Práctica de IDEs: Ferramentas de depuración

Nome e apelidos:

SECCIÓN 1: INTRODUCIÓN Á DEPURACIÓN

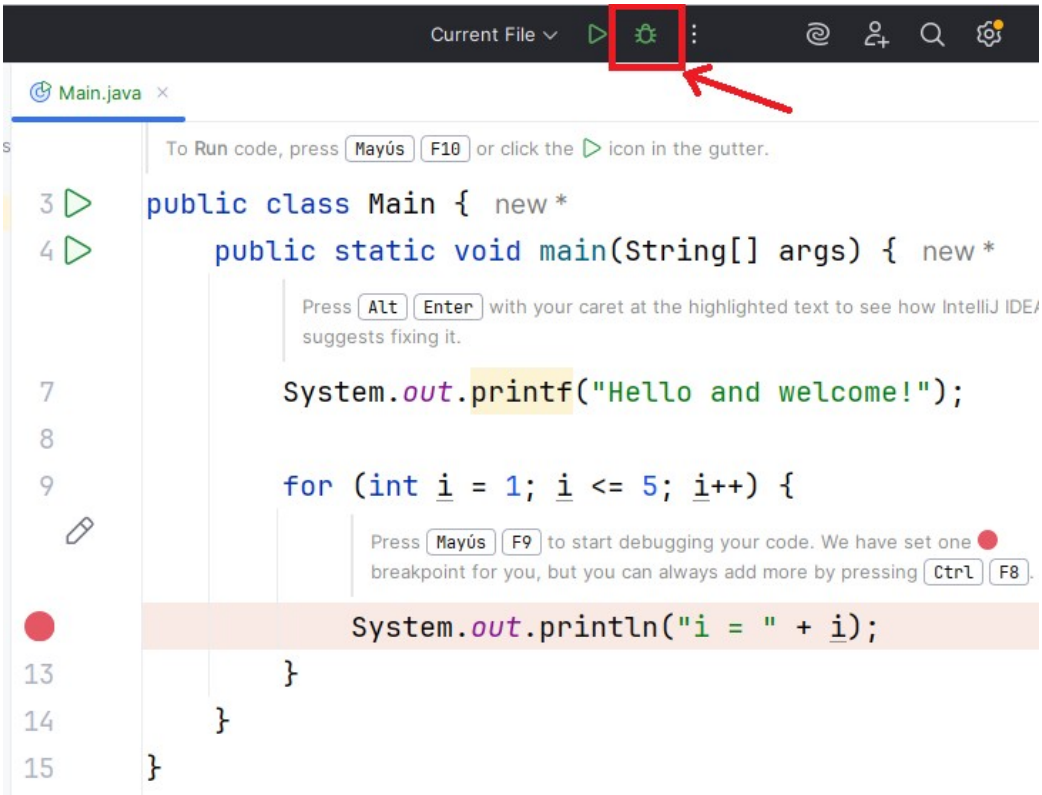
Abre IntelliJ IDEA e crea un novo proxecto.

Se xa tes outro proxecto aberto, podes pechalo primeiro desde o menú File > Close Project primeiro, ou ben ir directamente a File > New > Project.

Dalle o nome "**PedraPapelTesoira**".

No teu novo proxecto **marca** o *checkbox* "Add sample code".

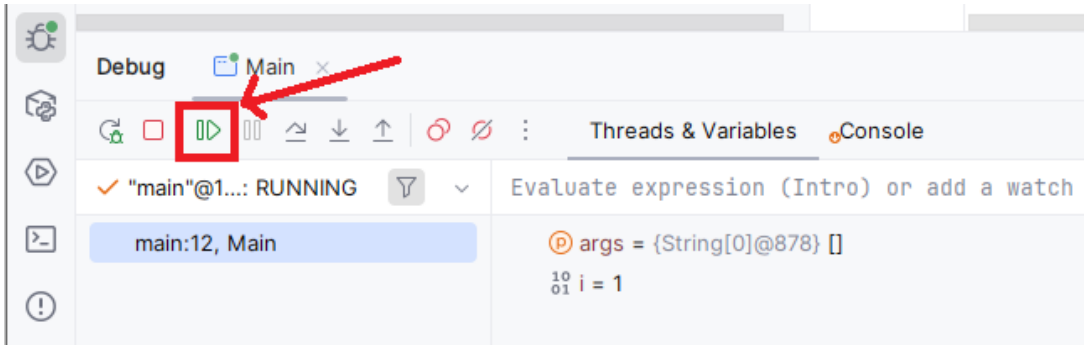
Cando todo estea listo, verás algo coma isto:



Agora, **preme** no botón dun "bichiño" que está marcado nun recadro vermello na imaxe anterior. Así, executarás o código Java en modo "depuración".

Cando executamos o código en modo “depuración” sucede algo especial: Non só se executa o programa, senón tamén un proceso adicional que serve para estudar como está a funcionar o programa en sí.

Ao executar o teu programa en modo “depuración”, aparecerá unha xanela coma a seguinte:



Clica no botón verde sinalado cun recadro na imaxe anterior. É o botón “Resume Program”

1.1. Que pasou?

Agora **segue clicando** nese botón “Resume Program” varias veces, ata que a xanela de depuración desapareza.

1.2. Dende que lanzas o programa (botón “bichiño”) ata que a xanela de depuración desaparece porque a execución remata, cantas veces tiveches que clicar no botón “Resume Program”?

❓ Fixácheste en que hai un valor no recadro da dereita que cambia cada vez que lle das a “Resume Program”?

Realmente, cando a xanela de depuración está aberta, significa que a execución do programa está pausada. Ao executar un programa en modo depuración, a execución se pausa automáticamente nos “puntos de ruptura”.

1.3. Que é un punto de ruptura?

🧠 Faille á pregunta á IA se queres, pero non esquezas engadir o contexo adecuado (e.g.: “estou aprendendo a depurar programas…”). Tamén, non copies e pegues a súa resposta. Le, entende, e engade esa información á túa intuición para plasmala coas túas palabras.

1.4. En que liña da clase Main.java hai agora mesmo un punto de ruptura?

1.5. Que liña de código (só unha) de Main.java modificarías para que durante a depuración, a execución se pausase 3 veces? (É dicir, teñas que pulsar 3 veces o botón "Resume Program" dende que lanzas o programa para que remate a execución)

Especifica o número de liña e como sería a modificación:

Agora, **elimina** o punto de ruptura da liña na que está. Debes clicar no circuliño vermello, e verás que desaparecerá.

1.6. Volve a depurar (executar co "bichiño") o programa. Pausouse a execución nalgún momento? Por que?

SECCIÓN 2: "RESUME" VS. "STEP OVER"

Agora **borra** todo o código de Main.java e escribe o seguinte programa:

Main.java

```
1  public class Main {  new *
2      public static void main(String[] args) {  new *
           System.out.println("Benvido/a");
4          System.out.println("Este é un programa de proba");
5          System.out.println("que che permitirá xogar a");
           System.out.println("un xogo.");
7      }
8  }
```

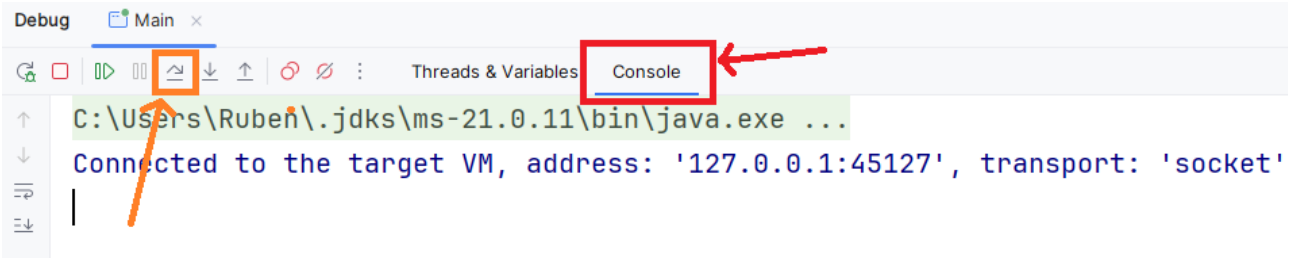
Engade tamén dous puntos de ruptura (*breakpoints*) nas liñas 3 e 6, como se ve na imaxe.

2.1. **Executa** o programa en modo depuración. En que liña se detivo a execución?

Observa que a liña na que está detido aparece resaltada en azul.

2.2. **Clica** en "Resume". En que liña se detivo a execución?

Agora, **volve** a executar o programa en modo depuración, pero cando se deteña a execución no primeiro *breakpoint*, **clica** na pestaña “Console” que se mostra a continuación cun recadro vermello.



Agora estás a ver a saída que produce o programa.

2.3. **Clica** no botón “Step Over” que está sinalado na imaxe cun recadro laranxa. Que saída ves na Consola? Copia e pega todo o contido:

2.4. En que liña está pausada a execución? Hai nesa liña algún *breakpoint*?

2.5. **Clica** dúas (2) veces máis en “Step Over”. Cal é a salida na Consola? Copia e pega todo o contido:

Agora xa podes rematar a execución do programa, empregando “Resume” ou “Step Over” varias veces.

2.6. Cal é a diferencia entre “Resume” e “Step Over”?

🧠 Non é necesario que lle preguntes á IA, pois co que fixeches debes ser capaz de deducilo. Pero, se o fas, escribe primeiro un borrador da túa resposta coa túa idea e logo contrasta iso coa IA

SECCIÓN 3: INSPECCIONANDO AS VARIABLES

Agora non só vamos a implementar un xogo de pedra-papel-tesoira, senón que usaremos as ferramentas de depuración para obter información privilexiada e gañar sempre.

Completa o código do teu programa anterior do seguinte xeito:

```
public static void main(String[] args) {  new *
    System.out.println("Benvido/a");
    System.out.println("Este é un programa de proba");
    System.out.println("que che permitirá xogar a");
    System.out.println("un xogo.");
    Random random = new Random();
    // 0 = pedra, 1 = papel, 2 = tesoiras
    int numeroAleatorioMaquina = random.nextInt( bound: 3);
    Scanner sc = new Scanner(System.in);
    System.out.println("Escolle pedra (0), papel (1) ou tesoiras (2)");
    int numeroXogador = sc.nextInt();
    if (numeroXogador == numeroAleatorioMaquina) {
        System.out.println("Empataches!");
    } else if (numeroXogador == 0) {
        System.out.println("Sacaches pedra");
        if (numeroAleatorioMaquina == 1) {
            System.out.println("A máquina sacou papel. Perdiches!");
        }
        if (numeroAleatorioMaquina == 2) {
            System.out.println("A máquina sacou tesoira. Gañaches!");
        }
    } else if (numeroXogador == 1) {
        System.out.println("Sacaches papel");
        if (numeroAleatorioMaquina == 0) {
            System.out.println("A máquina sacou pedra. Gañaches!");
        }
        if (numeroAleatorioMaquina == 2) {
            System.out.println("A máquina sacou tesoira. Perdiches!");
        }
    } else if (numeroXogador == 2) {
        System.out.println("Sacaches tesoira");
        if (numeroAleatorioMaquina == 0) {
            System.out.println("A máquina sacou pedra. Perdiches!");
        }
        if (numeroAleatorioMaquina == 1) {
            System.out.println("A máquina sacou papel. Gañaches!");
        }
    }
}
```

3.1. **Executa** o xogo (non fai falta que sexa en modo depuración). **Xoga**. Gañaches ou perdiches? Pega aquí a saída de execución:

Agora **elimina** os anteriores puntos de ruptura, se os tes, e **engade** un novo na liña 13.

Lanza en modo depuración o xogo. Vaise deter na liña 13. Na pestaña “Threads & Variables” podes ver o valor das variables que existen nese momento na execución.

3.2. Que valor ten “numeroAleatorioMaquina”?

Procede a continuar a execución con “Resume Program”.

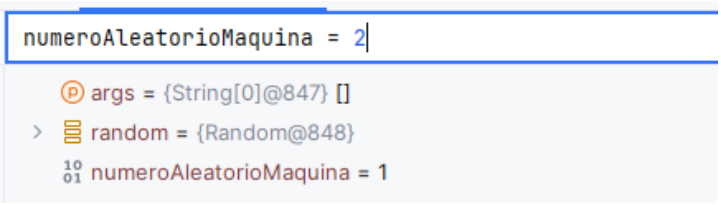
Agora **clica** na pestaña “Console” para ver a saída de execución. O programa está esperando a túa resposta.

3.3. Que número tes que poñer para gañar? **Escríbeo** e pega a saída de execución:

O potencial do depurador é que nos permite deter a execución en calqueira momento, e analizar o que está pasando por debaixo no programa e cales son os valores das variables. Isto é moi poderoso.

A continuación vamos a ir un paso máis aló. Non só vamos “inspeccionar” as variables. Vamos a realizar modificacións.

Lanza unha vez máis o xogo en modo depuración, ata que saia a xanela do punto de ruptura da liña 13. Logo, executa no pequeno recadro unha expresión coma a seguinte:



Como ves, “numeroAleatorioMaquina” era orixinalmente 1. Ao executar esa liña, cambiámolo a 2.

Fai o mesmo, modificando o valor a un distinto do que tivera orixinalmente.

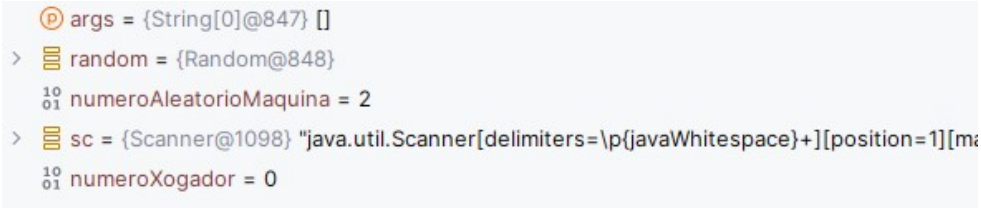
3.4. Que valor tiña “numeroAleatorioMaquina” e a que valor o modificaches?

Despois disto, completa o xogo se queres.

Elimina o *breakpoint* da liña 13 e **engade** outro novo na liña 16.

Volve a depurar o programa. **Introduce** a túa opción escollida e despois pulsa “Resume Program” para continuar a execución.

Cambia á pestaña “Threads & Variables” para ver algo coma o seguinte:



3.5. Cales son os valores de “numeroAleatorioMaquina” e “numeroXogador” na túa execución? Que pasaría se reanudases a execución?

3.6. A que número debes cambiar “numeroXogador” para gañar a partida? Se é preciso, faino.

SECCIÓN 4: “STEP INTO” E “STEP OUT”

Pecha o proxecto PedraPapelTesoiraContornos.

Crea un novo proxecto co nome “**XestorAforros**”. Non fai falla que marques “Add sample code”.

Crea unha nova clase “XestorAforros” co seguinte código:

```java
import java.util.LinkedList;
import java.util.List;

public class XestorAforros {
    private List<Integer> transaccions;

    public XestorAforros(int balanceInicial) {
        this.transaccions = new LinkedList<>();
        this.transaccions.add(balanceInicial);
    }

    public void rexistrarTransaccion(int cantidade) {
        this.transaccions.add(cantidade);
    }
}
```

Agora, no programa Main.java **pon** este código, cun punto de ruptura na liña que se mostra.

```java
public class Main {  new *
    public static void main(String[] args) {  new *
        XestorAforros xestor = new XestorAforros( balanceInicial: 50);
        xestor.rexistrarTransaccion( cantidade: -25);
        xestor.rexistrarTransaccion( cantidade: -40);
    }
}
```

4.1. Cal será o balance final desta conta de aforros?

4.2. Que número de elementos haberá, predeciblemente, no atributo “transaccions” ao final da execución?

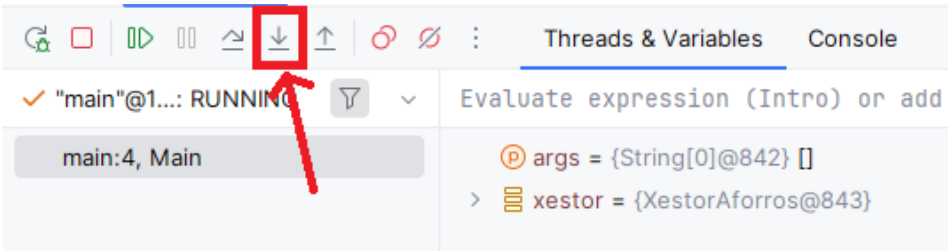
4.3. Cal é a diferencia entre “int” e “Integer”? Pódese facer unha “List” que conteña datos de tipo “int”?

🧠 Podes preguntar á IA sobre isto, pois é lóxico que non o saibas de primeiras. Asegúrate de interiorizar a resposta e adaptala ao teu modelo de pensamento. Serías capaz de explicarllo a outra persoa e que o entenda?

Executa agora o programa en modo depuración.

4.4. Coa execución detida no *breakpoint* da liña 4, que variables aparecen? Como é a variable “xestor” (escribe todo, incluido o que está entre chaves {})?

Agora **preme** no botón “Step Into” que se destaca a continuación:



4.5. A que liña de que ficheiro saltou a execución? Por que?

4.6. Que variables hai neste contexto de execución? Cal é o valor da variable “this”? Coincide co que explicaches na pregunta 4.4? Por que?

A continuación, **preme** no botón “Step Out”. Atópase á dereita de “Step Into”.

4.7. A que liña de que ficheiro saltou a execución?

4.8. De todo isto, que deduces que fan os botóns “Step Into” e “Step Out”? Como o relacionas co concepto de “pila de execución”?

⛔🧠 Non lle preguntes á IA. É interesante que só poñas o que diga a túa intuición.

Vamos crear un novo método na nosa sinxela clase de utilidade.

Agora **implementa** un novo método “`public int obterBalance()`” na clase “XestorAforros”. Debe sumar todas os números rexistrados en “transaccions” e devolver o resultado.

4.9. Escribe a continuación a túa implementación dese método.

⛔🧠 Non empregues inicialmente IA para facelo. É unha tarefa sinxela que deberías ser capaz de facer a estas alturas.

🧠 Cando a teñas feita, podes (1) pedirlla á IA e comparar críticamente o seu resultado co teu. Tamén podes (2) pedirlle á IA que che dea *feedback* sobre a túa implementación.

Engade a seguinte liña ao Main.java da aplicación:

```
System.out.println("O balance é: " + xestor.obterBalance());
```

Logo, detén e inspecciona a execución nesa mesma liña. Podes facelo premendo dúas veces “Step Over” despois de que a execución pare na liña 3, ou ben poñendo un *breakpoint* novo nesa liña.

4.10. **Preme** “Step Into” cando a execución estea pausada nesa liña. Que sucede?

4.11. Se escolles a segunda opción, a que liña de que ficheiro se trasladou a execución?

4.12. Se escolles a primeira opción, a que liña de que ficheiro se trasladou a execución?

SECCIÓN 5: EXTRA (NON AVALIABLE)

5.1. Cando a execución está pausada nun punto de ruptura, podes facer clic dereito e seleccionar “New Watch”. Despois, podes insertar o nome dunha variable. Que é isto? Para que serve?

5.2. Que problemas ten o código do programa pedra-papel-tesoira da sección 3? Faino de novo validando correctamente a entrada de usuario.

5.3. No pedra-papel-tesoira empréganse números (*ints*) para representar a opción de cada xogador. Isto é lexible? Cres que é boa práctica? Que outros tipos de dato se poderían empregar? Que vantaxes e desvantaxes ten cada un?

5.4. Implementa un novo pedra-papel-tesoira que teña un menú inicial no que che permita (1) xogar contra a máquina e (2) xogar contra outro humano por turnos na mesma consola. Implementa dita funcionalidade.

5.5. Que problemas lle ves ao teu “XestorAforros”? Como o mellorarías?

5.6. Como se pode depurar un programa escrito en Python?

5.7. Como se pode depurar un programa escrito en C?