

The logo for Oracle Academy. The word "ORACLE" is in a bold, orange, sans-serif font. Below it, the word "Academy" is in a smaller, dark grey, sans-serif font. The entire logo is centered on a light grey background, which is framed by dark grey horizontal bars at the top and bottom.

ORACLE

Academy

Java Foundations

2-2

¿Qué Está Haciendo mi Programa?

ORACLE
Academy



Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

Objetivos

- En esta lección se abordan los siguientes objetivos:
 - Comprender cómo se lee Java línea a línea
 - Definir y usar puntos de ruptura
 - Terminar sentencias con punto y coma (;)
 - Organizar el código usando espacios en blanco y otras convenciones
 - Crear comentarios



Lectura de un programa línea a línea

- Las líneas de los programas se leen de una en una

```
1 System.out.println("Line 1");
2 System.out.println("Line 2");
3 System.out.println("Line 3");
4 System.out.println("Line 4");
5 System.out.println("Line 5");
```

- En el ejemplo:
 - Se lee la línea 1...
 - Después, la línea 2...
 - Después, la línea 3...
 - Después, la línea 4...
 - Después, la línea 5...

Lectura línea a línea

- Normalmente, el lenguaje Java se lee línea a línea
- Pero hay algunos factores adicionales que se deben tener en cuenta
- Vamos a investigar sirviéndonos de...
 - Puntos de ruptura
 - Otras funciones de un IDE Java



Puntos de ruptura

- Defina un punto de ruptura en el código para
 - Pausar la ejecución de código
 - Ver el estado actual del programa
 - Facilitar la depuración
- Los puntos de ruptura afectan a la ejecución de código...
 - Cuando el código se ejecuta con el depurador
- Los puntos de ruptura no pueden afectar a la ejecución de código...
 - Cuando el código se está ejecutando con normalidad



ORACLE
Academy

JFo 2-2
¿Qué Está Haciendo mi Programa?

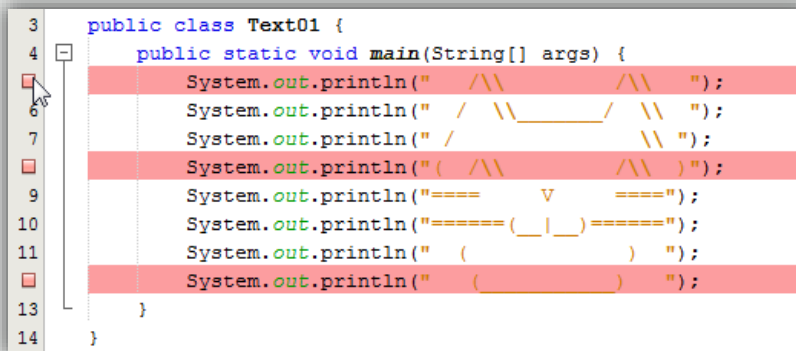
Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

6

Los pasos, las capturas de pantalla y los iconos de esta lección son de NetBeans. Si utiliza otro IDE Java, consulte la documentación correspondiente para conocer los pasos necesarios para agregar puntos de ruptura y depurar el código.

Definir la Animación de un Punto de Ruptura

- Para definir un punto de ruptura...
 - Coloque el cursor sobre un número en el margen izquierdo
 - Haga clic... y tendrá un punto de ruptura
 - Vuelva a hacer clic para eliminar un punto de ruptura
 - Puede configurar varios puntos de ruptura



```
3 public class Text01 {  
4     public static void main(String[] args) {  
5         System.out.println("  /\\"  
6         System.out.println(" /  \\"  
7         System.out.println(" /      \\"  
8         System.out.println("(  /\\"  
9         System.out.println("====  V  ===");  
10        System.out.println("===== (|) =====");  
11        System.out.println(" (      ) ");  
12        System.out.println(" (      ) ");  
13    }  
14 }
```

The screenshot shows a code editor with a Java class named Text01. A breakpoint is set at line 5, indicated by a red square in the left margin. The code contains several println statements with escape characters and symbols to create a visual animation.

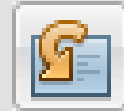
Ejercicio 1, Parte 1

- Cree un nuevo proyecto y agréguele el archivo `Text01.java`
- Defina un punto de ruptura en la línea 5 (la línea con la primera sentencia `print`)
- Ejecute el programa con normalidad
 - Los puntos de ruptura no deben tener ningún efecto

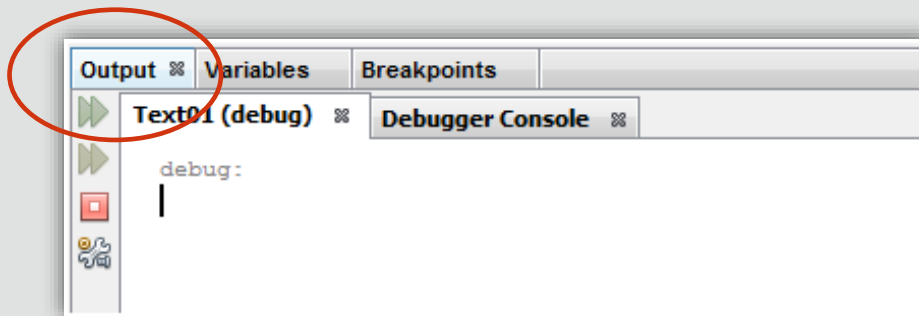


Ejercicio 1, Parte 2

- Ejecute el programa con el depurador:
 - Asegúrese de que aparezca la ventana de salida
 - Pulse Step Over cada vez que quiera pasar a la línea siguiente
- Observe cómo va apareciendo el gato línea a línea



Step Over



ORACLE
Academy

JFo 2-2
¿Qué Está Haciendo mi Programa?

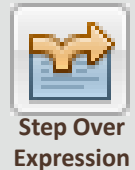
Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

9

Seleccione la ventana de salida haciendo clic en el botón Output que aparece en la esquina inferior izquierda del IDE. Haga clic en Step Over varias veces hasta que llegue al final del programa.

Ejercicio 1, Parte 3

- Modifique el código de manera que las tres primeras sentencias print aparezcan todas en la línea 5 (esto es lo que se conoce como eliminar espacios en blanco)
- Ejecute el programa con el depurador:
 - Asegúrese de que aparezca la ventana de salida
 - Haga clic en Step Over Expression cada vez que quiera pasar a la línea siguiente
 - Ignore el código complicado al final de la depuración
- Observe cómo va apareciendo el gato línea a línea
- Intente eliminar un punto y coma mientras está depurando el programa



¿Qué ocurre si lo hace antes de depurar?

ORACLE
Academy

JFo 2-2
¿Qué Está Haciendo mi Programa?

Copyright © 2022, Oracle y/o sus filiales. Oracle, Java y MySQL son marcas comerciales registradas de Oracle y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

10

Siga pulsando Step Over Expression hasta que aparezca el código complicado. Step Into Expression se parece a Step Over, con la diferencia de que permite analizar el código en mayor detalle. Pero hay ocasiones en las que no se desea tanto detalle.

Resultados de la Investigación, Parte 1



- Se puede decir que Java lee código línea a línea...
- Pero si hay varias sentencias en una misma línea, es más preciso decir que Java lee sentencia a sentencia
- Toda sentencia debe acabar en punto y coma (;)
 - Es fácil olvidarse de poner un punto y coma
 - En otros lenguajes (Python), el punto y coma no es tan importante

```
System.out.println("Meow") ;
```

- Editar código no tiene efecto alguno mientras se está ejecutando un programa
- Es necesario recompilar para que se apliquen los cambios



- ```

3 public class Text01 {
4 public static void main(String[] args) {
5 System.out.println(" /\ /\ ");
6 System.out.println(" / /\ ");
7 System.out.println("(/\ /\ ");
8 System.out.println("==== ▽ =====");
9 System.out.println("===== (|) =====");
10 System.out.println(" () ");
11
12 System.out.println(" () ");
13 }
14 }

```

*Este código funciona... pero es muy caótico*

**Este código funciona...  
pero es muy caótico**

# Espacio en Blanco

- Los espacios en blanco son espacios que están desprovistos de código:
  - Espacio entre palabras
  - Líneas en blanco
  - Sangrado delante de una línea de código

```
3 public class Text01 {
4 public static void main(String[] args) {
5 System.out.println(" /\\" /\\" ");
6
7 [red box]
8 [red box] System.out.println(" / \\" / \\");
9 System.out.println(" / \\");
10 }
11 }
```

Los espacios en blanco no incluyen los espacios en las sentencias print (hablaremos sobre las cadenas más adelante).

# Efectos de los Espacios en Blanco

- Los espacios en blanco ayudan a mantener el código organizado
- Los espacios en blanco no afectan al modo en el que se ejecuta un código
- Puede utilizar los espacios en blanco como prefiera
- Pero es muy recomendable sangrar debidamente porque...
  - Impide que haya problemas de lectura
  - Evita que se produzcan errores durante la programación

¡Ahhhhh!  
¡Qué código  
tan caótico!



# Sangrado y Corchetes Angulares

- Detrás de un corchete angular de apertura ( { ), se debe sangrar siempre con un separador adicional (4 espacios)
- Delante de un corchete angular de cierre ( } ), se debe dejar de sangrar con un separador adicional (4 espacios)
- El código comprendido entre corchetes angulares se denomina "bloque de código"
  - Al agregar un corchete angular de apertura ( { )...
  - Necesitará un corchete angular de cierre ( } ) tarde o temprano
  - Suele ser habitual olvidarse de colocar un corchete angular o que estos no estén emparejados

# Ejemplo de Bloque

```
public class Example
{
 public static void main(String[] args){
 System.out.println("Inner code");
 System.out.println("Inner code");
 {
 System.out.println("Inner-inner code");
 }
 }
}
```

*Estos corchetes angulares forman, a su vez, un bloque dentro de otro...*

*Este código también se sangra*



# Asistencia de sangrado del IDE

- Un IDE puede...
  - Codificar el ámbito de un bloque por colores (Greenfoot, BlueJ)
  - Sangrar automáticamente detrás de un corchete angular
  - Resaltar un corchete angular coincidente (como se muestra a continuación)
- Algunos comandos de Java necesitan corchetes angulares, aunque siempre puede agregar más

```
public class Example
{
 public static void main(String[] args) {
 System.out.println("Inner code");
 System.out.println("Inner code");
 {
 System.out.println("Inner-inner code");
 }
 }
}
```

Esta lección agrega corchetes angulares adicionales al código, a modo de ejemplo. Añadir corchetes angulares adicionales no es una práctica habitual.

## Ejercicio 2

- Cree un nuevo proyecto y agréguele el archivo `Text02.java`
- ¿Puede arreglar este programa y obtener los resultados siguientes?

1  
2  
3  
4

- Indicaciones:
  - Su IDE subraya el código problemático
  - Su IDE puede resaltar las llaves coincidentes
  - Su IDE tiene un acceso directo para corregir el sangrado

# Los comentarios

- Los programas cuidadosamente espaciados pueden ocupar demasiado y resultar difíciles de leer
- Puede agregar comentarios a un código para...
  - Ofrecer una explicación o facilitar información adicional al programador (comentar código)
  - Desactivar códigos y evitar que se ejecute sin tener que borrarlos (comentar código para excluirlos)

¡Ahhhhh! ¿Qué pasa con todo este código?



# Agregar Comentarios a un Código

- Comentarios de una sola línea...
  - Empiezan con dos barras inclinadas //
  - Terminan cuando acaba la línea

```
//A single line comment automatically ends when the line ends
System.out.println("This line prints");
```

# Agregar Comentarios a un Código

- Comentarios de varias líneas:

- Empiezan con una barra inclinada y una estrella `/*`
- Terminan con una barra inclinada y una estrella `*/`

```
/*A multi line comment...
continues for many lines...
System.out.println("This line does not print");
until the star-slash appears*/
System.out.println("This line prints");
```

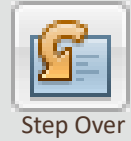
# Lectura línea a línea

- Podemos seguir investigando
- Vamos a investigar sirviéndonos de...
  - Bloques de código
  - Los comentarios
  - Puntos de ruptura
  - Otras funciones de su IDE



## Ejercicio 3

- Cree un nuevo proyecto y agréguele el archivo `Text03.java`
- Defina un punto de ruptura en la línea 11
- Ejecute el programa con el depurador:
  - Asegúrese de tener la ventana de salida seleccionada
  - Pulse Step Over cada vez que quiera pasar a la línea siguiente
- Observe como aparece la cara del gato, pero no las patas
- Escriba `drawLegs();` en la línea 19 y depure el programa
  - ¿Dónde puede agregar un punto de ruptura para ver las patas aparecer línea a línea?
  - ¿Qué resultado se obtiene al excluir las líneas mediante comentarios?



## Resultados de la Investigación, Parte 3



- Cuando Java lee línea a línea...
- Empieza dentro del bloque de código especial que se conoce como método principal

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

```
}//end method main
```

- No se ejecuta ningún otro código salvo que se especifique
  - En este ejercicio, el método principal debe llamar específicamente al bloque de código que imprime las patas
- El código comentado se ignora
  - Los comentarios se eliminan en código de byte



# Flujo del programa

1. Todos los programas Java comienzan con el método main
2. No se ejecuta ningún otro código a menos que se le llame

2) A continuación,  
vaya aquí

1) Comience aquí

```
public class Text03 {
 public static void drawLegs() {
 System.out.println(" || || ");
 System.out.println(" || || ");
 System.out.println(" (||) (||) ");
 }
 public static void main(String[] args) {
 System.out.println(" /\\"
 System.out.println(" / \\"
 System.out.println(" / \\"
 System.out.println("(/\\"
 System.out.println("==== V ===");
 System.out.println("=====(_|_)=====");
 System.out.println(" () ");
 System.out.println(" () ");
 drawLegs();
 }
}
```

# El método principal

- El método principal es un bloque de código especial
- Todos los programas de Java empiezan en el método principal
- Los programas deben tener solo 1 método principal
- Los métodos se explican más en detalle en la siguiente lección
  - drawLegs() es un ejemplo de método

```
public static void main(String[] args){
 //Your program starts here
} //end method main
```

# Objetivos

- Errores comunes:

- Falta el punto y coma (;)

```
System.out.println("Meow")
```

- Los {corchetes angulares} no están emparejados

```
{
 System.out.println("Meow");
```

- Mantenga el código organizado usando:

- Espacio en Blanco
- Corchetes angulares ( { } )
- Los comentarios

# Objetivos

- En esta lección, debe haber aprendido lo siguiente:
  - Comprender cómo se lee Java línea a línea
  - Definir y usar puntos de ruptura
  - Terminar sentencias con punto y coma (;)
  - Organizar el código usando espacios en blanco y otras convenciones
  - Crear comentarios



The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by a thin black border, with dark gray horizontal bars at the top and bottom.

# ORACLE

## Academy