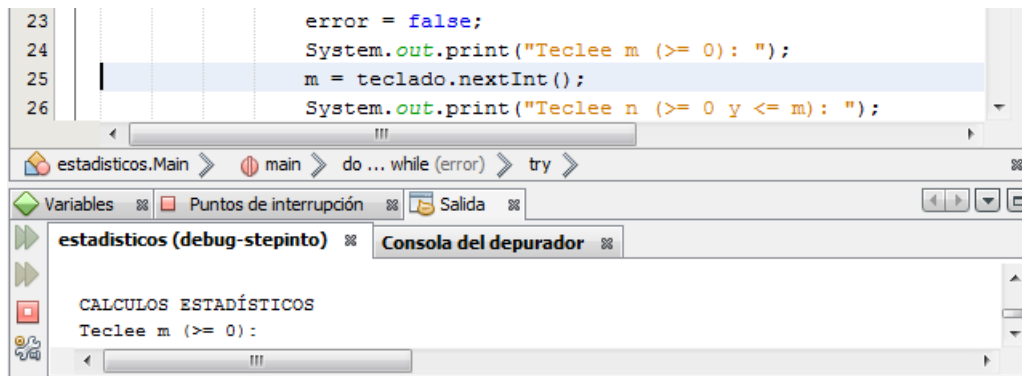
Zona 4: **Barra de ferramentas de depuración** para indicar o seguinte paso a realizar na depuración. Algunha das mesmas aparecen tamén activadas no menú principal *Depurar*.

Se non se ve algunha das ventás anteriores, pode accederse á opción *Ventana->Depuración* do menú principal e elixir a ventá que se desexa ver.



Durante o proceso de depuración pode ocorrer que a execución dunha liña de código precise dunha entrada por teclado e entón:

- A liña de código que ten a entrada pasa de ter fondo verde a ter fondo azul na ventá de edición.
- O proceso de depuración está detido ata que se teclee o dato na ventá de saída *estadísticos..*



Explicación rápida do significado de cada icona da barra de ferramentas de depuración (zona 4):

Icona	Descrición	Detalle
	Finalizar sesión do depurador (Maiúsculas+F5)	Finalizar instantaneamente a depuración.
	Pausa	Facer unha pausa no proceso de depuración.
	Continuar (F5)	Continuar a execución normal do programa despois de facer unha pausa.
	Continuar execución (F8)	Executa unha instrución. Se contén unha chamada a un método, este execútase completo sen parar en cada unha das instrucións.
	Continuar sobre a execución (Maiúsculas+F8)	Continuar sobre a execución. É un refinamento de F8 para expresións con chamadas a métodos. Cada vez que nunha sesión de depuración paso a paso se utiliza este comando sobre unha expresión con chamada a métodos, párase a depuración antes de executar a chamada ao método actual podendo ver o historial dos valores de retorno dos métodos inmediatamente previos e o valor dos argumentos do método actual na ventá de Variables locais. Pode ser útil cando os valores de retorno dun método non se gardan nunha variable e por tanto non poden ser inspeccionados en tempo de depuración.
	Paso a paso (F7)	Executar Paso a paso. Permite entrar na execución paso a paso do método da liña actual. De haber máis dun método na liña, poderase: escoller o método que se vai depurar paso a paso utilizando as teclas de movemento do cursor ou a tecla tab e confirmando con F7, executar normalmente (F7) Nesta versión de NetBeans, por defecto, Step Into (F7) entra na execución paso a paso dos métodos API de Java. De querer que non se abran estes métodos e se executen, haberá que premer F8 en lugar de F7.
	Executar e saír (Ctrl+F7)	No caso de estar depurando dentro dun método, Ctrl+F7 finaliza a sesión de depuración do método e volve ao método que chamou ao actual. No caso de utilizarse no método principal, finaliza a sesión de depuración.
	Executar ata o cursor F4	Executar ata o cursor e espera instrucións para continuar coa depuración.
	Aplicar cambios do código	Aplicar cambios no código.
		Premer para activar a recollida de lixo.

Inspección e modificación de variables, expresións e métodos

No código fonte

Durante unha sesión de depuración pódense ver o tipo e valor dunha variable situando o cursor sobre ela no código fonte:

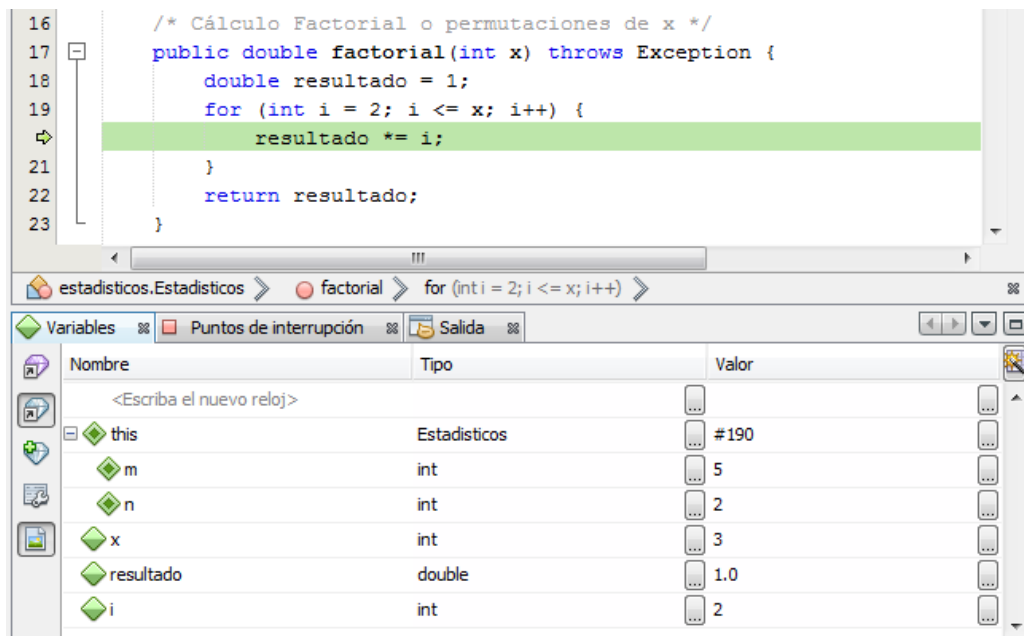
```


17 public double factorial(int x) throws Exception {
18     double resultado = 1;
19     for (int i = 2; i <= x; i++) {
20         resultado *= i;
21     }
22     return resultado;
23 }

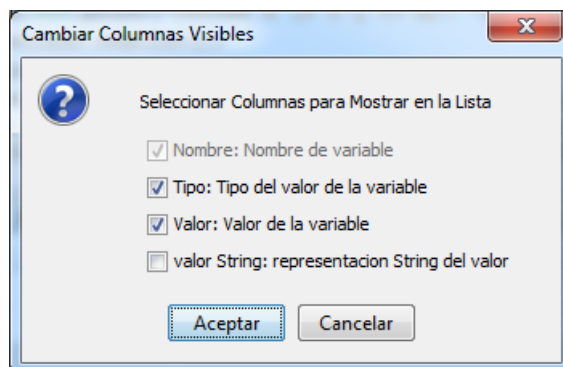
```

Na ventá Variables


Durante unha sesión de depuración pódese ver información sobre as variables locais na ventá *Variables* para variables locais e atributos de clase se existen.



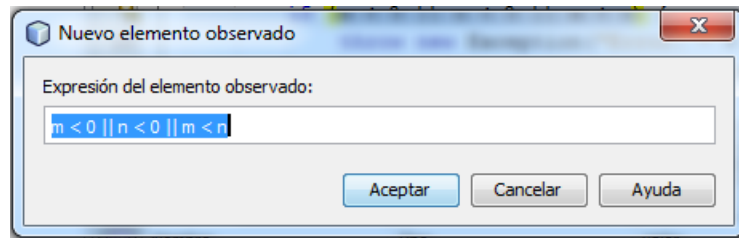
A icona situada á dereita  permite modificar a información que se ve.



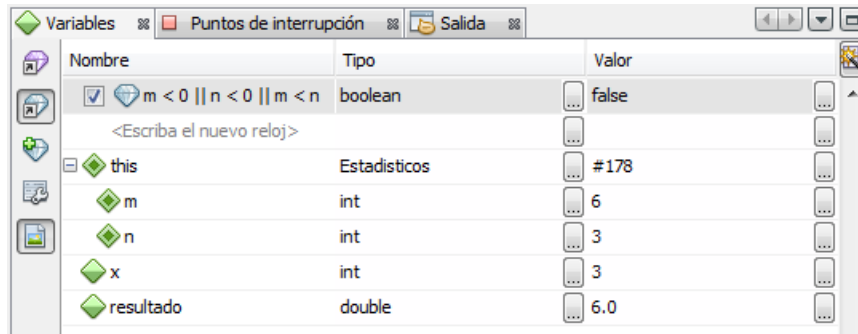
Ademais das variables locais pódense engadir expresións para observar (*watches*). Isto pódese facer de varias maneiras:

- Seleccionar a expresión no arquivo fonte editado, premer clic dereito e elixir *Nuevo elemento observado* ou premer Ctrl+Maiúsculas+F7.
- Seleccionar no menú principal *Depurar->Nuevo elemento observado*.
- Premer na icona  da ventá *Variables*.
- Teclear o novo elemento na liña da ventá *Variables* que pon *<Escriba el nuevo reloj>*.

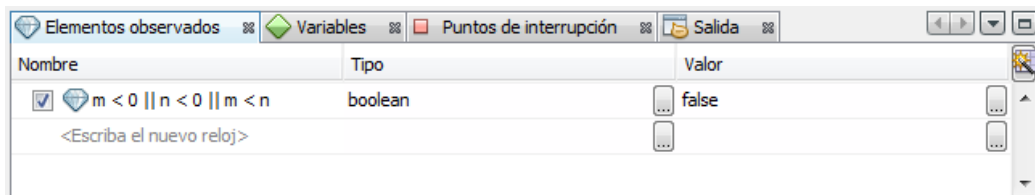
En calquera dos tres primeiros casos, aparece unha ventá na que ten que quedar definida a expresión que se quere observar.



A expresión aparece engadida na ventá *Variables* como un *watch*.



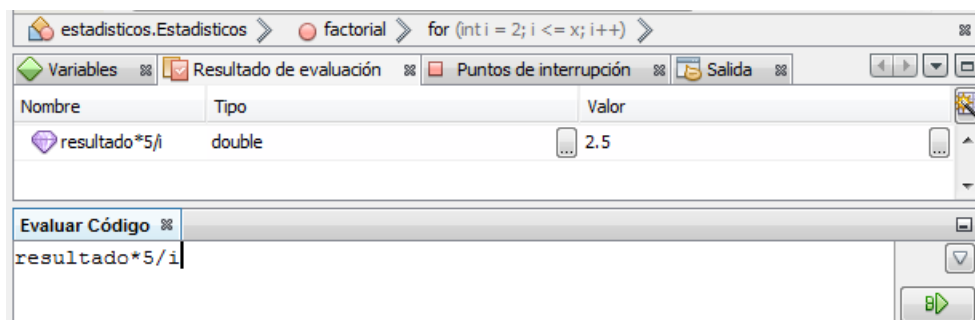
Os elementos observados poden moverse entre a ventá *Variables* e a ventá *Elementos observados* utilizando o interruptor




Na ventá *Variables* tamén se pode facer clic dereito sobre o nome dunha variable ou expresión e facer cambios como por exemplo: eliminar desa ventá unha variable ou expresión, eliminar todas, ver o valor noutro formato ou editar.

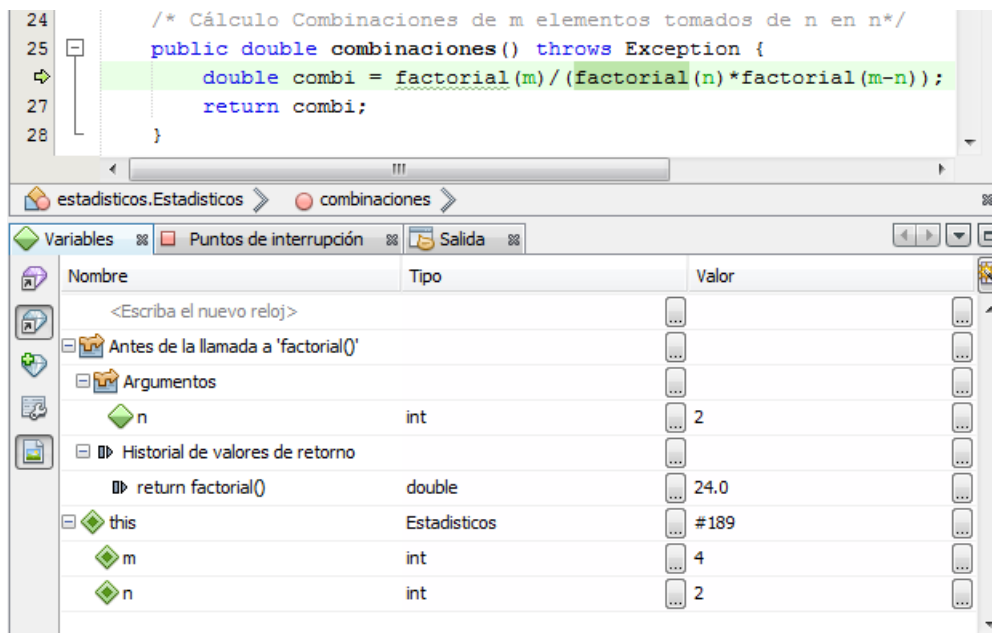
Co menú *Depuración->Evaluar expresión*

Durante unha sesión de depuración pódese ver o resultado dunha expresión dende o menú principal elixindo *Depurar->Evaluar expresión* ou premendo Ctrl+F9; ábrese a ventá *Evaluar Código* na que se pode teclear a expresión e avaliar o resultado nese momento premendo sobre o botón




Coa opción de depuración *Continuar sobre la ejecución*

Pódense ver os valores de retorno dos métodos previos (*Historial de valores de retorno*) e o valor dos parámetros do método seguinte (*Antes de la llamada a ...*), na ventá *Variables* cando nunha sesión de depuración se utiliza Maiúsculas+F8 (*Continuar sobre la ejecución*) ou se preme na icona  da barra de depuración sobre unha expresión con chamada a un método.



Modificación

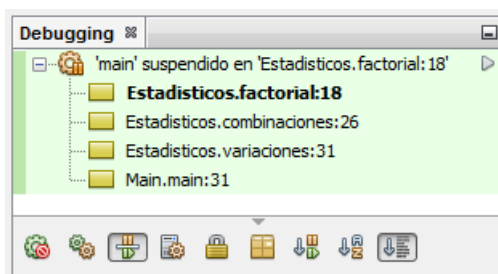
Na ventá de avaliar expresións, na de variables e na de elementos observados pódese modificar o valor dunha variable e continuar o proceso de depuración con ese valor. Para iso faise clic ao carón do valor actual da variable, ou utilízase a icona  cando exista, para teclear o novo valor, prémese en *Aceptar* e continúaase coa depuración.

Pila (call stack)

A utilización da pila de chamadas é especialmente útil cando se utilizan varios fíos ou subprocesos durante a execución. Cando se inicia unha sesión de depuración, ábrese automaticamente a ventá *Debugging* con información sobre os subprocesos existentes e a pila de chamadas de cada un dos subprocesos suspendidos ou pausados; só unha desas chamadas é a chamada actual.









Ventá *Debugging*

A ventá *Debugging* ten o seguinte aspecto:












Esta imaxe indica unha sesión de depuración na que a execución de `main()` queda interrompida na liña 31 por unha chamada ao método `variaciones()`; a execución de `variaciones()` queda interrompida na liña 31 por unha chamada ao método `combinaciones()`; a execución de `combinaciones()` queda interrompida na liña 26 por unha chamada ao método `factorial()` e `factorial()` é o método actual pausado na liña 18.

A última chamada realizada é a chamada que se considera actual e indícase na pila en letra grosa. De consultar as variables locais veríanse as da chamada actual. Se os arquivos fontes están dispoñibles, pódese facer clic dereito na chamada e elixir *Ir a fonte* para ver o código fonte da chamada. A carón de cada proceso aparece unha icona con información sobre o proceso:

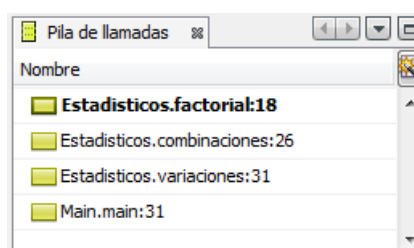
Icons	Description
	Indicates a thread that is running
	Indicates a thread suspended by hitting a breakpoint
	Indicates a suspended thread
	Indicates a thread group where all threads are running
	Indicates a thread group where all threads are suspended
	Indicates a thread group with running and suspended threads
	Indicates a call stack frame
	Indicates a call stack frame group

Pódese cambiar a vista desta información utilizando o menú de iconas da parte inferior:

Button	Description
	Show thread groups
	Shows or hides the controls for suspending and resuming threads.
	Show system threads
	Show suspended and current threads only
	Show monitors
	Show qualified names
	Sort by suspended/resumed state
	Sort by name
	Sort by default



Ver a pila

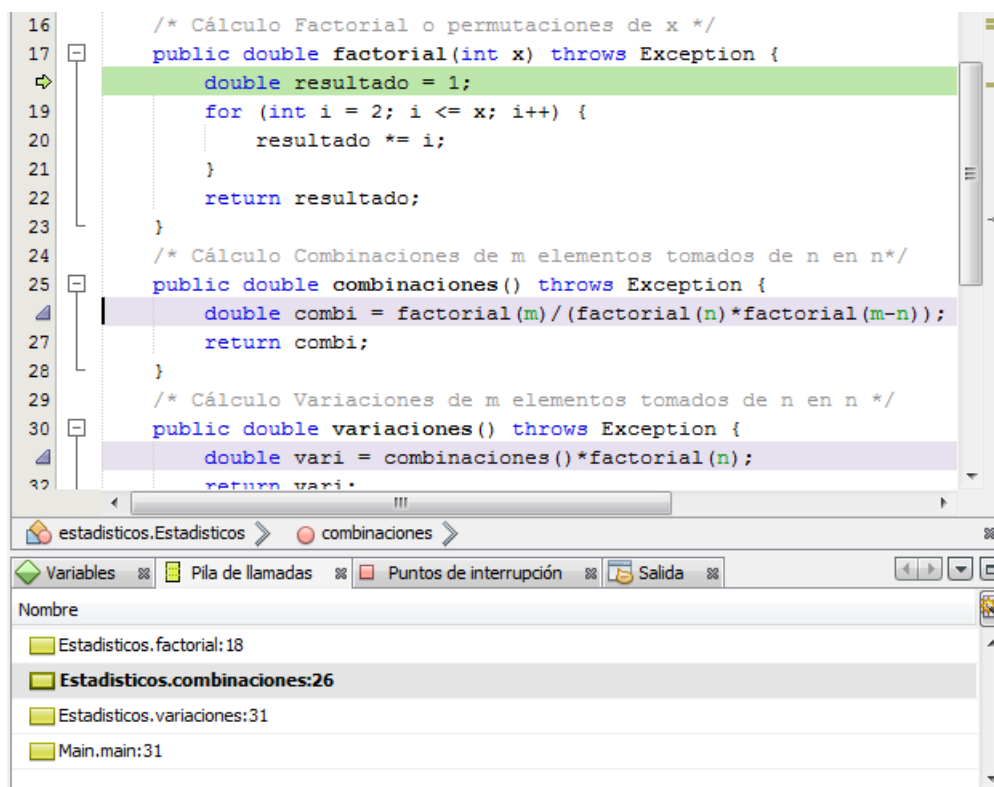
A información sobre a pila de chamadas tamén se pode ver indo ao menú principal e elixindo *Ventana->Depuración->Pila de llamadas* ou premer `Alt+Maiúsculas+3` e abrírase a ventá *Pila de llamadas* na zona de ventás. A información para cada chamada está marcada por unha icona e a descrición da chamada.



Cambios e movementos na pila

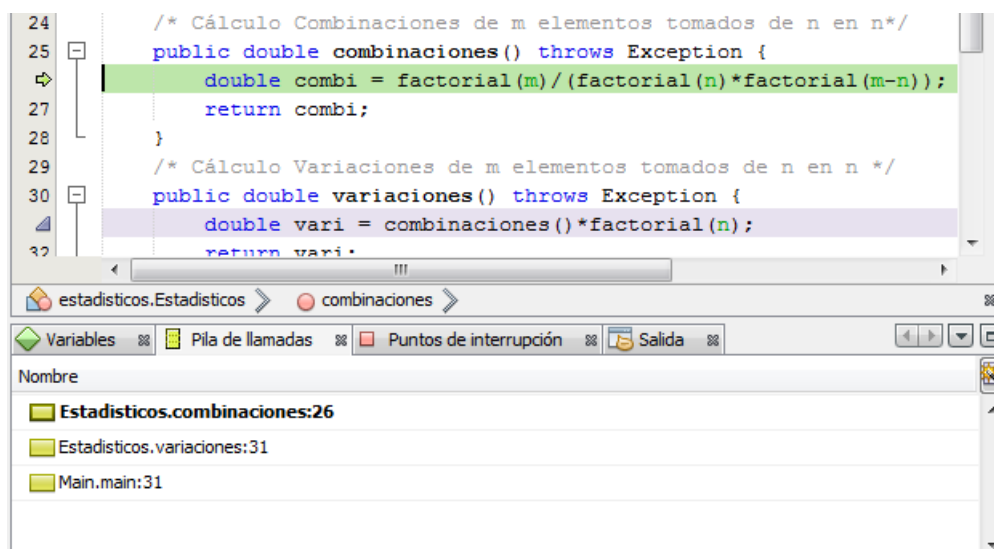
Utilizando *Depurar->Pila* do menú principal, pódese elixir *Asignar actual al emisor de la llamada (Make caller current)* ou *Asignar actual al receptor de la llamada (Make callee*

current) para moverse pola pila poñendo como chamada actual a chamada anterior á chamada actual ou a seguinte. Nese caso, na ventá de edición visualízase a liña de código corresponde á esa chamada marcada como , e na ventá de variables, veranse as variables correspondentes a esa chamada, pero non se cambia a cabeceira da pila, nin varían as chamadas da pila nin o marcador de depuración .



Para designar como actual calquera chamada da pila, débese iluminar a chamada na pila, facer clic dereito e elixir *Asignar actual*. Visualízase na ventá de edición, a liña de código corresponde á esa chamada e na ventá de variables as variables que se poidan ver nesa chamada, pero non cambia a cabeceira da pila, as chamadas da pila nin o marcador de depuración.

Utilizando dende o menú principal *Depurar->Pila ->Hacer emerger llamada superior (Top most Call)*, elimínase a chamada actual na cabeceira da pila, a seguinte chamada da pila pasa á cabeceira e por tanto a ser a chamada actual e o marcador da depuración no código fonte móvese á instrución que chamou á chamada á borrada. De seguir coa depuración, a chamada será repetida.



Iluminando unha chamada na ventá de depuración, facendo clic dereito nela e elixindo *Salta a aquí (Pop To Here)*, pódese poñer esa chamada como cabeceira da pila, elimínanse as chamadas posteriores da pila e o marcador de depuración actualízase. A eliminación dunha chamada da pila non implica que se eliminarán os efectos causados por esas chamadas. Por exemplo, se unha chamada abre a conexión cunha base de datos e esa chamada bórrase da pila, a base de datos permanecerá aberta.

Copiar pila en formato texto


Dende a ventá *Pila de llamadas*, pódese facer clic dereito e elixir *Copiar pila* para pasar a lista de elementos da pila ao portapapeles en formato texto. Por exemplo para a pila anterior:

```
estadisticos.Estadisticos.combinaciones(Estadisticos.java:26)
```

```
estadisticos.Estadisticos.variaciones(Estadisticos.java:31)
```

```
estadisticos.Main.main(Main.java:31)
```

Aplicar cambios no código

Pódense facer certas modificacións no código en tempo de depuración sen ter que reiniciar o programa. Para corrixir o código, débese corrixir na ventá de edición, ir ao menú principal e elixir *Depurar-> Aplicar cambios del código* ou premer en  na barra de ferramentas de depuración, para recompilar e facer a reparación do código fonte.

Consideracións xerais:

- Se hai erros durante a compilación, non se realizan os cambios e hai que arranxar os erros.
- Se non hai erros, o código obxecto resultante cambiarase polo que se estaba executando na depuración e:
 - se os cambios do código se fan dentro do método actual que se está a depurar, a pila de chamadas modifícase eliminando a chamada a ese método para permitir volver a chamar a ese módulo, pero
 - se os cambios se fan despois de haber chamado ao método, non se modificará a pila e para utilizar de novo o código modificado, debe eliminar as chamadas da pila que conteñan ese código.
- Están excluídas as seguintes modificacións:
 - Cambiar un modificador dun campo, un método ou unha clase.
 - Agregar ou quitar métodos ou campos.
 - Cambiar a xerarquía de clases.
 - Cambiar clases que non foron cargadas na máquina virtual.

Punto de interrupción

Definición

Un punto de interrupción ou ruptura ou breakpoint é unha marca no código fonte que indica ao depurador que se deteña nese punto e espere instrucións para continuar, podendo nese tempo facer operacións de inspección de variables, expresións ou código. Os puntos de interrupción de Java defínense a nivel global e afectan a tódolos proxectos que inclúan o código fonte que ten o punto de interrupción. Por exemplo, se *Estadisticos.java* tivera un punto de interrupción no método *factorial()*, cada vez que se depure un proxecto que inclúa esa clase, a sesión de depuración deteríase nese método. NetBeans permite varios tipos de puntos de interrupción:

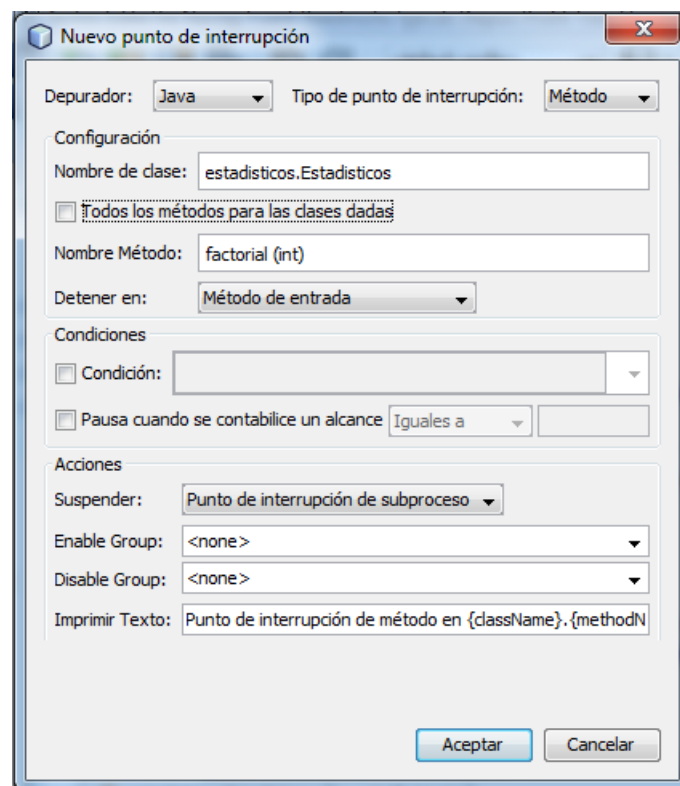
- Liña: para que se pare ao chegar a esa liña.

- Clase: para que se pare no momento de cargar a clase.
- Excepción: para que se pare cando se detecte unha excepción independentemente de se o programa controla esa excepción.
- Campo: para que se pare cando se accede e/ou modifica o campo dunha clase.
- Método: para que se pare cando se entra e/ou se sae dun método.
- Subproceso: para que se pare cando se empeza e/ou finaliza un subproceso ou fío (*thread*).

Establecer un punto de interrupción

Para establecer un punto de interrupción, haberá que seleccionar o elemento do código no que se desexa establecer o punto de interrupción e elixir no menú *Depurar->Nuevo punto de interrupción* ou premer Crtl+Maiúsculas+F8. Aparece a caixa de diálogo *Nuevo punto de interrupción* con información por defecto relacionada co elemento seleccionado e na que terán que facerse os axustes convenientes. O IDE indica o punto de interrupción establecido mediante unha icona na marxe esquerda do código fonte e os puntos de interrupción de liña indícaos ademais poñendo a liña con fondo roxo.

Na seguinte imaxe móstrase un exemplo de punto de interrupción de método para que a depuración se interrompa cando se entre no método factorial da clase Estadísticos:



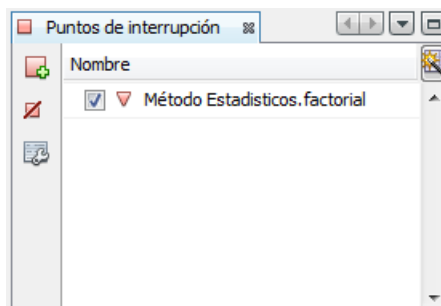
No código fonte aparecerá o punto de interrupción de método marcado cunha icona como aparece na seguinte imaxe:

```

16      /* Cálculo Factorial o permutaciones de x */
17      public double factorial(int x) throws Exception {
18          double resultado = 1;
19          for (int i = 2; i <= x; i++) {
20              resultado *= i;
21          }
22          return resultado;
23      }

```

Durante a sesión de depuración, NetBeans comproba a validez dos puntos de interrupción e se o encontra non válido indícao mediante a icona "rota" no código fonte e mostra unha mensaxe de erro na consola do depurador. Ademais é posible ver a ventá *Puntos de interrupción* con información sobre os puntos de interrupción. Se a ventá non está aberta pódese abrir dende o menú principal *Ventana->Depuración->Puntos de interrupción* ou premendo Alt+Maiúsculas+F5.



As iconas posibles para marcar os puntos de interrupción son:

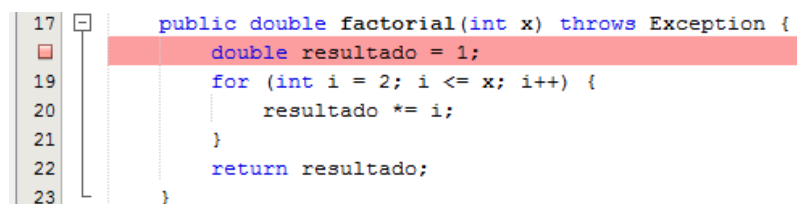
Annotation	Description
	Breakpoint
	Disabled breakpoint
	Invalid breakpoint
	Multiple breakpoints
	Method or field breakpoint
	Disabled method or field breakpoint
	Invalid method or field breakpoint
	Conditional breakpoint
	Disabled conditional breakpoint
	Invalid conditional breakpoint
	Program counter
	Program counter and one breakpoint
	Program counter and multiple breakpoints
	The call site or place in the source code from which the current call on the call stack was made
	Suspended threads
	Thread suspended by hitting a breakpoint

Un punto de interrupción de liña é un os máis utilizados e por iso hai varias maneiras de establecelos:

- O máis sinxelo é facer clic sobre a marxe esquerda da ventá de edición á altura da liña na que se desexa colocar o punto de interrupción.
- Colocar o cursor sobre a liña na que se encontre a instrución onde queremos poñer dito punto, facer clic co botón dereito e seleccionar a opción *Ocultar/Mostrar liña de punto de interrupción* ou premer as teclas Ctrl + F8.

- Colocar o cursor sobre a liña na que se encontre a instrución onde queremos poñer dito punto, e no menú principal elixir *Depurar-> Ocultar/Mostrar liña de punto de interrupción* ou premer as teclas Ctrl + F8.

No código fonte aparecerá o punto de interrupción marcado como se indica na seguinte imaxe.



```

17 public double factorial(int x) throws Exception {
18     double resultado = 1;
19     for (int i = 2; i <= x; i++) {
20         resultado *= i;
21     }
22     return resultado;
23 }

```

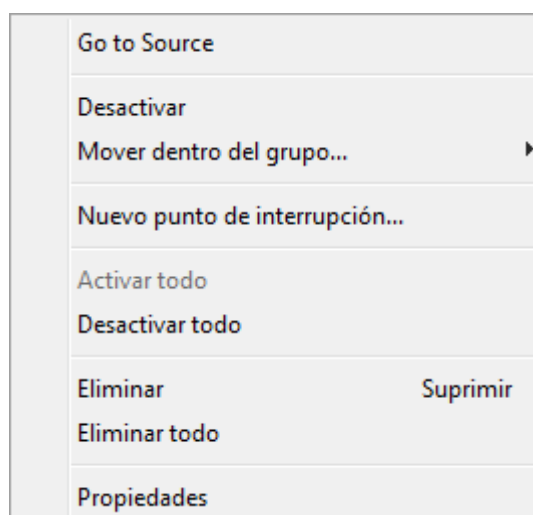
Executar ata o punto de interrupción

Para poder executar un proxecto principal ata un punto de interrupción, hai que ter definido o punto de interrupción e depurar o proxecto elixindo por exemplo no menú principal a opción *Depurar->Debug project(nome de proxecto)*, que executará o programa principal ata o primeiro punto de interrupción, ou excepción ou ata o final se non existen puntos de interrupción. A partir do punto de interrupción, pódese seguir depurando coas opcións xa vistas.

Modificar un punto de interrupción

Dende que se establece un punto de interrupción, pódese desactivar (queda rexistrado pero non en uso) se é que está activado, activar se é que estaba desactivado, eliminar (desaparece), ou modificar. Todas estas operacións empézanse a realizar dende a ventá *Puntos de interrupción* e son:

- Unha forma rápida de activar ou desactivar un punto é marcar ou desmarcar o textbox correspondente na ventá anterior.
- Unha forma rápida de eliminar un punto é colocar o cursor sobre o nome do punto de interrupción na ventá anterior e premer Supr.
- Tódalas operacións de modificación dun punto de interrupción pódense facer colocando o rato sobre o nome do punto de interrupción na ventá anterior e facendo clic co botón dereito. Aparece unha lista de operacións posibles:



- No caso de estar sobre un punto de interrupción activo, na ventá de opcións aparece dispoñible a opción *Desactivar*; se estivera desactivado, aparecería no seu lugar a opción *Activar*.

- As opcións de desactivar todo, activar todo ou eliminar todo terán efecto sobre tódolos puntos de interrupción da ventá.
- A opción *Propiedades* visualiza a ventá *Propiedades de punto de interrupción*, na que se pode axustar a configuración do punto de interrupción.
- Tamén se pode crear un novo punto de interrupción.

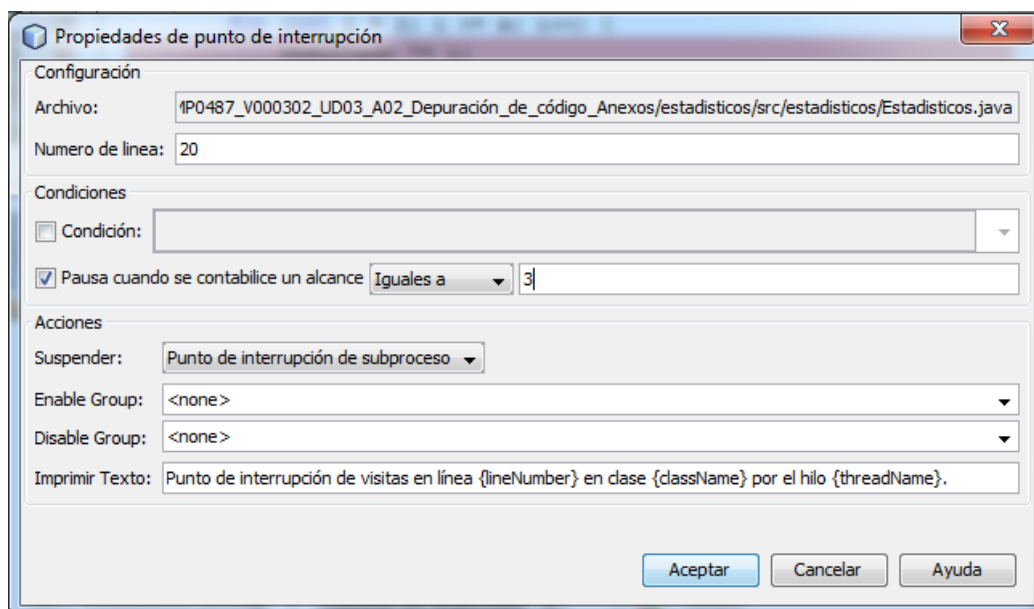
Poñer condicións ao punto de interrupción

Pódense poñer condicións para que un punto de interrupción pause unha depuración; algunhas son comúns para tódolos puntos de interrupción e outras depende de se non son de fío, se son de clase ou se son de excepción.

Condicións válidas para tódolos puntos de interrupción

Tódolos puntos de interrupción teñen a posibilidade de pausar unha depuración en función dunha frecuencia establecida, marcando o checkbox *Pausa cuando se contabilice un alcance*, seleccionando un criterio da lista despregable (*Igual a, es mayor que, es múltiplo de*) e establecendo un valor numérico para ese criterio na ventá de propiedades do punto.

A seguinte imaxe mostra a condición de que o punto de interrupción de liña situado na liña 20 de Estadisticos.java se active a terceira vez que se pase por ela durante unha sesión de depuración.

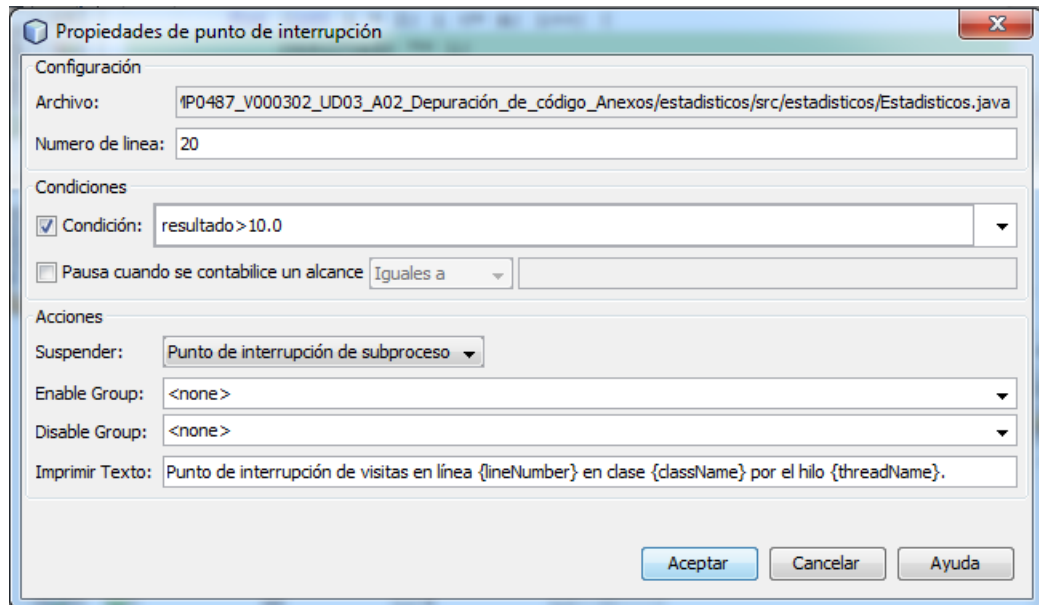


Condicións válidas para tódolos puntos agás os de tipos *thread*

Os puntos de interrupción que non son tipo fío, teñan a posibilidade de pausar unha depuración cando unha determinada condición é certa. Esta condición establécese na ventá de propiedades do punto de interrupción, seleccionando *Condición* e tecleando a condición. A condición debe seguir as normas de sintaxe de Java e pode incluír variables e métodos utilizados no contexto actual coas seguintes excepcións:

- As importacións son ignoradas. Débense usar nome completos como `obj instanceof java.lang.String`
- Non se pode acceder directamente a métodos e variables de clases externas. Débese utilizar `this.nome` ou `this.$1`

A seguinte imaxe mostra a condición de que o punto de interrupción de liña situado na liña 20 de Estadisticos.java se active cando a variable resultado teña un valor maior a 10 durante unha sesión de depuración.



Condicions específicas para os puntos de clase e excepción

Pódense dar as seguintes condicións específicas:



- Os puntos de interrupción de clases tamén permiten poñer como condición a exclusión dalgunha clase.
- Os puntos de interrupción de excepcións permiten poñer como condición un filtrado de clase a incluír ou excluír.

1.3 Tarefas

1.3.1 Tarefa 1. Depurar de forma básica

A tarefa consiste en realizar as seguintes depuracións sobre o proxecto Java *estadisticos*:

- Tarefa 1.1: Iniciar una depuración paso a paso do método *main*, tecleando os valores $m=5$ e $n=2$. Executar paso a paso soamente o método *factorial(5)*. Finalizar a depuración e ver o resultado final.
- Tarefa 1.2: Colocar o cursor na liña 36 de *Estadisticos.java* e iniciar unha depuración ata o cursor, e teclear os valores $m=15$ e $n=3$. Finalizar esta depuración e ver o resultado final.

Recórdase que para abortar unha sesión de depuración antes de que acabe normalmente, hai que premer en  na barra de depuración e para terminar de executar de forma normal un método antes de que acabe a execución paso a paso, hai que premer en  na barra de depuración.

1.3.2 Tarefa 2. Depurar facendo inspección de variables, expresións e métodos, e modificando valores

A tarefa consiste en realizar as seguintes depuracións sobre o proxecto Java *estadisticos*:

- Tarefa 2.1: Depurar para poder ver o valor de retorno dos métodos *factorial* na liña 26 de *Estadisticos.java*.

```
double combi = factorial(m)/(factorial(n)*factorial(m-n));
```

- Tarefa 2.2: Depurar para poder inspeccionar o valor da variable local *i* do método *factorial(5)* durante unha sesión de depuración.

- Tarefa 2.3: Depurar para que en tempo de depuración se poida modificar o valor da variable local *i* do método `factorial(5)` e ver os cambios realizados nas variables locais.
- Tarefa 2.4: Depurar para que durante unha sesión de depuración se poida ver o valor da expresión *m-n*.

1.3.3 Tarefa 3. Depurar inspeccionando a pila e facendo cambios nela

A tarefa consiste en depurar o proxecto Java *estadisticos* ata ter a pila con 4 chamadas para poder: cambiar a chamada actual, retroceder nas chamadas e volver a repetir algunhas.

1.3.4 Tarefa 4. Facer cambios no código mentres se depura

A tarefa consiste en forzar a modificación do código do proxecto Java *estadisticos* en tempo de depuración. A modificación a realizar pode ser para evitar que se calcule o factorial cando *m* ou *n* sexan maiores que 170 xa que se provocaría un desbordamento e retornaría o valor *Infinity*. A modificación debería de facerse en `Main.java`, no texto de información ao usuario, e no construtor de `Estadisticos.java`. Pode comprobarse que tecleando *m=171* e *n=171* prodúcese desbordamento en tódolos cálculos e tecleando *m=170* e *n=170* só se produce desbordamento no cálculo das variacións con repetición para o que non se utiliza o factorial.

1.3.5 Tarefa 5. Depurar utilizando puntos de interrupción

A tarefa consiste en facer as seguintes depuracións utilizando puntos de interrupción de liña, de método, de liña con condición e contador:

- Tarefa 5.1. Definir un punto de interrupción na liña 33 de `Main.java`. Executar unha depuración para ver a saída de resultados ata ese momento. Finalizar a depuración. Desactivar ese punto de interrupción.
- Tarefa 5.2. Definir un punto de interrupción que pare a depuración cada vez que se sae do método `factorial()` de `Estadisticos.java`. Executar a depuración e cada vez que se sae do método, ver como varía a pila na ventá da pila de chamadas, e ver os 9 valores dos elementos *x* e *resultado* na ventá de elementos observados. Eliminar ese punto de interrupción.
- Tarefa 5.3. Definir un punto de interrupción que pare a execución para os valores de *m=4* e *n=2*, na liña 22 de `Estadisticos.java`, se é a sétima vez que pasa por esa liña e o valor de *x* é igual a 4.
- Tarefa 5.4. Eliminar tódolos puntos de interrupción.