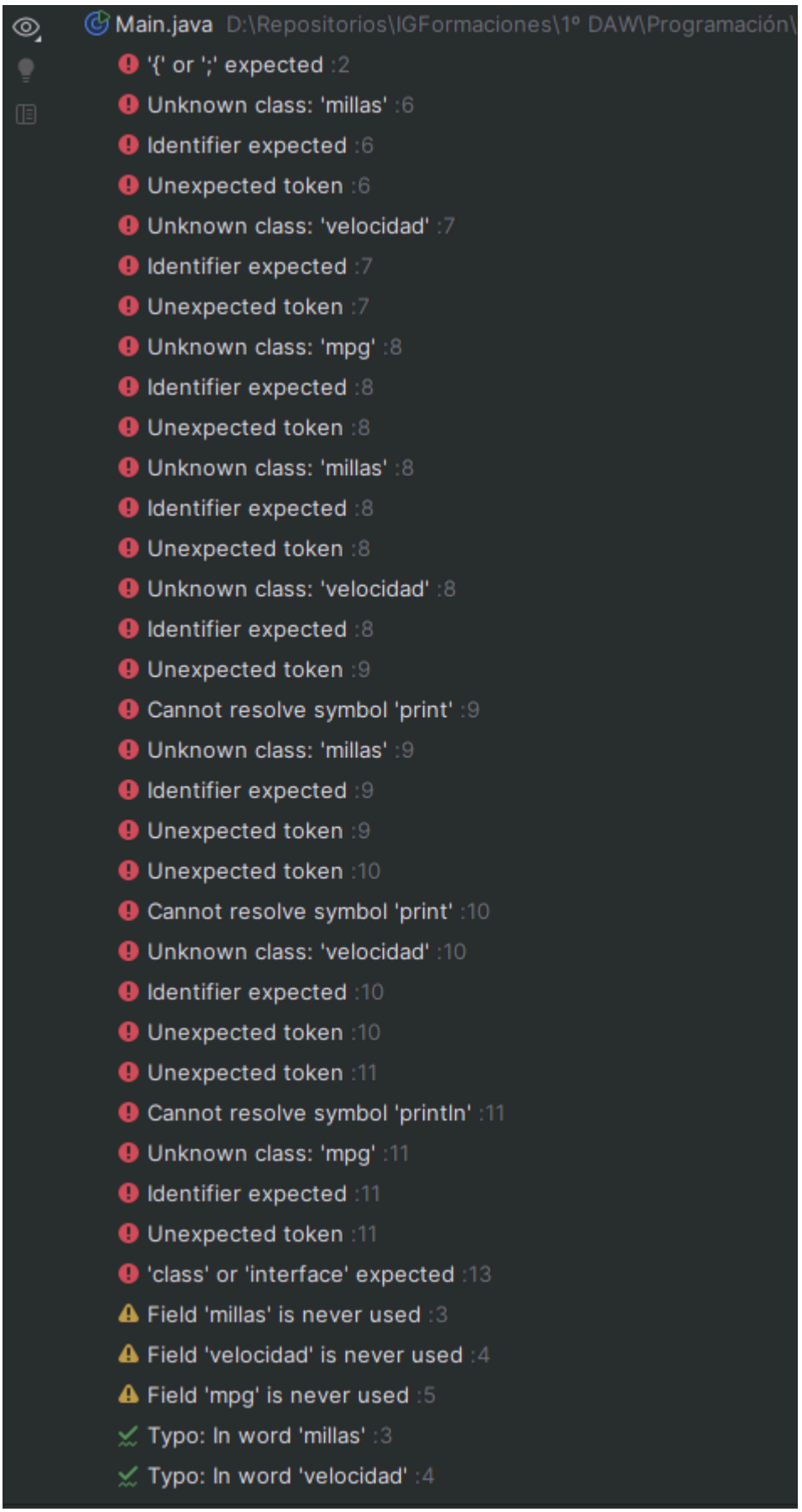
# UD2Tarea9 - Debug

Andrés Pérez Guardiola - 1st DAW

### 1. ¿Qué produce el siguiente código? Observa la cascada de errores que muestra el IntelliJ y realiza una captura de pantalla de los errores. ¿Es correcto el número de línea citado por el primer diagnóstico?

```
public class Main {
   public static void main(String[] args)
       double millas;
       double velocidad;
       double mpg;
       millas =341;
       velocidad =15.5;
       mpg =millas /velocidad;
       System.out.print(millas +" mi. / ");
       System.out.print(velocidad +" gal.");
       System.out.println(" = "+mpg +" mpg");
   }
}
```


Main.java D:\Repositorios\IGFormaciones\1º DAW\Programación\
'{' or ';' expected :2
Unknown class: 'millas' :6
Identifier expected :6
Unexpected token :6
Unknown class: 'velocidad' :7
Identifier expected :7
Unexpected token :7
Unknown class: 'mpg' :8
Identifier expected :8
Unexpected token :8
Unknown class: 'millas' :8
Identifier expected :8
Unexpected token :8
Unknown class: 'velocidad' :8
Identifier expected :8
Unexpected token :9
Cannot resolve symbol 'print' :9
Unknown class: 'millas' :9
Identifier expected :9
Unexpected token :9
Unexpected token :10
Cannot resolve symbol 'print' :10
Unknown class: 'velocidad' :10
Identifier expected :10
Unexpected token :10
Unexpected token :11
Cannot resolve symbol 'println' :11
Unknown class: 'mpg' :11
Identifier expected :11
Unexpected token :11
'class' or 'interface' expected :13
Field 'millas' is never used :3
Field 'velocidad' is never used :4
Field 'mpg' is never used :5
Typo: In word 'millas' :3
Typo: In word 'velocidad' :4

Toda la lista de errores que aparece en el IntelliJ viene porque en la línea 2 esperaba que se iniciara un bloque ({) o el cierre de una sentencia (;)  pero no encuentra ninguno de los dos, por lo que a partir de ahí se queja por absolutamente todo. El error se soluciona simplemente poniendo el inicio del bloque {.

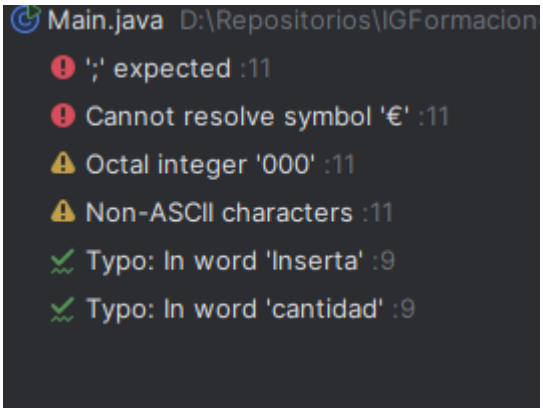
```
public class Main {
   public static void main(String[] args) {
       double millas;
       double velocidad;
       double mpg;
       millas =341;
       velocidad =15.5;
       mpg =millas /velocidad;
       System.out.print(millas +" mi. / ");
       System.out.print(velocidad +" gal.");
       System.out.println(" = "+mpg +" mpg");
   }
}
```

```
"C:\Program Files\Java\jdk-22\bin\java
341.0 mi. / 15.5 gal. = 22.0 mpg

Process finished with exit code 0
```

## 2. Corrige el siguiente código:

```java
import java.util.Scanner;

public class Main
{
   public static void main( String [] args )
   {
       double cost, qty;
       Scanner in = new Scanner(System.in);
       System.out.print("Inserta una cantidad: ");
       qty = in.nextDouble();
       cost = 15,000€;
       System.out.println("Total: " + qty * cost);
   }
}
```

En el ejemplo no aparece importada la clase Scanner, pero supongo que debería de estar la. Ahora cost se define como double y recibe un número separado por coma y un valor que no es un carácter ASCII.




El código corregido sería así.

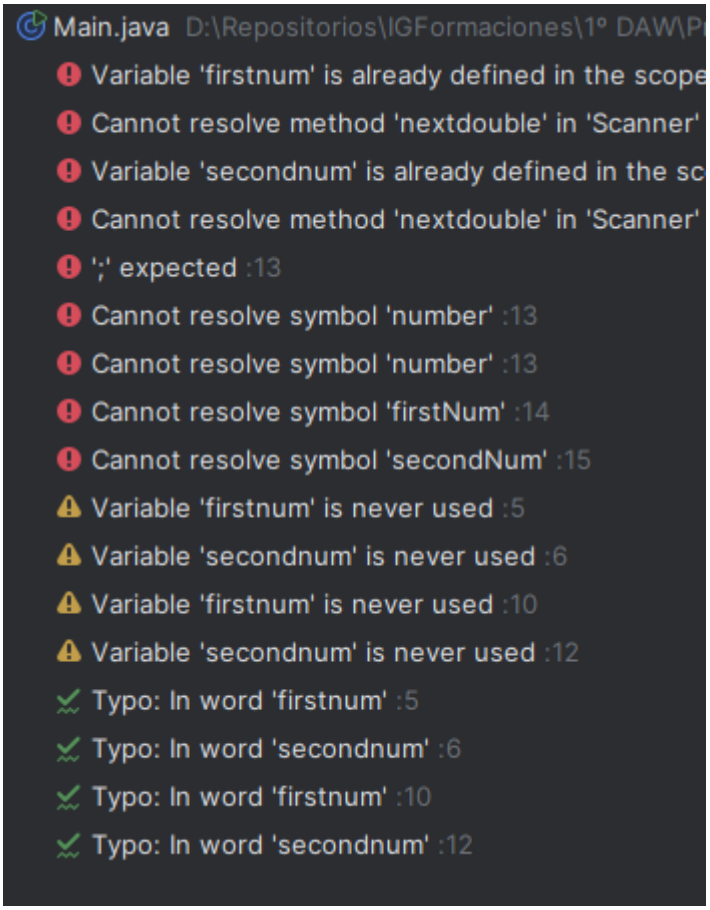
```java
import java.util.Scanner;

public class Main
{
   public static void main( String [] args )
   {
       double cost, qty;
       Scanner in = new Scanner(System.in);
       System.out.print("Inserta una cantidad: ");
       qty = in.nextDouble();
       cost = 15000;
       System.out.println("Total: " + qty * cost + "€");
```

```
    }
}
```

```
"C:\Program Files\Java\jdk-22\bin\java
Inserta una cantidad: 4
Total: 60000.0€

Process finished with exit code 0
```

### 3. El siguiente código produce errores, identifica cada uno de ellos y corrígelos:

```java
import java.util.Scanner;

public class Main {
   public static void main( String [] args ) {
       double firstnum;
       double secondnum;
       double product;
       Scanner in = new Scanner( System.in );
       System.out.print("Enter first number to multiply: " );
       double firstnum = in.nextdouble();
       System.out.print("Enter second number to multiply: " );
       double secondnum = in.nextdouble();
       product = 2number * 1number;
       System.out.print(firstNum + " * " );
       System.out.print(secondNum + " = " );
       System.out.println( product );
   }
}
```



El primer error se produce porque se ha definido ya previamente la variable firstnum. La segunda vez que aparece solo deberíamos inicializarla, no declararla nuevamente.

El segundo error es que no se ha puesto correctamente el nombre del método. Scanner tiene un método llamado nextDouble(); no nextdouble.

El tercer error tiene la misma causa que el primero, declaración de una variable ya declarada previamente.

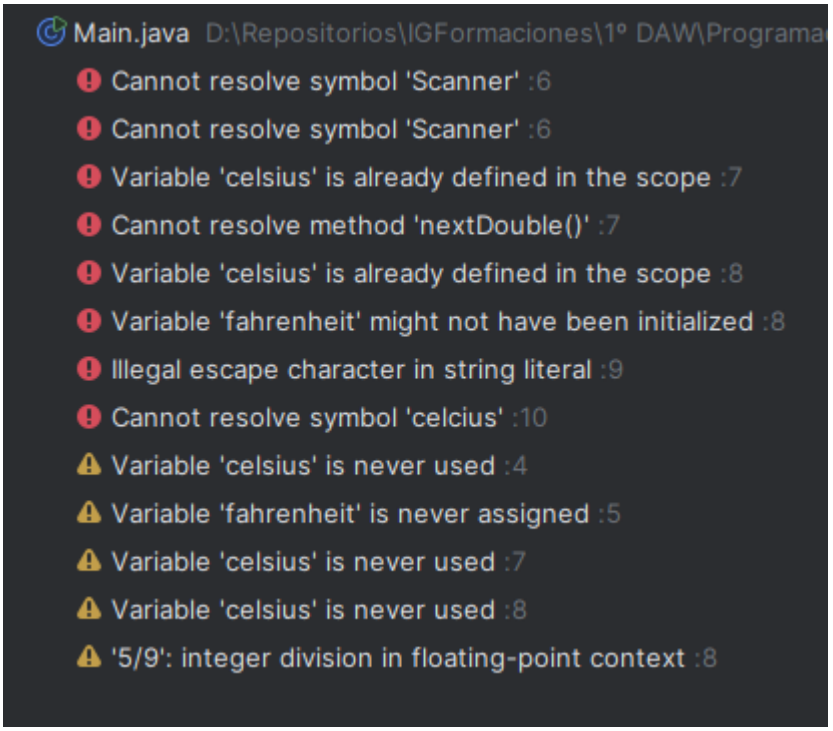
El cuarto error indica que esperaba un ; pero no lo encontró. Pero esto viene asociado a los siguientes errores. El problema es que se ha llamado incorrectamente a las variables firstnum and secondnum en la inicialización de product. Ninguna variable puede empezar por un número. Habría que cambiarlo por cómo se han llamado anteriormente, firstnum and secondnum.

Las advertencias que recibimos es que no usamos los valores firstnum and secondnum después de inicializarlas. Esto sucede porque en el System.out.print() del final hemos escrito de forma incorrecta el nombre de las variables. Por convención es aconsejable usar camelCase para las variables. Si corregimos el código sería:

```java
import java.util.Scanner;

public class Main {
   public static void main( String [] args ) {
       double firstNum;
       double secondNum;
       double product;
       Scanner in = new Scanner( System.in );
       System.out.print("Enter first number to multiply: " );
       firstNum = in.nextDouble();
       System.out.print("Enter second number to multiply: " );
       secondNum = in.nextDouble();
       product = secondNum * firstNum;
       System.out.print(firstNum + " * " );
       System.out.print(secondNum + " = " );
       System.out.println( product );
   }
}
```

## 4. Identifica y corrige los errores del siguiente código.

```java
public class Main
{
   public static void main( String [] args ) {
        double celsius;
        double fahrenheit;
        Scanner in = new Scanner(System.in);
        double celsius = in.nextDouble();
        double celsius = 5/9 * fahrenheit - 32;
        System.out.println(fahrenheit, "\U00B0F = ");
        System.out.println(celcius, "u00B0C");
   }
}
```




Los primeros errores son porque no está importada la clase scanner. Se corrige añadiendo la línea: `import java.util.Scanner;`

Lo segundo es la triple declaración de la variable celsius. La variable fahrenheit no se usa en todo el código. Según está escrito el programa, la declaración de celsius primera debería tomar el valor de temperatura en fahrenheit.

```java
double celsius = in.nextDouble();
```

Substituido por: `fahrenheit = in.nextDouble();`

En la declaración de la variable: `celsius = 5/9 * fahrenheit - 32;`
tenemos una operación entre literales integers que nos la marca como warning. Esta operación devuelve 0, de debería de especificar que sea una operación entre Doubles.

```
celsius = 5d/9d * fahrenheit - 32;
```

Esta variable no se usa nunca más durante el código porque está mal escrita en el System.out

```
System.out.println(celcius, "u00B0C");
```

Cambiar celcius por celsius

```
System.out.println(celsius, "u00B0C");
```

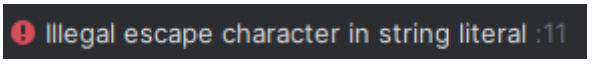
Esto devuelve otro error 

porque el método println() no toma dos argumentos. Se debe substituir la coma por un más.

```
System.out.println(celsius + "u00B0C");
```

Y lo mismo en el de arriba:

```
System.out.println(fahrenheit + "\U00B0F = ");
```

Luego tenemos el siguiente error 

debido a que para el carácter especial estamos usando

```
\U00B0F
```

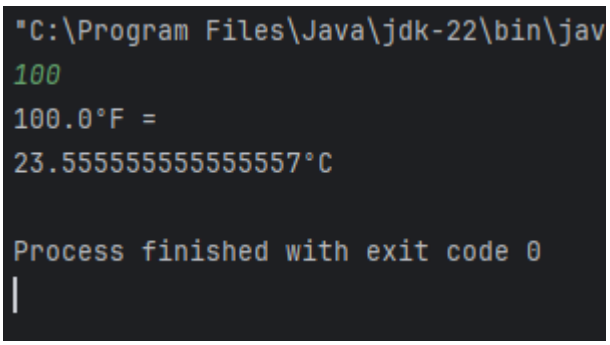
Y necesitamos que sea la u en minúsculas

Lo mismo sucede con el de abajo, que ni siquiera usamos el carácter de escape \u

```
"\u00B0C"
```

Ahora obtenemos un warning porque directamente no necesitamos usar \u00B0 para ese carácter, sino que podemos reemplazarlo directamente por º al ser un carácter ASCII válido.

```
import java.util.Scanner;

public class Main
{
   public static void main( String [] args ) {
        double celsius;
        double fahrenheit;
        Scanner in = new Scanner(System.in);
        fahrenheit = in.nextDouble();
        celsius = 5d/9d * fahrenheit - 32;
        System.out.println(fahrenheit + "°F = ");
        System.out.println(celsius + "°C");
   }
}
```




Cero Errores, cero warnings

## 5. Copia las clases Contador y ContadorMain en el IntelliJ. Pon un punto de ruptura en la clase Contador y sigue la ejecución del método contar. ¿Cuánto vale resultado cuando i=3? Realiza capturas de pantalla. Para lanzar el programa ejecuta la clase ContadorMain. ¿Cuánto vale resultado dentro de triContarDesde0 tras la segunda llamada a contar?

Cuandos se ejecuta obtenemos:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\
Ha contado 5050
Exception in thread "main" java.lang.ArithmeticException Create breakpoint : / by zero
    at Contador.triContarDesde0(Contador.java:27)
    at ContadorMain.main(ContadorMain.java:9)

Process finished with exit code 1
```

Primero obtenemos el primer print y luego obtenemos una excepción por división por cero en la línea 9 de ContadorMain que sigue por la función triContarDesde0 en la línea 27 de la clase Contador.

La línea 9 de ContadorMain.java:

```
contador.triContarDesde0();
```

La línea 27 de Contador.java:

```
resultado = resultado / (resultado - resultado);
```

Esta línea esta puesta dentro del método triContarDesde0():

```
public void triContarDesde0() {
    resultado = 0;
    this.contar();
    this.contar();
    this.contar();

    if (resultado %2 == 0) {
        // Forzamos una excepción de tipo ArithmeticException
        resultado = resultado / (resultado - resultado);
    }
}
```

Esto está pensado para lanzar una excepción de tipo ArithmeticException, aunque simplemente se podría lanzar la excepción sin necesidad de hacerlo tan rebuscado:

```
public void triContarDesde0() {
    resultado = 0;
    this.contar();
    this.contar();
```

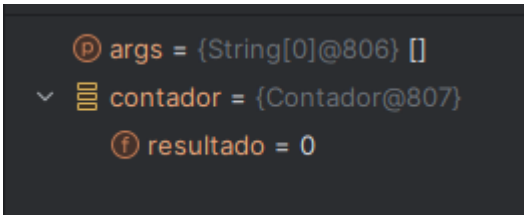
```
    this.contar();

    if (resultado %2 == 0) {
        // Forzamos una excepción de tipo ArithmeticException
        throw new ArithmeticException("ERROR: Resultado ha devuelto
un valor par.");
    }
}
```

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024
Ha contado 5050
Exception in thread "main" java.lang.ArithmeticException Create breakpoint : ERROR: Resultado ha devuelto un valor par.
    at Contador.triContarDesde0(Contador.java:27)
    at ContadorMain.main(ContadorMain.java:9)

Process finished with exit code 1
```

Esto parece que se debe a que algo sucede con el atributo resultados la clase. Para poder saber que sucede ponemos un par de breakpoint en el código, uno en cada llamada de los métodos de contar en la clase ContadorMain.java.

```
1  public class ContadorMain {  new *
2
3      public static void main(String[] args) {  new *
4          Contador contador = new Contador();
5
6          contador.contar();
7          System.out.println("Ha contado " + contador.getResultado());
8
9          contador.triContarDesde0();
10         System.out.println("Ha tricontado " + contador.getResultado());
11     }
12 }
13
```

Y le damos a debug:




Después de un par de llamadas (haciendo step-into para seguirlo dentro de la clase Contador.java:

```
public void contar() {  4 usages  new *
    for (int i = 0; i < 100; i++) {
        resultado += i + 1;
    }
}
```

```
this = {Contador@807}
i = 3
resultado = 6
```

Resultado va sumando el valor de i + 1, en este caso como resultado tiene ya el valor de 2 + 1 y 1 + 1, en el siguiente paso resultado valdrá 10:

```
this = {Contador@807}
i = 3
resultado = 10
```

El programa parece que funciona correctamente en esta ejecución.

Dejamos correr el programa bastantes iteraciones más:

```
this = {Contador@807}
resultado = 5050
```

Termina el bucle, la variable i que estaba definida sólo en el scope dentro del bloque for por lo que desaparece esta y solo queda resultado que es un atributo de la clase.

```
public void contar() {  4 usages  new *
    for (int i = 0; i < 100; i++) {
        resultado += i + 1;   resultado: 5050
    }
}
```

Una vez volvemos a ContadorMain(), seguimos la línea siguiente del programa que es

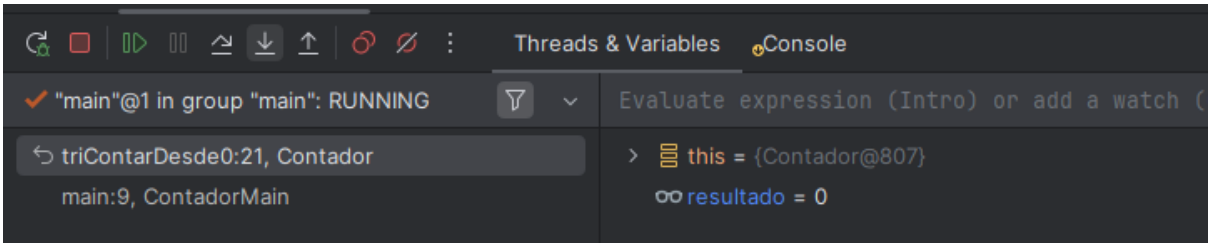
```
System.out.println("Ha contado " + contador.getResultado());
```

haciendo uso de un método de la clase contador, obtenemos el valor de la variable resultado guardado en el objeto contador. Obtenemos el 5050 y este se pinta correctamente en nuestra terminal.

Ahora empieza la segunda función de contar (triContarDesde0) y volvemos a saltar de Main a Contador.

```
    public void triContarDesde0() {  1 usage  new *
        resultado = 0;   resultado: 5050
        this.contar();
        this.contar();
        this.contar();

        if (resultado %2 == 0) {
            // Forzamos una excepción de tipo ArithmeticException
            throw new ArithmeticException("ERROR: Resultado ha devuelto un valor par.");
        }
    }
}
```

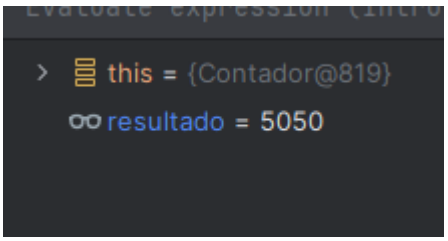
Empezando con resultado valiendo 5050 y la primera sentencia del método pone el atributo resultado a cero.




Seguimos y ahora el método llama al otro método de contar, este vuelve a ejecutar contar(). Si le pedimos que siga el programa (resume alcanzamos nuevamente el fallo). Por tanto, el fallo está después de terminar las tres llamadas de contar. Cambiamos los breakpoints a estar 3 llamadas:

```
18
19      public void triContarDesde0() {  1 usage  new *
20          resultado = 0;
            this.contar();
            this.contar();
            this.contar();
24
            if (resultado %2 == 0) {
26              // Forzamos una excepción de tipo ArithmeticException
27              throw new ArithmeticException("ERROR: Resultado ha devuelto un valor par.");
28          }
29      }
30  }
31
```

Ahora vamos comprobar que resultado obtenemos al finalizar cada uno de estos:




En el segundo:

```
18
19      public void triContarDesde0() {  1 usage   new *
20          resultado = 0;   resultado: 10100
21          this.contar();
22          this.contar();
23          this.contar();
24
25          if (resultado %2 == 0) {
26              // Forzamos una excepción de tipo ArithmeticException
27              throw new ArithmeticException("ERROR: Resultado ha devuelto un valor par.");
28          }
29      }
30  }
```

Básicamente obtenemos 5050 * 2, por lo que vamos a obtener en el tercero 15150 en el tercero:

```
19      public void triContarDesde0() {  1 usage   new *
20          resultado = 0;
21          this.contar();
22          this.contar();
23          this.contar();
24
25          if (resultado %2 == 0 = true ) {   resultado: 15150
26              // Forzamos una excepción de tipo ArithmeticException
27              throw new ArithmeticException("ERROR: Resultado ha devuelto un valor par.");
28          }
29      }
30  }
31
```

En este punto evaluamos si es un número par, y efectivamente es par, por lo que obtendremos una excepción.

**6. Ejecuta la clase Sumador. Pon un punto de ruptura en la línea 19 (int n=5). Entra en el método sumaImpares utilizando la función de debug step into y evalúa el valor de suma, cantidad e impar. ¿Funciona correctamente? Si no es así, corrige el programa.**

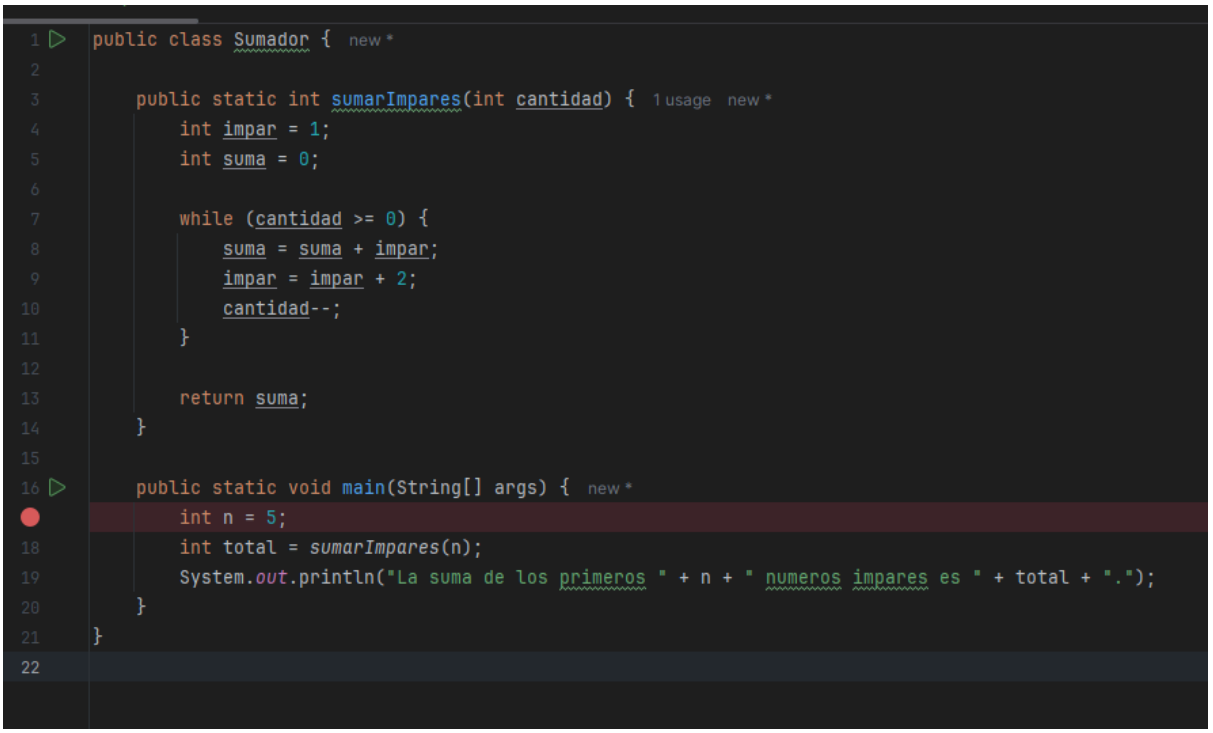
```
public class Sumador {

    public static int sumarImpares(int cantidad) {
        int impar = 1;
        int suma = 0;

        while (cantidad >= 0) {
            suma = suma + impar;
            impar = impar + 2;
            cantidad--;
        }

        return suma;
    }

    public static void main(String[] args) {
        int n = 5;
        int total = sumarImpares(n);
        System.out.println("La suma de los primeros " + n + "
numeros impares es " + total + ".");
    }
}
```

Este código no devuelve ninguna excepción durante su ejecución obteniendo:

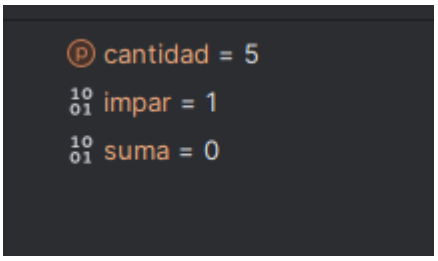
```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaa
La suma de los primeros 5 numeros impares es 36.

Process finished with exit code 0
```

Ahora, es posible que el resultado, pese a no fallar, sea incorrecto. Si ponemos el

breakpoint:

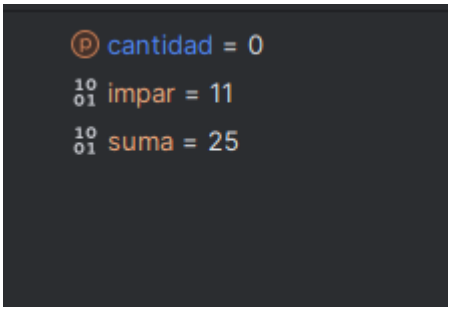


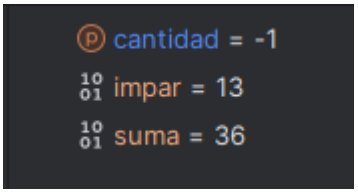
```java
public class Sumador {  new *

    public static int sumarImpares(int cantidad) {  1 usage  new *
        int impar = 1;
        int suma = 0;

        while (cantidad >= 0) {
            suma = suma + impar;
            impar = impar + 2;
            cantidad--;
        }

        return suma;
    }

    public static void main(String[] args) {  new *
        int n = 5;
        int total = sumarImpares(n);
        System.out.println("La suma de los primeros " + n + " numeros impares es " + total + ".");
    }
}
```




Antes del bucle while las variables que tenemos son las siguientes

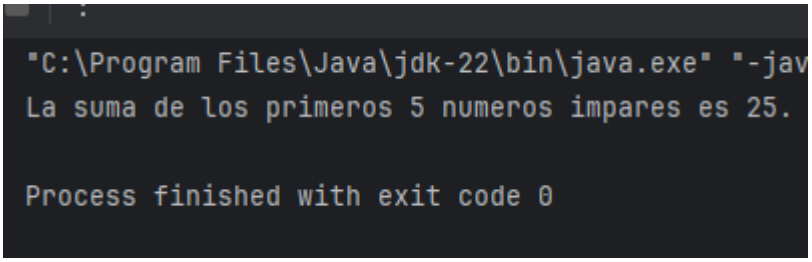
El problema es que se ejecuta cuando cantidad vale cero:




El resultado debería haber sido 25. El error en el código esta en la condición del bucle. Debería evaluar cantidad > 0:

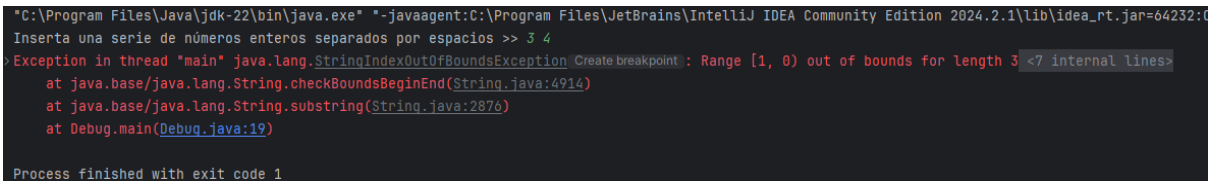
```java
while (cantidad > 0) {
    suma = suma + impar;
    impar = impar + 2;
    cantidad--;
}
```



## 7. La clase Debug contiene un método con errores lógicos. Pon puntos de ruptura y debuguea el código para entender qué hace. ¿Qué intenta realizar el programa? Corrige los errores para que funcione correctamente.

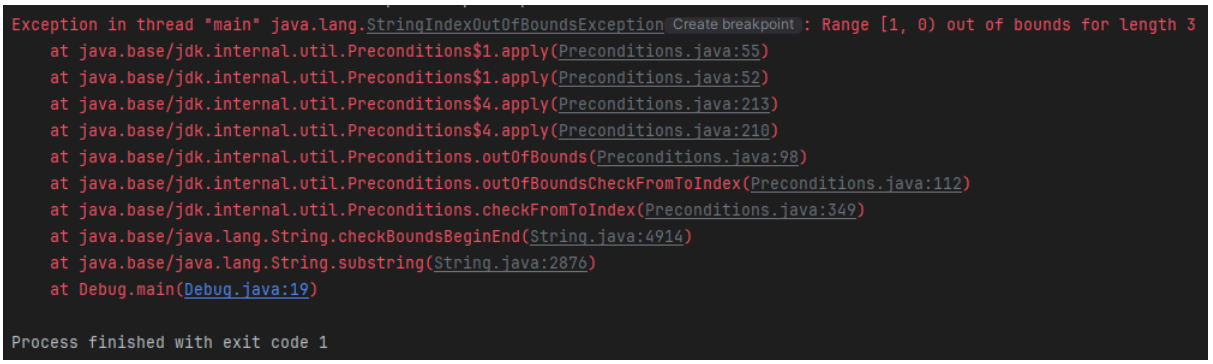
```
import java.util.Scanner;

public class Debug {

   public static void main( String [] args ) {
       int start = 0;
       int num;
       int ultimoEspacio = -1;
       int sum = 0;
       String partTexto;
       Scanner in = new Scanner(System.in);

       System.out.print("Inserta una serie de números enteros
separados por espacios >> ");
       String texto = in.nextLine();
       int longitud = texto.length();

       for(int i = 0; i <= longitud; ++i) {
           if(texto.charAt(i) == ' ') {
               partTexto = texto.substring(i, ultimoEspacio + 1);
               num = Integer.parseInt(partTexto);
               System.out.println("                " + num);
               sum = num;
               ultimoEspacio = i;
           }
       }

       partTexto = texto.substring(ultimoEspacio + 1, longitud);
       num = Integer.parseInt(partTexto);
       System.out.println("                " + num);
       sum = num;
       System.out.println("          -------------------" + "\nThe
sum of the integers is " + sum);
   }
}
```

El programa nos pide que introduzcamos una serie de valores enteros separados por espacios:

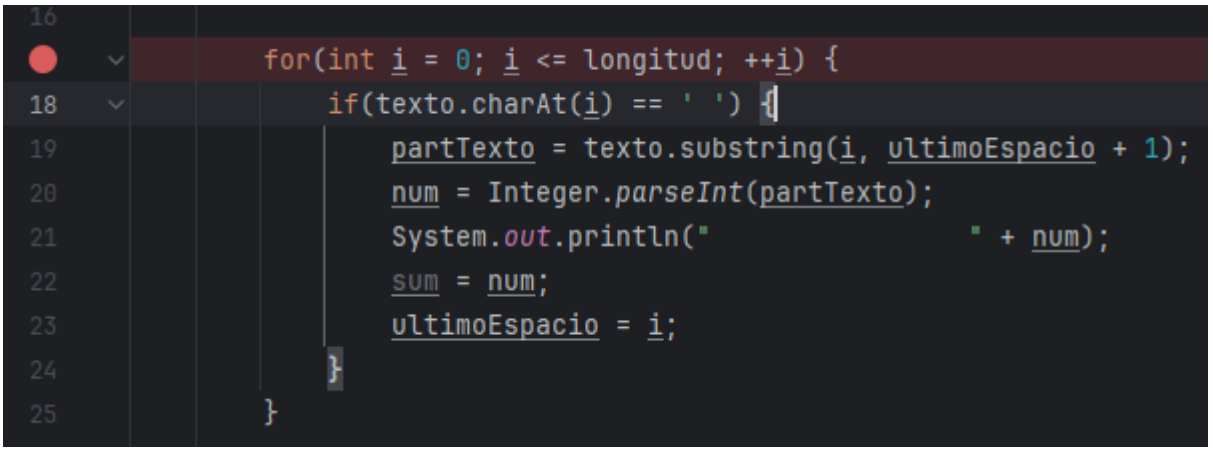



Obtenemos un StringIndexOutOfBoundsException en la línea 19.
El mensaje de error nos dice “ Range [1, 0) out of bounds for length 3”

Si mostramos toda la traza sale esto:



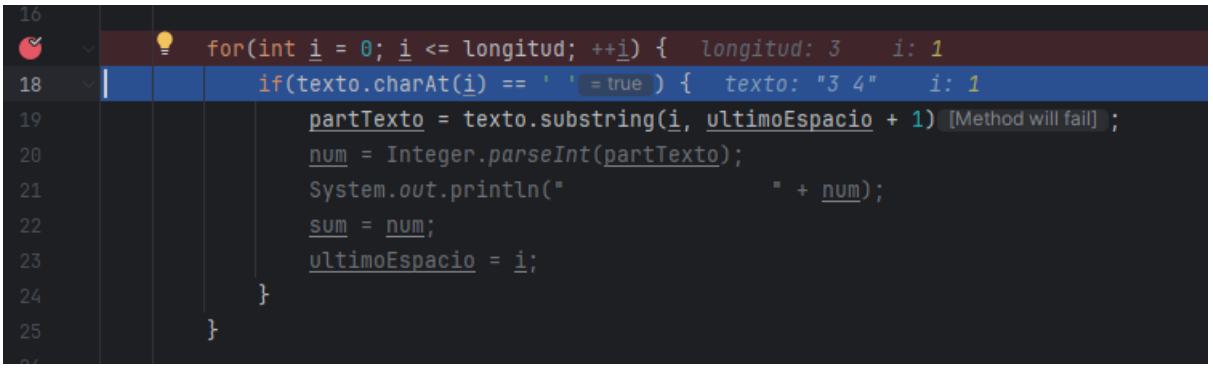

El error se produce al pedirle al método substring que se ejecute. Vamos a ver que recibe este método, ponemos un breakpoint justo al inicio del bucle for:



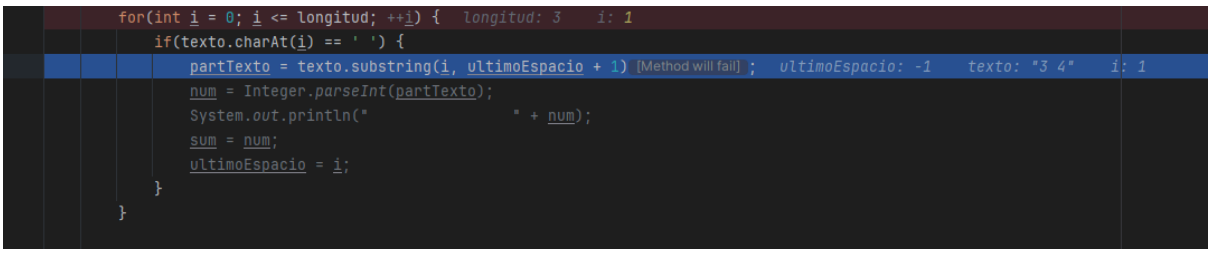

Después de darle exactamente los mismos parámetros de entrada vemos que justo entra en el bucle con las siguientes variables:


Vamos a olvidarnos del objeto Scanner (in). Por un lado tenemos texto “3 4”, por otro lado tenemos longitud = 3, start = 0 que es una variable que no usamos en todo el programa. ultimoExpacio = -1 y sum = 0.

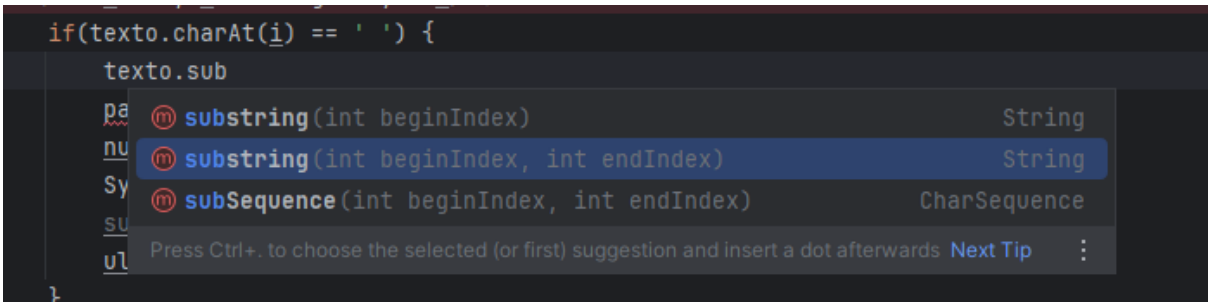
Seguimos en la primera iteración del bucle se crea la variable i = 0, cuando se comprueba si texto tiene espacio en esa posición devuelve falso, el carácter 0 en texto es 3. Seguimos en la segunda iteración:




La condición se evalúa como true, por lo que vamos a entrar dentro del bloque if:




Obtenemos un aviso de que el método fallará. Veamos qué argumentos toma el método substring:



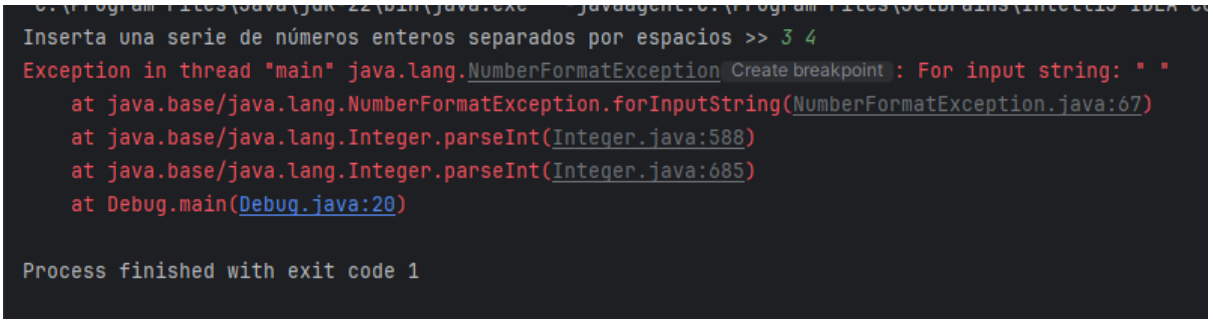

El método toma como argumentos 1 y 0.

```
Range [1, 0) out of bounds
```

El método debería de tomar como segundo argumento la longitud de la cadena texto menos i.

```
partTexto = texto.substring(i, texto.length());
```

Esto no produciría fallo en la línea, pero ahora fallaría en la siguiente línea:




Esto es porque fallamos en entender cómo debería funcionar el código. Al final imprimimos un mensaje.

```
System.out.println("          --------------------" + "\nThe sum of
the integers is " + sum);
```

Por lo que el objetivo del código es sumar una serie de números que le pasamos como texto separado por espacios. Substring está aquí para ir sacando cada número introducido por separado. Entonces, lo que está fallando es la línea de substring. Ahora mismo estamos extrayendo sólamente el carácter del espacio. Debemos ignorar ese carácter y seguir desde el siguiente que es un número.

```
partTexto = texto.substring(i + 1, longitud );
```

```
16
        for(int i = 0; i <= longitud; ++i) {
18          if(texto.charAt(i) == ' ') {
19              partTexto = texto.substring(i + 1, longitud );
                num = Integer.parseInt(partTexto);
21              System.out.println("                " + num);
22              sum = num;
23              ultimoEspacio = i;
```

Se seguimos ahora el programa:

```
args = {String[0]@1077} []
start = 0
ultimoEspacio = 1
sum = 4
> in = {Scanner@1078} "java.util.Scanner
> texto = "3 4"
longitud = 3
i = 3
```

```
    for(int i = 0; i <= longitud; ++i) {   longitud: 3    i: 3
        if(texto.charAt(i) [Method will fail] == ' ') {   texto: "3 4"    i: 3
            partTexto = texto.substring(i + 1, longitud );
            num = Integer.parseInt(partTexto);
            System.out.println("                " + num);
            sum = num;
```

El problema es que el bucle se sale de los límites. Debemos de cambiar la condición el bucle for:

```
for(int i = 0; i < longitud; ++i)
```

```
Connected to the target VM, address: '127.0.0.1:64375', transport: 'socket'
Inserta una serie de números enteros separados por espacios >> 3 4
                4
                4
        ------------------
The sum of the integers is 4
Disconnected from the target VM, address: '127.0.0.1:64375', transport: 'socket'

Process finished with exit code 0
```

Por lo menos termina el programa, pero no hace lo que queremos. Debería haber sumado 3 + 4.

Por otro lado, no ha fallado porque le estoy dando dos números, si le doy tres falla porque el primer substring devuelve “a b c”.substring(1+1, longitud) = “b c”, que luego da error al hacer parseInt(). Por lo tanto, está mal.

Para corregirlo se me ocurre que intente hacerlo al revés. En vez de ir cortando el string en substring desde el espacio hasta el final, se me ocurre por un lado cortar el texto y quedarme el lado izquierdo (el número). Aquí el valor de último espacio y i son los que delimitan donde estaba el espacio encontrado previamente y el actual que hemos encontrado, separando los valores poco a poco.

```
import java.util.Scanner;

public class Debug {

   public static void main( String [] args ) {
        int start = 0;
        int num;
        int ultimoEspacio = -1;
        int sum = 0;
        String partTextoLeft;
        Scanner in = new Scanner(System.in);

        System.out.print("Inserta una serie de números enteros
separados por espacios >> ");
        String texto = in.nextLine();
        int longitud = texto.length();

        for(int i = start; i < longitud; ++i) {
            if(texto.charAt(i) == ' ') {
                partTextoLeft = texto.substring(ultimoEspacio + 1,
i);
                num = Integer.parseInt(partTextoLeft);
                System.out.println("                " + num);
                sum = sum + num;
                ultimoEspacio = i;
            }

        }

        String ultimoNumero = texto.substring(ultimoEspacio + 1,
longitud);
        num = Integer.parseInt(ultimoNumero);
        System.out.println("                " + num);
        sum = sum + num;
```

```
        System.out.println("          --------------------" + "\nThe
sum of the integers is " + sum);
    }
}
```

Al final del bucle solo nos falta un valor por meter, ya que solo introducimos el valor a la izquierda del espacio recién encontrado. Le pedimos que añada el último número y ya los tendríamos todos.

La variable start no la usaba en todo el código, así que se la he puesto al bucle for para no tener un literal cero colgando y una variable sin usarse.

Otro fallo que tenía el código es que simplemente cambiaba el valor de sum por el valor del número encontrado. Le he pedido que le sume el valor ahora:

```
sum = sum + num;
```

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program File
Inserta una serie de números enteros separados por espacios >> 4 65 7
                4
                65
                7
          ------------------
The sum of the integers is 76

Process finished with exit code 0
```

**8. Una vez corregido el programa de la clase Debug, maneja la excepción con un try-catch que se produce al insertar la siguiente entrada: 55 3 w**

```
import java.util.Scanner;

public class Debug {

   public static void main( String [] args ) {
       int start = 0;
       int num;
       int ultimoEspacio = -1;
       int sum = 0;
       String partTextoLeft;
       Scanner in = new Scanner(System.in);

       System.out.print("Inserta una serie de números enteros
separados por espacios >> ");
       String texto = in.nextLine();
       int longitud = texto.length();

       try {

           for (int i = start; i < longitud; ++i) {
               if (texto.charAt(i) == ' ') {
                   partTextoLeft = texto.substring(ultimoEspacio +
1, i);
                   num = Integer.parseInt(partTextoLeft);
                   System.out.println("                " + num);
                   sum = sum + num;
                   ultimoEspacio = i;
               }

           }

           String ultimoNumero = texto.substring(ultimoEspacio +
1, longitud);
           num = Integer.parseInt(ultimoNumero);
           System.out.println("                " + num);
           sum = sum + num;
           System.out.println("         -------------------" +
"\nThe sum of the integers is " + sum);
       } catch(NumberFormatException e)
       {
           System.out.println("ERROR: " + e.getMessage() + " -
Expecting Number insted!");
       }
```

```
    }
}
```

```
C:\Program Files\Java\jdk-22\bin\java.exe "-javaagent:C:\Program Files
Inserta una serie de números enteros separados por espacios >> 55 3 w
                55
                3
ERROR: For input string: "w" - Expecting Number insted!

Process finished with exit code 0
```