Java – Trabajando con el depurador de código de NetBeans

Objetivos

Como ya estás familiarizado con NetBeans como herramienta de desarrollo de código Java, en esta sesión queremos que te familiarices con el uso del depurador de Java con unos ejercicios muy básicos de programación estructurada.

Queremos que emplees en el futuro esta poderosa herramienta para el proceso de depuración de código de todas las prácticas de la asignatura.

Introducción

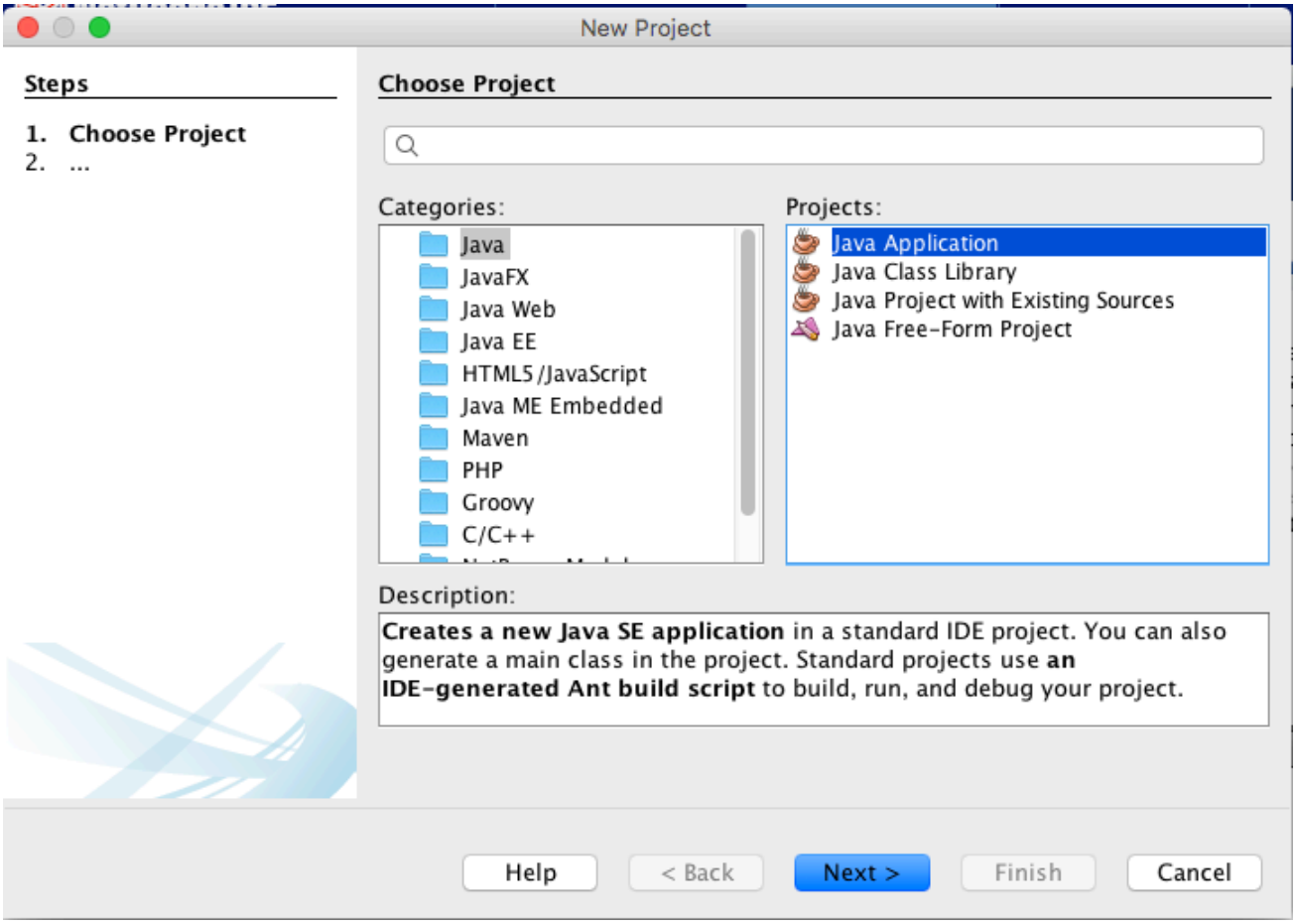
Cuando escribe programas, las cosas no siempre funcionan correctamente. Esa es, de hecho, una de las razones por las que necesitamos probar nuestras clases, queremos asegurarnos de que funcionen correctamente. Sin embargo, cuando las cosas no funcionan correctamente, puede ser difícil averiguar exactamente cuál es el problema para que podamos solucionarlo. NetBeans, como la mayoría de los otros entornos de programación, proporciona una herramienta particular para esto denominada depurador.

La depuración se puede definir como el proceso utilizado para examinar el código en busca de errores. La depuración se lleva a cabo configurando puntos de interrupción en el código y luego usando el depurador para ejecutarlo. Podemos ejecutar nuestro código una línea a la vez y examinar el estado de nuestra aplicación para descubrir cualquier problema.

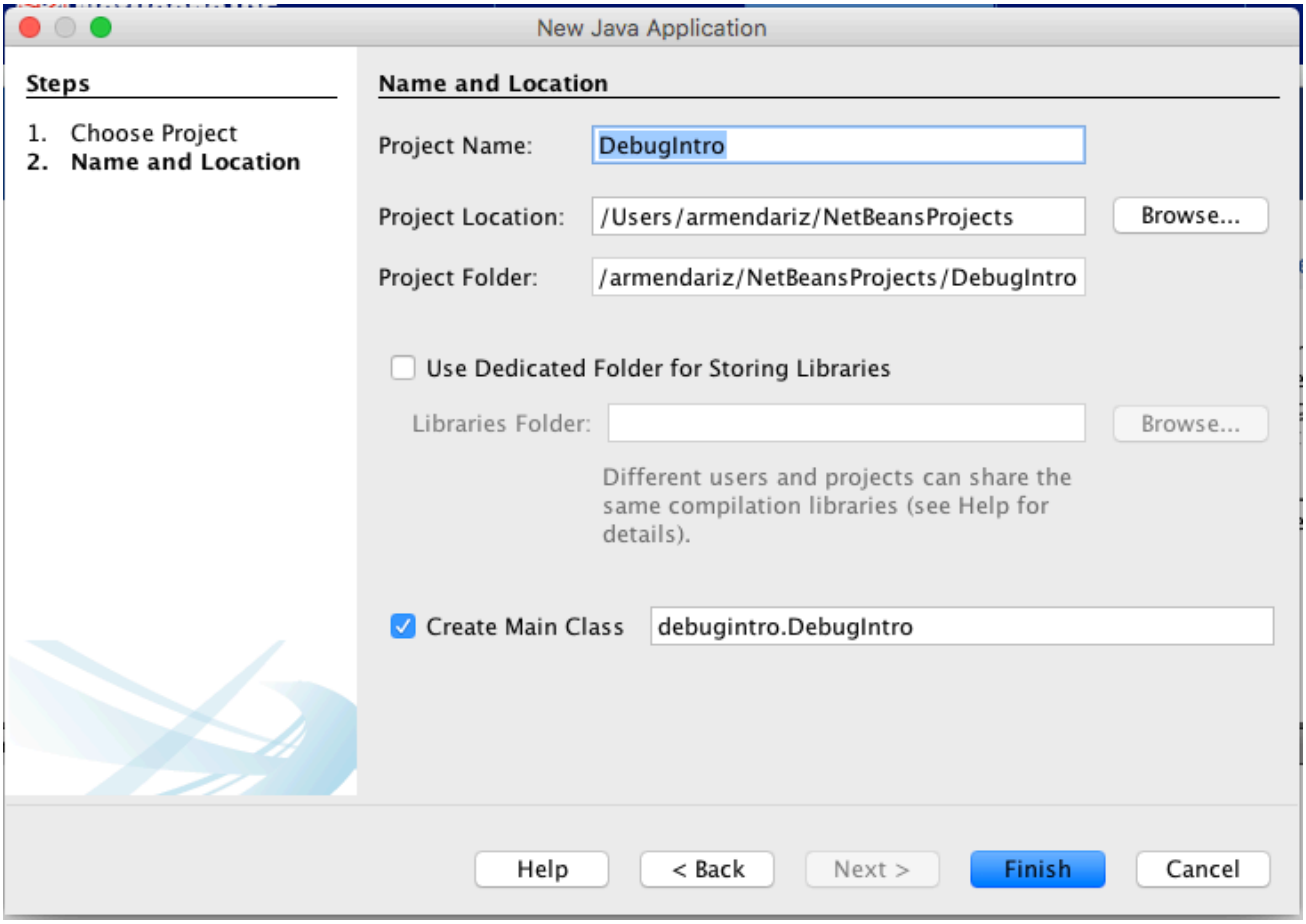
Cuando iniciamos una sesión de depuración en NetBeans, todas las ventanas de depurador relevantes aparecen automáticamente en la parte inferior de nuestra pantalla. Podemos depurar un proyecto completo, cualquier clase ejecutable y cualquier prueba JUnit.

Ejemplo práctico

Abre NetBeans y vete a "File" en el lado superior izquierdo de la barra de menú y seleccione "New Project". En el menú emergente que aparece, selecciona "Java" en "Categories" y "Java Application" en "Projects". Haz clic en el botón "Siguiente>".



Nombra el proyecto como "DebugInfo" y selecciona la ubicación que consideres apropiada



Presiona "Finish". Se ha creado un proyecto denominado "DebugIntro" en el panel izquierdo de proyectos.

Remplaza el código `main` con el siguiente contenido:

```
public static int main(String[] args) {

// TODO code application logic here


Scanner sc;


System.out.println("Enter any month in number (Ex: 1 for January, etc)");

sc = new Scanner(System.in);

float month = Integer.parseInt(sc.next());


if (month / 2 == 0) {
```

```
System.out.println("The month is Even");
} else {
System.out.println("The month is Odd");
}

// switch statement to return the name of the month
switch (month) {
case 1:
System.out.println("January");
break;
case 2:
System.out.println("February");
break;
case 3:
System.out.println("March");
break;
case 4:
System.out.println("April");
break;
case 5:
System.out.println("May");
break;
case 6:
System.out.println("June");
break;
case 7:
```

```
System.out.println("July");
break;
case 8:
System.out.println("August");
case 9:
System.out.println("September");
break;
case 10:
System.out.println("October");
break;
case 11:
System.out.println("November");
break;
case 12:
System.out.println("December");
break;
default:
System.out.println("Invalid month.");
break;
}

//while loop to calculate time to new year
int j = 0;
int i = month;
while (i != 0) {
i += 1;
```

```
j++;
System.out.println("Calculating time to New Year...");
}

System.out.println("Total months left for New Year:" + j);
}
```

Nota: Recuerda que puedes formatear tu código haciendo clic derecho en cualquier lugar dentro de la clase y seleccionando "Format". Alineará tu código correctamente.

Ten en cuenta que hay unos pocos **errores de compilación** en el programa.

Nota: el compilador siempre se ejecuta detrás de la escena. Por lo tanto, no hay opción de "Compilar" en este entorno para elegir.

El primer error de compilación está en el método `main`.

Cambia `int` a `void` en el método principal para eliminar el error de compilación. El método principal siempre es `void` ya que no devuelve ningún valor.

Nota: El método principal es siempre:

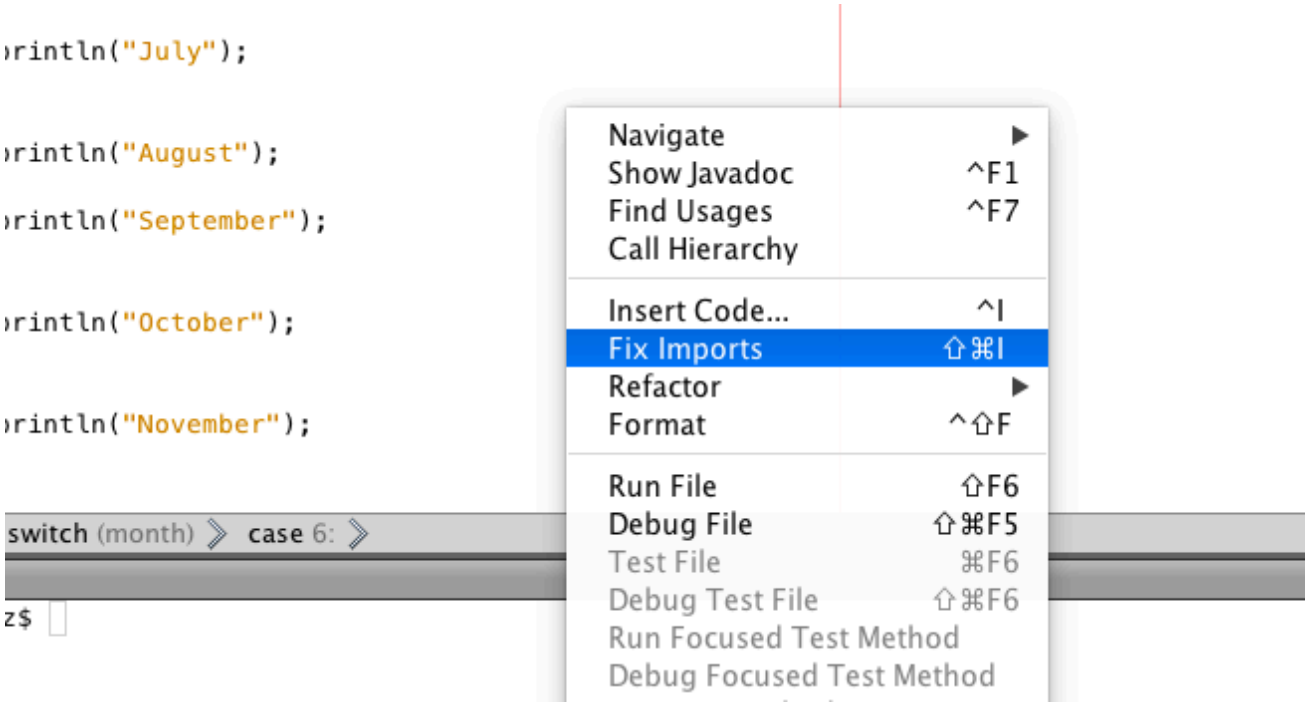
- `public`: para que se pueda acceder fuera de la clase.
- `static`: para que pueda invocarse sin crear ningún objeto.
- `void`: ya que no devuelve ningún valor.

El segundo error de compilación está en la declaración `switch`. Este error se debe al hecho de que la instrucción `switch` no funciona con `float`.

Al comienzo del método principal, cambia la variable `month` por el tipo de datos `int` en vez de `float`.

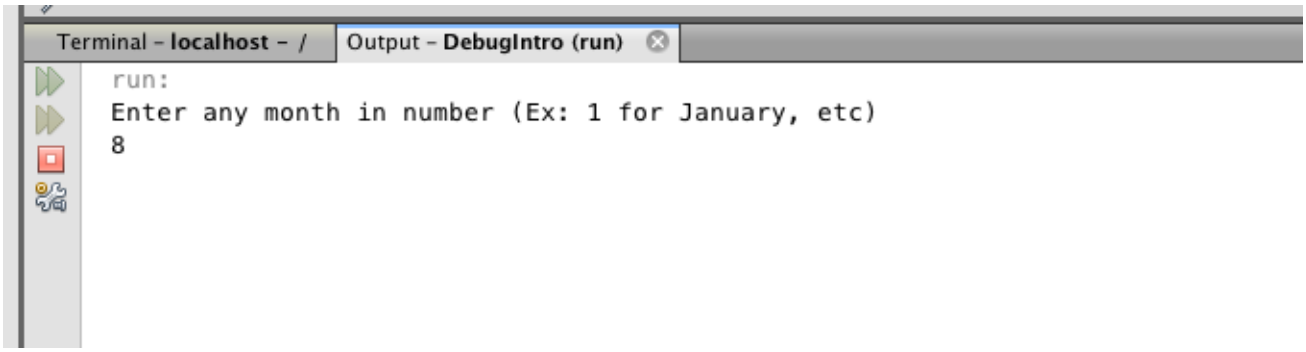
Nota: Un conmutador funciona con los tipos de datos primitivos `byte`, `short`, `char` e `int`.

Para eliminar otros errores, haz clic con el botón derecho en la ventana del editor de código fuente y seleccione "Fix Imports".



Asegúrate de que "`java.util.Scanner`" aparece en la ventana pop-up y selecciona "OK".

Ahora no debería haber errores de compilación en el programa. Guarda la clase y haz clic en el icono "Run" en la parte superior de la barra de menú o haciendo clic con el botón derecho en el proyecto y seleccione "Run". Cuando el programa solicita el dato de entrada, introduce 8 en el panel de salida ("output") situado en la parte inferior del IDE y presiona "Enter".



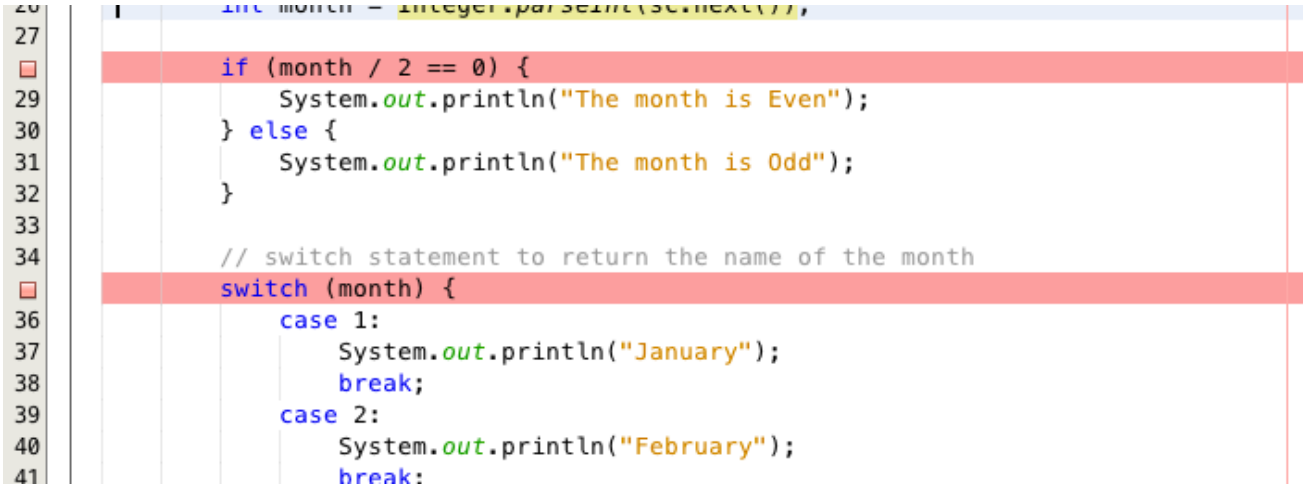
A pesar de que el mes introducido es par (agosto), la salida que se muestra es "`The month is Odd`" y "`August September`". Seguidamente, el programa entra en un ciclo infinito. Para salir del bucle infinito, haga clic en el botón de cerrar en el margen inferior del IDE.

Ahora vamos a rastrear la causa del problema utilizando el depurador. No resuelva el problema simplemente mirando el código. El objetivo de este ejercicio es aprender a usar el depurador para resolver problemas más difíciles que este.

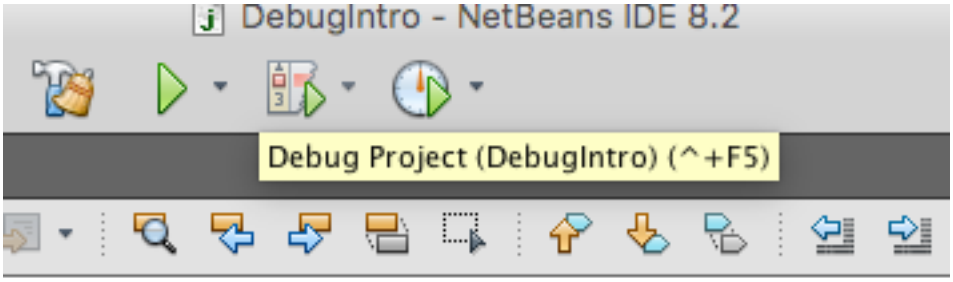
Queremos avanzar por el programa, paso a paso, para descubrir qué está mal dentro del código. Para hacer esto, primero podemos establecer un punto de interrupción (*breakpoint*) en el código. Un punto de interrupción es simplemente una bandera que le dice a NetBeans: "Para aquí si estás depurando".

Un punto de interrupción es un mecanismo de depuración que le dice al programa que cuando llegue a este punto, desea detener el programa. Para establecer un punto de ruptura, vaya a la línea de código que desea configurar y luego haga clic en el borde gris de la izquierda. Esto pondrá un punto de interrupción en esta línea.

Vamos a rastrear el programa para ver por qué ocurren estos errores colocando un punto de interrupción en la instrucción "`if`", la instrucción "`switch`" y el bucle "`while`". Puedes hacer esto haciendo clic en la franja gris izquierda de la ventana del editor. (Haz clic de nuevo para eliminar el punto de ruptura).

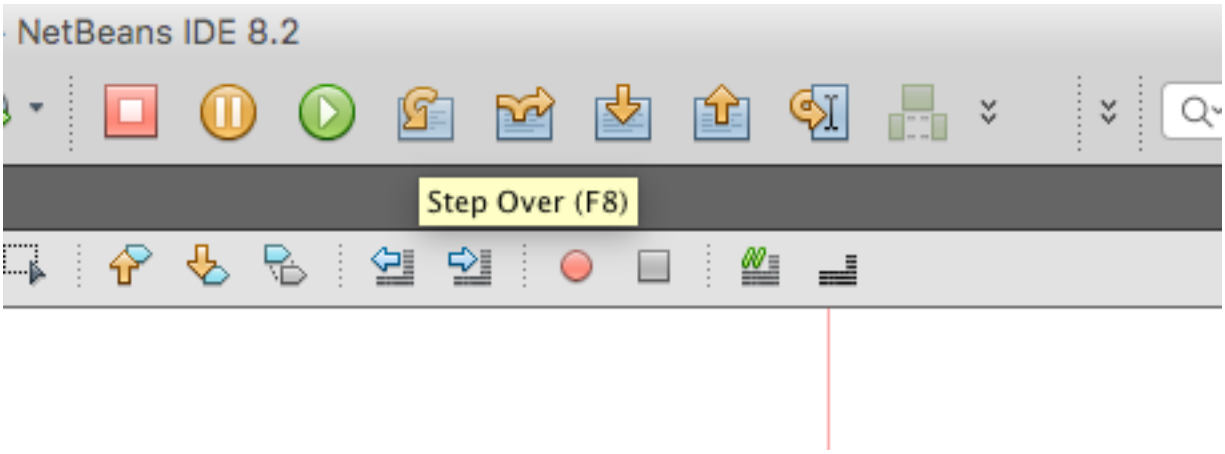


Haz click en el icono "Debug Main Project" en la barra de menús superior.



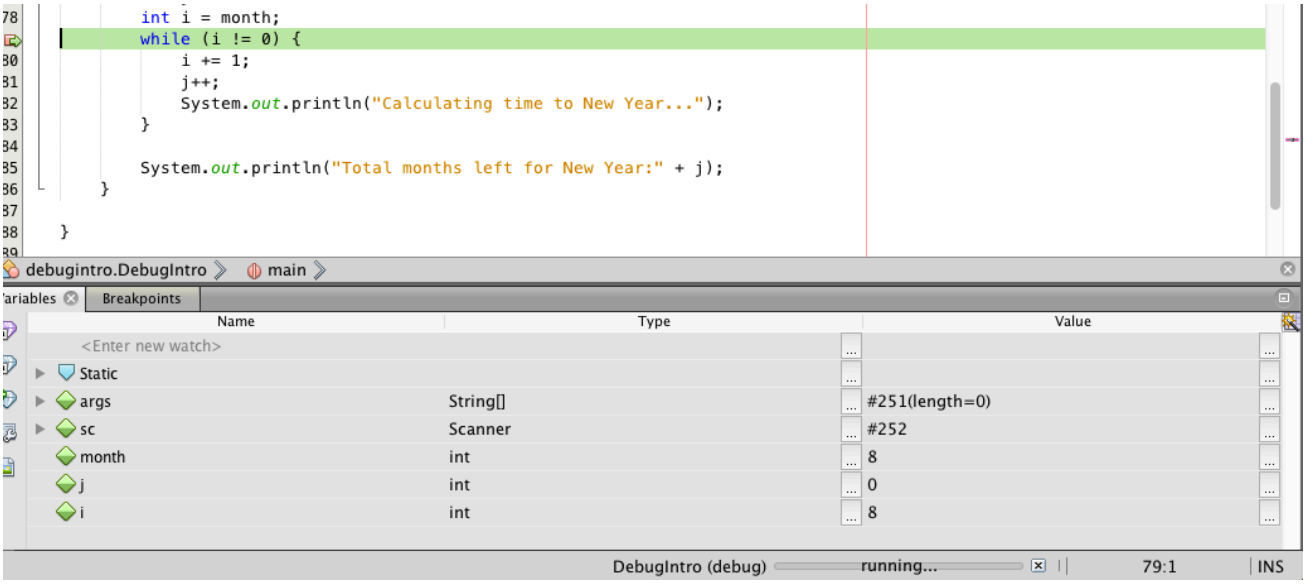

```
m.out.println("February");
:
```

Una vez más, introduce "8" en el panel de Output en la parte inferior del IDE y presiona Enter. Para recorrer el programa paso a paso, emplearás el ícono de "Step Over" de la barra de herramientas en el panel "Debug" de esta ventana, que se encuentra en la parte superior del IDE.

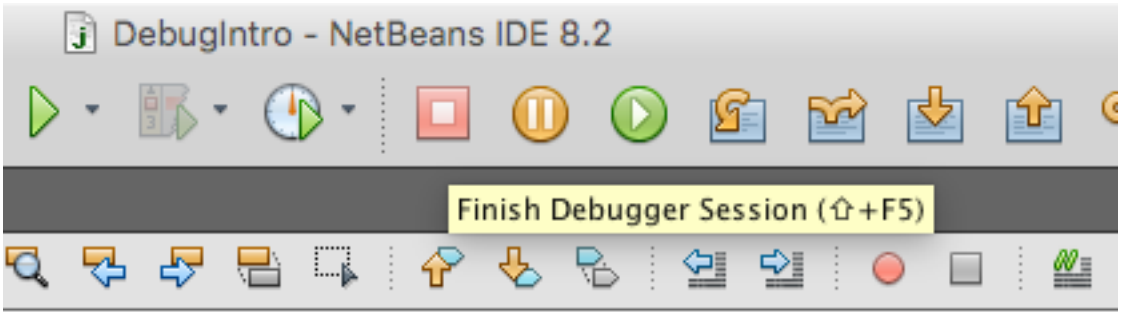


Haz clic en "Step Over" y continúa haciendo clic en "Step Over" y comprobarás que:

1. a. en lugar de ejecutarse la rama "`if`" en el método principal, se ejecuta la instrucción del bloque "`else`".
2. Se ejecuta el caso "8" y el caso "9" en la instrucción "`switch`".
3. El programa nunca sale del ciclo "`while`" (¡el valor de "`i`" sigue aumentando!) y, por lo tanto, este ciclo se ejecuta infinitas veces. De hecho, haga clic en la pestaña Variables en la parte inferior del IDE y podrá observar que los valores de "`i`" van del 8, al 9, al 10, al 11, etc., y no llega a cero; es un ejemplo de ciclo infinito clásico. Este proceso de observación de la variable te permite observar el comportamiento de su programa fácilmente.



Como ya hemos descubierto el origen del problema, detenemos la depuración haciendo clic en el icono "Finish" en la parte superior del IDE.



Encontramos en el momento de la depuración que la condición de la instrucción "`if`" no está escrita correctamente. Para corregir este error, reemplazamos el operador '`/`' en la instrucción "`if`" con el operador '`%`'.

Nota: Sabemos que '`/`' es el operador de la división. Por otro lado, el '`%`' es el operador resto de la división entera. Por ejemplo, si 8 está dividido por 2, el operador '`/`' devolverá 4 y el operador '`%`' devolverá 0.

Durante la depuración, también hemos notado que junto con el caso "8" en la sentencia "`switch`", el caso "9" también se ejecuta. Esto se debe al hecho de que no hay instrucción "`break`"; se ejecuta el caso "8" y, por lo tanto, inmediatamente después de la ejecución del caso "8", el control se desplaza al caso "9". Para corregir este error, incluye el comando "`break` en el caso "8" de la instrucción "`break`".

Por último, también hemos descubierto al depurar que el bucle "`while`" se ejecuta infinitamente. Mira de cerca y encontrarás que como necesitamos calcular los meses restantes para el nuevo año,

la condición "`i!= 0`" no tiene ningún sentido. Debe ser "`i!= 12`". Por lo tanto, para deshacerse de este error, hay que remplazar "`i!= 0`" con "`i != 12`" en el ciclo "`while`".

Guarda el proyecto y ejecute el proyecto nuevamente haciendo clic en el icono "Run" en la parte superior de la barra de menú o haciendo clic con el botón derecho en el proyecto y seleccione "Run".

Ingresa el mes de tu elección y ahora obtendrá la salida correcta sin más errores.

Ejercicios para realizar

En "MiAulario" tienes una carpeta con los recursos para esta práctica en la "Sesión 1" en el fichero "DebuggingExercises.zip".

Ejercicio 1

En el fichero "DebuggingExercise1":

- Añade los comandos para que el usuario entre su altura en metros y su peso en kilogramos y calcula el IMC con la formula siguiente: weight / (height*height).
- ¿Qué sucede cuando se entra un texto en vez de números?

Ejercicio 2:

El código de la clase "DebuggingExercise2" calcula la serie de Fibonacci para un elemento dado. Sin embargo, aunque el código compila, su ejecución da fallos y no da el buen resultado.

- Haz que funcione de manera correcta el programa

Ejercicio 3:

El código de "DebuggingExercise3" busca el elemento máximo de un *array*. Sin embargo, aunque el código compila, su ejecución da fallos y no da el buen resultado.

- Haz que funcione de manera correcta el programa

Ejercicio 4:

El código "DebuggingExercise4" multiplica dos matrices cuadradas. Sin embargo, aunque el código compila, su ejecución da fallos y no da el resultado correcto.

- Haz que funcione de manera correcta el programa

Ejercicio 5:

Completa el código de "Exercise1" para que calcule el resultado de la serie de Fibonacci usando una función recursiva.

Ejercicio 6:

Completa el código de "Exercise2" para que emplee el algoritmo de búsqueda dicotómica de un elemento dentro de un *array* ordenado. La generación del *array* está ya implementada.

- Escribe la función binarySearch(), y añade el comando para que el usuario busque un numero de su elección

Ejercicio 7:

Completa el código de "Exercise3" para que multiplique dos matrices no cuadradas.

- Añade los comandos para que el usuario entre el número de filas y de columnas de la primera matriz. El tamaño de la segunda matriz automáticamente será el inverso. Por ejemplo, si la primera matriz tiene 3 filas y 2 columnas (3,2), la segunda matriz será de tamaño (2,3). También añade la posibilidad al usuario para que pueda rellenar las dos matrices.
- Escribe el método `multiplyMatrix()` que es una extensión del ejercicio 4 para tener en cuenta los tamaños distintos de la matriz. En el ejemplo anterior, la multiplicación de una matriz (3,2) con una matriz (2,3) dará una matriz (3,3). Y la multiplicación de una matriz (2,3) con una (3,2) dará una matriz (2,2);