

fundaciónesplai
ciudadanía comprometida

T Systems

Let's power
higher performance



@mihifidem
creativity is intelligence having fun

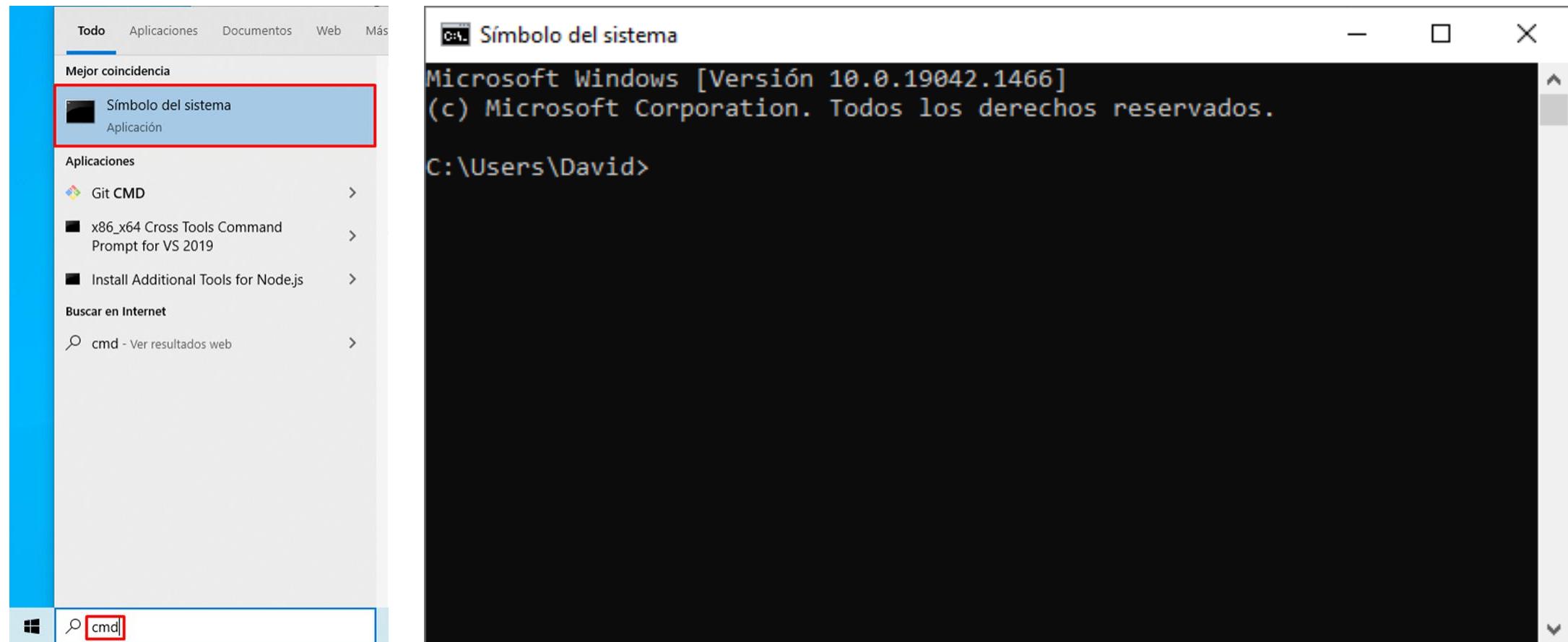
Programación estructurada y programación funcional en Java

1. Presentación



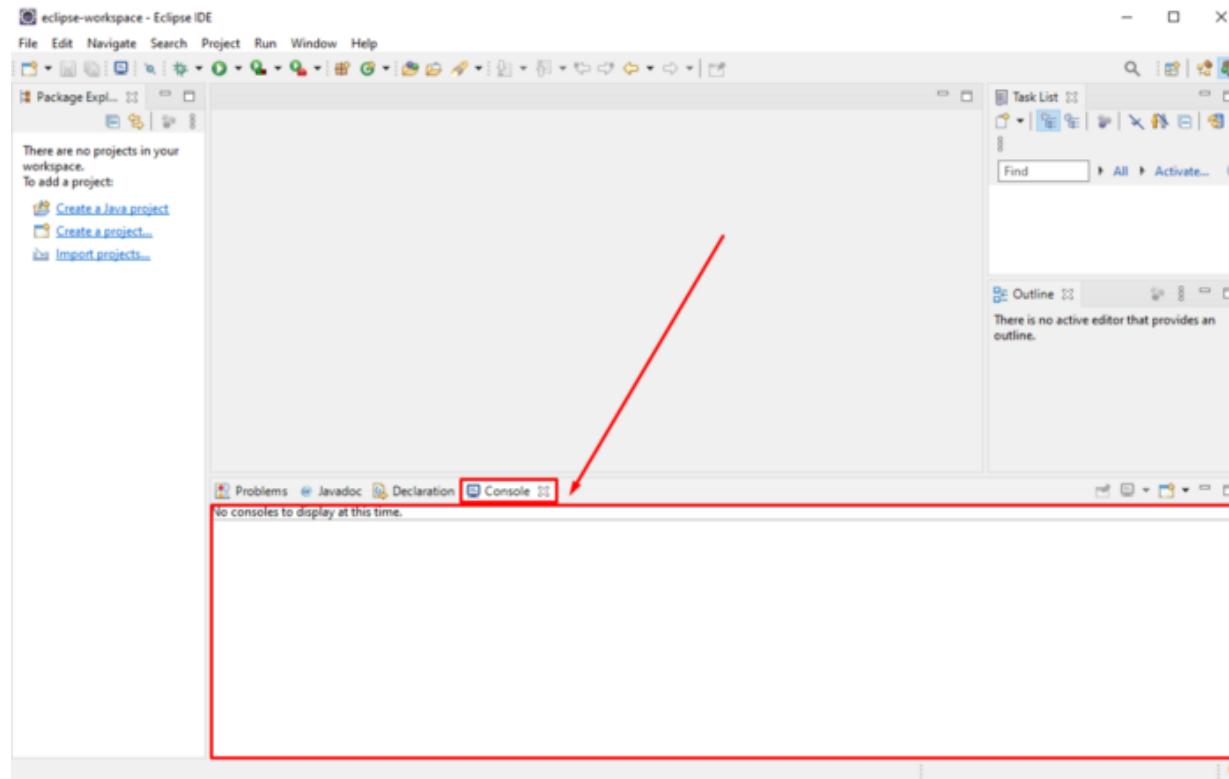
1. Presentación

1.1 Command prompt (CMD)



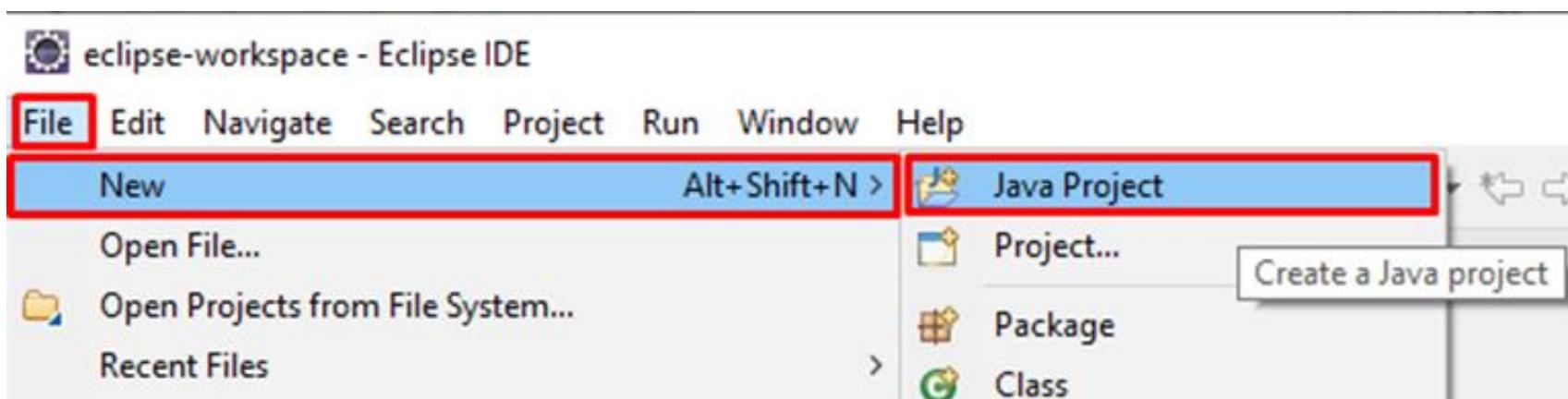
1. Presentación

1.1 Command prompt (CMD)



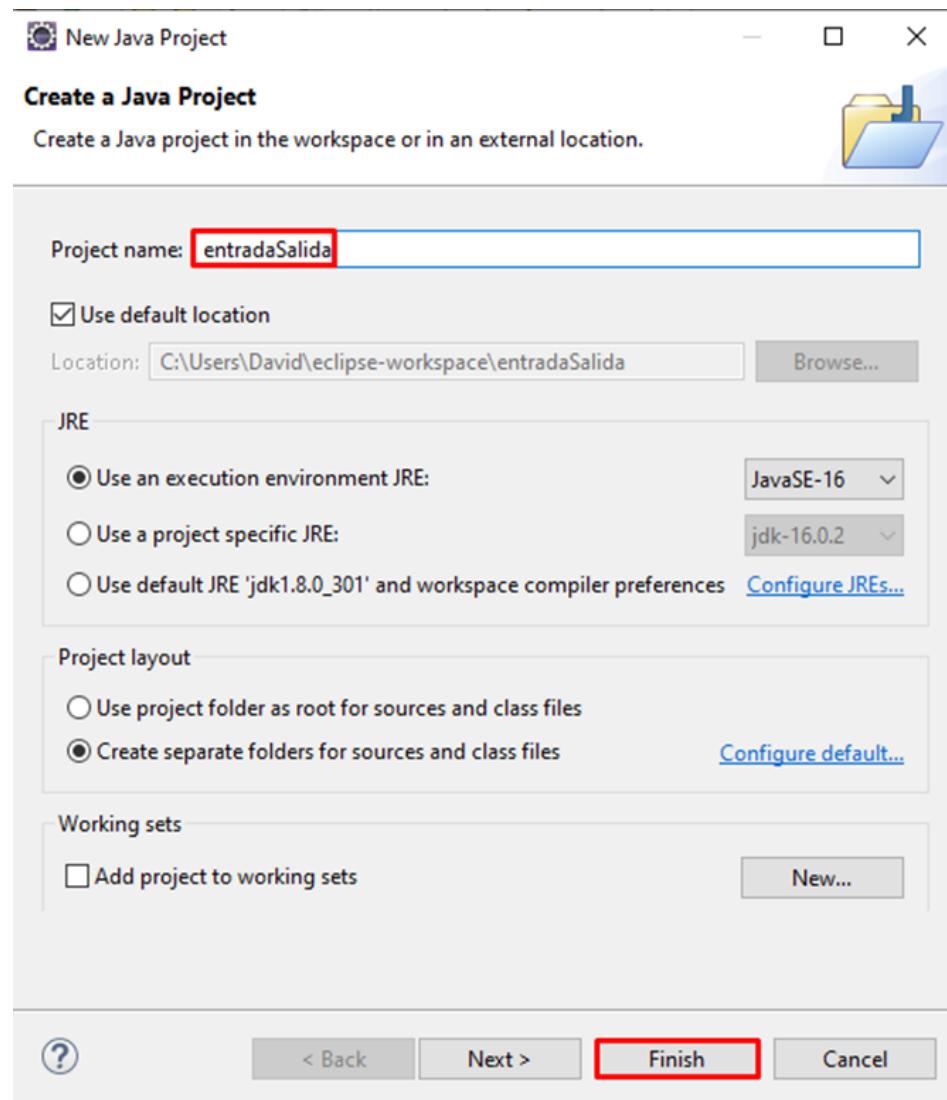
1. Presentación

1.1 Command prompt (CMD)



1. Presentación

1.1 Command prompt (CMD)



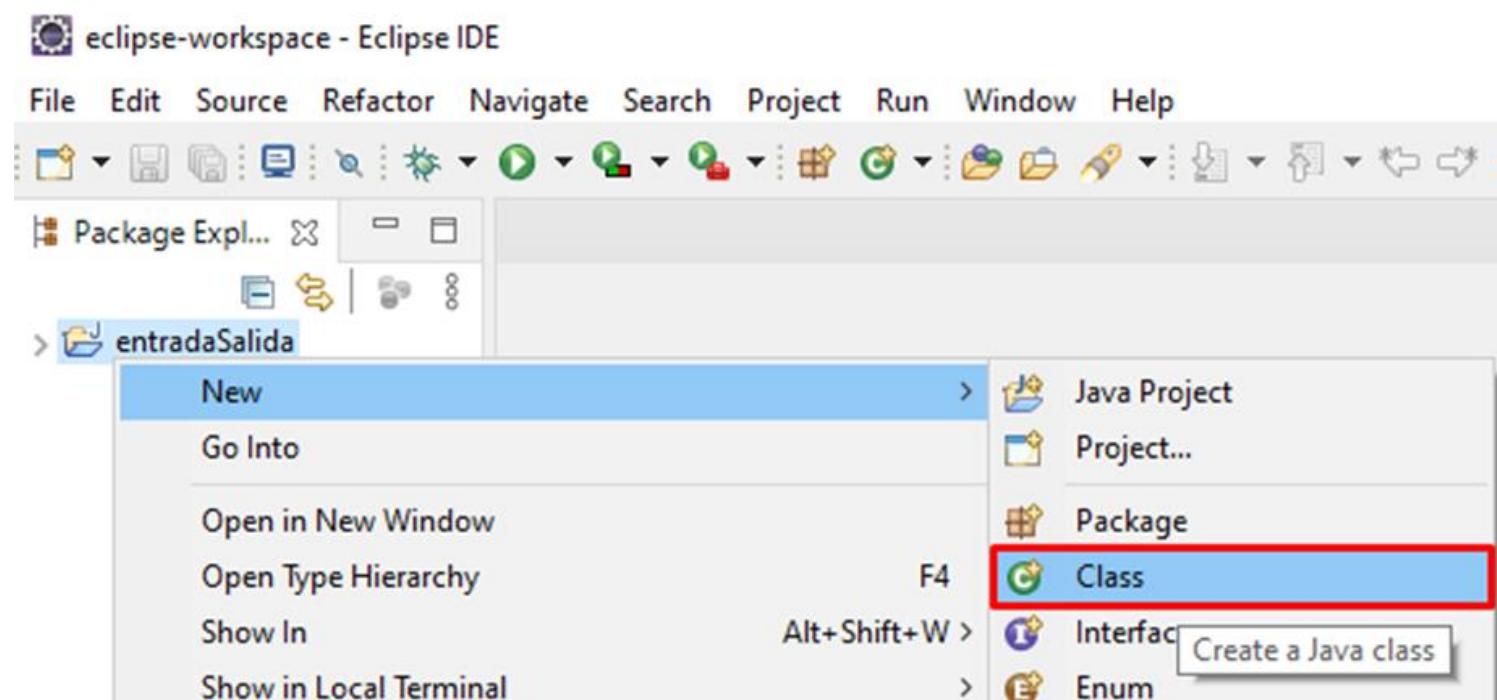
1. Presentación

1.1 Command prompt (CMD)



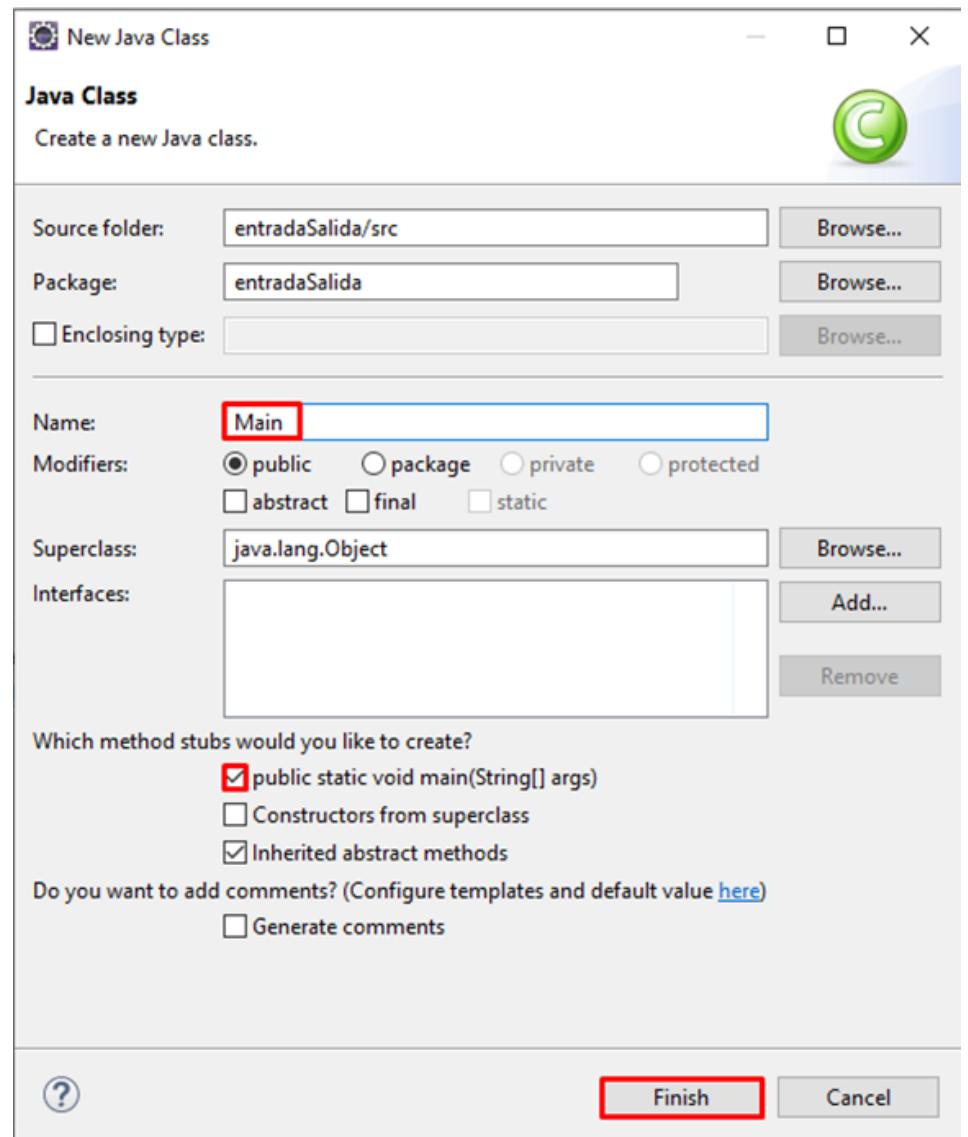
1. Presentación

1.1 Command prompt (CMD)



1. Presentación

1.1 Command prompt (CMD)



1. Presentación

1.1 Command prompt (CMD)

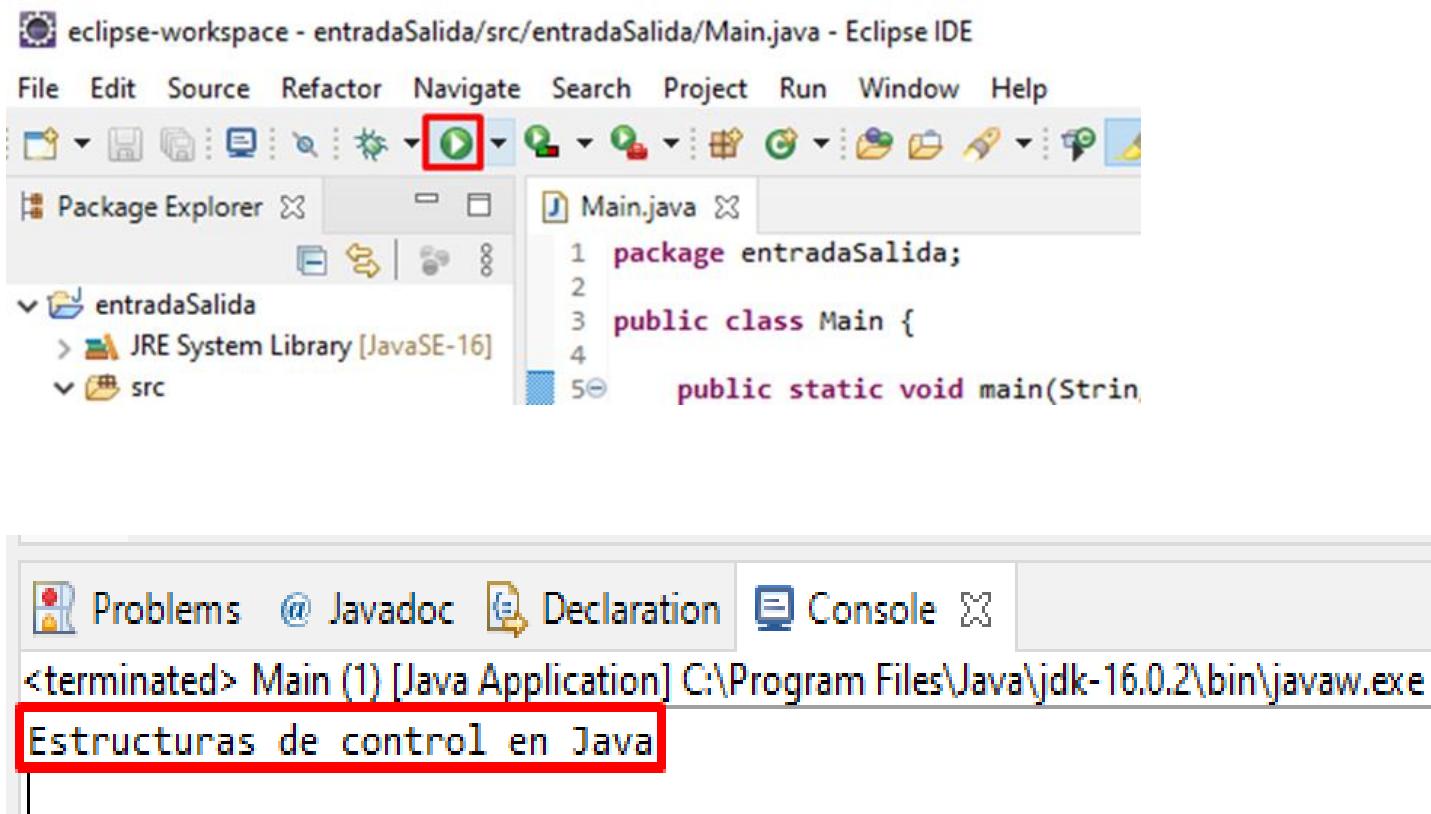
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a project structure under the package `entradaSalida`. It includes a JRE System Library entry for `[JavaSE-16]` and a source folder `src` containing a package named `entradaSalida` which contains a file named `Main.java`. On the right, the Main.java code editor window is open, showing the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Estructuras de control en Java");
7     }
8
9 }
```

The line `System.out.println("Estructuras de control en Java");` is highlighted with a red rectangle.

1. Presentación

1.1 Command prompt (CMD)



1. Presentación

1.2 Comentarios en Java

```
Main.java X
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // Comentario de una sola línea
7         System.out.println("Estructuras de control en Java");
8     }
9
10 }
```

```
Main.java X
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Estructuras de control en Java"); // Comentario de una sola línea
7     }
8
9 }
```

The screenshot shows the Eclipse IDE interface with several tabs at the top: Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of a terminated Java application named 'Main'. The output text is 'Estructuras de control en Java'.

```
Problems @ Javadoc Declaration Console X
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Estructuras de control en Java
```

1. Presentación

1.2 Comentarios en Java

```
J Main.java ✘
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Estructuras de control en Java");
7
8         /*
9             Comentario
10            de
11            varias
12            líneas
13        */
14    }
15
16 }
```

```
J Main.java ✘
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Estructuras de control en Java");
7
8         /*
9             * Comentario
10            * de
11            * varias
12            * líneas
13        */
14    }
15
16 }
```

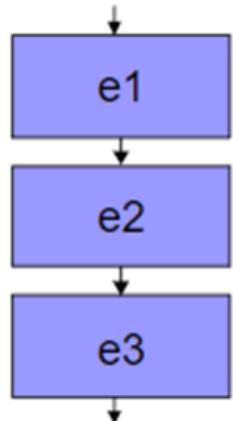
The screenshot shows a Java development environment. At the top is a code editor window titled "Main.java" containing Java code. Below the code editor is a tab bar with "Problems", "@ Javadoc", "Declaration", and "Console". The "Console" tab is active, showing the output of a Java application named "Main". The console output is: "<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe" followed by the text "Estructuras de control en Java".

```
Problems @ Javadoc Declaration Console ✘
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Estructuras de control en Java
```

1. Presentación

1.3 Tipos de estructuras de control de flujo

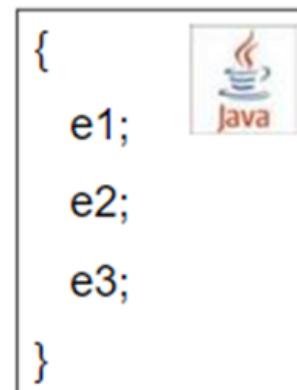
Programación secuencial Secuenciación



Comienza

e1;
e2;
e3;

Termina

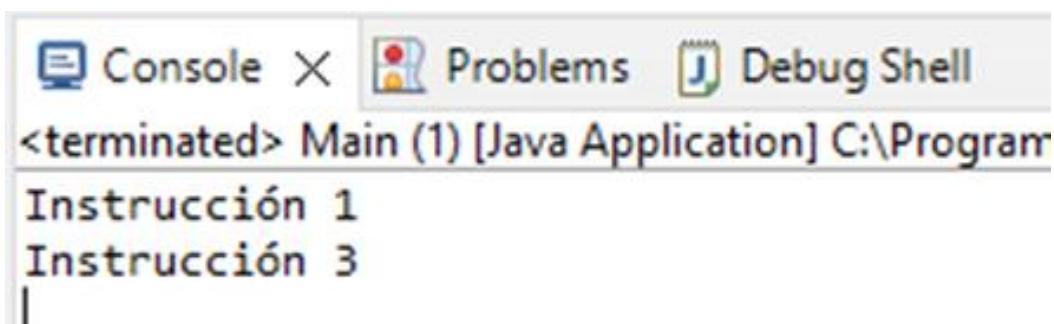


```
Console X Problems Debug Shell  
<terminated> Main (1) [Java Application] C:\Progra  
Instrucción 1  
Instrucción 2  
Instrucción 3  
Instrucción 4
```

1. Presentación

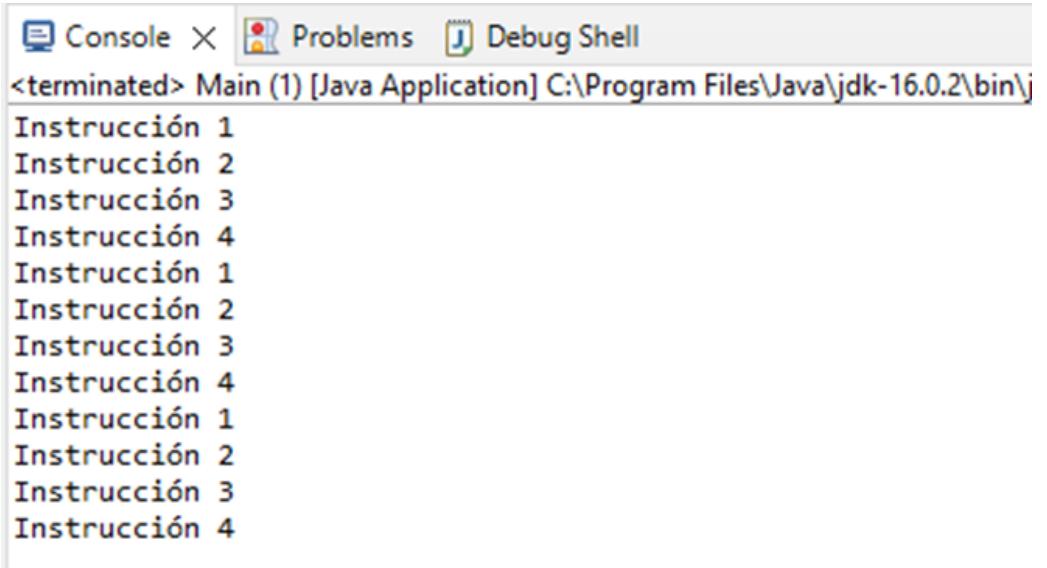
1.3 Tipos de estructuras de control de flujo

Estructura condicional



The screenshot shows the Eclipse IDE's Console view. The tabs at the top are 'Console' (selected), 'Problems', and 'Debug Shell'. The console output window displays the following text:
<terminated> Main (1) [Java Application] C:\Program
Instrucción 1
Instrucción 3
|

Estructuras repetitivas

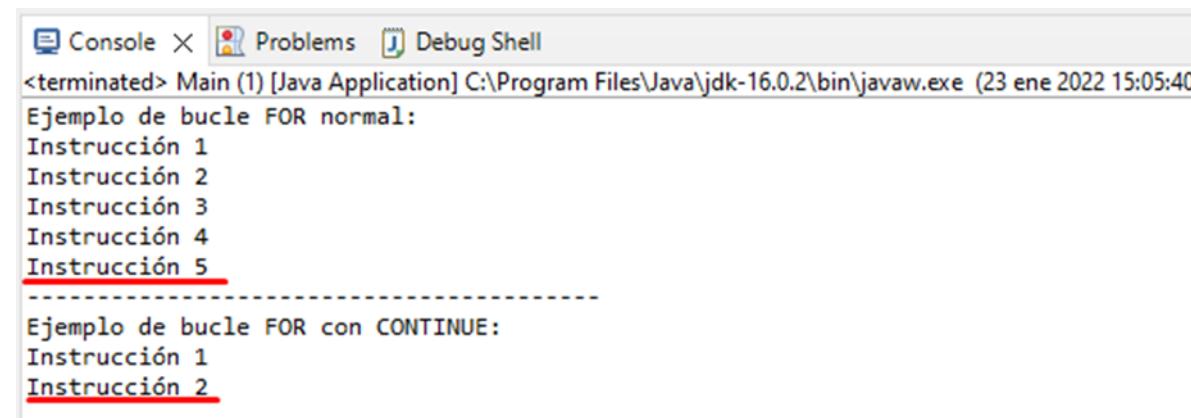


The screenshot shows the Eclipse IDE's Console view. The tabs at the top are 'Console' (selected), 'Problems', and 'Debug Shell'. The console output window displays the following text:
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\j
Instrucción 1
Instrucción 2
Instrucción 3
Instrucción 4
Instrucción 1
Instrucción 2
Instrucción 3
Instrucción 4
Instrucción 1
Instrucción 2
Instrucción 3
Instrucción 4

1. Presentación

1.3 Tipos de estructuras de control de flujo

Instrucciones de salto



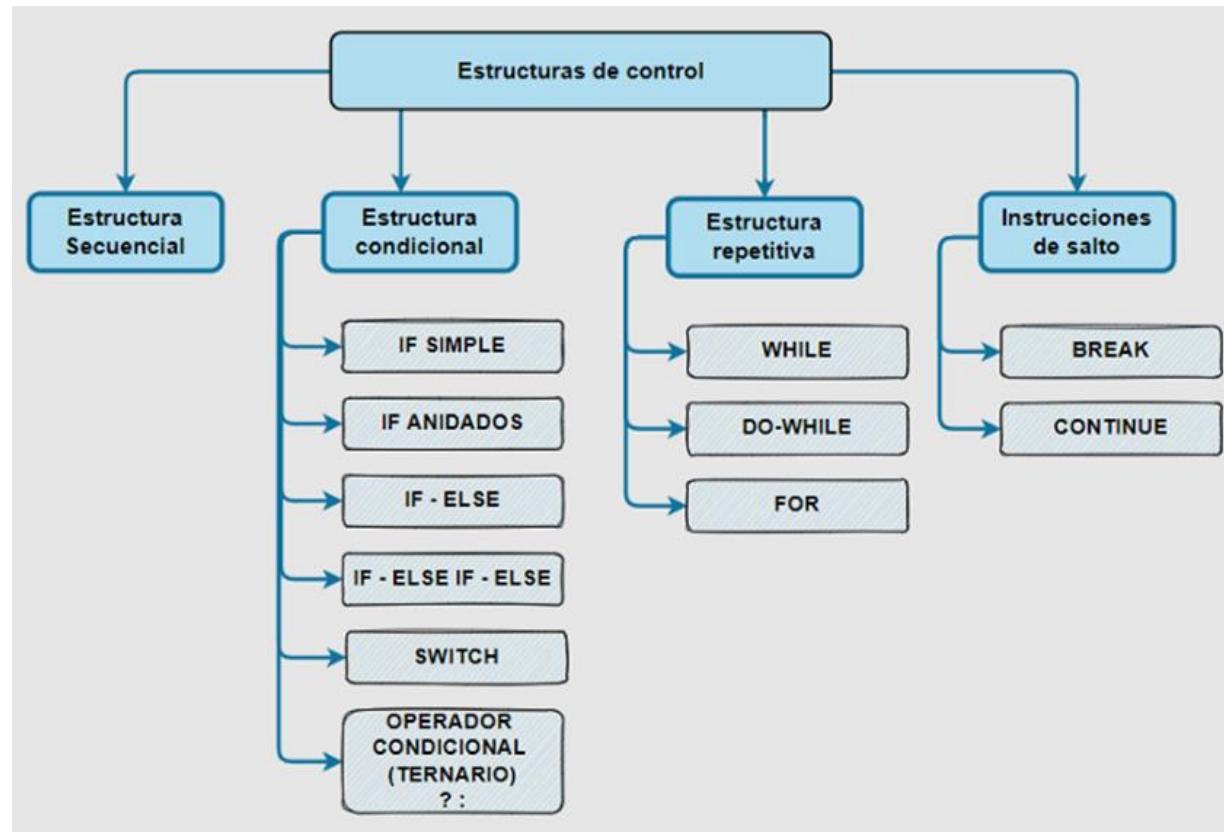
The screenshot shows a Java application running in a terminal window. The title bar indicates it's a Java Application named Main (1). The console output is as follows:

```
Console X Problems Debug Shell
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (23 ene 2022 15:05:40)
Ejemplo de bucle FOR normal:
Instrucción 1
Instrucción 2
Instrucción 3
Instrucción 4
Instrucción 5
-----
Ejemplo de bucle FOR con CONTINUE:
Instrucción 1
Instrucción 2
```

The word "Instrucción 5" is underlined in red, and "Instrucción 2" is underlined in red.

1. Presentación

1.4 Esquema de categorías y las distintas estructuras de control en Java

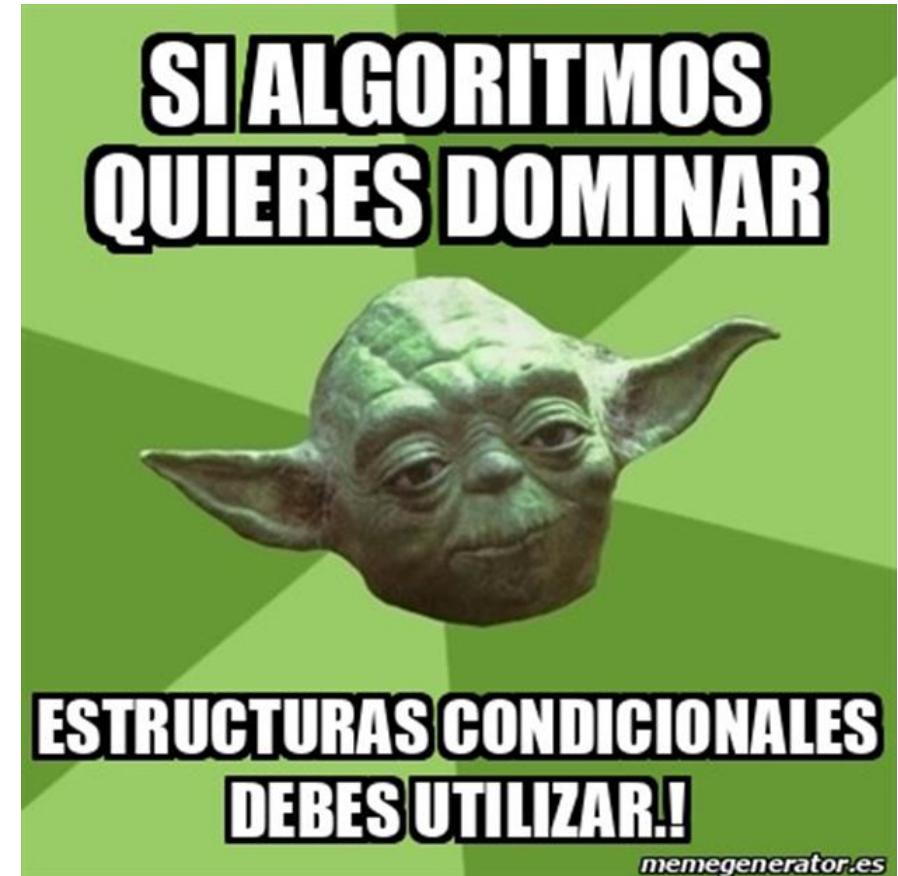


2. Estructuras condicionales/selección



2. Estructuras condicionales/selección

2.1 Introducción a las estructuras condiciones



2. Estructuras condicionales/selección

2.1 Introducción a las estructuras condiciones



2. Estructuras condicionales/selección

2.2 Profundizando en las instrucciones condicionales

Boolean



ONE
VALUE



CERO
VALUE

2. Estructuras condicionales/selección

2.2 Profundizando en las instrucciones condicionales

<https://jsbin.com/?html,output>

The screenshot shows a browser-based code editor and console interface. At the top, there's a navigation bar with tabs for File, Add library, HTML, CSS, JavaScript, Console, and Output. Below the tabs, the JavaScript tab is selected, indicated by a dropdown menu. The main area contains two lines of code:

```
JavaScript
console.log(Boolean(0)); // false
console.log(Boolean(1)); // true
```

To the right, under the Console tab, the output is displayed:

Console
false
true

A blue arrow points from the bottom right corner of the code area towards the bottom right corner of the output area.

2. Estructuras condicionales/selección

2.2 Profundizando en las instrucciones condicionales

Código 2.2_1

The screenshot shows a Java code editor window titled "Main.java". The code is as follows:

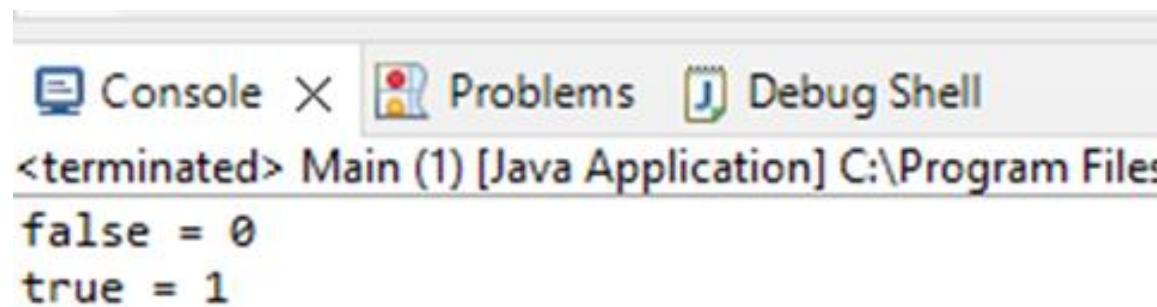
```
1
2
3 public class Main {
4
5     public static void main(String[] args) {
6         boolean b = false;
7         int i = (int)b;
8     }
9
10}
```

A red error marker is present on line 7, indicating a casting error. A tooltip box is displayed over the error, containing the message "Cannot cast from boolean to int" and the instruction "Press 'F2' for focus".

2. Estructuras condicionales/selección

2.2 Profundizando en las instrucciones condicionales

Código 2.2_2

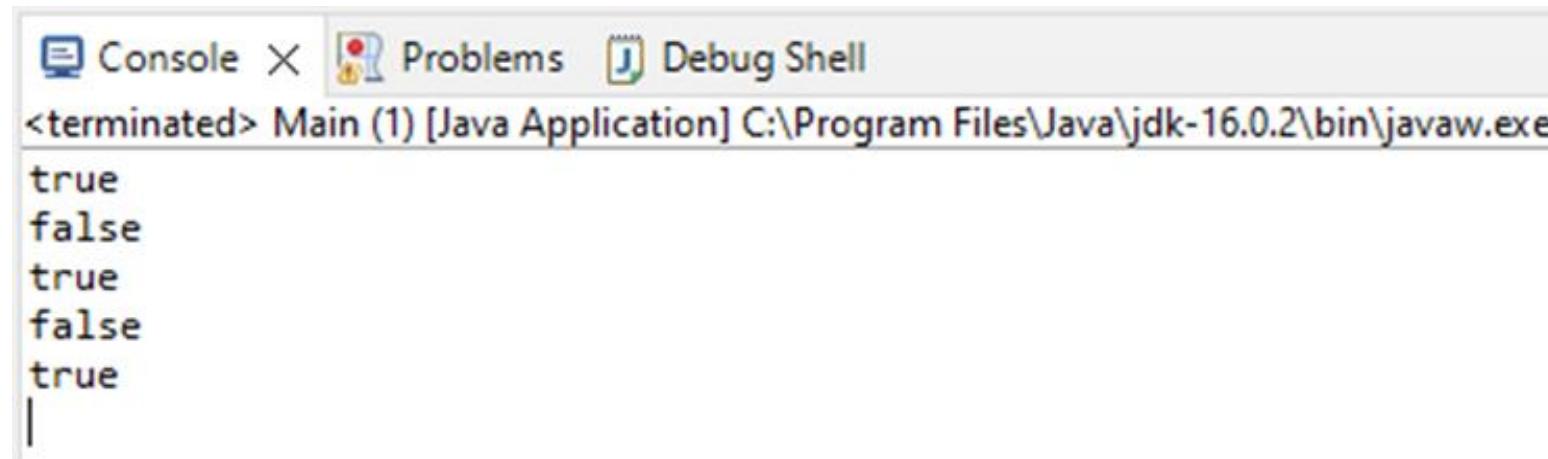


```
<terminated> Main (1) [Java Application] C:\Program Files
false = 0
true = 1
```

2. Estructuras condicionales/selección

2.3 ¿Cómo realizamos una operación condicional en Java?

Código 2.3_1



The screenshot shows a Java application running in an IDE. The console tab is active, displaying the following output:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
true
false
true
false
true
|
```

2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

The screenshot shows a Java code editor with the following code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int a = 1, b = 1, c = 3;
7         System.out.println(a == b); // En este caso, el resultado de la condición es TRUE, ya que A y B son
8                         // iguales.
9
10        boolean resultado = (a == c); // Variable de tipo boolean en la que almacenamos el resultado de comparar si A y C son iguales. (==).
11
12        System.out.println(resultado); // En este caso, el resultado de la condición es FALSE, ya que A y C no son iguales.
13
14        System.out.println((b != c)); // En este caso, el resultado de la condición es TRUE, ya que B no es igual (!=) a C
15
16        resultado = (1 < 0);
17        System.out.println(resultado); // Compara si 1 es menor que 0, en este caso retorna false ya que 1 es más grande que 0
18
19        resultado = (1 == a && 2 == 2); // Comparando dos condiciones
20        System.out.println(resultado); // Compara si 1 es igual a 1 Y a su vez si 2 es igual a 2. Como ambas son TRUE, retorna TRUE
21
22    }
23}
24
```

A red arrow points from the text "Hacemos doble click en esta zona para poner un break point y posteriormente pulsamos sobre el bichito para activar el debugger" to the line number 19 in the code.

A red box highlights the "Run" button in the toolbar, which is the second button from the left.

The code includes several comments in Spanish explaining the logic of each line.

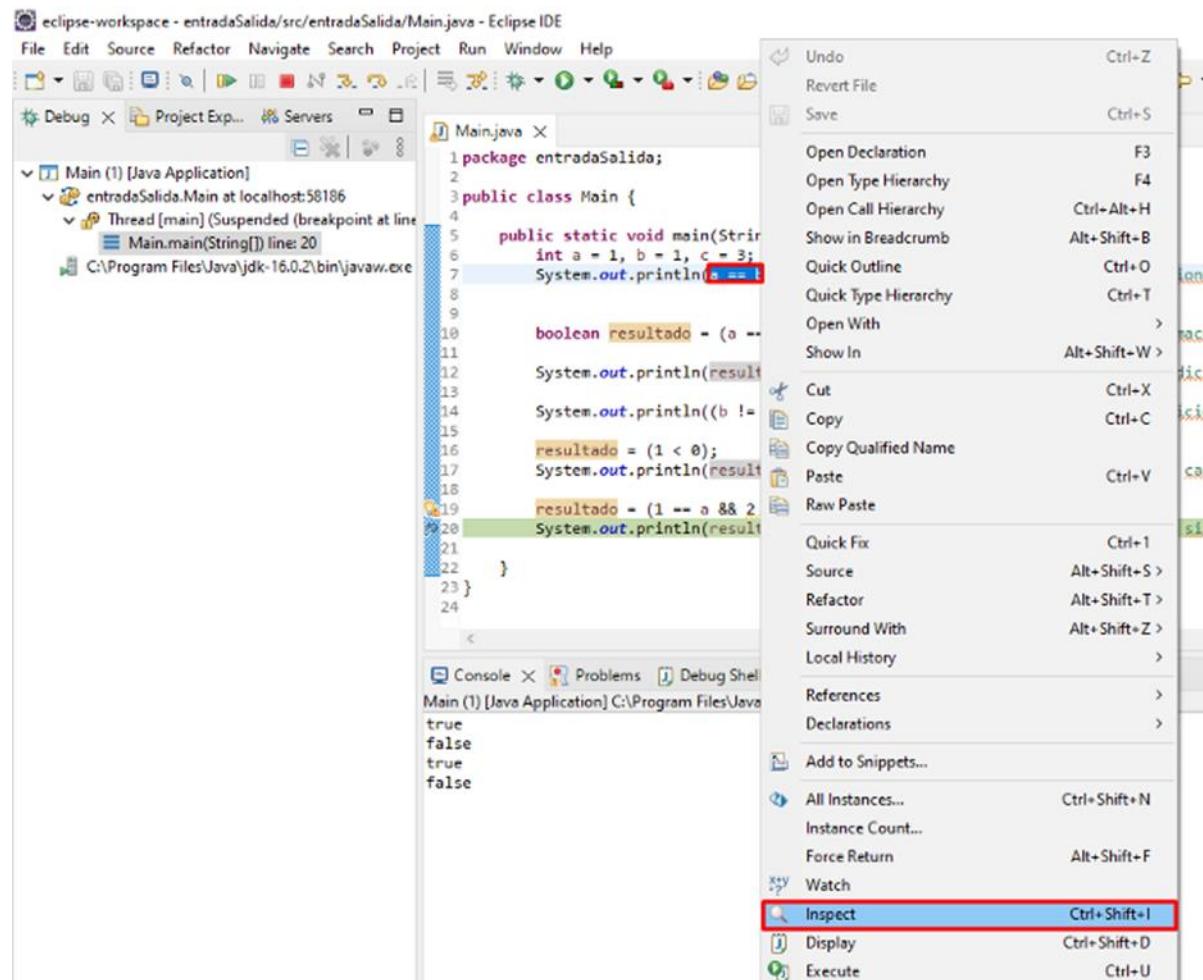
2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

```
eclipse-workspace - entradaSalida/src/entradaSalida/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug X Project Exp... Servers
Main.java X
Main (1) [Java Application]
entradSalida.Main at localhost:58186
Thread [main] (Suspended (breakpoint at line 20))
  Main.main(String[])
C:/Program Files/Java/jdk-16.0.2/bin/java.exe
Main.java:1: package entradaSalida;
^
Main.java:2: import java.util.*;
^
Main.java:3: public class Main {
^
Main.java:4:     public static void main(String[] args) {
^
Main.java:5:         int a = 1, b = 1, c = 3;
^
Main.java:6:         System.out.println(a == b); // En este caso, el resultado de la condición es TRUE, ya que A y B son
Main.java:7:             // iguales
^
Main.java:8: 
Main.java:9:         boolean resultado = (a == c); // Variable de tipo boolean en la que almacenamos el resultado de comparar si A y C son iguales (==).
^
Main.java:10:        System.out.println(resultado); // En este caso, el resultado de la condición es FALSE, ya que A y C no son iguales
^
Main.java:11:        System.out.println(b != c)); // En este caso, el resultado de la condición es TRUE, ya que B no es igual (!=) a C
^
Main.java:12:        resultado = (1 < 0);
^
Main.java:13:        System.out.println(resultado); // Compara si 1 es menor que 0, en este caso retorna false ya que 1 es más grande que 0
^
Main.java:14:        resultado = (1 == a && 2 == b); // Comparando dos condiciones
Main.java:15:        System.out.println(resultado); // Compara si 1 es igual a 1 Y si 2 es igual a 2. Como ambas son TRUE, retorna TRUE
^
Main.java:16:    }
^
Main.java:17: }
^
Main.java:18: }
```

2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

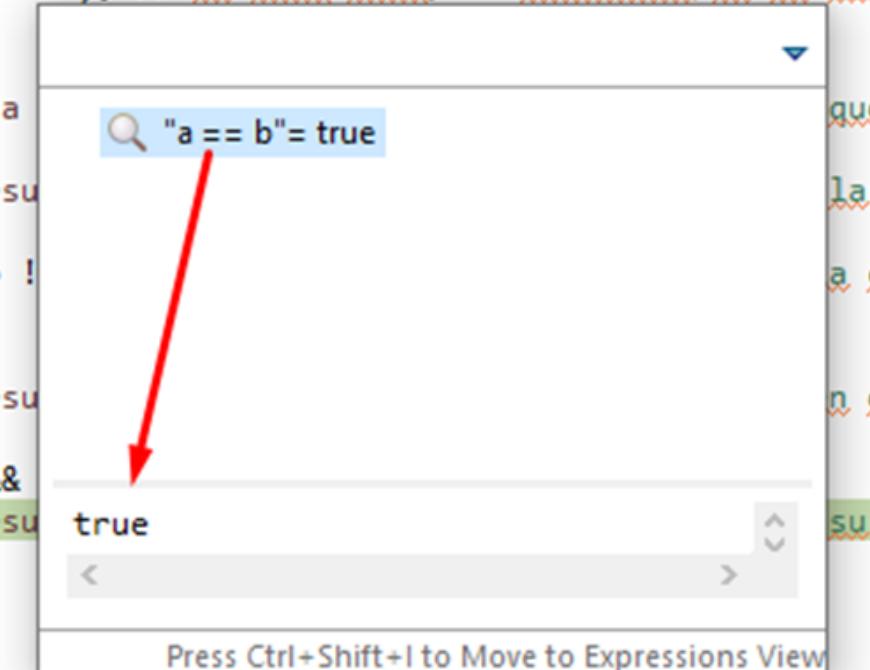


2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

```
public static void main(String[] args) {
    int a = 1, b = 1, c = 3;
    System.out.println(a == b); // En este caso, el resultado de la con
                                que
                                la
                                a c
                                n e
                                su
boolean resultado = (a
    System.out.println(resu
    System.out.println((b !
        resultado = (1 < 0);
        System.out.println(resu
        resultado = (1 == a &&
        System.out.println(resu
}

```



"a == b" = true

true

Press Ctrl+Shift+I to Move to Expressions View

2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - entradaSalida/src/entradaSalida/Main.java - Eclipse IDE
- Toolbar:** Standard Eclipse toolbar with various icons for file operations, search, and project management.
- Left Sidebar:** Project Explorer showing a single Java file named Main.java under a terminated Java application.
- Code Editor:** The Main.java file contains the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int a = 1, b = 1, c = 3;
7         System.out.println(a == b); // En este caso, el resultado de la condición es TRUE, ya que A y
8                         // son iguales
9
10        boolean resultado = (a == c); // Variable de tipo boolean en la que almacenamos el resultado de
11        System.out.println(resultado); // En este caso, el resultado de la condición es FALSE, ya que
12
13        System.out.println((b != c)); // En este caso, el resultado de la condición es TRUE, ya que B no
14                         // es igual a C
15        resultado = (1 < 0);
16        System.out.println(resultado); // Compara si 1 es menor que 0, en este caso retorna false ya que
17
18        resultado = (1 == a && 2 == 2); // Comparando dos condiciones
19        System.out.println(resultado); // Compara si 1 es igual a 1 Y si 2 es igual a 2. Como
20
21    }
22}
23}
24}
```
- Bottom Panel:** Shows the Console tab with the output of the program:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (23 ene 2022 20:56:28)
true
false
true
false
true
```

2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - entradaSalida/src/entradaSalida/Main.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbars:** Standard, Debug, Project Explorer, Servers, Task List, Problems, Outline, Properties, Status Bar.
- Left Sidebar:** Shows the project structure under "Main (1) [Java Application]" and the current thread "Thread [main] (Suspended)" at line 20.
- Central Area:** Displays the code for `Main.java`. The code includes:

```
package entradaSalida;
public class Main {
    public static void main(String[] args) {
        int a = 1, b = 1, c = 3;
        System.out.println(a == b); // En este caso el resultado es true porque a y b tienen el mismo valor
        boolean resultado = (a == c); // Varias operaciones de comparación
        System.out.println(resultado); // En este caso el resultado es false porque a no es igual a c
        System.out.println((b != c)); // En este caso el resultado es true porque b no es igual a c
        resultado = (1 < 0);
        System.out.println(resultado); // Como resultado es menor que 0, el resultado es false
        resultado = (1 == a && 2 == 2); // Como los resultados de las comparaciones son true, el resultado es true
        System.out.println(resultado); // Como resultado es true, el resultado es true
```
- Variables View (highlighted with a red box):** Shows the current values of variables:

Name	Value
args	String[0] (id=20)
a	1
b	1
c	3
resultado	true

2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

eclipse-workspace - entradaSalida/src/entradaSalida/Main.java - Eclipse IDE

Main.java X

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int a = 1, b = 1, c = 3;
7         System.out.println(a == b); // En este
8                                     // iguales
9
10        boolean resultado = (a == c); // Variat
11
12        System.out.println(resultado); // En ei
13
14        System.out.println((b != c)); // En est
15
16        resultado = (1 < 0);
17        System.out.println(resultado); // Compr
18
19        resultado = (1 == a && 2 == 2); // Compr
20        System.out.println(resultado); // Compr
21
22    }
23
24 }
```

Variables X Breakpoints

Name	Value
no method return value	
args	String[0] (id=20)
a	1
b	1
c	3
resultado	true

Console X Problems Debug Shell

```
Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (23 ene 2022 20:58:13)
true
false
true
false
```

2. Estructuras condicionales/selección

2.4 Aprendiendo a debuggear

```
eclipse-workspace - entradaSalida/src/entradaSalida/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
D X P S Main.java X
Main (1) [Java Application]
entréeSalida.Main at pc
Thread [main] [Suspended]
Main.main(String[])
C:\Program Files\Java\jdk
Main.java
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int a = 1, b = 1, c = 3;
7         System.out.println(a == b); // En este caso son iguales
8
9         boolean resultado = (a == c); // Variante de if
10        System.out.println(resultado); // En este caso es false
11
12        System.out.println((b != c)); // En este caso es true
13
14        resultado = (1 < 0);
15        System.out.println(resultado); // Imprime false
16
17        resultado = (1 == a && 2 == 2); // Combinación
18        System.out.println(resultado); // Combinación
19
20    }
21
22 }
23 }
```

Variables

Name	Value
no method return value	
args	String[0] (id=20)
a	1
b	1
c	3
resultado	true

Console

```
true
false
true
false
true
```

Console

```
<terminated> Main (1) [Java Application] C:\Prog
true
false
true
false
false
```

2. Estructuras condicionales/selección

2.5 Estructura condicional IF (Unidireccional)

The diagram illustrates the structure of a Java if statement. It shows a code editor window with the file 'Main.java'. The code defines a class 'Main' with a main method. Inside the main method, there is an if statement: 'if(condicion) { // Instrucciones }'. A green bracket on the left side of the if keyword indicates the start of the conditional structure. A blue box encloses the code within the if block, labeled 'define el tipo de estructura condicional sobre la que estamos trabajando. En este caso un IF (si)'. A red bracket above the if block indicates the condition being evaluated. A blue box to the right of the if block contains explanatory text in Spanish: 'nos retornará true o false. Solamente si nos devuelve true nos permitirá ejecutar el bloque de instrucciones que tenemos englobado entre las llaves {}'. Below this, another blue box contains the text: 'dentro de este bloque añadiremos las instrucciones que se ejecutarán cuando se cumpla la condición'.

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         boolean condicion = true;    nos retornará true o false. Solamente si nos devuelve
7         if(condicion) {            true nos permitirá ejecutar el bloque de instrucciones
8             // Instrucciones      que tenemos englobado entre las llaves {}
9         }
10    }
11
12 }
```

define el tipo de estructura condicional sobre la que estamos trabajando. En este caso un IF (si)

nos retornará true o false. Solamente si nos devuelve true nos permitirá ejecutar el bloque de instrucciones que tenemos englobado entre las llaves {}

dentro de este bloque añadiremos las instrucciones que se ejecutarán cuando se cumpla la condición

2. Estructuras condicionales/selección

2.5 Estructura condicional IF (Unidireccional)

Código 2.5_1

Código 2.5_2

2. Estructuras condicionales/selección

2.5 Estructura condicional IF (Unidireccional)

The screenshot shows a Java code editor and a terminal window. The code in Main.java is as follows:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int nota = 3;
7         System.out.println(nota < 5); // En este
8         // Solamente ejecutará las instrucciones
9         if(nota < 5) {
10             System.out.println("Suspendido");
11         }
12     }
13 }
14 }
```

The line `System.out.println(nota < 5);` is highlighted with a blue box and has a blue arrow pointing to the word `true` in the terminal. The line `System.out.println("Suspendido");` is highlighted with a red box and has a red arrow pointing to the text `Suspendido` in the terminal.

Console output:

```
<terminated> Main
true
Suspendido
```

2. Estructuras condicionales/selección

2.5 Estructura condicional IF (Unidireccional)

The screenshot shows an IDE interface with two main windows: 'Main.java' and 'Console'. In the 'Main.java' window, the code is as follows:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int nota = 5;
7         System.out.println(nota < 5); // En este caso, el resultado de
8         // Solamente ejecutara las instrucciones que están dentro del
9         if(nota < 5) {
10             System.out.println("Suspendido");
11         }
12     }
13 }
14 }
```

The line `System.out.println(nota < 5);` is highlighted with a blue box. A blue arrow points from this highlighted line to the output in the 'Console' window.

In the 'Console' window, the output is:

```
<terminated> Main
false
```

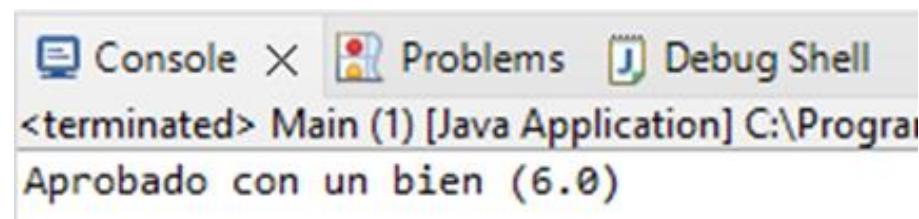
The screenshot shows an IDE interface with three tabs: 'Console', 'Problems', and 'Debug Shell'. The 'Console' tab is active and displays the output of a Java application named 'Main'.

```
Console X Problems Debug Shell
<terminated> Main (1) [Java Application] C:\Program
Aprobado
```

2. Estructuras condicionales/selección

2.6 Estructura condicional IF con estructuras anidadas

Código 2.6_1



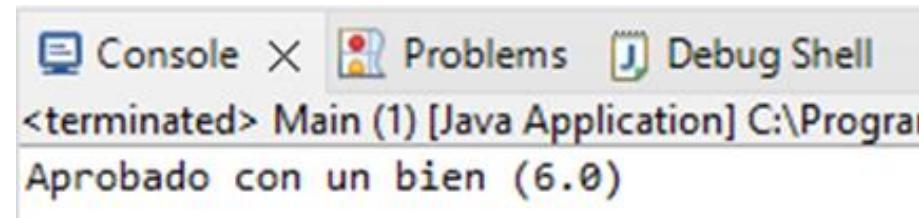
The screenshot shows a software interface with a toolbar at the top featuring 'Console' (selected), 'Problems', and 'Debug Shell'. Below the toolbar, the text '<terminated> Main (1) [Java Application] C:\Program' is displayed. In the main area, the output 'Aprobado con un bien (6.0)' is shown.

```
<terminated> Main (1) [Java Application] C:\Program  
Aprobado con un bien (6.0)
```

2. Estructuras condicionales/selección

2.6 Estructura condicional IF con estructuras anidadas

Código 2.6_2



The screenshot shows a software interface with a toolbar at the top featuring 'Console', 'Problems', and 'Debug Shell' tabs. The 'Console' tab is active, displaying the text: '<terminated> Main (1) [Java Application] C:\Program'. Below this, the output of the program is shown: 'Aprobado con un bien (6.0)'. The entire window has a light gray background.

```
<terminated> Main (1) [Java Application] C:\Program
Aprobado con un bien (6.0)
```

2. Estructuras condicionales/selección

2.6 Estructura condicional IF con estructuras anidadas

The screenshot shows a Java code editor with a file named Main.java. The code defines a class Main with a main method. Inside the main method, a variable nota is initialized to 10. The code then uses nested if statements to determine the grade based on the value of nota. The nested if statements are highlighted with red boxes. The output of the program is shown in the console tab at the bottom, where it prints "Aprobado con un matricula de honor (10.0)" and "Enhorabuena!".

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double nota = 10;
7
8         if(nota < 5) {
9             System.out.println("Suspendido");
10        }
11
12         if(nota >= 5) {
13             System.out.print("Aprobado"); // Se ejecutará siempre que sea aprobado
14             // EJEMPLO DE IFS ANIDADOS
15             if(nota >= 5 && nota < 6)
16                 System.out.println(" con un suficiente (" + nota + ")");
17             if(nota >= 6 && nota < 7)
18                 System.out.println(" con un bien (" + nota + ")");
19             if(nota >= 7 && nota < 9)
20                 System.out.println(" con un notable (" + nota + ")");
21             if(nota >= 9 && nota <10)
22                 System.out.println(" con un excelente (" + nota + ")");
23             if(nota == 10)
24                 System.out.println(" con un matricula de honor (" + nota + ")");
25                 System.out.println("Enhorabuena!");
26         }
27     }
28 }
29
```

Console X Problems Debug Shell
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (23 ene 2022 14:31:59 – 14:31:59)
Aprobado con un matricula de honor (10.0)
Enhorabuena!

2. Estructuras condicionales/selección

2.6 Estructura condicional IF con estructuras anidadas

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double nota = 6;
7
8         if(nota < 5) {
9             System.out.println("Suspendido");
10        }
11
12        if(nota >= 5) {
13            System.out.print("Aprobado"); // Se ejecutará siempre que sea aprobado
14            // EJEMPLO DE IFS ANIDADOS
15            if(nota >= 5 & nota < 6)
16                System.out.println(" con un suficiente (" + nota + ")");
17            if(nota >= 6 & nota < 7)
18                System.out.println(" con un bien (" + nota + ")");
19            if(nota >= 7 & nota < 9)
20                System.out.println(" con un notable (" + nota + ")");
21            if(nota >= 9 & nota < 10)
22                System.out.println(" con un excelente (" + nota + ")");
23            if(nota == 10)
24                System.out.println(" con un matricula de honor (" + nota + ")");
25            System.out.println("Enhorabuena!");
26        }
27    }
28}
```

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (23 ene 2022 14:31:12 - 14)
Aprobado con un bien (6.0)
Enhorabuena!
```

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double nota = 6;
7
8         if(nota < 5) {
9             System.out.println("Suspendido");
10        }
11
12        if(nota >= 5) {
13            System.out.print("Aprobado"); // Se ejecutará siempre que sea aprobado
14            // EJEMPLO DE IFS ANIDADOS
15            if(nota >= 5 & nota < 6)
16                System.out.println(" con un suficiente (" + nota + ")");
17            if(nota >= 6 & nota < 7)
18                System.out.println(" con un bien (" + nota + ")");
19            if(nota >= 7 & nota < 9)
20                System.out.println(" con un notable (" + nota + ")");
21            if(nota >= 9 & nota < 10)
22                System.out.println(" con un excelente (" + nota + ")");
23            if(nota == 10)
24                System.out.println(" con un matricula de honor (" + nota + ")");
25            System.out.println("Enhorabuena!");
26        }
27    }
28}
```

Console X Problems Debug Shell
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (23 ene 2022 14:33:58 – 14:33:58)
Aprobado con un bien (6.0)
Enhorabuena!

2. Estructuras condicionales/selección

2.7 Analizando el rendimiento de muchos IFs frente a algunas de sus alternativas

Código 2.7_1

Código 2.7_2

2. Estructuras condicionales/selección

2.7 Analizando el rendimiento de muchos IFs frente a algunas de sus alternativas

The screenshot shows a Java application running in a terminal window. The terminal tabs at the top are 'Console', 'Problems', and 'Debug Shell'. The current tab is 'Console'. The output shows the application has terminated. The text output is:

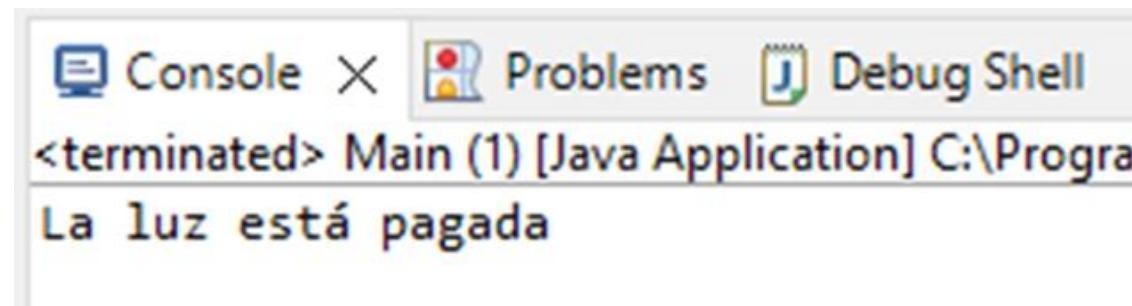
```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Tiempo total de ejecución de un triple IF es: 8542ms
Tiempo total de ejecución de un IF + IF ELSE es: 8306ms
Tiempo total de ejecución de un IF IFELSE ELSE es: 8298ms
```

The last three lines of text are highlighted with red boxes.

2. Estructuras condicionales/selección

2.8 Estructura condicional IF-ELSE (Bidireccional)

Código 2.8_1



The screenshot shows a software interface with a toolbar at the top featuring 'Console' (selected), 'Problems', and 'Debug Shell'. Below the toolbar, the text '<terminated> Main (1) [Java Application] C:\Progra' is displayed, followed by the output 'La luz está pagada'.

```
Console × Problems Debug Shell
<terminated> Main (1) [Java Application] C:\Progra
La luz está pagada
```

2. Estructuras condicionales/selección

2.9 La ley del “70% 30%”, como organizar las estructuras condicionales

Código 2.9_1

The screenshot shows a Java application window titled "Console". The title bar includes standard window controls (minimize, maximize, close) and the title "Console X". The main area displays the output of a Java application. The output starts with "<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe". Below this, two lines of text are displayed: "Cuando se ejecuta el primer boque del if-else tarda 8322 ms." and "Cuando se ejecuta el segundo boque del if-else tarda 8606 ms.". A red rectangular box highlights the number "8322 ms." in the first line. Another red rectangular box highlights the number "8606 ms." in the second line. At the bottom, the text "Por tanto, la diferencia entre ejecutar el primer bloque y el segundo 5000 veces es 284 ms." is shown, with the number "284 ms." highlighted by a red rectangular box.

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Cuando se ejecuta el primer boque del if-else tarda 8322 ms.
Cuando se ejecuta el segundo boque del if-else tarda 8606 ms.

Por tanto, la diferencia entre ejecutar el primer bloque y el segundo 5000 veces es 284 ms.
```

2. Estructuras condicionales/selección

2.10 Estructura condicional IF-ELSE IF- ELSE (Tridireccional o más...)

Código 2.10_1

The screenshot shows a Java code editor and a terminal window. The code in Main.java is as follows:

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         double number = 55;
6
7         if (number > 0) {
8             System.out.println("¡Es positivo!");
9         } else if (number < 0) {
10            System.out.println("¡Es negativo!");
11        } else {
12            System.out.println("¡Es cero, na de ná!");
13        }
14    }
15}
```

The line `System.out.println("¡Es positivo!");` is highlighted with a red box and has a red arrow pointing from it to the terminal window. The terminal window shows the output: `<terminated> Main (1) [J] ¡Es positivo!`.

2. Estructuras condicionales/selección

2.10 Estructura condicional IF-ELSE IF- ELSE (Tridireccional o más...)

Main.java

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         double number = -25;
6
7         if (number > 0) {
8             System.out.println("¡Es positivo!");
9         } else if (number < 0) {
10            System.out.println("¡Es negativo!");
11        } else {
12            System.out.println("¡Es cero, na de ná!");
13        }
14    }
15 }
```

Console

```
<terminated> Main (1)
¡Es negativo!
```

Main.java

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         double number = 0;
6
7         if (number > 0) {
8             System.out.println("¡Es positivo!");
9         } else if (number < 0) {
10            System.out.println("¡Es negativo!");
11        } else {
12            System.out.println("¡Es cero, na de ná!");
13        }
14    }
15 }
```

Console

```
<terminated> Main (1) [Java]
¡Es cero, na de ná!
```

2. Estructuras condicionales/selección

2.10 Estructura condicional IF-ELSE IF- ELSE (Tridireccional o más...)

Código 2.10_2

The screenshot shows an IDE interface with two main panes. On the left is the code editor for a file named Main.java, which contains the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         int cp = 8241; // CP de Manresa
6
7         if (cp >= 8000 && cp <= 8080) { // 08000 - 08080 = Barcelona
8             System.out.println("¡Código postal de Barcelona!");
9         } else if ((cp >= 8171 && cp <= 8174) || (cp >= 8195 && cp <= 8198))
10             System.out.println("¡Código postal de Sant Cugat del Vallés!");
11         } else if (cp >= 8240 && cp <= 8248) { //08240 - 08248 = Manresa
12             System.out.println("¡Código postal de Manresa!");
13         } else {
14             System.out.println("No conozco este código postal :(");
15         }
16     }
17 }
```

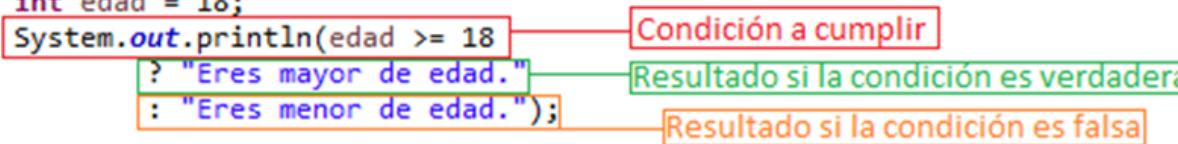
The code uses a series of nested if-else statements to check the value of the variable cp against different ranges. The range cp = 8241 is explicitly defined. The ranges for Barcelona (8000-8080), Sant Cugat del Vallés (8171-8174 and 8195-8198), and Manresa (8240-8248) are determined by the logic in the if-else statements. The output pane on the right shows the result of running the program with cp set to 8241, which is "¡Código postal de Manresa!", highlighted with a red box.

2. Estructuras condicionales/selección

2.11 Estructura condicional operador ternario/condicional (“bidireccional” en principio)

```
public class OperadorTernario {  
    public static void main(String[] args) {  
        int edad = 18;  
        System.out.println(edad >= 18 ? "Eres mayor de edad." : "Eres menor de edad.");  
    }  
}
```

www.javadesde0.com



The diagram illustrates the flow of the ternary operator. It starts with the condition `edad >= 18`, which is highlighted in a red box and labeled "Condición a cumplir". A green arrow points from this box to the expression `? "Eres mayor de edad."`, which is highlighted in a green box and labeled "Resultado si la condición es verdadera". Another green arrow points from this expression to the final result. A third green arrow points from the expression `? "Eres mayor de edad."` to the expression `: "Eres menor de edad."`, which is highlighted in a blue box and labeled "Resultado si la condición es falsa".

2. Estructuras condicionales/selección

2.11 Estructura condicional operador ternario/condicional (“bidireccional” en principio)

Código 2.11_1

```
Main.java X
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int edad = 18;
7         // Ejemplo de operador ternario ?: directamente desde un System.out.println();
8         System.out.println(
9             edad >= 18 // Condición
10            ? "Eres mayor de edad." // Instrucción que se ejecuta cuando es true
11            : "Eres menor de edad." // Instrucción que se ejecuta cuando es false
12        );
13
14         // Ejemplo de operador ternario ?: almacenado en un boolean y posteriormente imprimido desde System.out.println()
15         boolean esMayordeEdadBoolean = (edad >= 18) ? true: false;
16         System.out.println(esMayordeEdadBoolean);
17
18         // Ejemplo de operador ternario ?: almacenado sobre un String y posteriormente imprimido de System.out.println()
19         String esMayordeEdadString = (edad >= 18) ? "Eres mayor de edad." : "Eres menor de edad.";
20         System.out.println(esMayordeEdadString);
21     }
22 }
```

Console X
<terminated> Main (1) [Java]
Eres mayor de edad.
true
Eres mayor de edad.

2. Estructuras condicionales/selección

2.11 Estructura condicional operador ternario/condicional (“bidireccional” en principio)



2. Estructuras condicionales/selección

2.12 Desmontando el mito que el operador condicional solo puede ser bidireccionalidad

Código 2.12_1

```
Main.java X
entraSalida > src > entraSalida > Main > main(String[]): void
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int edad = 15;
7         System.out.println(edad>65?"No puedes trabajar!":edad>18?"A Trabajar!":"¡Al cole!");
8     }
9 }
10
```

Console X

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (25 ene 2022 22:46:45 – 22:46:45)
¡Al cole!
```

2. Estructuras condicionales/selección

2.13 Operador condicional SWITCH

Código 2.13_1

2. Estructuras condicionales/selección

2.13 Operador condicional SWITCH

The screenshot shows an IDE interface with two tabs: 'Main.java X' and 'Console X'. The Main.java tab displays a Java program that prints the day of the week and its type ('laborable' or 'festivo'). The Console tab shows the output for day 1 ('Lunes'), which is highlighted with a red box. A red arrow points from the highlighted code in the editor to the corresponding output line in the console.

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         int day = 1; // Dia de la semana
6         String dayType = "";
7
8         switch (day)
9         {
10             case 1:
11                 System.out.println("Lunes");
12                 dayType = "laborable";
13                 break;
14             case 2:
15                 System.out.println("Martes");
16                 dayType = "laborable";
17                 break;
18             case 3:
19                 System.out.println("Miércoles");
20                 dayType = "laborable";
21                 break;
22             case 4:
23                 System.out.println("Jueves");
24                 dayType = "laborable";
25                 break;
26             case 5:
27                 System.out.println("Viernes");
28                 dayType = "laborable";
29                 break;
30             case 6:
31                 System.out.println("Sábado");
32                 dayType = "festivo";
33                 break;
34             case 7:
35                 System.out.println("Domingo");
36                 dayType = "festivo";
37                 break;
38
39             default:
40                 System.out.println("Error! Día invalido");
41                 break;
42 }
```

Console Output:

```
Lunes
1 es un laborable
```

2. Estructuras condicionales/selección

2.13 Operador condicional SWITCH

Código 2.13_2

2. Estructuras condicionales/selección

2.13 Operador condicional SWITCH

The screenshot shows a Java code editor with a file named Main.java. The code defines a class Main with a main method. It declares an integer variable day and initializes it to 3. A switch statement is used to determine the day of the week based on the value of day. The case blocks for days 1 through 6 print the day name and set dayType to "laborable". The case block for day 7 prints the day name and sets dayType to "festivo". A default case prints an error message. The code also includes a comment explaining the purpose of the System.out.println statements. When run, the program outputs the days from Wednesday to Sunday, followed by an error message indicating that day 3 is a holiday.

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         int day = 3; // Día de la semana
6         String dayType = "";
7
8         switch (day)
9         {
10             case 1:
11                 System.out.println("Lunes");
12                 dayType = "laborable";
13             case 2:
14                 System.out.println("Martes");
15                 dayType = "laborable";
16             case 3:
17                 System.out.println("Miércoles");
18                 dayType = "laborable";
19             case 4:
20                 System.out.println("Jueves");
21                 dayType = "laborable";
22             case 5:
23                 System.out.println("Viernes");
24                 dayType = "laborable";
25             case 6:
26                 System.out.println("Sábado");
27                 dayType = "festivo";
28             case 7:
29                 System.out.println("Domingo");
30                 dayType = "festivo";
31
32             default:
33                 System.out.println("Error! Día invalido");
34         }
35
36         // Mensaje que imprime el num del día y si es laborable
37         System.out.println(day +" es un "+ dayType);
38     }
39 }
```

Miércoles
Jueves
Viernes
Sábado
Domingo
Error! Día invalido
3 es un festivo

2. Estructuras condicionales/selección

2.14 Analizando el funcionamiento de la condición en SWITCH

The screenshot shows a Java code editor window titled "Main.java X". The code is as follows:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         boolean interruptor = true;
7
8         switch (interruptor) {
9             case true:
10        }
11    }
12}
```

A tooltip is displayed over the line "switch (interruptor) {" with the message: "Cannot switch on a value of type boolean. Only convertible int values, strings or enum variables are permitted".

2. Estructuras condicionales/selección

2.14 Analizando el funcionamiento de la condición en SWITCH

The diagram shows a Java code editor with the file 'Main.java' open. The code defines a class 'Main' with a main method that prints different messages based on the value of the variable 'num'. A blue arrow points from the 'switch' keyword to the text 'variable o expresión entera'. Red arrows point from each 'case' label to the text 'identificadores'.

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int num = 1;
7
8         switch (num) → variable o expresión entera
9         {
10             case 1: → identificadores
11                 System.out.println("Uno");
12                 break;
13             case 2: → identificadores
14                 System.out.println("Dos");
15                 break;
16             case 3: → identificadores
17                 System.out.println("Otro número");
18                 break;
19         }
20     }
21 }
```

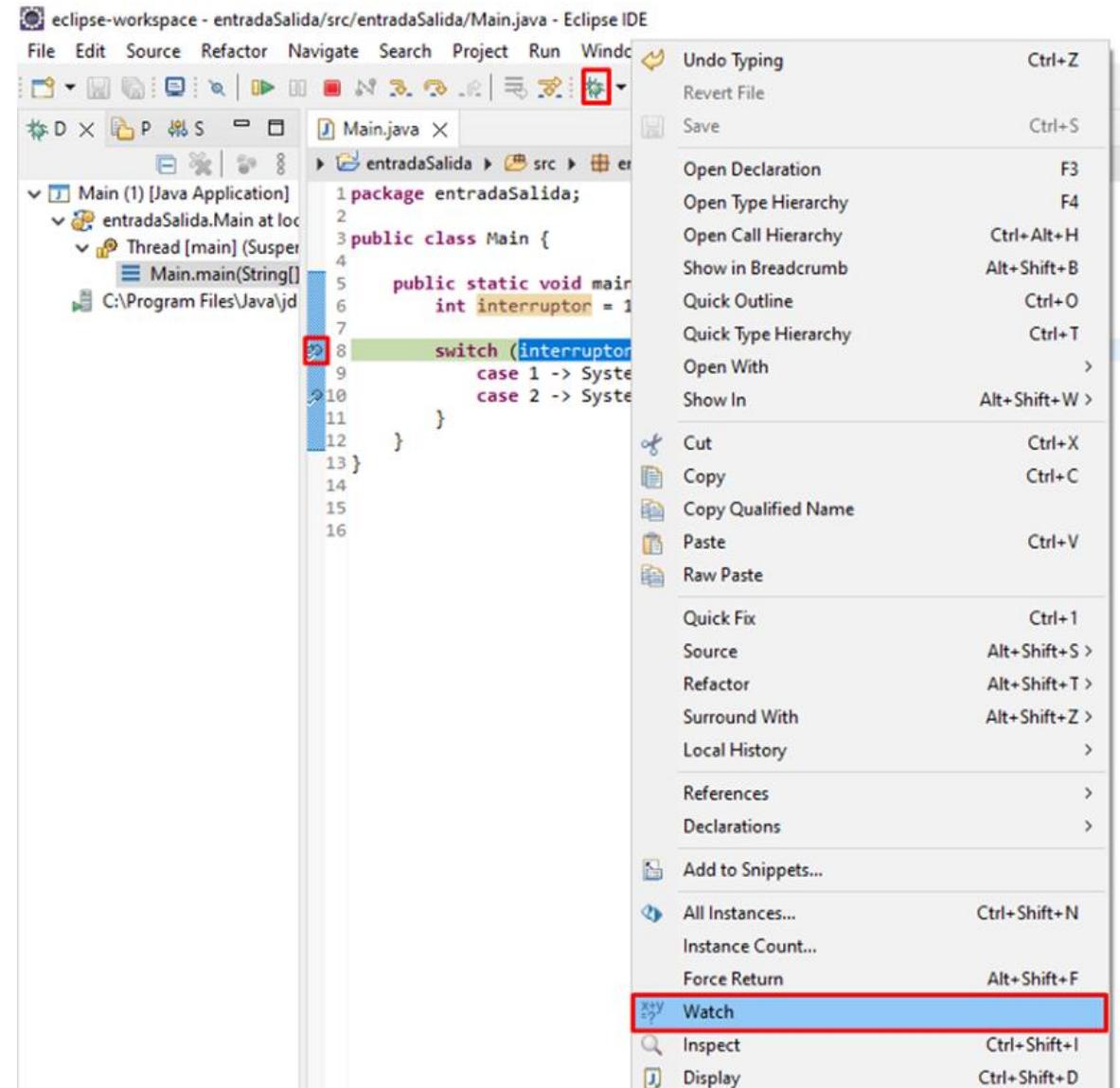
2. Estructuras condicionales/selección

2.14 Analizando el funcionamiento de la condición en SWITCH

Código 2.14_1

2. Estructuras condicionales/selección

2.14 Analizando el funcionamiento de la condición en SWITCH



2. Estructuras condicionales/selección

2.14 Analizando el funcionamiento de la condición en SWITCH

The screenshot shows a Java code editor and an attached Expressions tool window.

Main.java:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int interruptor = 1;
7
8         switch (interruptor) {
9             case 1 -> System.out.println("a");
10            case 2 -> System.out.println("b");
11        }
12    }
13}
```

Expressions:

Name	Value
interruptor	1
interruptor==1	true
interruptor==2	false

2. Estructuras condicionales/selección

2.14 Analizando el funcionamiento de la condición en SWITCH

Código 2.14_2

The screenshot shows a Java development environment with two tabs: 'Main.java X' and 'Console X'. The code in 'Main.java' is as follows:

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         char op = '*'; // Dia de la semana
6
7         switch (op)
8         {
9             case '+':
10                System.out.println("Suma");
11                //break;
12            case '-':
13                System.out.println("Resta");
14                //break;
15            default:
16                System.out.println("Error!");
17                // break;
18            case '/':
19                System.out.println("División");
20                // break;
21        }
22    }
23 }
```

The 'Console' tab shows the output: '<terminated> Main | Error! División'. A red box highlights the 'default' case block, and another red box highlights the output 'Error! División' in the console, with a red arrow pointing from the highlighted code to the highlighted output.

2. Estructuras condicionales/selección

2.14 Analizando el funcionamiento de la condición en SWITCH

The screenshot shows a Java development environment with two windows:

- Main.java**: The code is as follows:

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         char op = '*'; // Dia de la semana
6
7         switch (op)
8         {
9             case '+':
10                System.out.println("Suma");
11                break;
12             case '-':
13                System.out.println("Resta");
14                break;
15             default:
16                 System.out.println("Error!");
17                 break;
18             case '/':
19                 System.out.println("División");
20                 break;
21         }
22     }
23 }
```

- Console**: The output is:

```
<terminated> Main (
    Error!
```

Annotations in the code editor highlight the assignment to `char op = '*'` and the `default` case in the switch statement. A red arrow points from the `default` case to the word `Error!` in the console output.

2. Estructuras condicionales/selección

2.15 Ejemplo de SWITCH con cascada de CASES

Código 2.15_1

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         int day = 1; // Dia de la semana
6         String dayType = "";
7
8         switch (day)
9         {
10             // Ejemplo de case con multiples opciones
11             case 1:
12             case 2:
13             case 3:
14             case 4:
15             case 5:
16                 dayType = "laborable";
17                 break;
18             case 6:
19             case 7:
20
21                 dayType = "festivo";
22                 break;
23
24             default:
25                 System.out.println("Error! Día invalido");
26                 break;
27         }
28
29         System.out.println(dayType);
30     }
31 }
```

The screenshot shows an IDE interface with two main windows: 'Main.java X' and 'Console X'. The code in 'Main.java' demonstrates a switch statement with multiple cases (1-5) leading to the same value ('laborable'). The 'Console' window shows the output 'laborable'.

2. Estructuras condicionales/selección

2.16 Estructura de control SWITCH mejorada (disponible a partir de la versión 14 de Java)

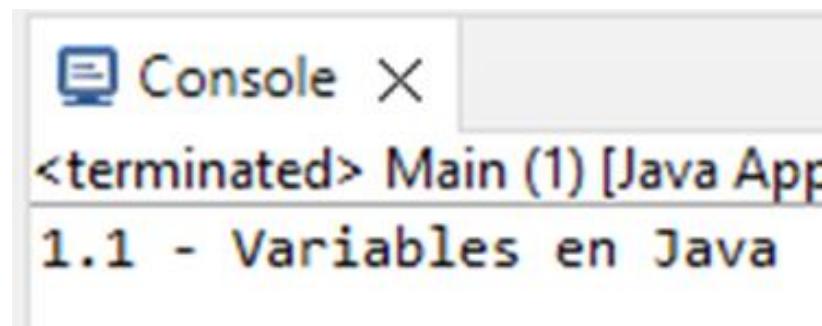
Código 2.16_1

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         //Solo compatible con la versión 14 de JDK en adelante
7         int day = 1;
8
9         switch (day) {
10             case 1, 2, 3, 4, 5:
11                 System.out.println("Laborable"); break;
12             case 6, 7:
13                 System.out.println("Festivo"); break;
14             default:
15                 System.err.println("¡No es un día de la semana!");
16         }
17
18         String dayType = "";
19         day = 6;
20         switch (day) {
21             case 1, 2, 3, 4, 5 -> dayType = "Laborable";
22             case 6, 7 -> dayType = "Festivo";
23             default -> System.err.println("¡No es un día de la semana!");
24         }
25         System.out.println(dayType);
26     }
27 }
```

2. Estructuras condicionales/selección

2.17 SWITCH dentro de SWITCH

Código 2.17_1



The image shows a screenshot of a Java application's console window. The title bar says "Console X". The main area displays the following text:
<terminated> Main (1) [Java App]
1.1 - Variables en Java

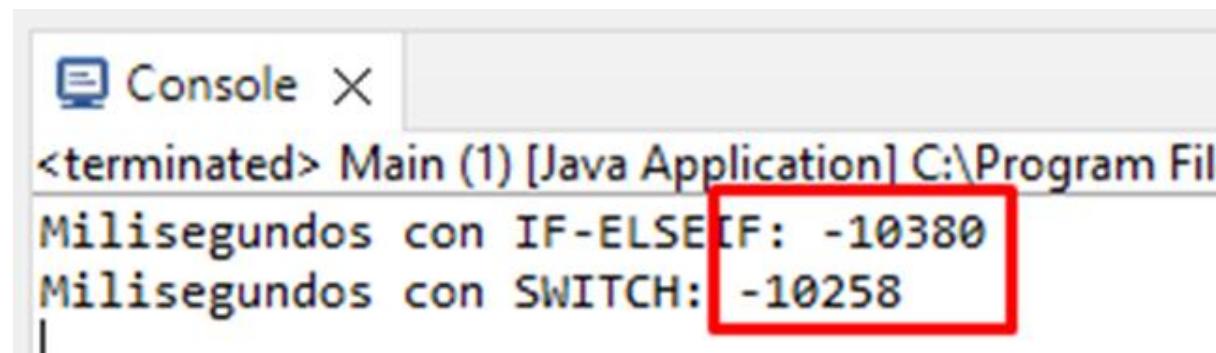
2. Estructuras condicionales/selección

2.18 ¿Cuándo usar un SWITCH frente a las otras alternativas (IF, IF-ELSE, IF-ELSE IF-ELSE o TERNARY OPERATOR)?

2. Estructuras condicionales/selección

2.19 Enfrentando a IF-ELSE con SWITCH para analizar su performance

Código 2.19_1



The screenshot shows a Java application window with a single tab labeled "Console". The console output is as follows:

```
Console X
<terminated> Main (1) [Java Application] C:\Program Fil
Milisegundos con IF-ELSE: -10380
Milisegundos con SWITCH: -10258
```

The line "Milisegundos con SWITCH: -10258" is highlighted with a red rectangular box.

2. Estructuras condicionales/selección

2.20 Ejercicios de estructuras condicionales

EJERCICIO 1: A PARTIR DEL SIGUIENTE CÓDIGO QUE SIMULA EL LANZAMIENTO DE UNA MONEDA AL AIRE:

```
package entradaSalida;

public class Main {

    public static void main(String[] args) {
        int num = (int) Math.floor(Math.random()
* 2); // Devuelve 1 o 0 aleatoriamente
        System.out.println(num);
    }
}
```

2. Estructuras condicionales/selección

2.20 Ejercicios de estructuras condicionales

REALIZA UN PROGRAMA QUE MUESTRE:

- CRUZ CUANDO EL VALOR SEA 0 Y CARA CUANDO EL VALOR SEA 1.
- PRIMERAMENTE, REALIZA EL EJERCICIO SOLAMENTE CON IFS (UNIDIRECCIONALES)
- LUEGO, REALIZA EL EJERCICIO SOLAMENTE CON LA ESTRUCTURA CONDICIONAL IF-ELSE
- POSTERIORMENTE, REALIZA EL EJERCICIO CON LA ESTRUCTURA CONDICIONAL DEL OPERADOR TERNARIO

2. Estructuras condicionales/selección

2.20 Ejercicios de estructuras condicionales

EJERCICIO 2: A PARTIR DE VALOR DE UNA VARIABLE EN LA QUE ASIGNAMOS UNA TEMPERATURA MEDIA DE UN PAÍS. REALIZA UN PROGRAMA QUE MUESTRE LAS SIGUIENTES TEMPERATURAS:

Temperatura	Categoría del clima	Tipo de clima
menor de 10 °C	Climas fríos	Polar - Cuando es menor que 5
		Alta montaña - Cuando es mayor o igual que 5 pero menor que 10
Igual o mayor que 10 °C y menor que 20 °C	Climas templados	Oceánico - Cuando es mayor o igual que 10 pero menor que 13,5
		Mediterráneo - Cuando es mayor o igual que 13,5 pero menor que 16,5
	Climas cálidos	Continental - Cuando es mayor o igual que 16,5 pero menor que 20
Igual o mayor que 20 °C		Ecuatorial - Cuando es mayor o igual que 20 pero menor que 23,5
		Tropical - Cuando es mayor o igual que 23,5 pero menor que 26,5
		Desértico - Cuando es mayor o igual a 26,5

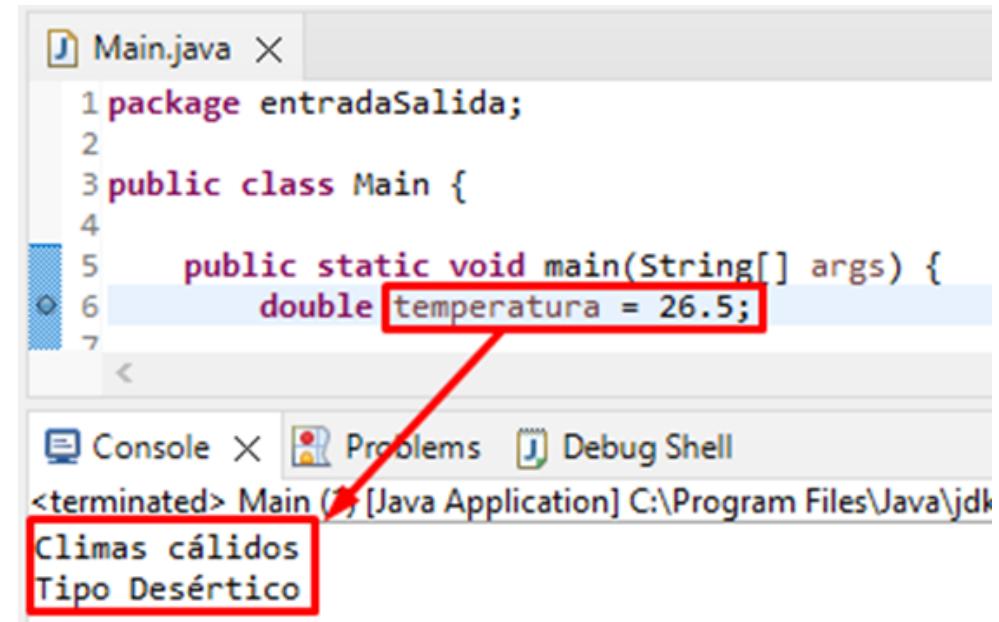
2. Estructuras condicionales/selección

2.20 Ejercicios de estructuras condicionales

EJERCICIO 2:

- PRIMERAMENTE, REALIZA EL EJERCICIO SOLAMENTE CON IFS ANIDADOS.
- POSTERIORMENTE, REALIZA EL EJERCICIO CON IF-ELSE IF-ELSE

EL RESULTADO CUANDO PONGAMOS UNA TEMPERATURA DE 26.5 GRADO SERÁ EL SIGUIENTE:



The screenshot shows an IDE interface with two main panes. The top pane is a code editor for a file named 'Main.java' containing the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double temperatura = 26.5;
7     }
}
```

The variable declaration 'double temperatura = 26.5;' is highlighted with a red box. A red arrow points from this highlighted code down to the console output in the bottom pane. The bottom pane is a 'Console X' tab showing the output of the program:

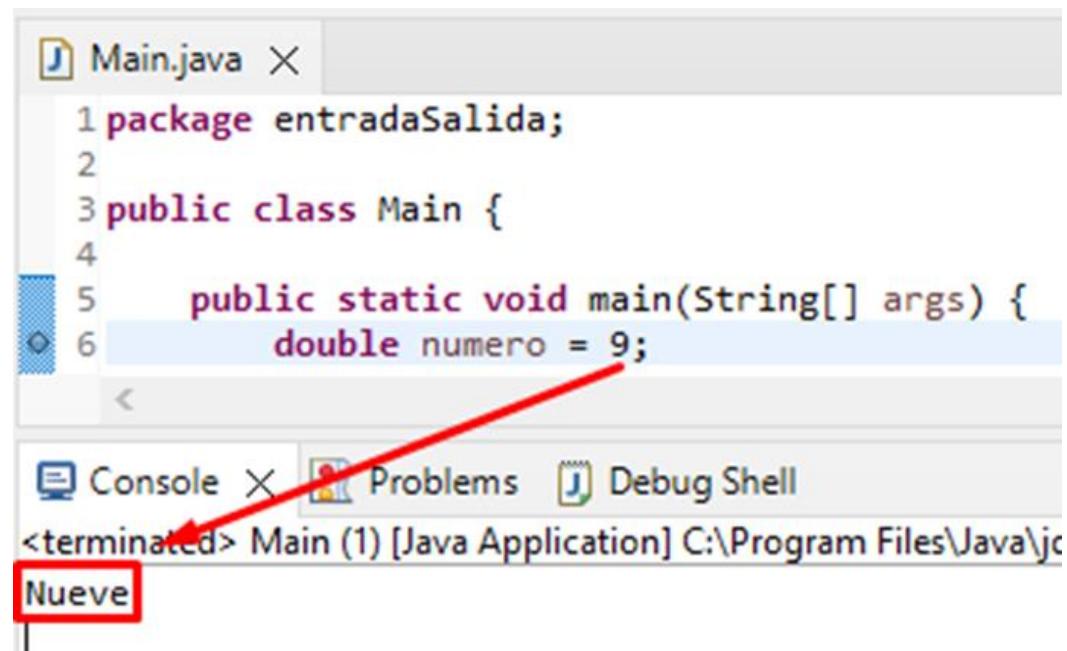
```
<terminated> Main () [Java Application] C:\Program Files\Java\jdk
Climas cálidos
Tipo Desértico
```

The lines 'Climas cálidos' and 'Tipo Desértico' are also highlighted with a red box.

2. Estructuras condicionales/selección

2.20 Ejercicios de estructuras condicionales

EJERCICIO 3: CREA UN PROGRAMA CON (SWITCH CASE) QUE CUANDO INTRODUZCAMOS EN UNA VARIABLE LOS VALORES NÚMERICOS DEL 1 AL 9 LO IMPRIMIRÁ EN TEXTO. POR EJEMPLO, SI INTRODUCCIMOS UN 9 LO MOSTRARÁ DE LA SIGUIENTE MANERA:



The screenshot shows a Java development environment. In the top panel, there is a code editor titled "Main.java" containing the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         double numero = 9;
```

A red arrow points from the word "numero" in line 6 to the output window below. In the bottom panel, there is a terminal window titled "Console" showing the output of the program:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jc
Nueve
```

The word "Nueve" is highlighted with a red box.

2. Estructuras condicionales/selección

2.20 Ejercicios de estructuras condicionales

EJERCICIO 4: REALIZA EL SIGUIENTE IF-ELSE IF-ELSE UTILIZANDO EL OPERADOR TERNARIO.

```
package entradaSalida;

public class Main {
    public static void main(String[] args) {
        double number = 55;

        if (number > 0) {
            System.out.println("¡Es positivo!");
        } else if (number < 0) {
            System.out.println("¡Es negativo!");
        } else {
            System.out.println("¡Es cero, na de ná!");
        }
    }
}
```

3. Estructuras repetitivas/iterativas

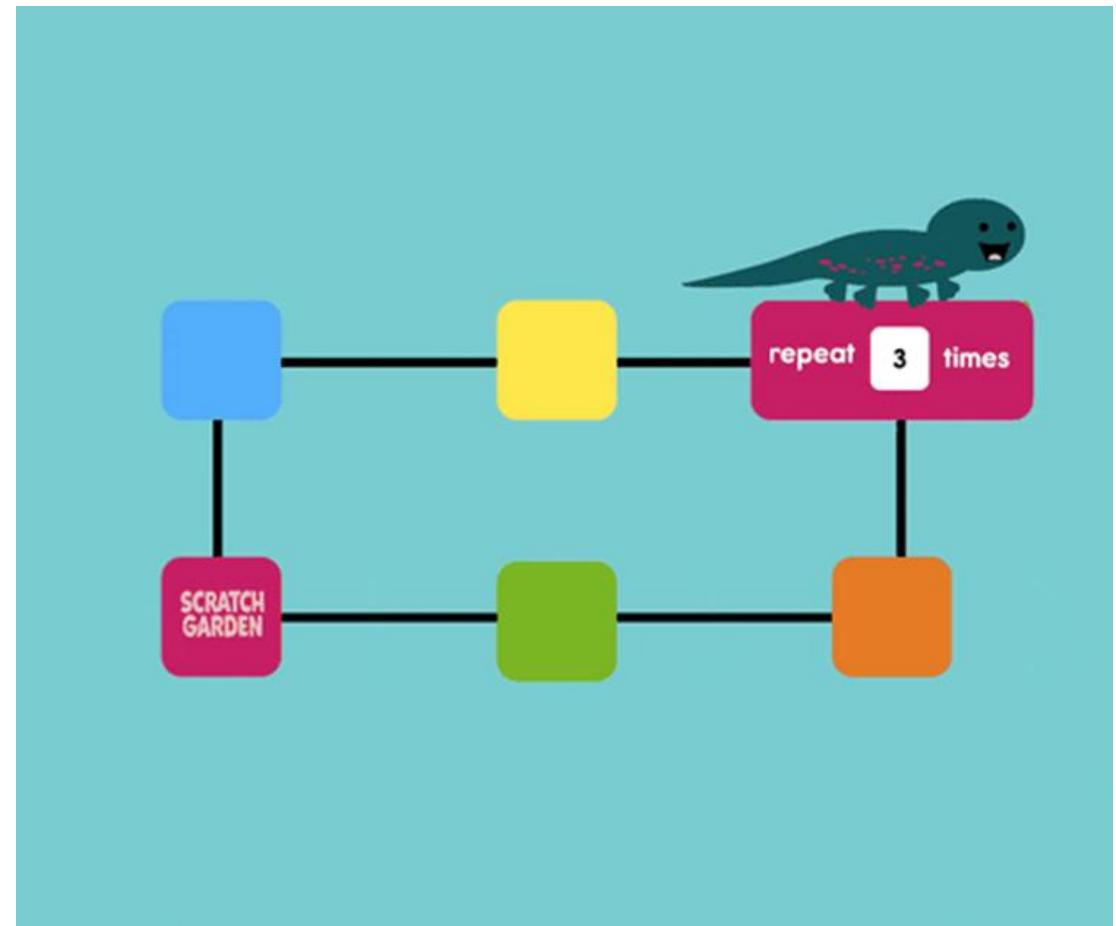
3. Estructuras repetitivas/iterativas

3.1 Introducción a las estructuras repetitivas



3. Estructuras repetitivas/iterativas

3.1 Introducción a las estructuras repetitivas



3. Estructuras repetitivas/iterativas

3.2 Estructura repetitiva FOR



3. Estructuras repetitivas/iterativas

3.2 Estructura repetitiva FOR

Código 3.2_1

The screenshot shows a Java development environment with two main windows: a code editor and a console window.

Code Editor (Main.java):

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int iteraciones = 100;
7         for(int i = 0; i < iteraciones; i++) {
8             System.out.println(i + ". I will not yell \"fire\" in a crowded classroom");
9         }
10    }
11}
12
```

Console Window:

```
<terminated> Main () [Java Application] C:\Program Files\Java\jd
0. I will not yell "fire" in a crowded classroom
1. I will not yell "fire" in a crowded classroom
2. I will not yell "fire" in a crowded classroom
3. I will not yell "fire" in a crowded classroom
4. I will not yell "fire" in a crowded classroom
5. I will not yell "fire" in a crowded classroom
6. I will not yell "fire" in a crowded classroom
7. I will not yell "fire" in a crowded classroom
8. I will not yell "fire" in a crowded classroom
9. I will not yell "fire" in a crowded classroom
10. I will not yell "fire" in a crowded classroom
11. I will not yell "fire" in a crowded classroom
```

3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR

A screenshot of an IDE showing a Java file named Main.java. The code contains a for loop that prints "Iteración i" for i from 0 to 4. A red arrow points from the condition 'i < 5' in the for loop to the output window.

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int i = 0; i<5 ;i++) {
7             System.out.println("Iteración " + i);
8         }
9     }
10 }
```

Console output:

```
<terminated> Main (1)
Iteración 0
Iteración 1
Iteración 2
Iteración 3
Iteración 4
```

A screenshot of an IDE showing a Java file named Main.java. The code contains a for loop that prints "Iteración i" for i from 1 to 5. A red arrow points from the condition 'i < 6' in the for loop to the output window.

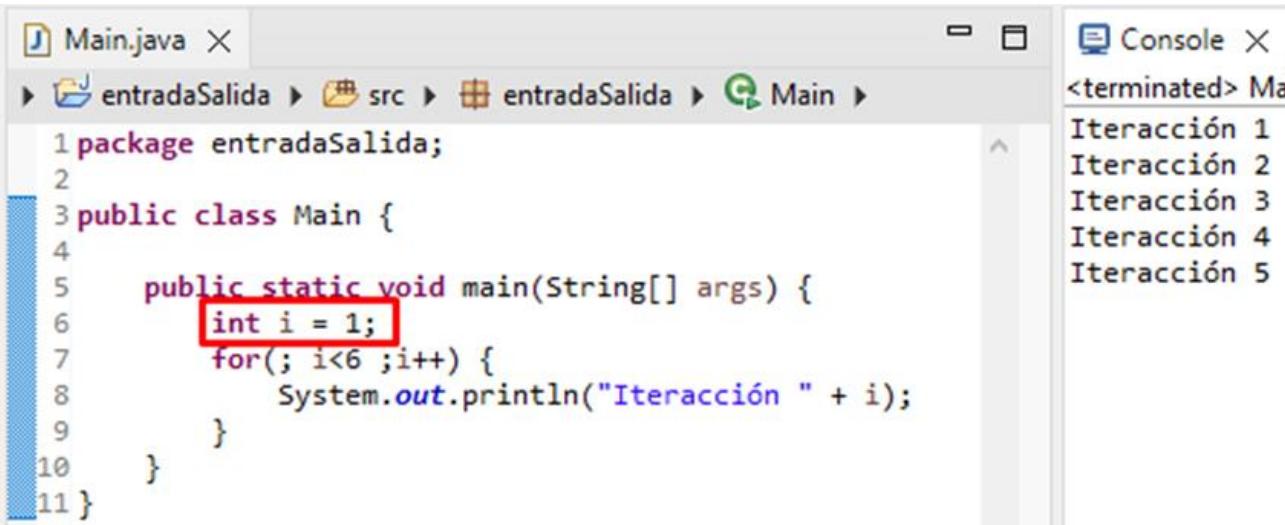
```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int i = 1; i<6 ;i++) {
7             System.out.println("Iteración " + i);
8         }
9     }
10 }
```

Console output:

```
<terminated> Main (1)
Iteración 1
Iteración 2
Iteración 3
Iteración 4
Iteración 5
```

3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR



The screenshot shows a Java development environment with two windows. On the left, the code editor displays Main.java with the following content:

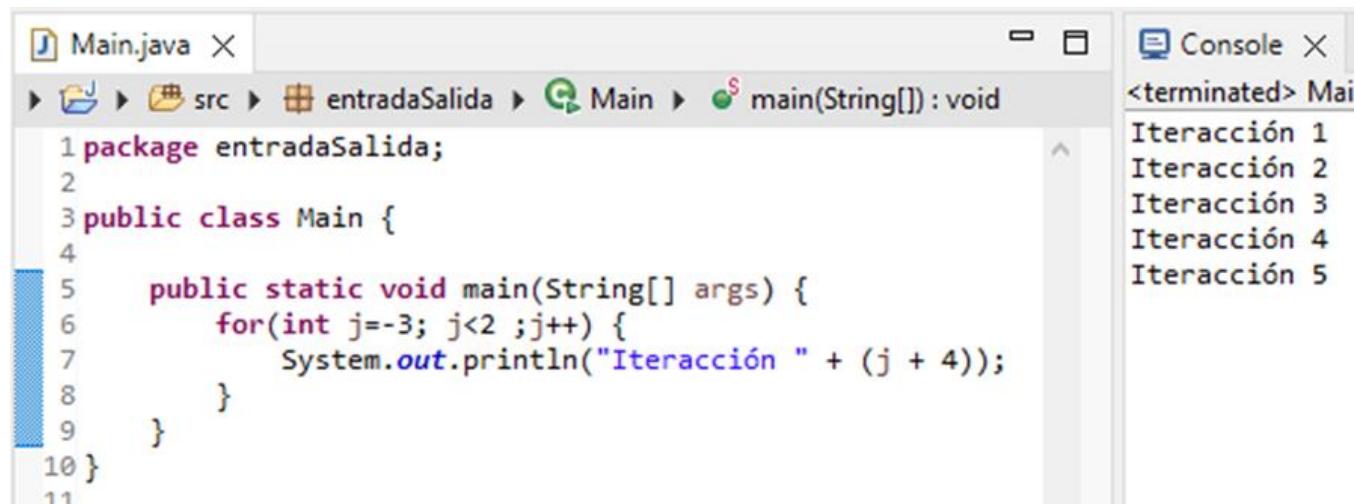
```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         int i = 1;
7         for(; i<6 ;i++) {
8             System.out.println("Iteración " + i);
9         }
10    }
11 }
```

The line `int i = 1;` is highlighted with a red box. On the right, the console window shows the output of the program:

```
<terminated> Mai
Iteración 1
Iteración 2
Iteración 3
Iteración 4
Iteración 5
```

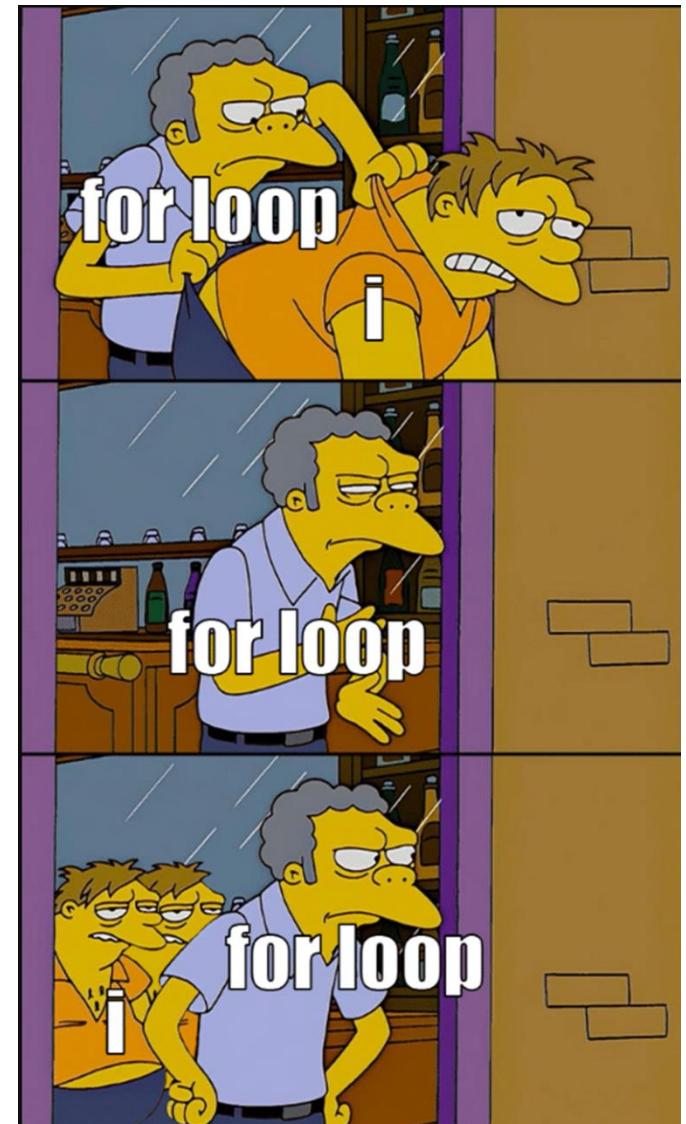
3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR



The screenshot shows an IDE interface with two main panes. The left pane is the code editor for a file named Main.java, located in the package entradaSalida. The code contains a main method with a for loop that prints five iterations of the string "Iteración" followed by a value from 1 to 5. The right pane is the console window, which displays the output of the program: "Iteración 1", "Iteración 2", "Iteración 3", "Iteración 4", and "Iteración 5".

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int j=-3; j<2 ;j++) {
7             System.out.println("Iteración " + (j + 4));
8         }
9     }
10}
```



3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR

The image shows a Java development environment with two panes. The left pane displays the code for a class named Main in a package called entradaSalida. The code contains a main method with a for loop that prints iteration numbers from -3 to 2. The condition j<2 in the for loop is highlighted with a red box. The right pane shows the execution of the program in a terminal window, where the output is 12 lines of "Iteración <number>" followed by the message "<terminated> Main (1) [Java Application]".

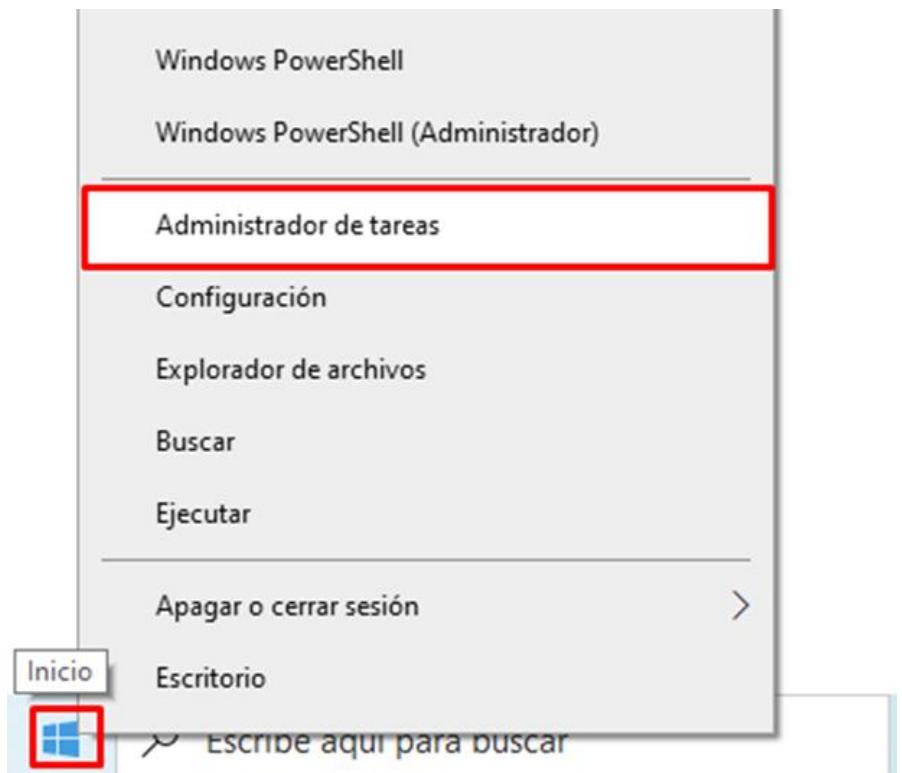
```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int j=-3; j<2 ;j++) {
7             System.out.println("Iteración " + (j + 4));
8         }
9     }
10 }
```

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int j=-3; j<2 ;j++) {
7             System.out.println("Iteración " + (j + 4));
8         }
9     }
10 }
11
12
```

```
Iteración 403858
Iteración 403859
Iteración 403860
Iteración 403861
Iteración 403862
Iteración 403863
Iteración 403864
Iteración 403865
Iteración 403866
Iteración 403867
Iteración 403868
Iteración 403869
Iteración 403870
Iteración 403871
```

3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR



3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR

The screenshot illustrates the execution of a Java application that prints a sequence of numbers from 976044 to 976049. On the left, the Java code in the IDE shows a simple for-loop:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int j=3; j++); {
7             System.out.println("Iteración " + (j + 4));
8         }
9     }
10}
11
12
```

The middle section shows the Windows Task Manager's Performance tab. A red box highlights the CPU usage, which is at 100% 3.68 GHz. The Task Manager also displays memory, disk, and network usage.

The right section shows the Java application's console output. A red arrow points to the close button of the console window. The output is:

```
Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Iteración 976044
Iteración 976045
Iteración 976046
Iteración 976047
Iteración 976048
Iteración 976049
```

3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR

A screenshot of a Java IDE showing a code editor and a console window. The code editor displays the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int i=-3; i<0; i++){
7             System.out.println("Iteración " + i);
8         }
9     }
10}
```

The line `i++` is highlighted with a red box. The console window shows the output of the program:

```
<terminated> Main
Iteración -3
Iteración -2
Iteración -1
```

A screenshot of a Java IDE showing a code editor and a console window. The code editor displays the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int i=-3; i<0; i--){
7             System.out.println("Iteración " + i);
8         }
9     }
10}
```

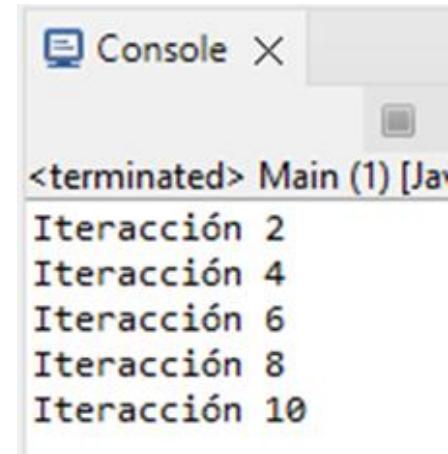
The line `i--` is highlighted with a red box. The console window shows the output of the program:

```
<terminated> Main (1) [Ja
Iteración -500883
Iteración -500884
Iteración -500885
Iteración -500886
Iteración -500887
Iteración -500888
Iteración -500889
Iteración -500890
Iteración -500891
Iteración -500892
```

3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR

Código 3.3_1



The image shows a screenshot of a Java console window titled "Console". The window displays the output of a program named "Main (1)". The output consists of five lines of text: "Iteración 2", "Iteración 4", "Iteración 6", "Iteración 8", and "Iteración 10".

```
<terminated> Main (1) [Java]
Iteración 2
Iteración 4
Iteración 6
Iteración 8
Iteración 10
```

3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR

The screenshot shows an IDE interface with two main windows: 'Main.java X' and 'Console X'.

Main.java X: This window displays the Java code for a class named 'Main'. The code contains a single method 'main' with the following content:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int i=2; i<=10; ) {
7             System.out.println("Iteración " + i);
8             i+=2;
9         }
10    }
11 }
```

The line 'i+=2;' is highlighted with a red box.

Console X: This window shows the output of the program. It displays five lines of text, each starting with 'Iteración' followed by an even number from 2 to 10, indicating the iterations of the loop.

```
<terminated> Main (1)
Iteración 2
Iteración 4
Iteración 6
Iteración 8
Iteración 10
```

3. Estructuras repetitivas/iterativas

3.3 Estructura de una condición de un FOR

Main.java X

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int i=2; i<=10; ) {
7             i+=2;
8             System.out.println("Iteración " + i);
9         }
10    }
11 }
```

The code shows a for loop with an initial value of 2, a condition of $i \leq 10$, and an increment of $i+2$. The increment line is highlighted with a red box.

<terminated> Main
Iteración 4
Iteración 6
Iteración 8
Iteración 10
Iteración 12

Main.java X

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         for(int i=2; i<=10; ++i) {
7             System.out.println("Iteración " + i);
8         }
9     }
10 }
```

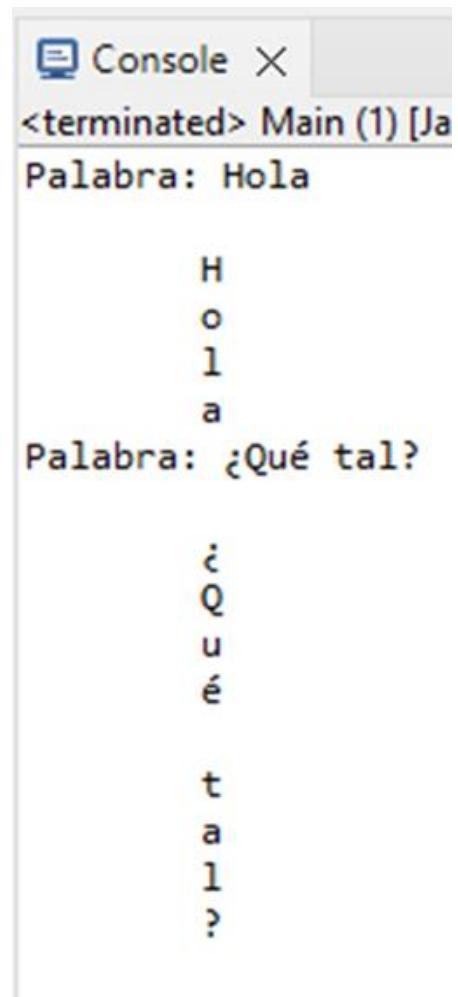
The code shows a for loop with an initial value of 2, a condition of $i \leq 10$, and an increment of $++i$. The increment line is highlighted with a red box.

<terminated> Main
Iteración 2
Iteración 4
Iteración 6
Iteración 8
Iteración 10

3. Estructuras repetitivas/iterativas

3.4 FOR dentro de FOR

Código 3.4_1



The screenshot shows a Java console window titled "Console X". It displays the output of a program that prints characters from two input strings. The first string is "Hola" and the second is "¿Qué tal?". The characters are printed one by one, starting with "H" and "o" for the first word, and "¿" and "Q" for the second word.

```
Console X
<terminated> Main (1) [Ja
Palabra: Hola
H
o
l
a
Palabra: ¿Qué tal?
¿
Q
u
é
t
a
l
?
```

3. Estructuras repetitivas/iterativas

3.5 Estructura de repetición ForEach (para cada), disponible a partir de la versión 5 de Java

```
for (String string : array) {  
}
```

3. Estructuras repetitivas/iterativas

3.5 Estructura de repetición ForEach (para cada), disponible a partir de la versión 5 de Java

Código 3.5_1

The screenshot shows an IDE interface with three main panes. The left pane displays the code for Main.java:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         String[] array = {"Hola", "¿Qué tal?"};
8
9         for (String string : array) {
10            System.out.println(string);
11        }
12    }
13}
14
```

The middle pane shows the code editor with the line `String[] array = {"Hola", "¿Qué tal?"};` highlighted in blue. The right pane shows the console output:

```
<terminated> Main
Hola
¿Qué tal?
```

A red arrow points from the highlighted code in the editor to the corresponding output in the console. Another red box highlights the `System.out.println(string);` line in the editor.

3. Estructuras repetitivas/iterativas

3.5 Estructura de repetición ForEach (para cada), disponible a partir de la versión 5 de Java

Código 3.5_2

The screenshot shows an IDE interface with two tabs: 'Main.java' and 'Console'. The 'Main.java' tab displays the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         String[] array = {"Hola", "¿Qué tal?"};
8
9         for (String string : array) {
10             System.out.println("\n" + string + "\n");
11             char[] letras = string.toCharArray();
12             for (char letra : letras) {
13                 System.out.println(letra);
14             }
15         }
16     }
17 }
```

The 'Console' tab shows the output of the program:

```
Hola
H
o
l
a
¿Qué tal?
¿
Q
u
é
t
a
l
?
```

Annotations with arrows point from the highlighted code lines to the corresponding output in the console. A blue arrow points from the first `System.out.println` call to the first line of output ('Hola'). A red arrow points from the inner loop's `System.out.println` call to the character-by-character output of 'Hola'. Another red arrow points from the second `System.out.println` call to the first line of output ('¿Qué tal?'). A blue arrow points from the second inner loop's `System.out.println` call to the character-by-character output of '¿Qué tal?'.

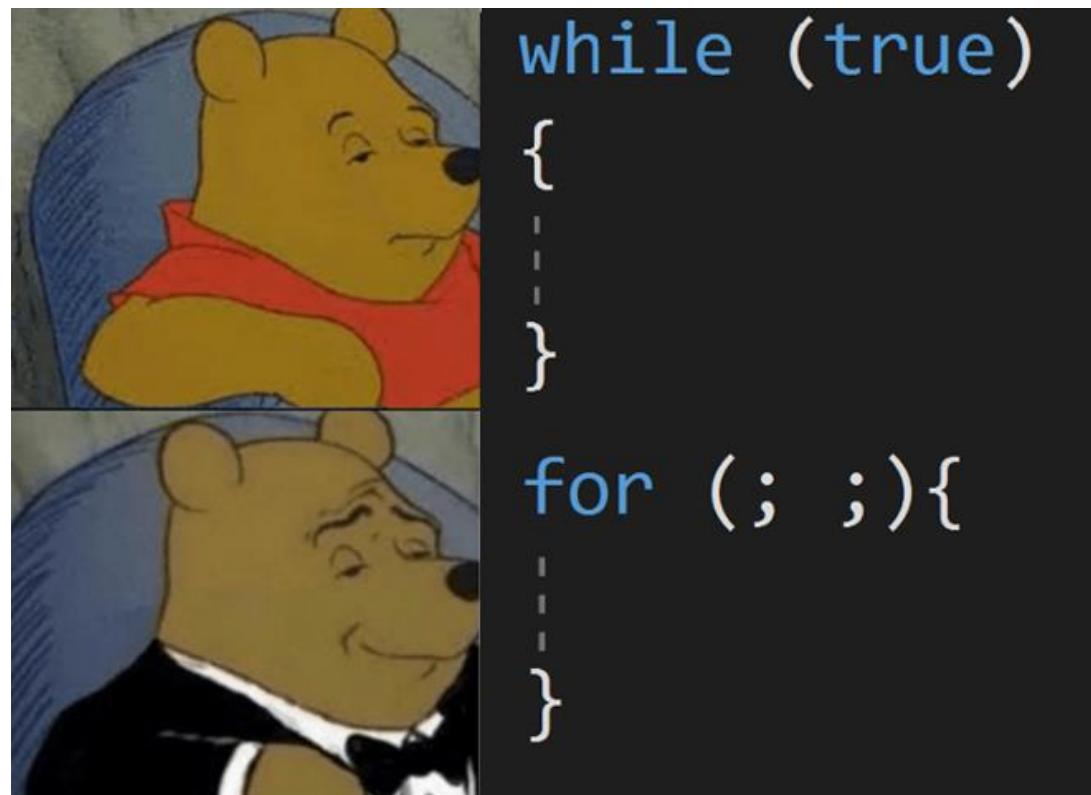
3. Estructuras repetitivas/iterativas

3.6 Estructura de control WHILE

Código 3.6_1

3. Estructuras repetitivas/iterativas

3.6 Estructura de control WHILE



```
Main.java X
Main.java | 1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         boolean interruptor = true;
8
9         while (interruptor) {
10             System.out.println("No hay luz");
11         }
12     }
13 }
14
```

Console X
No consoles to display at this time.

3. Estructuras repetitivas/iterativas

3.6 Estructura de control WHILE

Código 3.6_2

The screenshot shows a Java development environment with two panes. The left pane, titled 'Main.java X', displays the following code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         boolean interruptor = true;
7         int i = 1;
8
9         while (interruptor) {
10             System.out.println(i + ".I will not yell \"fire\" in a crowded classroom");
11             if(i==5) {
12                 System.err.println("Fin del programa");
13                 interruptor = false;
14             }
15         }
16     }
17 }
18 }
```

The right pane, titled 'Console X', shows the output of the program:

```
<terminated> Main (I) [Java Application] C:\Program Files\Java\j
1.I will not yell "fire" in a crowded classroom
2.I will not yell "fire" in a crowded classroom
3.I will not yell "fire" in a crowded classroom
4.I will not yell "fire" in a crowded classroom
5.I will not yell "fire" in a crowded classroom
Fin del programa
```

3. Estructuras repetitivas/iterativas

3.6 Estructura de control WHILE

Código 3.6_3

The screenshot shows a Java IDE interface with two tabs: 'Main.java X' and 'Console X'. The 'Main.java X' tab displays the following code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         int i = 1;
8
9         while (i<=5) {
10             System.out.println(i + ".I will not yell \"fire\" in a crowded classroom");
11             i++;
12         }
13     }
14 }
```

The 'Console X' tab shows the output of the program:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
1.I will not yell "fire" in a crowded classroom
2.I will not yell "fire" in a crowded classroom
3.I will not yell "fire" in a crowded classroom
4.I will not yell "fire" in a crowded classroom
5.I will not yell "fire" in a crowded classroom
```

3. Estructuras repetitivas/iterativas

3.7 Estructura de control DO WHILE

Código 3.7_1

The screenshot shows a Java development environment with two windows. On the left is the code editor for `Main.java`, which contains the following Java code:

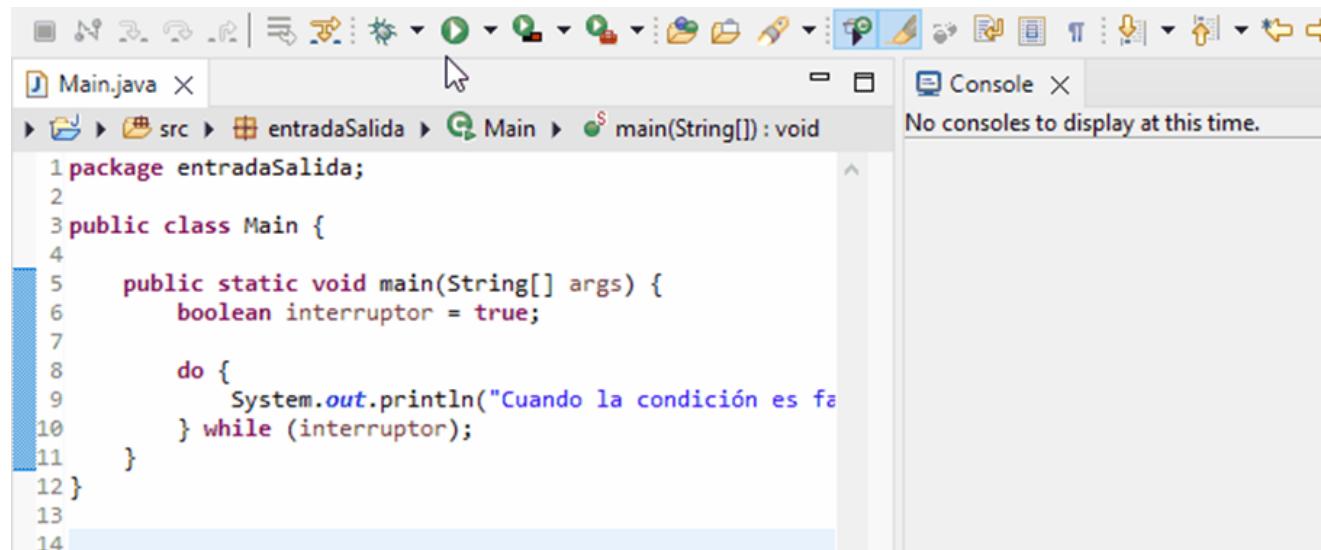
```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         boolean interruptor = false;
7
8         do {
9             System.out.println("Cuando la condición es f"
10        } while (interruptor);
11    }
12}
```

The variable `interruptor` is highlighted with a red box. On the right is the `Console X` window, which displays the output of the application's execution:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk
Cuando la condición es false se ejecuta 1 sola vez
```

3. Estructuras repetitivas/iterativas

3.7 Estructura de control DO WHILE



The screenshot shows a Java development environment with the following details:

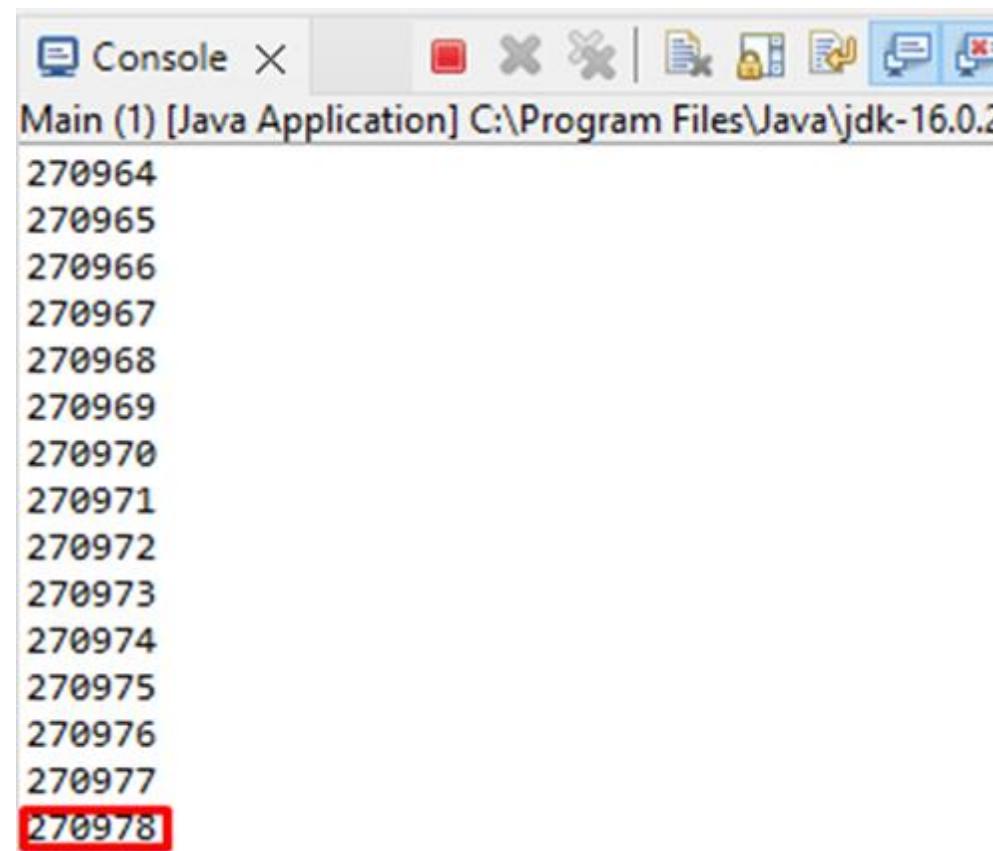
- Title Bar:** Shows the file name "Main.java" and various tool icons.
- Project Explorer:** Displays the package structure: "src > entradaSalida > Main".
- Code Editor:** Contains the Java code for a "do-while" loop:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         boolean interruptor = true;
7
8         do {
9             System.out.println("Cuando la condición es fa");
10        } while (interruptor);
11    }
12}
13
14
```
- Console:** A panel titled "Console X" with the message "No consoles to display at this time."

3. Estructuras repetitivas/iterativas

3.7 Estructura de control DO WHILE

Código 3.7_2



Console X

Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2

```
270964
270965
270966
270967
270968
270969
270970
270971
270972
270973
270974
270975
270976
270977
270978
```

3. Estructuras repetitivas/iterativas

3.8 Ejercicios sobre estructuras iterativas

EJERCICIO 1: REALIZA UN BUCLE CON FOR Y CON WHILE QUE MUESTRE SOLAMENTE LOS NÚMEROS QUE SEAN DIVISIBLES POR 3.

3. Estructuras repetitivas/iterativas

3.8 Ejercicios sobre estructuras iterativas

EJERCICIO 2: REALIZA UN BUCLE QUE EJECUTE UN MENSAJE TANTO SI SE CUMPLE LA CONDICIÓN COMO SI NO Y LUEGO SE EJECUTE 3 VECES. EL EJERCICIO SE DEBE REALIZAR CON DO WHILE Y CON FOR.

4. Instrucciones de salto: break y continue

4. Instrucciones de salto: break y continue

4.1 Presentación de las instrucciones de salto

4. Instrucciones de salto: break y continue

4.2 Ejemplo de uso de las instrucciones de salto break y continue

Código 4.2_1

The screenshot shows a Java application window with three tabs: Console, Problems, and Debug Shell. The Console tab is active and displays the following output:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Ejemplo de bucle FOR normal:
Instrucción 1
Instrucción 2
Instrucción 3
Instrucción 4
Instrucción 5
-----
Ejemplo de bucle FOR con CONTINUE:
Instrucción 1
InSTRUCCIÓN 2
Instrucción 4
Instrucción 5
-----
Ejemplo de bucle FOR con BREAK:
Instrucción 1
InSTRUCCIÓN 2
```

The output demonstrates three examples of for loops. The first example shows a normal loop with five iterations. The second example shows a loop where the second iteration is skipped using the `CONTINUE` statement, resulting in only four iterations. The third example shows a loop where the second iteration is terminated early using the `BREAK` statement, also resulting in only four iterations. The word "STRUCCIÓN" is misspelled as "InSTRUCCIÓN" in the output.

4. Instrucciones de salto: break y continue

4.3 Ejercicios con instrucciones de salto

EJERCICIO 1: REALIZA UN WHILE Y SAL DEL BUCLE CUANDO ESTÉS EN LA 3^a ITERRACIÓN.

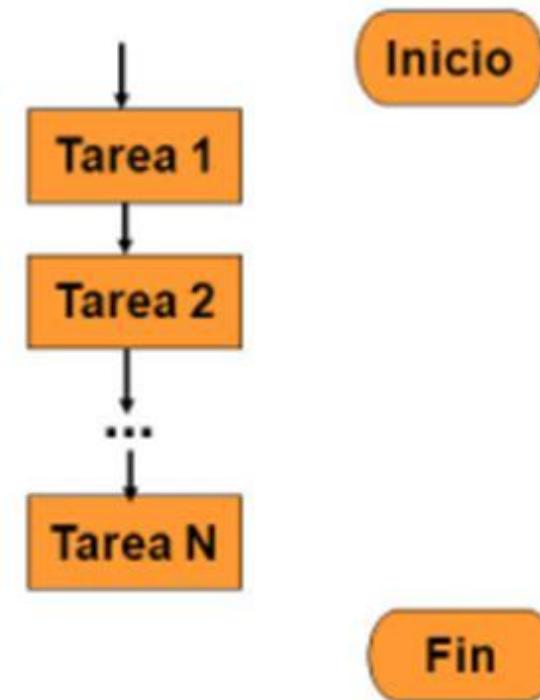
EJERCICIO 2: REALIZA UN BUCLE FOR QUE IMPRIMA UN TEXTO ACOMPAÑADO DEL NÚMERO DE LA ITERACIÓN Y SALTA SOLAMENTE LA 5^a ITERRACIÓN

5. Estructura secuencial y programación funcional

5. Estructura secuencial y programación funcional

5.1 Introducción a la estructura secuencial

Estructura Secuencial:



*Representación en
Diagrama de Flujo*

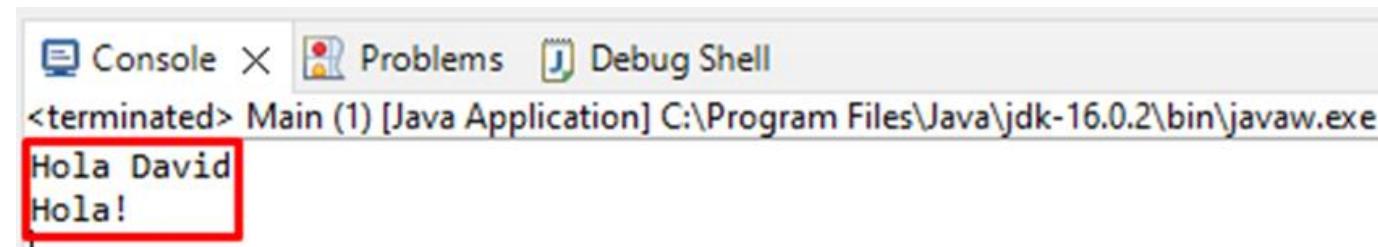
5. Estructura secuencial y programación funcional

5.2 Introducción a la programación funcional

5. Estructura secuencial y programación funcional

5.3 Programación funcional: Creando métodos void (sin return) en la misma clase

Código 5.3_1



The screenshot shows a Java application running in a console window. The window title bar reads "Console X Problems Debug Shell". Below the title bar, it says "<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe". The main content area of the console displays two lines of text: "Hola David" and "Hola!". The first line, "Hola David", is highlighted with a red rectangular box.

5. Estructura secuencial y programación funcional

5.3 Programación funcional: Creando métodos void (sin return) en la misma clase

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // Forma 1
7         Main main = new Main(); //Instanciamos es decir declaramos/construimos la clase a la que pertenece dicho método
8         main.saludarA("David"); //Llamamos al método correspondiente
9
10    // Forma 2: Cuando el método tiene static
11    saludoGenerico();
12
13
14    // Forma 1: Cuando NO asignamos static al método
15    public void saludarA(String nombre){ //
16        System.out.println("Hola " + nombre);
17    }
18
19    // Forma 2: Cuando asignamos static al método
20    public static void saludoGenerico(){
21        System.out.println("Hola!");
22    }
23
24 }
```

The screenshot shows a Java code editor with the file 'Main.java' open. The code defines a class 'Main' with a static method 'main'. It also contains two other methods: 'saludarA' (non-static) and 'saludoGenerico' (static). Annotations with arrows explain the usage of static methods:

- A red box highlights the instantiation of the class and the call to a non-static method: 'Main main = new Main();' and 'main.saludarA("David");'. An arrow points from this box to the first annotation: 'Forma 1: Cuando NO asignamos static al método'.
- A blue box highlights the declaration of a static method: 'public static void saludoGenerico()'. An arrow points from this box to the second annotation: 'Forma 2: Cuando asignamos static al método'.

5. Estructura secuencial y programación funcional

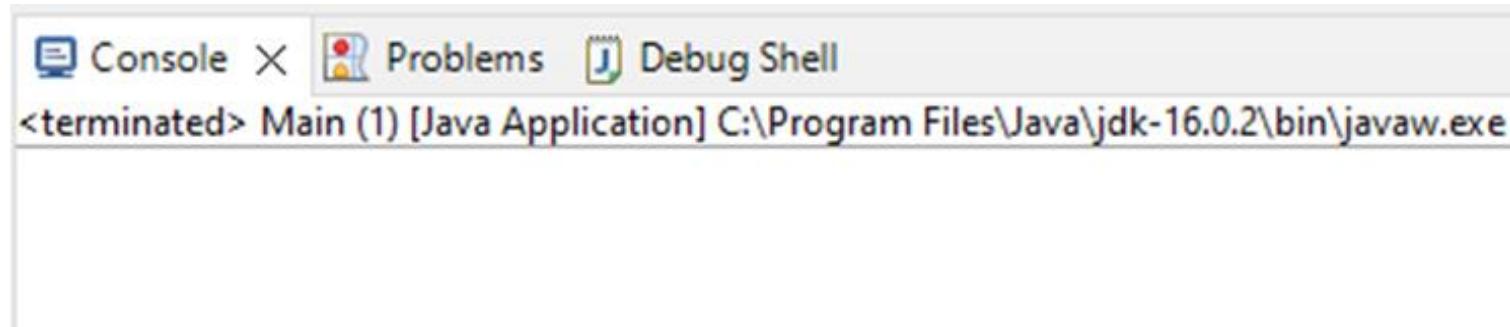
5.4 Programación funcional: Creando métodos (sin void) con return en la misma clase

```
1 Main.java X
2
3 package entradaSalida;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         // Forma 1
9         Main main = new Main(); //Instanciamos es decir declaramos/construimos la clase a la que pertenece dicho método.
10        main.saludarA("David"); //Llamamos al método correspondiente.
11
12        // Forma 2: Cuando el método tiene static
13        saludоГenerico();
14    }
15
16    // Forma 1: Cuando NO asignamos static al método
17    public void saludarA(String nombre){ //
18        System.out.println("Hola " + nombre);
19    }
20
21    // Forma 2: Cuando asignamos static al método
22    public static void saludоГenericо(){
23        System.out.println("Hola!");
24    }
}
```

5. Estructura secuencial y programación funcional

5.4 Programación funcional: Creando métodos (sin void) con return en la misma clase

Código 5.4_1



5. Estructura secuencial y programación funcional

5.4 Programación funcional: Creando métodos (sin void) con return en la misma clase

Código 5.4_2

```
Main.java X
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // Forma 1
7         Main main = new Main(); //Instanciamos es decir declaramos/construimos la clase a la que pertenece dicho método
8         System.out.println(main.saludarA("David")); //llamamos al método correspondiente
9
10        // Forma 2: Cuando el método tiene static
11        System.out.println(saludoGenerico());
12    }
13
14    // Forma 1: Cuando NO asignamos static al método
15    public String saludarA(String nombre){
16        return "Hola " + nombre;
17    }
18
19    // Forma 2: Cuando asignamos static al método
20    public static String saludoGenerico(){
21        return "Hola!";
22    }
23
24}
```

The screenshot shows an IDE interface with two main sections: a code editor and a terminal window.

Code Editor: The file `Main.java` is open. It contains Java code demonstrating functional programming concepts like static methods and method references. A red box highlights the line `System.out.println(main.saludarA("David"));`, with a tooltip explaining it calls the method on the instance `main`. Another red box highlights the line `System.out.println(saludoGenerico());`, with a tooltip explaining it calls the static method `saludoGenerico`.

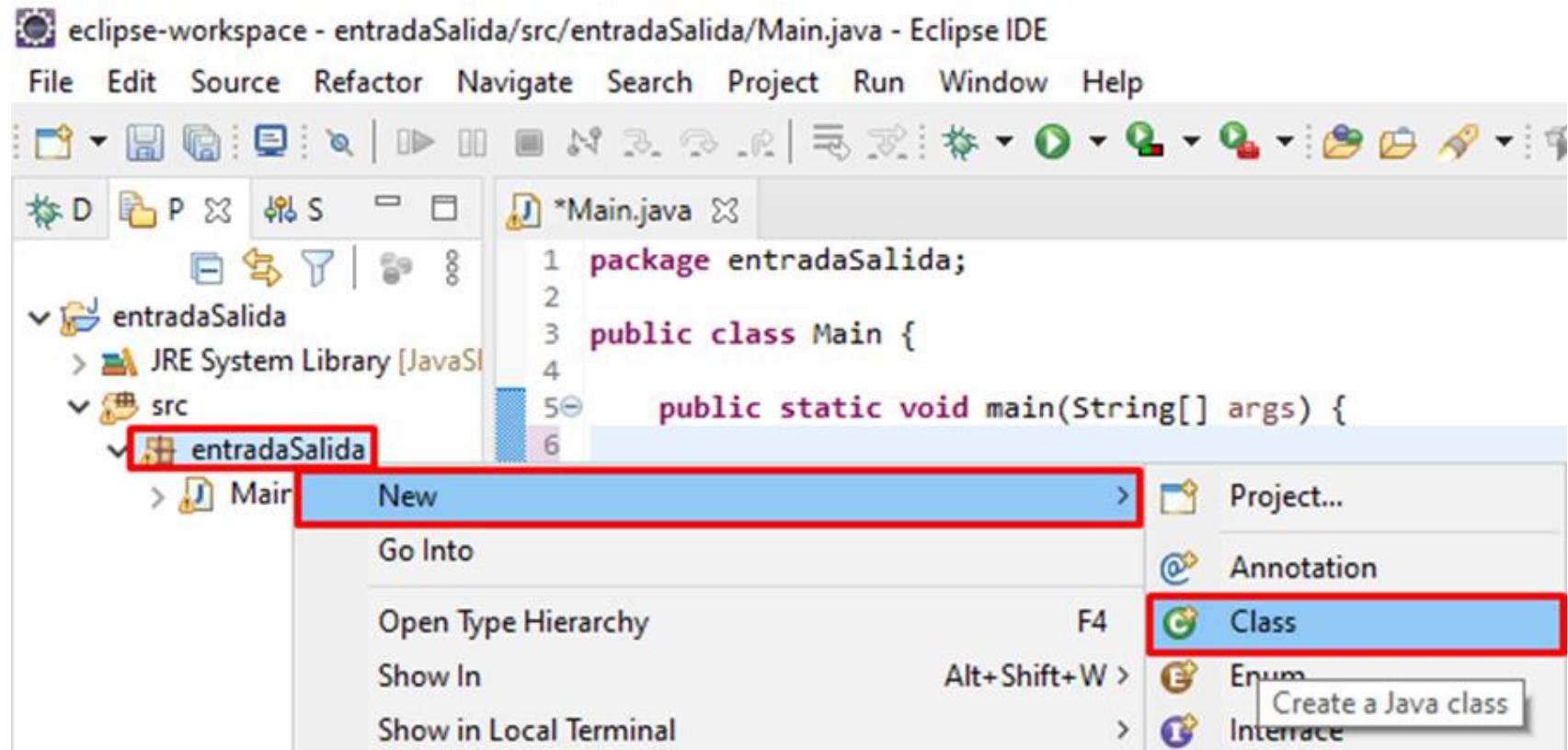
Terminal Window: The title bar says `Console X`. The output shows the results of running the application:

```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Hola David
Hola!
```

The last two lines, `Hola David` and `Hola!`, are highlighted with a red box.

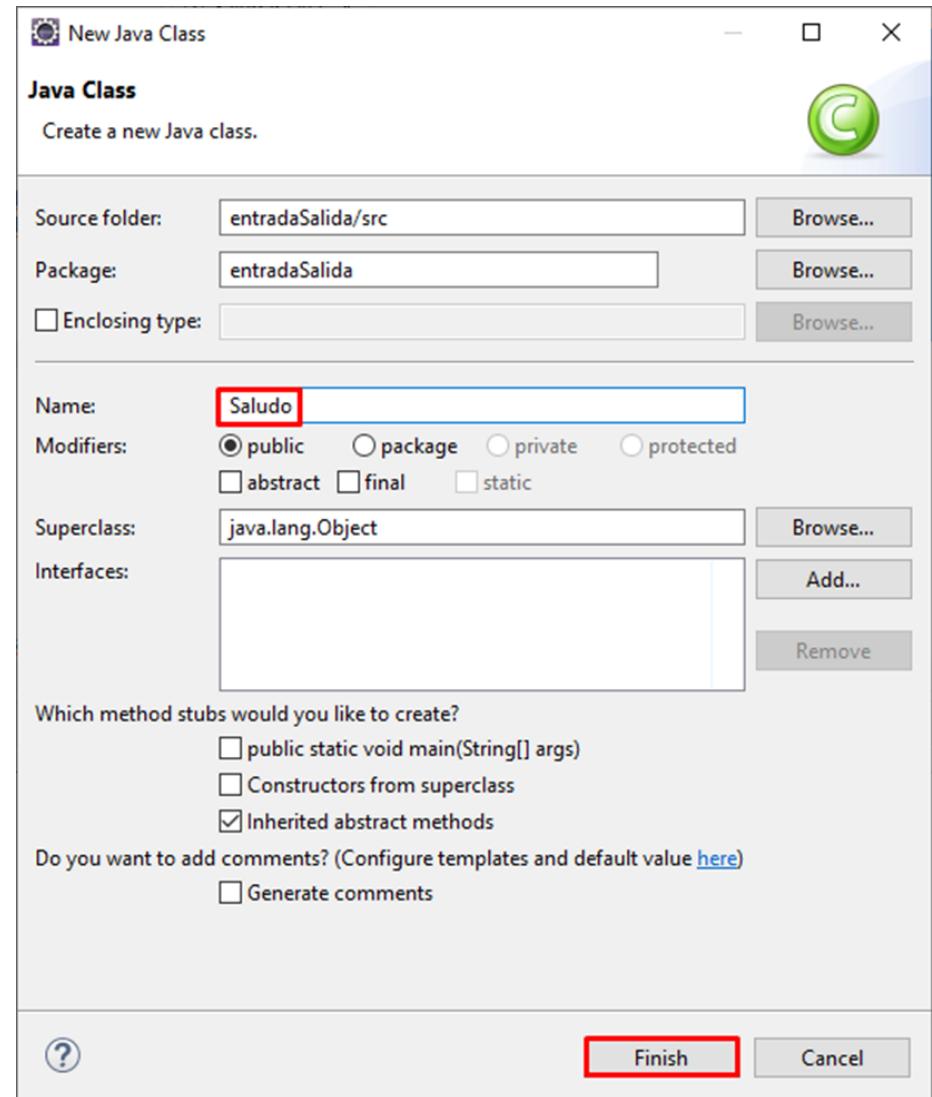
5. Estructura secuencial y programación funcional

5.5 Programación funcional: Creando métodos void (sin return) en la misma clase



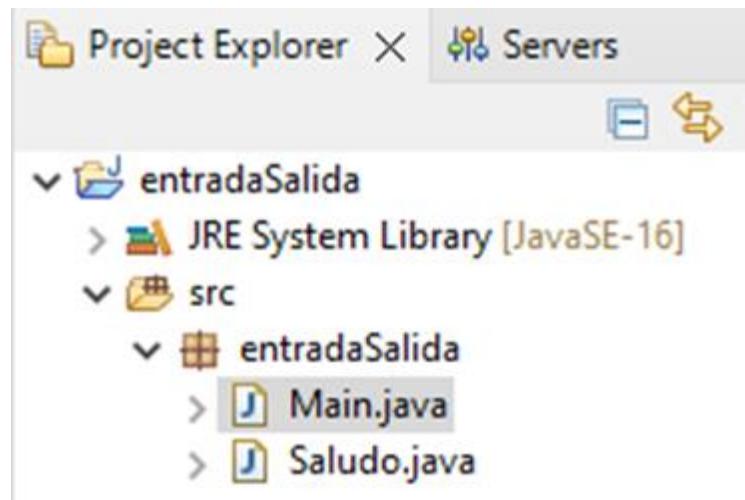
5. Estructura secuencial y programación funcional

5.5 Programación funcional: Creando métodos void (sin return) en la misma clase



5. Estructura secuencial y programación funcional

5.5 Programación funcional: Creando métodos void (sin return) en la misma clase



5. Estructura secuencial y programación funcional

5.5 Programación funcional: Creando métodos void (sin return) en la misma clase

Código 5.5_1

Código 5.5_2

```
Main.java X
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         // Forma 1: Cuando el método no tiene static
7         Saludo saludo = new Saludo(); //Instanciamos (llamamos) a la clase
8         saludo.saludoA("David"); // Ejecutamos el método
9
10    // Forma 2: Cuando el método tiene static
11    Saludo.saludoGenerico(); // Clase.metodo();
12
13 }
14 
```

```
Saludo.java X
1 package entradaSalida;
2
3 public class Saludo {
4
5     // Forma 1: Cuando NO asignamos static al método
6     public void saludoA(String nombre){ //
7         System.out.println("Hola " + nombre);
8     }
9
10    // Forma 2: Cuando asignamos static al método
11    public static void saludoGenerico(){
12        System.out.println("Hola!");
13    }
14 
```

5. Estructura secuencial y programación funcional

5.5 Programación funcional: Creando métodos void (sin return) en la misma clase

Código 5.5_3

The screenshot shows an IDE interface with two main windows. On the left, the code editor displays `Main.java` with the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4     public static void main(String[] args) {
5         System.out.println(saludar("David"));
6         System.out.println(saludar("Juan"));
7     }
8
9     public static String saludar(String txt) {
10        return txt.equals("David")?txt:null; //
11    }
12 }
```

The code uses a functional programming style with a `saludar` function that returns `null` if the input is "David". The `main` method calls this function twice with arguments "David" and "Juan".

On the right, the `Console` window shows the output of the program:

```
<terminated> Main
David
null
```

Two red arrows point from the highlighted lines in the code editor to the corresponding output lines in the console window, illustrating the execution flow.

5. Estructura secuencial y programación funcional

5.6 Ejercicios de programación funcional

EJERCICIO 1: REALIZA UNA CLASE PRINCIPAL LLAMADA MAIN QUE CONTENDRÁ DOS MÉTODOS (DENTRO DE LA MISMA CLASE) CUYOS MÉTODOS TENDRÁS POSTERIORMENTE QUE INSTANCIAR. LOS MÉTODOS DEBEN CUMPLIR LAS SIGUIENTES CONDICIONES:

- GETNOMBRE: MOSTRARÁ POR PANTALLA TU NOMBRE SIN SER PASADO COMO PARAMETRO. ESTE MÉTODO DEBE NO RETORNARNOS NADA (Y, POR TANTO, DEBERÁ UTILIZAR VOID Y NO RETURN).
- GETCODIGOPOSTAL MOSTRARÁ POR PANTALLA TU CÓDIGO POSTAL. DICHO CÓDIGO POSTAL SERÁ PASADO COMO PARAMETRO. ESTE MÉTODO DEBE RETORNARNOS EL CÓDIGO POSTAL (Y POR TANTO, DEBERÁ UTILIZAR VOID Y NO RETURN). Y FINALMENTE, IMPRIME (CON LA CLASE SYSTEM.OUT.PRINLNT) EL CÓDIGO POSTAL POR PANTALLA.

5. Estructura secuencial y programación funcional

5.6 Ejercicios de programación funcional

EJERCICIO 2: REALIZA LO MISMO QUE HEMOS REALIZADO ANTERIORMENTE, PERO PASANDO DICHOS MÉTODOS A UNA NUEVA CLASE LLAMADA USUARIO.

EJERCICIO 3: REALIZA UN PROGRAMA QUE SOLAMENTE DEVUELVA UNA CADENA DE TEXTO SI EL RESULTADO DE DICHA CADENA ES JAVA. EN CASO CONTRARIO, DEVOLVERÁ UN NULL.

6. ENTRADA DE DATOS CON SCANNER

6. ENTRADA DE DATOS CON SCANNER

6.1 ¿Para que utilizamos la clase Scanner?

6. ENTRADA DE DATOS CON SCANNER

6.2 Importando librería necesaria para trabajar con Scanner

```
import java.util.Scanner;
```

6. ENTRADA DE DATOS CON SCANNER

6.3 Aprendiendo a usar la API de Java

A screenshot of a Google search results page. The search bar at the top contains the query "jdk 14 api". Below the search bar are navigation links for "Todo", "Videos", "Noticias", "Imágenes", "Maps", "Más", and "Herramientas". A message indicates "Aproximadamente 9.600.000 resultados (0,52 segundos)". The first result is a link to "JDK 14 Documentation - Home - Oracle Help Center", which is highlighted with a red box. Below the link, a snippet of text reads: "The documentation for JDK 14 includes developer guides, API documentation, and release notes." Another link, "Oracle JDK Migration Guide - JAR File Specification", is also visible.

6. ENTRADA DE DATOS CON SCANNER

6.3 Aprendiendo a usar la API de Java



The screenshot shows a browser window displaying the Java SE 14 Documentation for the `Scanner` class. The URL is docs.oracle.com/en/java/javase/14/. The search bar at the top has "Java SE 14" selected, and the term "Scanner" is typed. A red box highlights the search term "Scanner". Below the search bar, the results for "Scanner (Java SE 14 & JDK 14)" are shown, with another red box highlighting this text.

**Module java.base
Package java.util**

Class Scanner

`java.lang.Object
java.util.Scanner`

All Implemented Interfaces:
`Closeable, AutoCloseable, Iterator<String>`

`public final class Scanner
extends Object
implements Iterator<String>, Closeable`

A simple text scanner which can parse primitive types and strings using regular expressions.

A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various next methods.

For example, this code allows a user to read a number from System.in:

```
Scanner sc = new Scanner(System.in);  
int i = sc.nextInt();
```

6. ENTRADA DE DATOS CON SCANNER

6.4 Realizando el autoimport de Scanner

The screenshot shows a Java code editor with the file `Main.java` open. The code is as follows:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Scanner sc = new Scanner();
7     }
8 }
```

A red arrow points from the word `Scanner` in line 6 to a dropdown menu. The menu contains the following options:

- Import 'Scanner' (java.util)
- >Create class 'Scanner'
- Create record 'Scanner'
- Create interface 'Scanner'
- Change to 'JSpinner' (javax.swing)
- Change to 'Signer' (java.security)

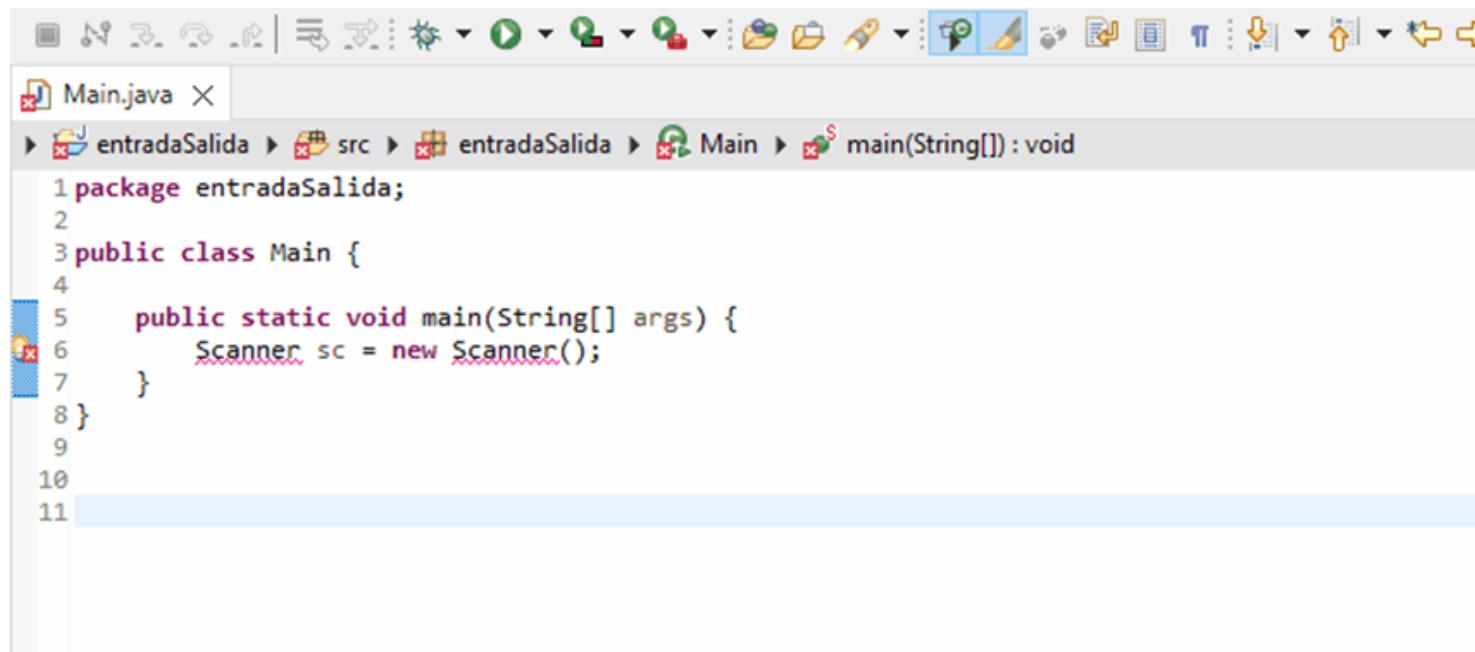
The option `- Import 'Scanner' (java.util)` is highlighted with a red box. A red arrow points from this box to the `import java.util.Scanner;` statement in the code completion preview on the right.

Code completion preview:

```
... package entradaSalida;
import java.util.Scanner;
public class Main { ... }
```

6. ENTRADA DE DATOS CON SCANNER

6.4 Realizando el autoimport de Scanner



The screenshot shows a Java IDE interface with the following details:

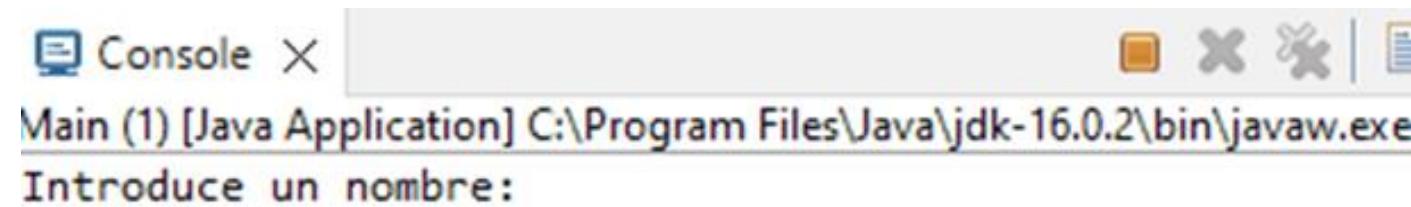
- Title Bar:** Shows the file name "Main.java X".
- Project Explorer:** Displays the package structure: "entradaSalida > src > entradaSalida > Main".
- Code Editor:** Contains the following Java code:

```
1 package entradaSalida;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         Scanner sc = new Scanner();
7     }
8 }
9
10
11
```

6. ENTRADA DE DATOS CON SCANNER

6.5 Ejemplo de cómo capturar un String con Scanner

Código 6.5_1



6. ENTRADA DE DATOS CON SCANNER

6.6 Explicando cómo y porque utilizar el método close al finalizar el uso de Scanner

Código 6.6_1

The screenshot shows an IDE interface with two panes: 'Main.java' and 'Console X'.

Main.java:

```
1 package entradaSalida;
2
3 import java.util.Scanner; // Importamos la clase Scanner
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9
10        System.out.print("Introduce un nombre: ");
11        String name = sc.nextLine();
12        System.out.println("Hola " + name + "!");
13
14        sc.close(); // Cerramos el uso de Scanner por lo tanto no se podra usar mas
15
16        System.out.print("Introduce un nombre: ");
17        name = sc.nextLine();
18        System.out.println("Hola " + name + "!");
19    }
}
```

Console X:

```
<terminated> Main [1] [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Introduce un nombre: Test1
Hola Test1!
Introduce un nombre: exception in thread "main" java.lang.IllegalStateException: Scanner closed
at java.base/java.util.Scanner.ensureOpen(Scanner.java:1150)
at java.base/java.util.Scanner.findWithinHorizon(Scanner.java:1761)
at java.base/java.util.Scanner.nextLine(Scanner.java:1649)
at entradaSalida.Main.main(Main.java:17)
```

A red box highlights the line `sc.close();`. A green box highlights the second block of code in the console output where the program fails to read from the scanner again.

6. ENTRADA DE DATOS CON SCANNER

6.6 Explicando cómo y porque utilizar el método close al finalizar el uso de Scanner

Method Summary		
All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
void	close()	Closes this scanner.
Pattern	delimiter()	Returns the Pattern this Scanner is currently using to match delimiters.
Stream<MatchResult>	findAll(String patString)	Returns a stream of match results that match the provided pattern string.

6. ENTRADA DE DATOS CON SCANNER

6.6 Explicando cómo y porque utilizar el método close al finalizar el uso de Scanner

```
close
public void close() {
    Closes this scanner.

    If this scanner has not yet been closed then if its underlying readable also implements the Closeable interface then the readable's close method will be invoked. If this scanner is already closed then invoking this method will have no effect.

    Attempting to perform search operations after a scanner has been closed will result in an IllegalStateException.

    Specified by:
    close in interface AutoCloseable
```

```
Introduce un nombre: Test1
Hola Test1!
Introduce un nombre: Exception in thread "main" java.lang.IllegalStateException: Scanner closed
        at java.base/java.util.Scanner.ensureOpen(Scanner.java:1150)
        at java.base/java.util.Scanner.findWithinHorizon(Scanner.java:1781)
        at java.base/java.util.Scanner.nextLine(Scanner.java:1649)
        at entradaSalida.Main.main(Main.java:17)
```



6. ENTRADA DE DATOS CON SCANNER

6.6 Explicando cómo y porque utilizar el método close al finalizar el uso de Scanner

¿Qué tendríamos que hacer aquí?

Código 6,6_2

```
Main.java X
entraSalida > src > entraSalida > Main > main(String[]): void
package entradaSalida;
import java.util.Scanner; // importamos la clase Scanner del package java.util
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // Creando Scanner
        System.out.print("Introduce un nombre: ");
        String name = sc.nextLine();
        System.out.print("Introduce tu apellido: ");
        String apellido = sc.nextLine();
        System.out.println("Hola " + name + " " + apellido + "!");
    }
}

Console X
<terminated> Main (1) [Java Application]
Introduce un nombre: David
Introduce tu apellido: Bernal
Hola David Bernal!
```

6. ENTRADA DE DATOS CON SCANNER

6.7 Métodos a utilizar en función del tipo de dato que queremos capturar

- nextInt() método para leer un valor int.
- nextLong() método para leer un valor long.
- nextShort() método para leer un valor short .
- nextDouble() método para leer un valor double.
- nextByte() método para leer un valor byte.
- nextBoolean() método para leer un valor boolean.
- nextLine() método para leer un String.

6. ENTRADA DE DATOS CON SCANNER

6.8 Protegiendo nuestro código de excepciones cuando trabajamos con Scanner

Código 6.8_1



The screenshot shows a Java development environment with two windows. On the left, the code editor window displays the file `Main.java` with the following content:

```
1 package entradaSalida;
2
3 import java.util.Scanner;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         System.out.print("Introduce un número entero: ");
9         Scanner sc = new Scanner(System.in); // Creando Scanner
10        int num = sc.nextInt();
11        System.out.println("Genial, me gusta el " + num);
12        sc.close(); // Cerrando Scanner
13    }
14}
15
```

On the right, the console window shows the application's output:

```
<terminated> Main (1) [Java Application]
Introduce un número entero: 3
Genial, me gusta el 3
```

The user input "3" and the program's response "Genial, me gusta el 3" are highlighted with a red box.

6. ENTRADA DE DATOS CON SCANNER

6.8 Protegiendo nuestro código de excepciones cuando trabajamos con Scanner

The screenshot shows a Java development environment with two panes. On the left, the code editor displays `Main.java` with the following content:

```
1 package entradaSalida;
2
3 import java.util.Scanner;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         System.out.print("Introduce un número entero: ");
9         Scanner sc = new Scanner(System.in); // Creando Scanner
10        int num = sc.nextInt();
11        System.out.println("Genial, me gusta el " + num);
12        sc.close(); // Cerrando Scanner
13    }
14}
```

On the right, the console window shows the output of running the program:

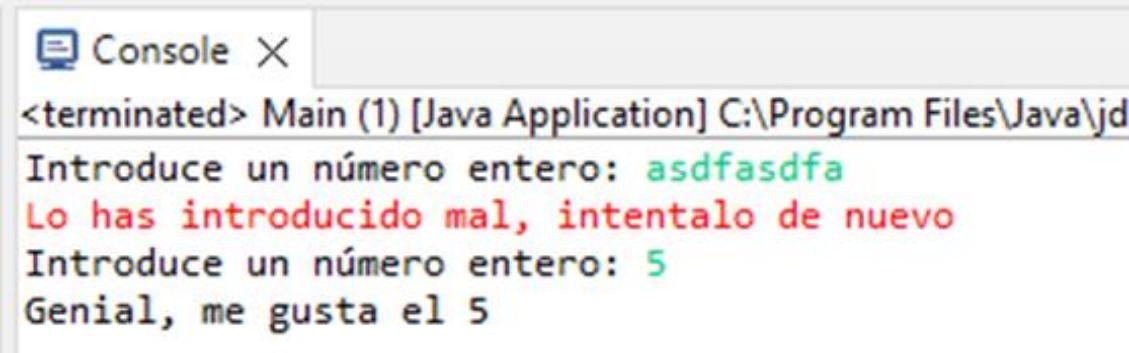
```
<terminated> Main (1) [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe
Introduce un número entero: que número v que ná!
Exception in thread "main" java.util.InputMismatchException
at java.base/java.util.Scanner.throwFor(Scanner.java:939)
at java.base/java.util.Scanner.next(Scanner.java:1594)
at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
at entradaSalida.Main.main(Main.java:10)
```

A red box highlights the stack trace in the console output.

6. ENTRADA DE DATOS CON SCANNER

6.8 Protegiendo nuestro código de excepciones cuando trabajamos con Scanner

Código 6.8_2



The screenshot shows a Java application window titled "Console X". The title bar includes the text "<terminated> Main (1) [Java Application] C:\Program Files\Java\jd". The main area of the window displays the following interaction:

```
Introduce un número entero: asdfasdfa
Lo has introducido mal, intentalo de nuevo
Introduce un número entero: 5
Genial, me gusta el 5
```

The first input "asdfasdfa" is highlighted in green, indicating it was successfully read by the scanner. The error message "Lo has introducido mal, intentalo de nuevo" is displayed in red. The second input "5" is also highlighted in green, showing successful reading after the exception was handled.

6. ENTRADA DE DATOS CON SCANNER

6.9 Ejercicios con Scanner

EJERCICIO 1: REALIZA UN EJEMPLO DE SCANNER CON TODOS LOS MÉTODOS SIGUIENTE:

- nextInt() método para leer un valor int.
- nextLong() método para leer un valor long.
- nextShort() método para leer un valor short .
- nextDouble() método para leer un valor double.
- nextByte() método para leer un valor byte.
- nextBoolean() método para leer un valor boolean.
- nextLine() método para leer un String.

6. ENTRADA DE DATOS CON SCANNER

6.9 Ejercicios con Scanner

EJERCICIO 2: REALIZA UN TAMAGOCHI QUE MUESTRE LAS SIGUIENTES OPCIONES:

- 1. SALUDAR
- 2. COMER
- 3. DORMIR
- 4. CANTAR
- 5. APAGAR TAMAGOCHI

Y QUE UTILIZANDO SCANNER CAPTURÉ EL VALOR INTRODUCIDO POR EL USUARIO E INTERACTURE CON EL. EL PROGRAMA DEBE SEGUIR FUNCIONANDO TANTO CUANDO EL USUARIO INTERACTUÉ UNA O VARIAS VECES CON EL TAMAGOCHI COMO CUANDO SE INTRODUZCA UN VALOR NO VÁLIDO. CUANDO SE INTRODUZCA UN VALOR INCORRECTO, DEBERÁ MOSTRARSE UN MENSAJE POR LA CONSOLA EN ROJO DONDE APAREZCA QUE EL VALOR ES INCORRECTO Y PIDIENDONOS QUE INTRODUZCAMOS NUEVAMENTE UNA OPCIÓN. EN EL CASO DE QUE EL USUARIO APAGUÉ EL TAMAGOCHI (OPCIÓN 5) FINALIZAREMOS EL FLUJO DE EJECUCIÓN DE NUESTRO PROGRAMA.

AUNQUE NO ES OBLIGATORIO, SI QUERÉIS, PODÉIS TRABAJAR CON DIBUJOS ASCIIS QUE NOS PERMITIRÁN REALIZAR COSAS COMO LA SIGUIENTE

6. ENTRADA DE DATOS CON SCANNER

6.9 Ejercicios con Scanner

