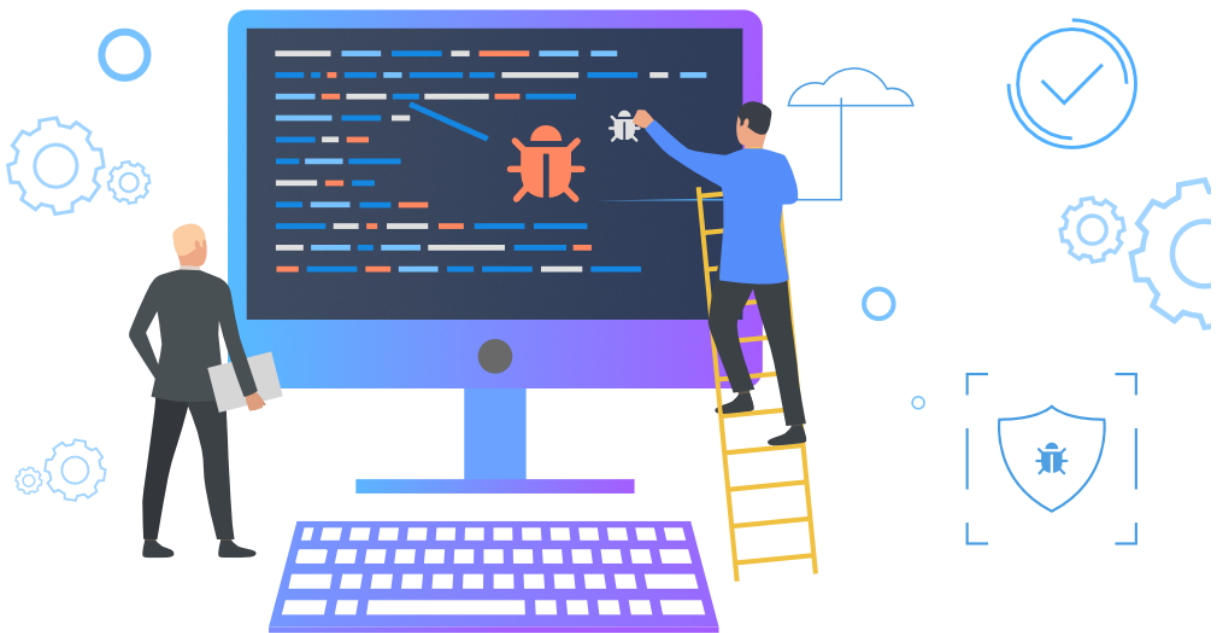


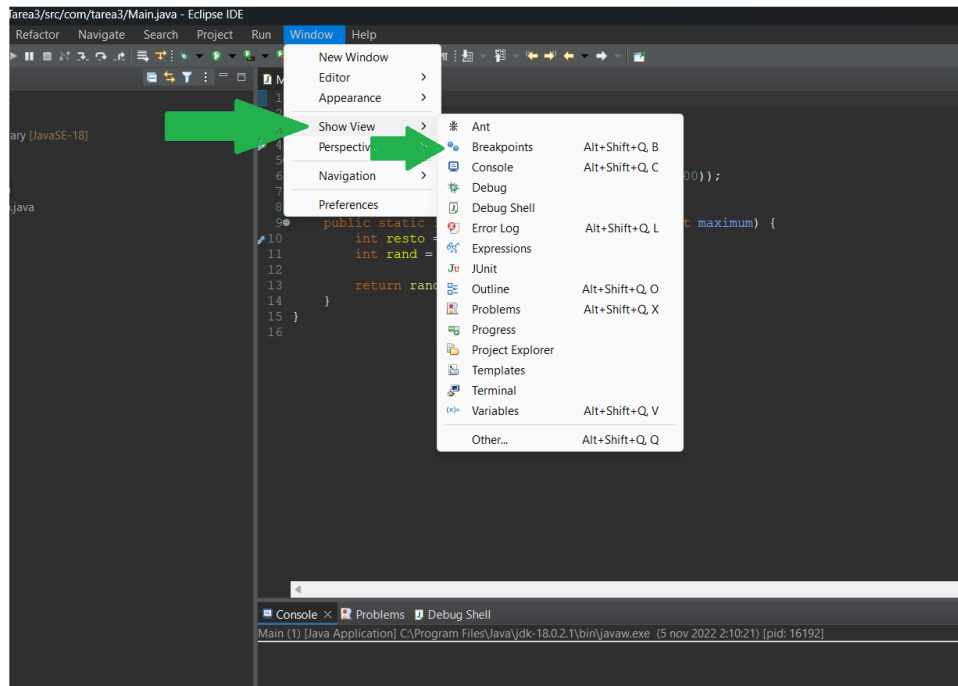
por Bruno Emiliano Mercado Sarsano
1ºDAM

USO DE LA FUNCIÓN DEBUG

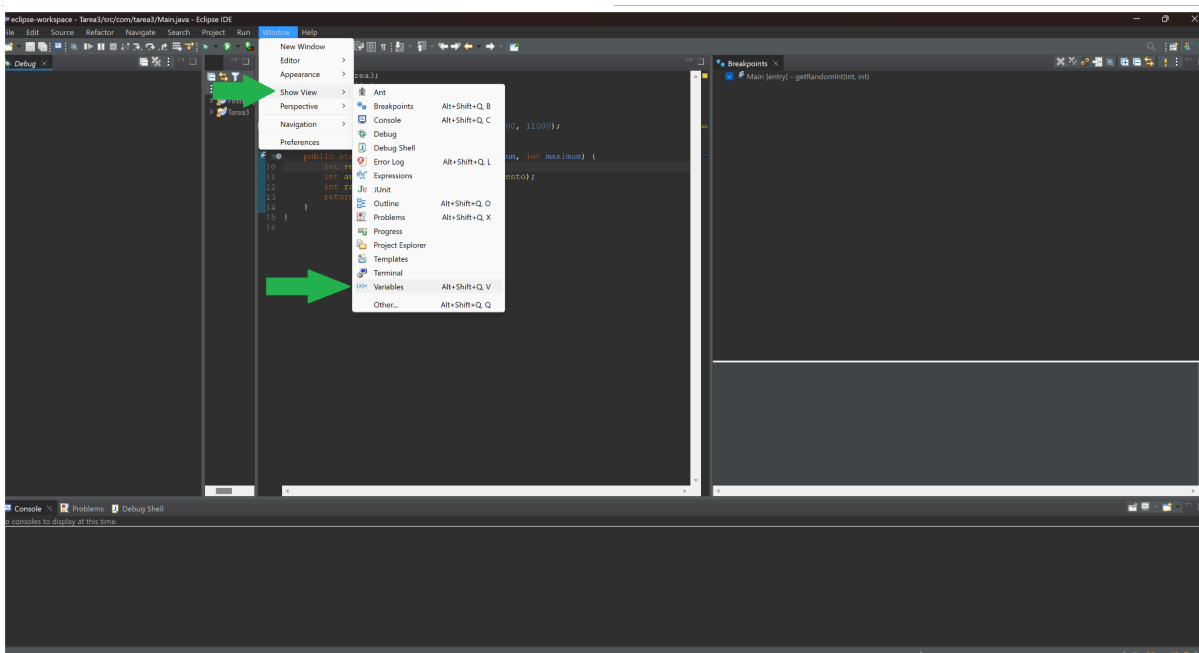


La función **debug** nos puede servir en diversas ocasiones. Por ejemplo cuando no tenemos claro lo que está haciendo el programa (esperamos un resultado y obtenemos otro) o simplemente por mera curiosidad.

Para hacer uso de la función debug necesitamos un programa. En algunos IDEs como en eclipse, no tenemos abierta la ventana por defecto, por lo tanto la abriremos siguiendo las siguientes instrucciones.



Ahora, para poder ver cómo se actualizan los valores de las variables del programa, abriremos la ventana variables de la siguiente forma.

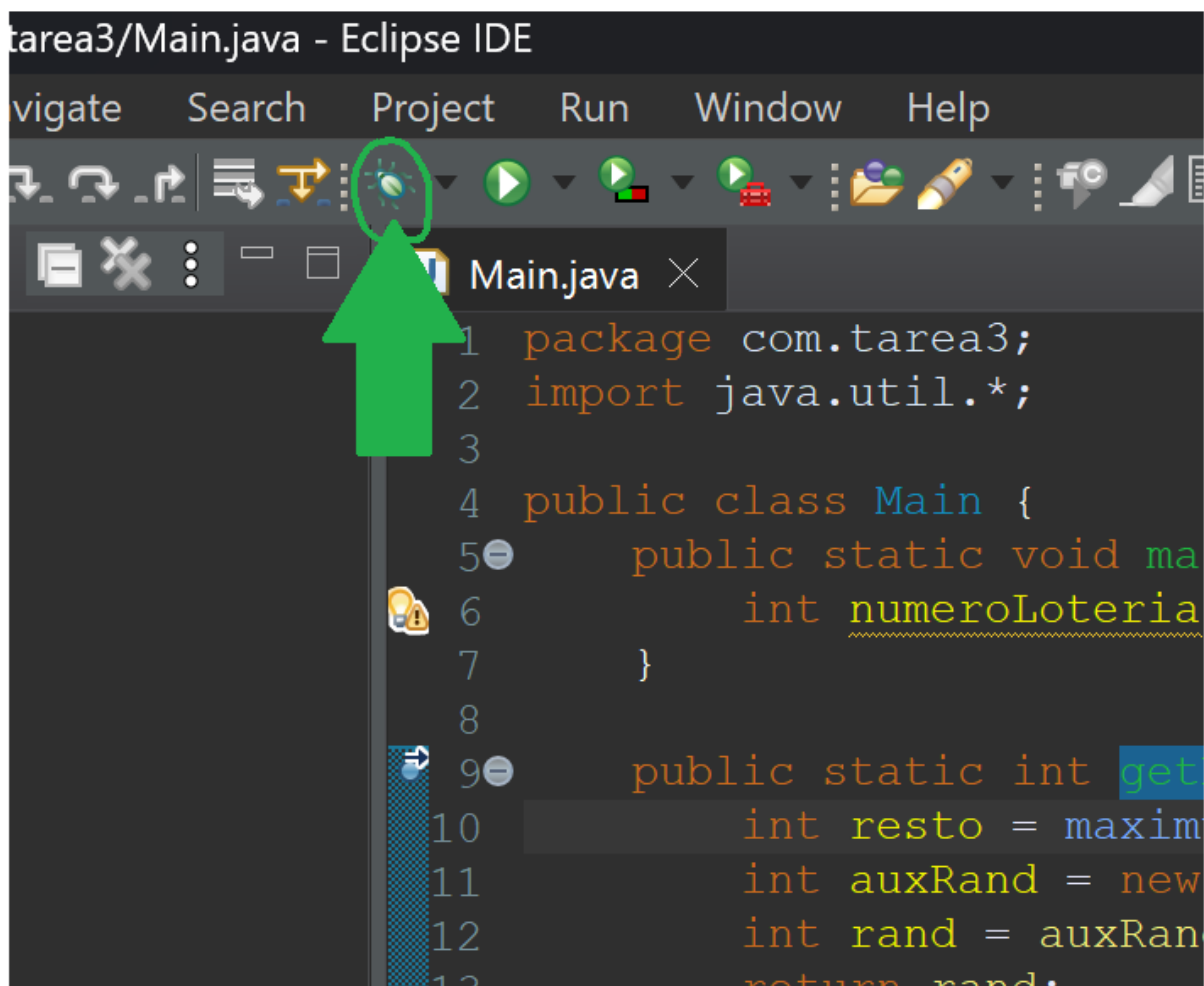


El siguiente paso sería colocar un checkpoint. Haremos doble click izquierdo sobre el lado izquierdo de la línea que nos convenga. En mi caso, quiero depurar mi función getRandomInt para ver detenidamente el valor de mis variables, por eso, coloco el checkpoint en la primera línea del método (línea 9).

```
4 public class Main {
5     public static void main(String[] args) {
6         int numeroLoteria = getRandomInt(10000, 11000);
7     }
8
9     public static int getRandomInt(int minimum, int maximum) {
10        int resto = maximum-minimum;
11        int auxRand = new Random().nextInt(resto);
12        int rand = auxRand + minimum;
13        return rand;
14    }
15 }
16
```

Ya estamos listos para ejecutar la función debug (depurar).

Lo que hará el depurador será ejecutar nuestro programa y una vez llegado al checkpoint, se detendrá.





Una vez llegados a este punto, tendremos que indicarle al IDE como queremos seguir la depuración. A nosotros nos interesa ir línea por línea, por lo tanto haremos F6 (step over).

La línea subrayada en gris transparente nos indica en qué línea nos encontramos. Como podemos observar en la ventana “Variables”, a la izquierda tenemos el nombre de las variables y a la derecha, sus valores. Estas se irán actualizando cada vez que ejecutemos step over. En caso de declarar nuevas variables, estas se mostrarán en nuevas filas.

```
1 package com.tarea3;
2 import java.util.*;
3
4 public class Main {
5     public static void main(String[] args) {
6         int numeroLoteria = getRandomInt(10000, 11000);
7     }
8
9     public static int getRandomInt(int minimum, int maximum) {
10        int resto = maximum-minimum;
11        int auxRand = new Random().nextInt(resto);
12        int rand = auxRand + minimum;
13        return rand;
14    }
15 }
16
```

Name	Value
no method return value	
minimum	10000
maximum	11000

Las siguientes imágenes representan el ciclo de vida de la función getRandomInt línea por línea.

Name	Value
no method return value	
minimum	10000
maximum	11000
resto	1000

(x)= Variables X	
Name	Value
nextInt() returned	576
minimum	10000
maximum	11000
resto	1000
auxRand	576

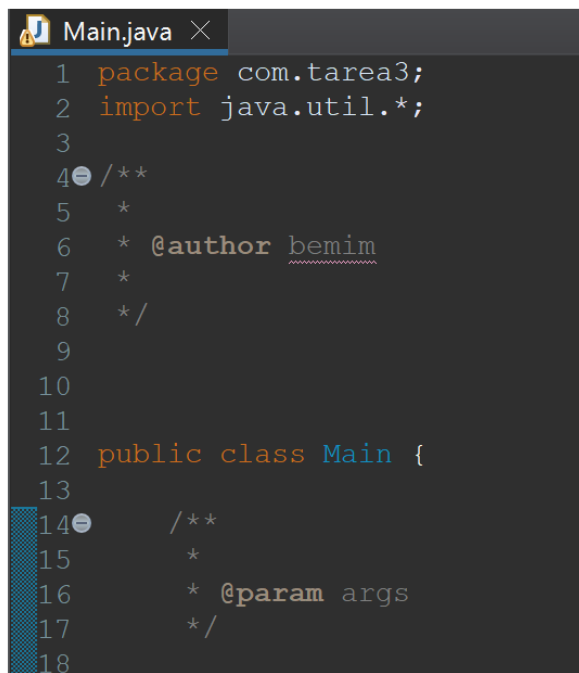
(x)= Variables X	
Name	Value
no method return value	
minimum	10000
maximum	11000
resto	1000
auxRand	576
rand	10576

(x)= Variables X	
Name	Value
getRandomInt() returned	10421
args	String[0] (id=25)

Como podemos observar, se han creado nuevas variables (resto, auxRand y rand) y tienen los valores correspondientes a las operaciones realizadas en la función. getRandomInt devolvió 10421, respetando el mínimo y máximo enviados por parámetro. getRandomInt(10000, 11000)

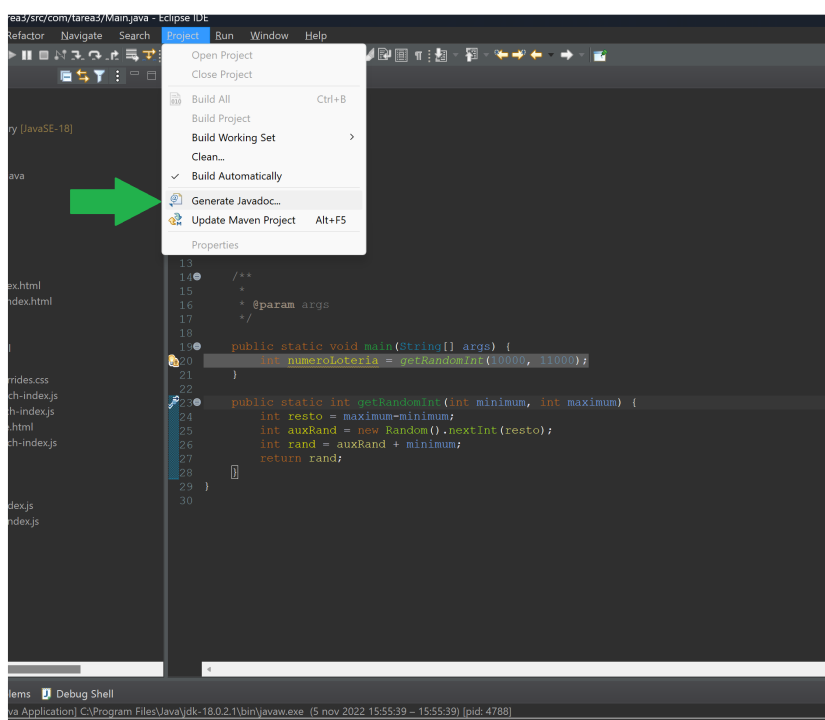
COMO CREAR EL JAVADOC

Escriba “/**” + enter fuera de la clase para generar la autoría del proyecto. Luego, hágalo dentro de la clase. Esto es para mejorar la presentación del Javadoc.

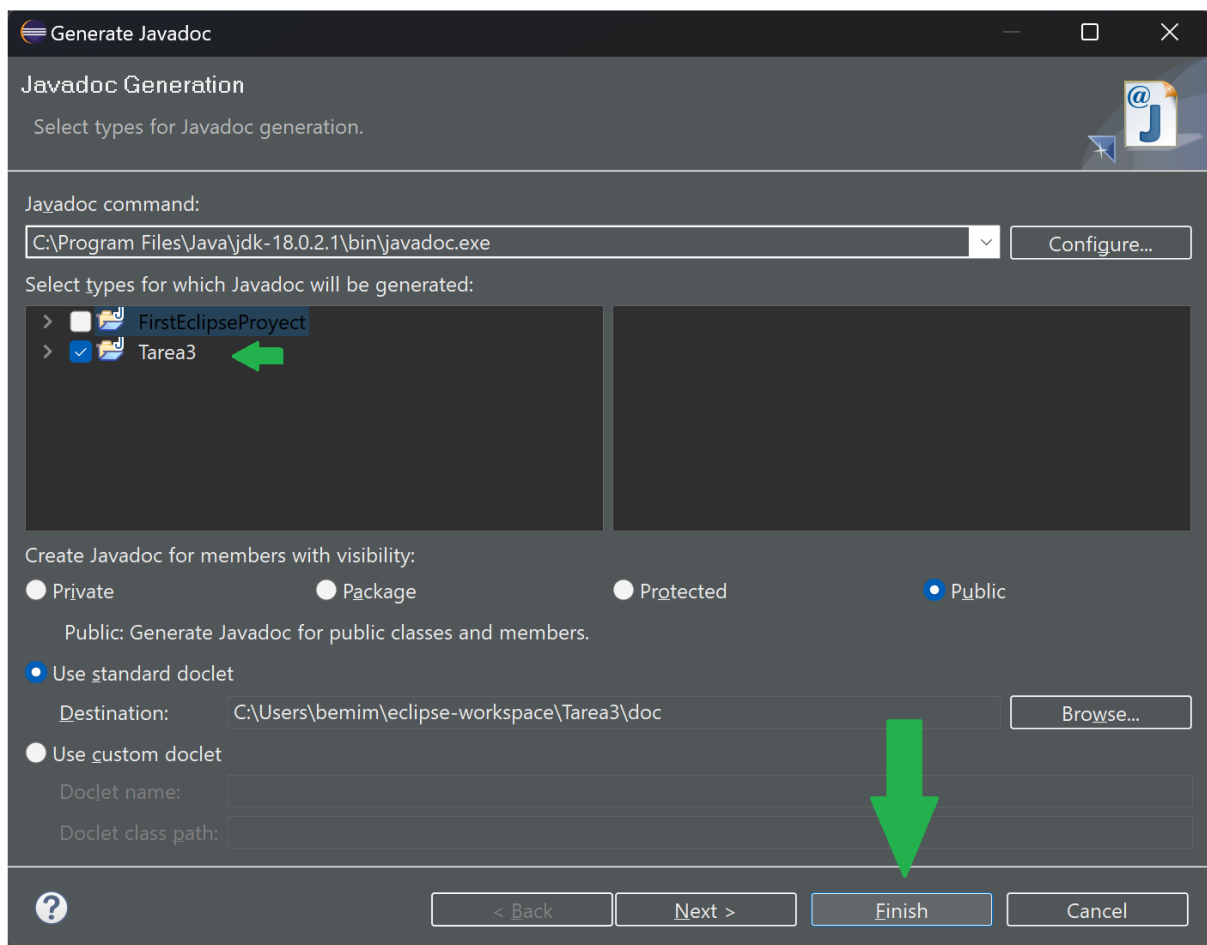


```
1 package com.tarea3;
2 import java.util.*;
3
4 /**
5  *
6  * @author bemim
7  *
8  */
9
10
11
12 public class Main {
13
14     /**
15      *
16      * @param args
17      */
18
```

Para generar el Javadoc, haga click en Project y luego en Generate Javadoc.



Luego, elija el proyecto del cual quiera generar el javadoc (en mi caso, Tarea3) y clickee en Finish



Por último, haga click en Finish y dele un nombre.
Si ha seguido todos los pasos, debería tener el javadoc listo (se crea una carpeta doc automáticamente en la carpeta del proyecto).

Aqui un ejemplo:
Javadoc creado a partir del proyecto Tarea3.

MODULE

PACKAGE

CLASS

USE

TREE

INDEX

HELP

SUMMARY: NESTED | FIELD | CONSTR | METHODDETAIL: FIELD | CONSTR | METHOD

Module Tarea3

Package com.tarea3

Class Main

java.lang.Object[Ⓜ]
com.tarea3.Main

```
public class Main
extends ObjectⓂ
```

Author:
bemim

Constructor Summary

Constructors

Constructor	Description
Main()	

Method Summary

All MethodsStatic MethodsConcrete Methods

Modifier and Type	Method	Description
static int	getRandomInt (int minimum, int maximum)	
static void	main (String [Ⓜ] [] args)	

Methods inherited from class java.lang.Object[Ⓜ]

[equals](#)[Ⓜ], [getClass](#)[Ⓜ], [hashCode](#)[Ⓜ], [notify](#)[Ⓜ], [notifyAll](#)[Ⓜ], [toString](#)[Ⓜ], [wait](#)[Ⓜ], [wait](#)[Ⓜ], [wait](#)[Ⓜ]

Constructor Details

Main

```
public Main()
```

Method Details

main

```
public static void main(StringⓂ[] args)
```

Parameters:
args -

getRandomInt

```
public static int getRandomInt(int minimum,
                               int maximum)
```


En el javadoc podemos observar quien es el autor de la clase, los métodos que contiene, su método constructor, si pertenece a algún paquete, de qué clase hereda y todo tipo de detalles sobre cualquier característica del proyecto.