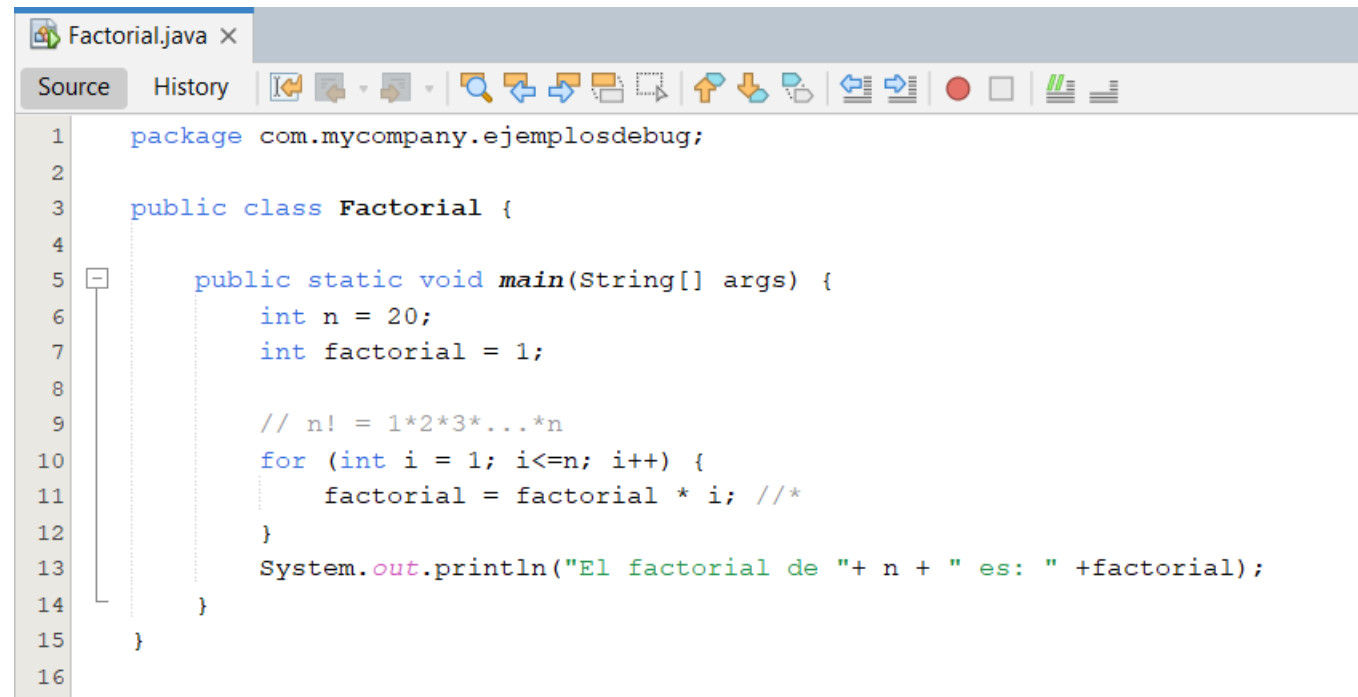


ACTIVIDAD GUIADA DEPURACIÓN

Paso 0: Escribe el siguiente programa Java en NetBeans



The screenshot shows the NetBeans IDE with a file named 'Factorial.java' open. The code is written in Java and calculates the factorial of a number n (set to 20). The code is as follows:

```
1 package com.mycompany.ejemplosdebug;
2
3 public class Factorial {
4
5     public static void main(String[] args) {
6         int n = 20;
7         int factorial = 1;
8
9         // n! = 1*2*3*...*n
10        for (int i = 1; i<=n; i++) {
11            factorial = factorial * i; /*
12        }
13        System.out.println("El factorial de "+ n + " es: " +factorial);
14    }
15 }
16
```

ACTIVIDAD GUIADA

DEPURACIÓN

El programa anterior calcula e imprime el factorial de un número n .

$$n! = 1 * 2 * 3 * \dots * n.$$

Ejecuta el programa al completo y observa que tiene un error lógico ya que produce una respuesta incorrecta para $n = 20$. La solución que se muestra por consola es la siguiente:

```
El factorial de 20 es: -2102132736
```

¿¿un número negativo??

ACTIVIDAD GUIADA DEPURACIÓN

Paso 1: Establecer un punto de interrupción

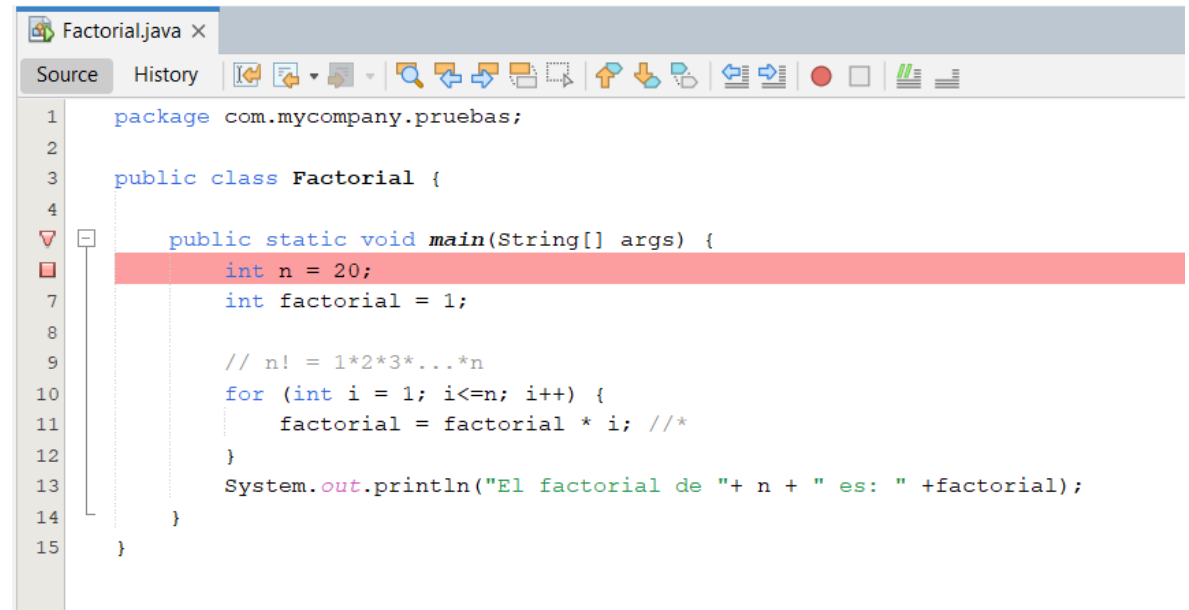
Un punto de interrupción suspende la ejecución del programa para que puedas examinar los estados internos del programa.

Antes de iniciar el depurador, debes establecer al menos un punto de interrupción para suspender la ejecución dentro del programa.

ACTIVIDAD GUIADA DEPURACIÓN

Según donde establezcas el breakpoint haciendo clic en el margen izquierdo, aparecerá en dicho margen un **triángulo invertido** (cuando el breakpoint se establece sobre la definición de una clase o método), o un **cuadrado rojo** (cuando se establece en cualquier otra línea de código).

Se puede establecer más de un breakpoint, pero no es lo normal. Con un solo breakpoint es suficiente para poder depurar nuestros programas. Para nuestra actividad, establece un solo punto de interrupción en la línea 5 donde se declara el método `main()`



The screenshot shows an IDE window titled 'Factorial.java'. The code is as follows:

```
1 package com.mycompany.pruebas;
2
3 public class Factorial {
4
5     public static void main(String[] args) {
6         int n = 20;
7         int factorial = 1;
8
9         // n! = 1*2*3*...*n
10        for (int i = 1; i<=n; i++) {
11            factorial = factorial * i; /**
12        }
13        System.out.println("El factorial de "+ n + " es: " +factorial);
14    }
15 }
```

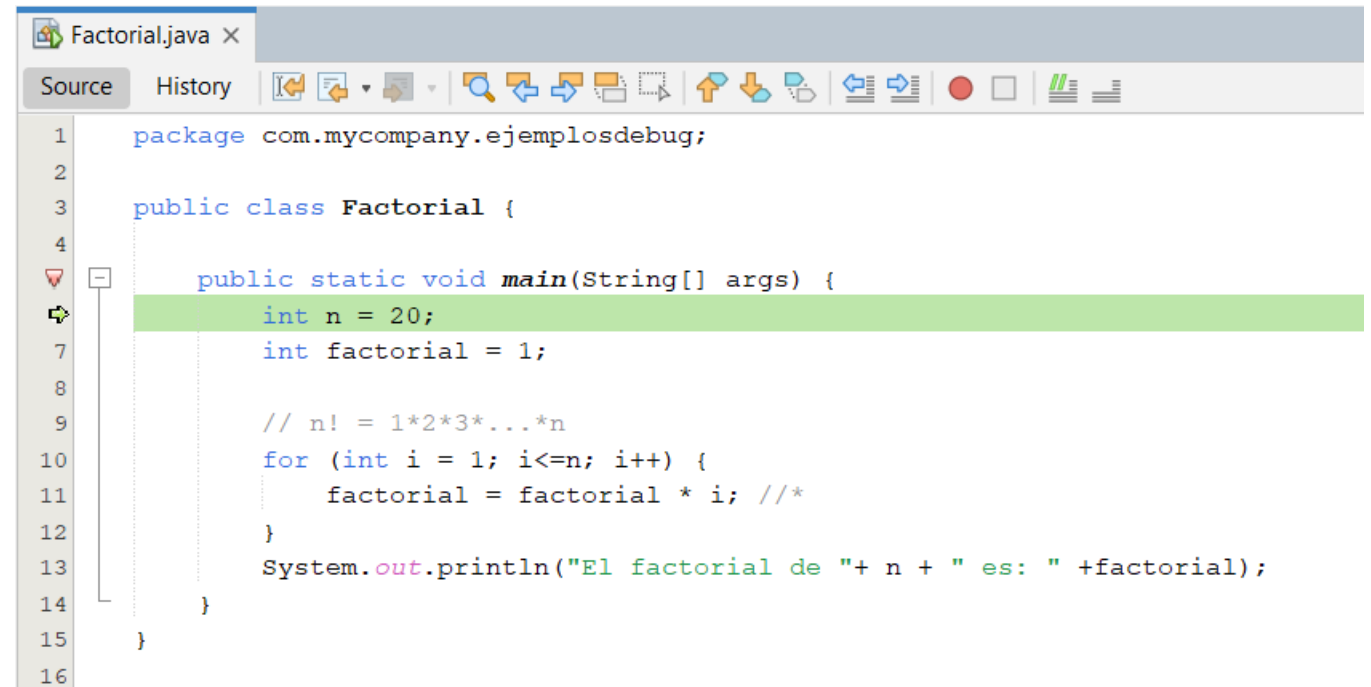
A breakpoint is set on line 5, indicated by a red square icon in the left margin. The line containing the `main` method signature is highlighted in red.

ACTIVIDAD GUIADA DEPURACIÓN

Paso 2: Comienza a depurar

Haz clic derecho en cualquier parte del código fuente ⇒ "Debug File". El programa comienza la ejecución pero suspende su operación en el punto de interrupción, es decir, el método main ().

Como se ilustra en la siguiente imagen, la línea resaltada en verde (también señalada por una flecha verde en el margen izquierdo) indica la instrucción que se ejecutará en el siguiente paso.

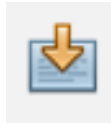


```
1 package com.mycompany.ejemplosdebug;
2
3 public class Factorial {
4
5     public static void main(String[] args) {
6         int n = 20;
7         int factorial = 1;
8
9         // n! = 1*2*3*...*n
10        for (int i = 1; i<=n; i++) {
11            factorial = factorial * i; /**
12        }
13        System.out.println("El factorial de " + n + " es: " +factorial);
14    }
15 }
16
```

ACTIVIDAD GUIADA DEPURACIÓN

Paso 3: Paso a paso y ver las variables y salidas

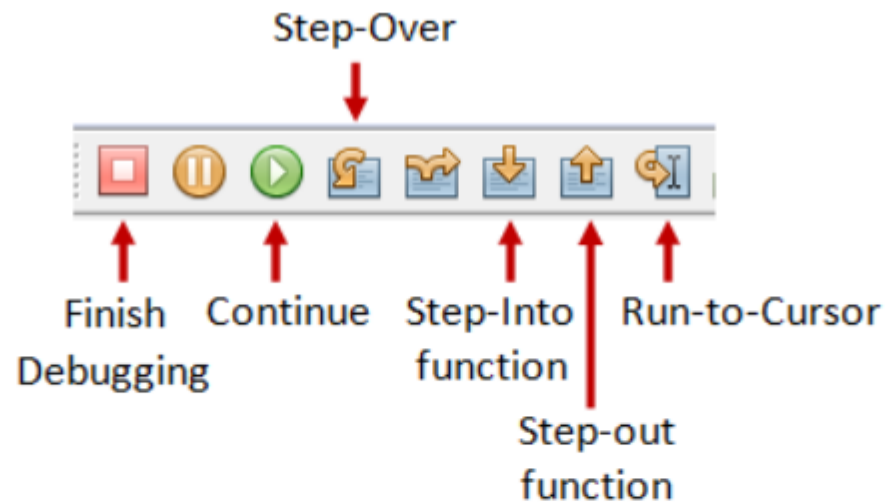
Haz clic en el botón "Step into" para avanzar línea a línea en el programa.



para avanzar línea a línea en el programa.

En cada uno de los pasos, examina el valor de las variables (en el panel "Variables") y las salidas producidas por su programa (en el panel "Output"), si las hay. También puedes colocar el cursor en cualquier variable para inspeccionar el contenido de la variable.

Notifications	Variables ×	Breakpoints	Output
	Name	Type	Value
	<Enter new watch>		...
	> Static		...
	> args	String[]	... #77(length=0)
	n	int	... 20
	factorial	int	... -2102132736








Notifications			Variables ×			Breakpoints			Output		
			Name			Type			Value		
			<Enter new watch>						...		
			> Static						...		
			> args			String[]			... #77(length=0)		
			n			int			... 20		
			factorial			int			... 24		
			i			int			... 4		

El paso a paso a través del programa y la observación de los valores de las variables internas y las salidas producidas es el medio definitivo para la depuración de programas, **¡ya que es exactamente la forma en la que la computadora ejecuta el programa!**

ACTIVIDAD GUIADA

DEPURACIÓN

	Step Over. Ejecuta una línea de código. Si la instrucción es una llamada a un método, ejecuta el método sin entrar dentro del código del método.
	Step Into. Ejecuta una línea de código. Si la instrucción es una llamada a un método, entra dentro del método y continúa la ejecución por la primera línea de dicho método.
	Step Out. Ejecuta una línea de código. Si la línea de código actual se encuentra dentro de un método, se ejecutarán todas las instrucciones que queden del método y se vuelve a la instrucción desde la que se llamó al método.
	Continue. La ejecución del programa continúa hasta el siguiente breakpoint. Si no existe un breakpoint se ejecuta hasta el final.
	Finish Debugger Session. Termina la depuración del programa.

ACTIVIDAD GUIADA DEPURACIÓN

Otras características del depurador:

Cambio del valor de las variables en tiempo de ejecución

Puedes modificar el valor de una variable ingresando un nuevo valor en el panel "Variable" en tiempo de ejecución. Esto es útil para modificar temporalmente el comportamiento de un programa, sin cambiar el código fuente.

Añadir un watch a una variable o expresión

Se trata de una forma para monitorizar el valor de una variable o incluso de una expresión lógica compleja, ej: `if ((A || B) && (C || D))`