PROGRAMACIÓN BÁSICA CON JAVA

Índice de contenido

1.

Variables

1. Variables.

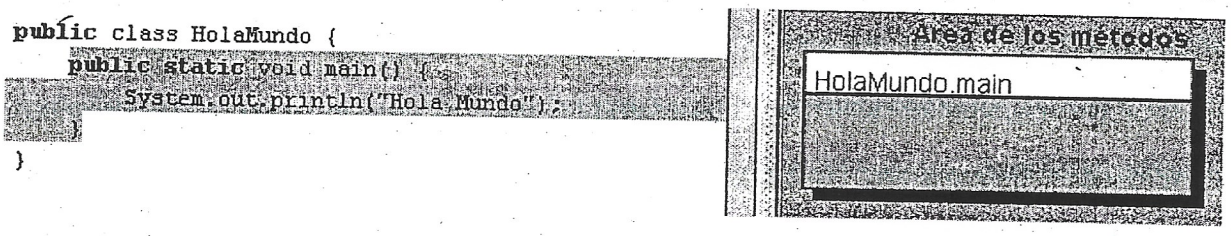
1.1. HolaMundo.

1.1.1. Código.

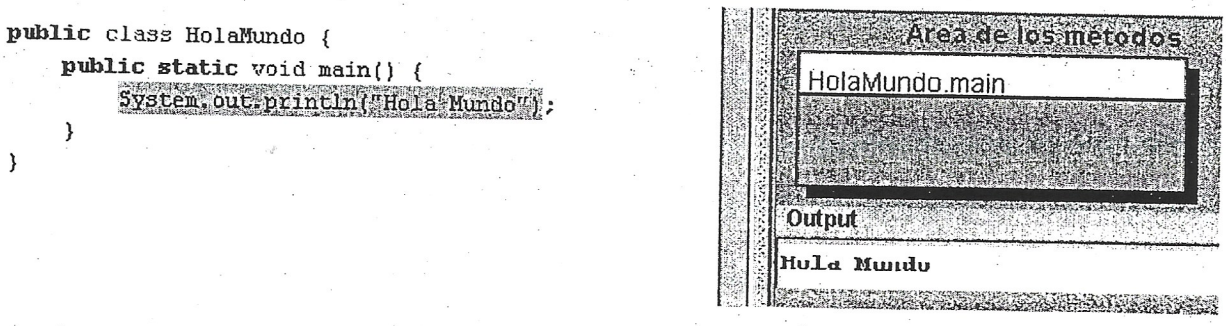
```
1 public class HolaMundo {
2     public static void main() {
3         System.out.println("Hola Mundo");
4     }
5 }
```

1.1.2. Ejecución.

1. El programa empieza en el método de clase main. Se crea un registro de activación.

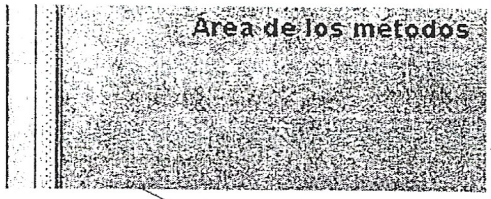


2. "System.out" es la ventana de visualización asociada al programa, también conocida como salida del programa. En ella se muestra el mensaje "Hola Mundo". Las comillas no aparesen en la salida y se realiza un salto de línea.



3. Se destruye el registro de activación para man y termina la ejecución del programa.

```
public class HolaMundo {
    public static void main() {
        System.out.println("Hola Mundo");
    }
}
```



1.2. Variables y Constantes.

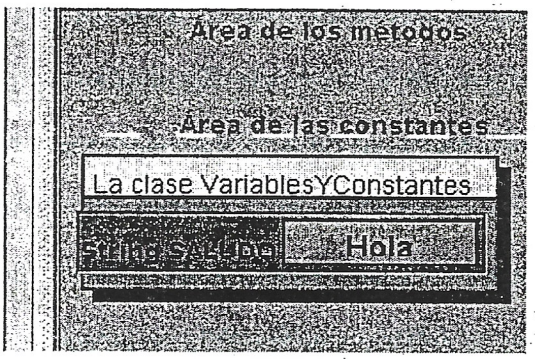
1.2.1. Código.

```
 1 public class VariablesYConstantes {
 2     public static final String SALUDO = "Hola";
 3     public static void main() {
 4         int a = 2;
 5         System.out.println("a");
 6         System.out.println(a);
 7         System.out.println("El valor de a es: " + a);
 8         System.out.println(SALUDO);
 9     }
10 }
```

1.2.2. Ejecución.

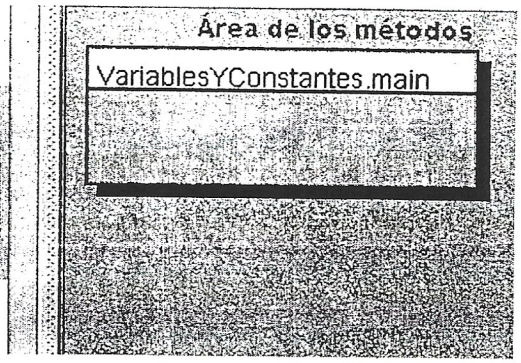
1 Define una constante llamada SALUDO con el valor "HOLA". Las constantes nunca cambian su valor y generalmente se escriben en mayúsculas.

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```



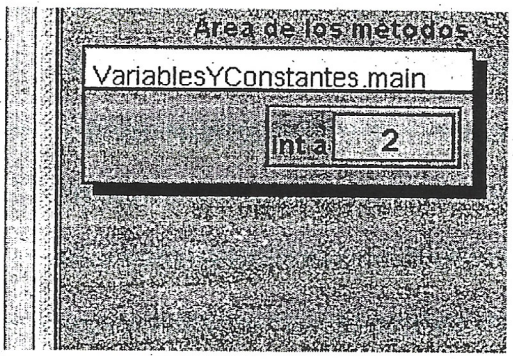
2 Se crea un registro de activación para el método de clase main.

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```



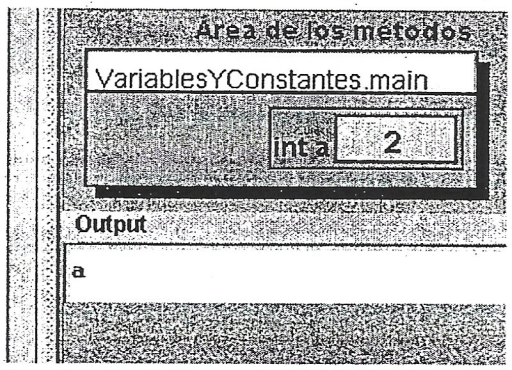
3 Se crea una variable entera llamada "a" en el registro de activación de main y se le asigna el valor "2".

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```



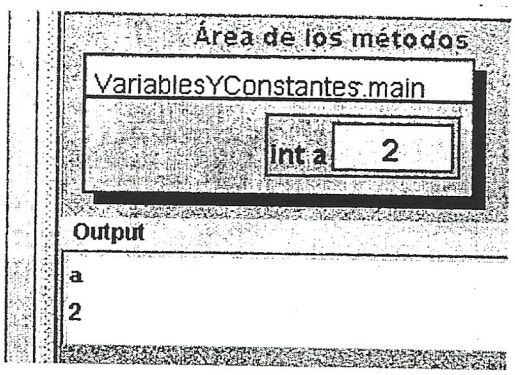
4 Muestra el texto "a" en la salida del programa (las comillas no aparecen) y realiza un salto de línea.

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```



5 Muestra el valor de la variable "a", o sea "2", y realiza un salto de línea.

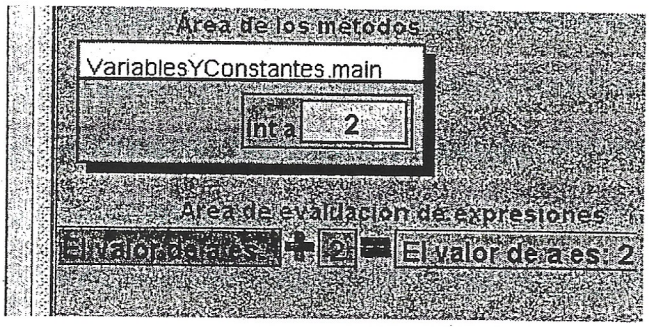
```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```



6 Muestra un letrero y el valor de "a".

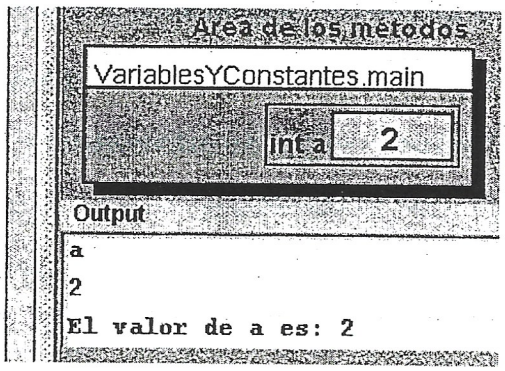
6.1 Realiza la expresión ("El valor de a es: " + a), que pega las dos expresiones como si fueran texto. El resultado es "El valor de a es: 2".

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a)
        System.out.println(SALUDO);
    }
}
```



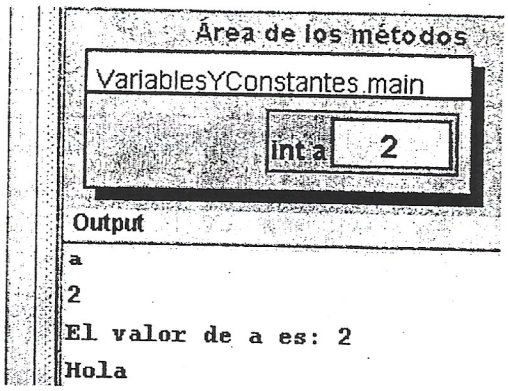
6.2 Muestra en salida "El valor de a es: 2" y realiza un salto de línea.

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```



7 Muestra el valor de la constante "SALUDO", o sea "Hola", y realiza un salto de línea.

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```



8 Se destruye el registro de activación para man y termina la ejecución del programa.

```
public class VariablesYConstantes {
    public static final String SALUDO = "Hola";
    public static void main() {
        int a = 2;
        System.out.println("a");
        System.out.println(a);
        System.out.println("El valor de a es: " + a);
        System.out.println(SALUDO);
    }
}
```

1.2.3. Comentarios.

Los nombres para clases, variables y métodos se conocen como **identificadores** y deben seguir las siguientes reglas:

1. Empiezan con un caracter alfabético, como a, b, etc.
2. Continúan con caracteres alfanuméricos, como son 0, 1,2 o a, b, c, etc.
3. No son ninguna de las siguientes palabras reservadas por el lenguaje: abstract, continue, for, new, switch, assert, default, goto,package, synchronized, boolean, do, if,private, this, break, double, implements, protected, throw, byte, else, import, public, throws, case, enum, instanceof, return, transient, catch, extends, int, short, try, char, final, interface, static, void, class, finally, long, strictfp , volatile, const, float, native, super, while.

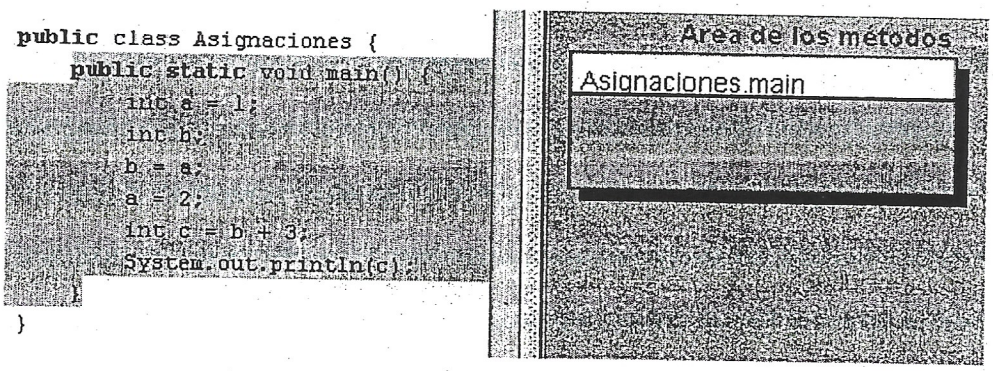
1.3. Asignaciones.

1.3.1. Código.

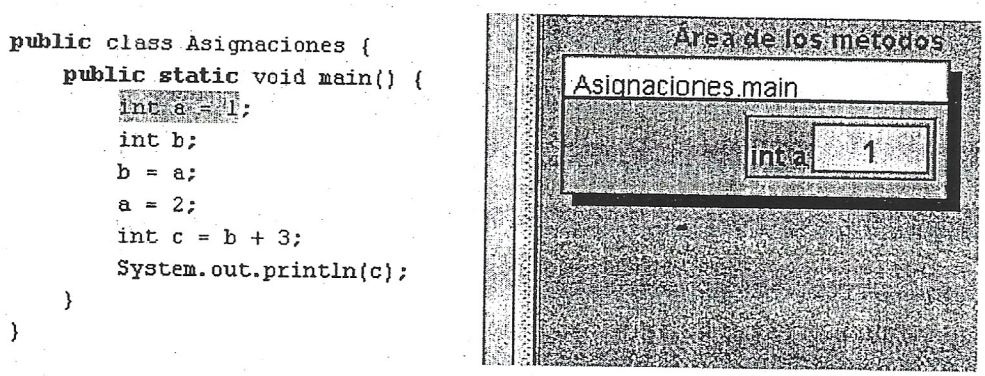
```
1  public class Asignaciones {
2      public static void main() {
3          int a = 1;
4          int b;
5          b = a;
6          a = 2;
7          int c = b + 3;
8          System.out.println(c);
9      }
10 }
```

1.3.2. Ejecución.

1 Se crea un registro de activación para el método de clase main.

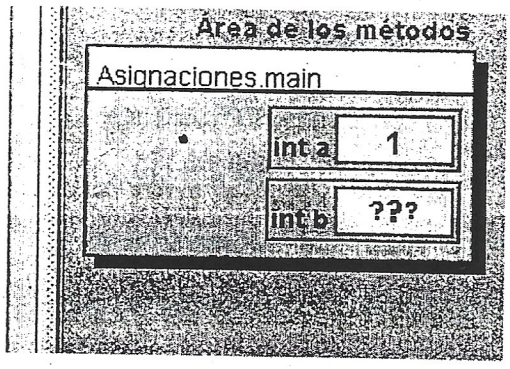


2 Se crea una variable entera llamada "a" en el registro de activación de main y se le asigna el valor "1".



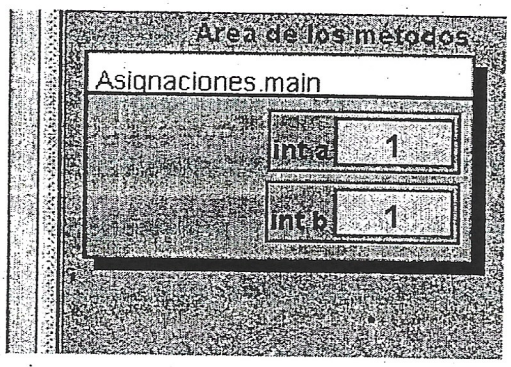
3 Se crea una variable entera llamada "b" en el registro de activación de main sin asignarle valor.

```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



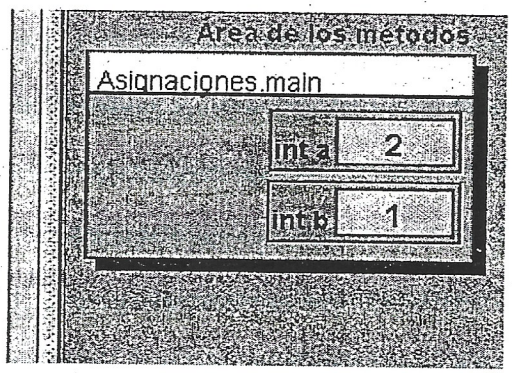
4 Asigna a "b" el valor que contiene "a", o sea 1. La variable "a" mantiene su valor y ambas se mantienen independientes una de otra.

```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



5 Asigna el valor "2" a la variable "a"; su valor anterior se descarta.

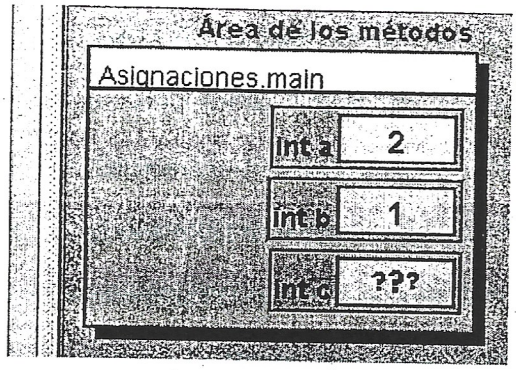
```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



6 Crea la variable "c" y le asigna valor.

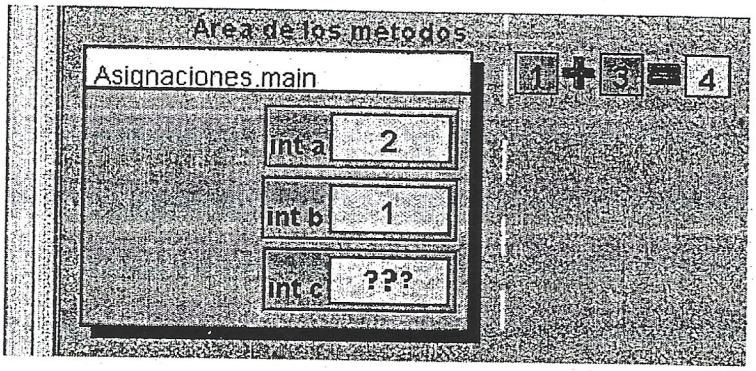
6.1 Se crea la variable "c", pero aún no le asigna valor.

```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



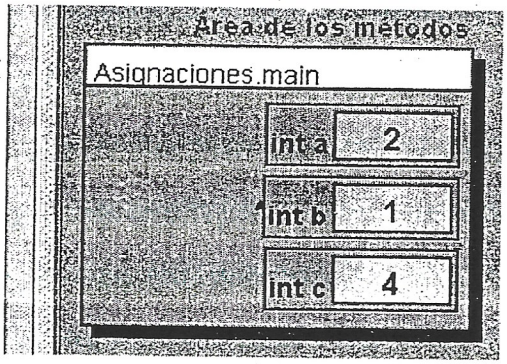
6.2 Calcula la expresión (b + 3), sustituyendo el valor actual de "b", o sea 1. Se calcula (1 + 3), obteniéndose "4".

```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



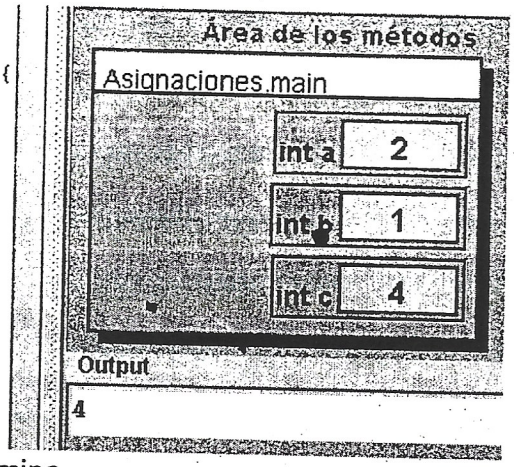
6.3 Asigna el valor obtenido (4) a la variable "c".

```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



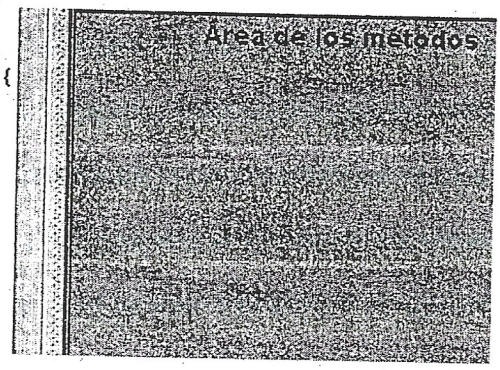
7 Muestra en la salida el valor de la variable "c" y realiza un salto de línea.

```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



8 Elimina el registro de activación y termina.

```
public class Asignaciones {
    public static void main() {
        int a = 1;
        int b;
        b = a;
        a = 2;
        int c = b + 3;
        System.out.println(c);
    }
}
```



1.4. Lectura de Datos desde el Teclado.

1.4.1. Código.

```
1  import jeliot.io.*; // Output e Input están en este paquete
2
3  public class LecturaDeDatos {
4      public static void main(String[] args) {
5          // Lee información del teclado.
6          String s = Input.readString(); // texto
7          char c = Input.readChar(); // un caracter
8          int i = Input.readInt(); // un número entero
9          double d = Input.readDouble(); // un número de punto flotante
10         System.out.println(s);
11         // Jeliot también puede mostrar así:
12         Output.println(c);
13     }
14 }
```

2.

Operadores.

2. Operadores.

2.1. Enteros.

2.1.1. Código.

```
 1 public class Enteros {
 2     // 1. Se crea un registro de activación.
 3     public static void main() {
 4         // 2. Crea la variable a y le asigna 3.
 5         int a = 3;
 6         // 3. Crea la variable b y le asigna 71.
 7         int b = 71;
 8         // 4. Muestra el resultado de una suma.
 9         // 4.1. Se calcula la expresión (b + a).
10         // 4.2. Al sustituir a y b se calcula (71 + 3), obteniendo 74.
11         // 4.3. Concatena "b + a = " con 74, obteniendo "b + a = 74".
12         // 4.4. Despliega "b + a = 74".
13         System.out.println("b + a = " + (b + a));
14         // 5. Muestra el resultado de una resta.
15         // 5.1. Se calcula la expresión (b - a).
16         // 5.2. Al sustituir a y b se calcula (71 - 3), obteniendo 68.
17         // 5.3. Concatena "b - a = " con 68, obteniendo "b - a = 68".
18         // 5.4. Despliega "b - a = 68".
19         System.out.println("b - a = " + (b - a));
20         // 6. Muestra el resultado de una multiplicación.
21         // 6.1. Se calcula la expresión (b * a).
22         // 6.2. Al sustituir a y b se calcula  (71 * 3), obteniendo 213,
23         // 6.3. Concatena "b * a = " con 213, obteniendo "b + a = 213".
24         // 6.4. Despliega "b * a =  213".
25         System.out.println("b * a = " + (b * a));
26         // 7. Muestra el cociente de una división.
27         // 7.1. Se calcula la expresión (b / a).
28         // 7.2. Al sustituir a y b se calcula (71 / 3), obteniendo 23,
29         //      que es el cociente entero de dividir 71 entre 3.
30         // 7.3. Concatena "b / a = " con 23, obteniendo "b / a = 23".
31         // 7.4. Despliega "b / a = 23".
32         System.out.println("b / a = " + (b / a));
33         // 8. Muestra el resultado de calcular un residuo.
34         // 8.1. Se calcula la expresión (b % a).
35         // 8.2. Al sustituir a y b se calcula (71 % 3), obteniendo 2,
36         //      que es el residuo de dividir 71 entre 3.
36         // 8.3. Concatena "b % a = " con 2, obteniendo "b % a =  2".
38         // 8.4. Despliega "b % a =  2".
39         System.out.println("b % a = " + (b % a));
40         // 9. Muestra el resultado de calcular un preincremento.
41         // 9.1. Se calcula la expresión (++a). Se incrementa a y se toma
42         //      el nuevo valor, o sea 4.
43         // 9.2. Concatena "++a = " con 4, obteniendo "++a = 4".
44         // 9.3. Despliega "++a = 4".
```

```
45          System.out.println("++a = " + (++a));
46          // 10. Muestra el resultado posterior al preincremento.
47          // 10.1. Concatena "a = " con el valor de a que es 4, obteniendo
48          //       "a = 4".
49          // 10.2. Despliega "a = 4".
50          System.out.println("a = " + a);
51          // 11. Muestra el resultado de calcular un postincremento.
52          // 11.1. Se calcula la expresión (a++). Se incrementa a y se toma
53          //       el valor previo al incremento, o sea 4.
54          // 11.2. Concatena "a++ = " con 4, obteniendo "a++ = 4".
55          // 11.3. Despliega "a++ = 4".
56          System.out.println("a++ = " + (a++));
57          // 12. Muestra el resultado posterior al postincremento.
58          // 12.1. Concatena "a = " con el valor de a que es 5, obteniendo
59          //       "a = 5".
60          // 12.2. Despliega "a = 5".
61          System.out.println("a = " + a);
62          // 13. Muestra el resultado de calcular un predecremento.
63          // 13.1. Se calcula la expresión (--a). Se decrementa a y se toma
64          //       el nuevo valor, o sea 4.
65          // 13.2. Concatena "--a = " con 4, obteniendo "--a = 4".
66          // 13.3. Despliega "--a = 4".
67          System.out.println("--a = " + (--a));
68          // 14. Muestra el resultado posterior al predecremento.
69          // 14.1. Concatena "a = " con el valor de a que es 4, obteniendo
70          //       "a = 4".
71          // 14.2. Despliega "a = 4".
72          System.out.println("a = " + a);
73          // 15. Muestra el resultado de calcular un postdecremento.
74          // 15.1. Se calcula la expresión (a--). Se decrementa a y se toma
75          //       el valor previo al decremento, o sea 4.
76          // 15.2. Concatena "a-- = " con 4, obteniendo "a-- = 4".
77          // 15.3. Despliega "a-- = 4".
78          System.out.println("a-- = " + (a--));
79          // 16. Muestra el resultado posterior al postdecremento.
80          // 16.1. Concatena "a = " con el valor de a que es 3, obteniendo
81          //       "a = 3".
82          // 16.2. Despliega "a = 3".
83          System.out.println("a = " + a);
84          // 17. Muestra el resultado de cambiarle el signo a un número.
85          // 17.1. Calcula la expresión (-a), que da -3.
86          // 17.2. Concatena "-a = " con -3, obteniendo "-a = -3".
87          // 17.3. Despliega "-a = -3".
88          System.out.println("-a = " + (-a));
89          // 18. Se destruye el registro de  activación para main y
90          //    termina la  ejecución del programa.
91      }
92  }
```

2.1.2. Comentarios.

La operación "+a" convierte un dato de tipo char, byte o short en int. (Ver tabla más adelante.)

Están también los operadores de asignación del tipo **a += b**, que son equivalentes a tener **a = a + b**. Se tienen por el mismo estilo los operadores -=, *=, /= y %=.

2.2. Flotantes.

2.2.1. Código.

```
1  public class Flotantes {
2      // 1. Se crea un registro de activación.
3      public static void main() {
4          // 2. Crea la variable a y le asigna 3.2.
5          double a = 3.2;
6          // 3. Crea la variable b y le asigna 71.0.
7          double b = 7.1e1;
8          // 4. Muestra el resultado de una suma.
9          // 4.1. Se calcula la expresión (b + a).
10         // 4.2. Al sustituir a y b se calcula (71.0 + 3.2), obteniendo 74.2.
11         // 4.3. Concatena "b + a = " con 74.2, obteniendo "b + a = 74.2".
12         // 4.4. Despliega "b + a = 74.2".
13         System.out.println("b + a = " + (b + a));
14         // 5. Muestra el resultado de una resta.
15         // 5.1. Se calcula la expresión (b - a).
16         // 5.2. Al sustituir a y b se calcula (71.0 - 3.2), obteniendo 68.8.
17         // 5.3. Concatena "b - a = " con 68.8, obteniendo "b - a = 68.8".
18         // 5.4. Despliega "b - a = 68.8".
19         System.out.println("b - a = " + (b - a));
20         // 6. Muestra el resultado de una multiplicación.
21         // 6.1. Se calcula la expresión (b * a).
22         // 6.2. Al sustituir a y b se calcula  (71.0 * 3.2), obteniendo
23         //      227.20000000000002, que es el resultado de multiplicar
24         //      71.0 * 3.2.
25         // 6.3. Concatena "b * a = " con 227.2, obteniendo
26         //      "b * a =  227.20000000000002".
27         // 6.4. Despliega "b * a =  227.20000000000002".
28         System.out.println("b * a = " + (b * a));
29         // 7. Muestra el resultado de una división.
30         // 7.1. Se calcula la expresión (b / a).
31         // 7.2. Al sustituir a y b se calcula (71.0 / 3.2), obteniendo
32         //      22.1875, que es el resultado de dividir 71.0 entre 3.2.
33         // 7.3. Concatena "b / a = " con 22.1875, obteniendo
34         //      "b / a = 22.1875".
35         // 7.4. Despliega "b / a = 22.1875".
36         System.out.println("b / a = " + (b / a));
36         // 8. Muestra el resultado de calcular un residuo.
38         // 8.1. Se calcula la expresión (b % a).
```

```
39          // 8.2. Al sustituir a y b se calcula (71.0 % 3.2), obteniendo
40          //      0.5999999999999961, que es el residuo de dividir 71.0
41          //      entre 3.2.
42          // 8.3. Concatena "b % a = " con 0.5999999999999961, obteniendo
43          //      "b % a =  0.5999999999999961".
44          // 8.4. Despliega "b % a =  0.5999999999999961".
45          System.out.println("b % a = " + (b % a));
46          // 9. Se destruye el registro de  activación para main y
47          //    termina la  ejecución del programa.
48      }
49  }
```

2.2.2. Comentarios.

El tipo double también soporta los operadores de incremento(++), decremento(--), conversión(+) , cambio de signo(-) y asignación (-=, *=, /= y %=).

2.3. Caracteres.

2.3.1. Código.

```
1   public class Caracteres {
2       // 1. Se crea un registro de activación.
3       public static void main() {
4           // 2. Crea la variable a y le asigna el caracter 'y'.
5           char a = 'y';
6           // 3. Crea la variable b y le asigna el caracter '4'.
7           char b = '4';
8           // 4. Crea la variable c y le asigna el caracter '&'.
9           char c = '&';
10          // 5. Crea la variable d y le asigna un espacio en blanco.
11          char d = ' ';
12          // Todas las operaciones aritméticas toman el código Unicode del
13          // caracter para operar, excepto ++ y -- que toman el caracter.
14          // 6. Muestra el valor de la variable a.
15          // 6.1. Se calcula la expresión "a = " + a, obteniendo "a = y".
16          // 6.2. Despliega "a = y".
17          //      variable a.
18          System.out.println("a = " + a);
19          // 7. Muestra el valor entero de la variable a.
20          // 7.1. Se calcula la expresión (int)a, es decir se toma el
21          //      valor de a y se interpreta como si fuera su código
22          //      Unicode, o sea 121.
23          // 7.2. Se calcula la expresión "a = " + 121, obteniendo "a = 121".
24          // 7.3. Despliega "a = 121".
25          System.out.println("a = " + (int)a);
26          // 7. Muestra el resultado de sumar dos caracteres.
27          // 7.1. Se calcula la expresión (a + b), tomando los valores Unicode
28          //      para a(121) y b(52), obteniéndose 173.
29          // 7.2. Se calcula la expresión "a + b = " + 173, obteniendo
30          //      "a + b = 173".
31          // 7.3. Despliega "a + b = 173".
```

```
32          System.out.println("a + b = " + (a + b));
33      }
34  }
```

2.3.2. Comentarios.

La expresión **(int)a** que sirve para interpretar una expresión como si tuviera un tipo de datos diferente se conoce como **cast** o, en Español, conversión.

2.4. Cadenas.

2.4.1. Código.

```
1  public class Cadenas {
2      public static void main() {
3          String base = "Cadena base";
4          // Pega las cadenas "prefijo" y base.
5          System.out.println("prefijo" + base);//prefijoCadena base
6          // Obtiene una subcadena que va desde el caracter 2 hasta el 5.
7          System.out.println("Cadena base".substring(2,6)); //dena
8          // Regresa verdadero si las dos cadenas contienen el mismo texto.
9          System.out.println("Cadena base".equals("Cadena base")); //true
10         System.out.println("Cadena base".equals("inicio")); //false
11         // Verdadero si la cadena termina con "ase".
12         System.out.println("Cadena base".endsWith("ase")); //true
13         // Verdadero si la cadena comienza con ase.
14         System.out.println("Cadena base".startsWith("Cad")); //true
15         // Regresa el caracter en la posición 7; la primera posición es la 0.
16         System.out.println("Cadena base".charAt(7)); //b
17         // Es como "equals", pero no hace distinción de mayúsculas y
18         // minúsculas.
19         System.out.println("tela".equalsIgnoreCase("TELA")); //true
20         // Busca el caracter 'a' desde el inicio de la cadena y regresa la
21         // posición donde lo encontró. Si no aparece, regresa un valor
22         // negativo.
23         System.out.println("Cadena base".indexOf('a')); //1
24         // Igual que el anterior, pero busca a partir de la posición 3.
25         System.out.println("Cadena base".indexOf('a', 3)); //5
26         // Busca subcadenas. También hay una variante que busca a partir de
27         // cierta posición.
28         System.out.println("Cadena base".indexOf("de")); //2
29         // Busca desde el final. Tiene las mismas variantes que "indexOf".
30         System.out.println("Cadena base".lastIndexOf('a')); //8
31         // Regresa el número de posiciones que ocupa la cadena; o sea, su
32         // longitud.
33         System.out.println("Cadena base".length()); //11
34         // Convierte todas las letras de la cadena en su equivalente en
35         // minúscula. El resto de los caracteres no se modifican.
36         System.out.println("Cadena base".toLowerCase()); //cadena base
37         // Convierte todas las letras de la cadena en su equivalente en
38         // mayúscula. El resto de los caracteres no se modifican.
39         System.out.println("Cadena base".toUpperCase());//CADENA BASE
```

```
40          System.out.println(" espacios " + "hola"); // espacios hola
41          // Remueve los espacios al inicio y al final de una cadena.
42          System.out.println(" espacios ".trim() + "hola");//espacioshola
43      }
44  }
```

2.5. Relacionales.

2.5.1. Código.

```
1  public class Relacionales {
2      public static void main() {
3          System.out.println("3 < 7 = " + (3 < 7)); //true
4          System.out.println("3 <= 7 = " + (3 <= 7)); //true
5          System.out.println("3 > 7 = " + (3 > 7)); //false
6          System.out.println("3 >= 7 = " + (3 >= 7)); //false
7          System.out.println("3 == 7 = " + (3 == 7)); //false
8          System.out.println("3 != 7 = " + (3 != 7)); //true
9
10         System.out.println("7 < 3 = " + (7 < 3)); //false
11         System.out.println("7 <= 3 = " + (7 <= 3)); //false
12         System.out.println("7 > 3 = " + (7 > 3)); //true
13         System.out.println("7 >= 3 = " + (7 >= 3)); //true
14         System.out.println("7 == 3 = " + (7 == 3)); //false
15         System.out.println("7 != 3 = " + (7 != 3)); //true
16
17         System.out.println("7 < 7 = " + (7 < 7)); //false
18         System.out.println("7 <= 7 = " + (7 <= 7)); //true
19         System.out.println("7 > 7 = " + (7 > 7)); //false
20         System.out.println("7 >= 7 = " + (7 >= 7)); //true
21         System.out.println("7 == 7 = " + (7 == 7)); //true
22         System.out.println("7 != 7 = " + (7 != 7)); //false
23     }
24 }
```

2.5.2. Nombre de los Operadores.

Operador	*Nombre*
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual que
!=	Diferente de

2.6. Lógicos.

2.6.1. Código.

```
1  public class Logicos {
2      public static void main() {
3          boolean a = true;
4          boolean b = false;
5
6          System.out.println("!true = " + (!a)); //false
7          System.out.println("!false = " + (!b)); //false
8
9          System.out.println("true && true = " + (true && true)); //true
10         System.out.println("true && false = " + (true && false)); //false
11         System.out.println("false && true = " + (false && true)); //false
12         System.out.println("false && false = " + (false && false)); //false
13
14         System.out.println("true || true = " + (true || true)); //true
15         System.out.println("true || false = " + (true || false)); //true
16         System.out.println("false || true = " + (false || true)); //true
17         System.out.println("false || false = " + (false || false)); //false
18
19         System.out.println("true ^ true = " + (true ^ true)); //false
20         System.out.println("true ^ false = " + (true ^ false)); //true
21         System.out.println("false ^ true = " + (false ^ true)); //true
22         System.out.println("false ^ false = " + (false ^ false)); //false
23     }
24 }
```

2.6.2. Nombre de los Operadores.

Operador	*Nombre*
!	No
&&, &	Y
||, |	O
^	O exclusiva

2.6.3. Comentarios.

Los operadores && y || evalúan solo cuando es necesario, es decir que siempre evalúan el primer operando, pero.

- && solo evalúa el segundo operando cuando el primero es verdadero.
- || solo evalúa el segundo operando cuando el primero es falso.

Por su parte los operadores & y | son equivalentes a los anteriores, pero siempre evalúan ambos operandos. El operador ^ también evalúa los dos operandos. Tenemos también los operadores de asignación &=, |= y ^=, que siguen la misma lógica indicada en 2.1.2.

2.7. Operadores de Bits.

2.7.1. Código.

```
1  public class Bits {
2      public static void main() {
3          // Todas estas operaciones operan sobre la representación binaria
4          // de los números.
5          int a = 0; // bits: 32 ceros
6          int b = -1; // bits: 32 unos
7
8          System.out.println("~0 = " + (~a)); //-1
9          System.out.println("~-1 = " + (~b)); //0
10
11         System.out.println("-1 & -1 = " + (-1 & -1)); //-1
12         System.out.println("-1 & 0 = " + (-1 & 0)); //0
13         System.out.println("0 & -1 = " + (0 & -1)); //0
14         System.out.println("0 & 0 = " + (0 & 0)); //0
15
16         System.out.println("-1 | -1 = " + (-1 | -1)); //-1
17         System.out.println("-1 | 0 = " + (-1 | 0)); //-1
18         System.out.println("0 | -1 = " + (0 | -1)); //-1
19         System.out.println("0 | 0 = " + (0 | 0)); //0
20
21         System.out.println("-1 ^ -1 = " + (-1 ^ -1)); //0
22         System.out.println("-1 ^ 0 = " + (-1 ^ 0)); //-1
23         System.out.println("0 ^ -1 = " + (0 ^ -1)); //-1
24         System.out.println("0 ^ 0 = " + (0 ^ 0)); //0
25
26         // Corrimientos. Toma bits 1:1, 2:10, 4: 100
27         System.out.println("1 << 1 = " + (1 << 1)); //2, bits 10
28         System.out.println("2 << 2 = " + (1 << 2)); //4, bits 100
29         System.out.println("4 >> 1 = " + (4 >> 1)); //2 llena con 0
30         System.out.println("4 >> 2 = " + (4 >> 2)); //1 llena con 0
31         System.out.println("-1 >> 1 = " + (-1 >> 31)); //-1, llena con 1
32         System.out.println("4 >>> 1 = " + (4 >> 1)); //2 llena con 0
33         System.out.println("4 >>> 2 = " + (4 >> 2)); //1 llena con 0
34         System.out.println("-1 >>> 2 = " + (-1 >>> 31)); //1, llena con 0
35     }
36 }
```

2.7.2. Nombre de los Operadores.

Operador	*Nombre*
~	No bit a bit
&	Y bit a bit
|	O bit a bit
^	O exclusiva bit a bit
<<	Corrimiento de bits a la izquierda

Operador	*Nombre*
>>	Corrimiento a la derecha llenando con el bit del signo
>>>	Corrimiento a la derecha llenando con 0

2.7.3. Comentarios.

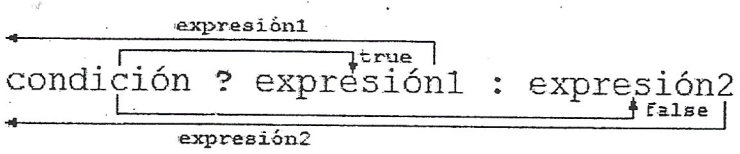
Tenemos también los operadores de asignación &=, |=, ^=, <<=, >>= y >>>= que siguen la misma lógica indicada en 2.1.2.

2.8. Operador Condicional.

2.8.1. Código.

```
1  public class Condicionales {
2      // 1. Se crea un registro de activación para main.
3      public static void main() {
4          // 2. Se crea la variable calificación de tipo double, con el
5          //    valor 3.0.
6          double calificación = 8.0;
7          // 3. Crea la variable mensaje. Evalúa la condición
8          //    (calificación >= 7.0); como es verdadera, asigna "Aprobaste".
9          String mensaje = (calificación >= 7.0) ? "Aprobaste" : "Reprobaste";
10         // 4. cambia el valor de calificación.
11         calificación = 3.0;
12         // 5. Crea la variable mensaje2. Evalúa la condición
13         //    (calificación >= 7.0); como es false, asigna "Reprobaste".
14         String mensaje2 = (calificación >= 7.0) ? "Aprobaste" : "Reprobaste";
15         // 6. Muestra el valor de mensaje, que es "Aprobaste".
16         System.out.println(mensaje);
17         // 7. Muestra el valor de mensaje2, que es "Reprobaste".
18         System.out.println(mensaje2);
19         // 8. Elimina el registro de activación y termina.
20     }
21 }
```

2.8.2. Resumen del Operador Condicional.



2.9. Tipos de Datos Básicos de Java.

	Mínimo Valor Negativo	*Máximo Valor Negativo*	*Mínimo Valor Positivo*	*Máximo Valor Positivo*
byte	-128	-1	0	127
short	-32,768	-1	0	32,767
int	2,147,483,648	-1	0	2,147,483,647
long	-9,223,372,036,854,755,808L	-1	0	9,223,372,036,854,755,807L
float	-3.4028234663852886E38f	-1.401298464324817E-45f	1.401298464324817E-45f	3.4028234663852886E38f
double	-1.7976931348623157E308	-4.94065645841246544E-324	4.94065645841246544E-324	1.7976931348623157E308
char			0	65535
boolean			false	true

2.10. Precedencia de Operadores.

Los operadores que aparecen en los primeros renglones se evalúan antes que los mostrados en los últimos renglones.

Tipo	*Operadores*
Posfijos	arreglo[índice] ref.miembro método(parámetros) expresión++ expresión--
Unarios	++expresión --expresión +expresión -expresión ~expresión !expresión
Creación y cast	new Clase(parámetros) (tipo)expresión
Multiplicativos	expresión * expresión expresión % expresión expresión / expresión
Aditivos	expresión + expresión expresión - expresión
Corrimientos	expresión << expresión expresión >> expresión expresión >>> expresión
Relacionales	expresión < expresión expresión > expresión expresión <= expresión expresión >= expresión referencia instanceof expresión
De igualdad	expresión == expresión expresión != expresión
Y completa	expresión & expresión
O exclusiva	expresión ^ expresión
O completa	expresión | expresión
Y rápida	expresión && expresión

Tipo	*Operadores*
O rápida	expresión || expresión
Condicional	expresión ? expresión : expresión
Asignación	expresión = expresión expresión += expresión expresión -= expresión expresión *= expresión expresión /= expresión expresión %= expresión expresión &= expresión expresión ^=expresión expresión |= expresión expresión <<= expresión expresión >>= expresión expresión >>>= expresión

3.

Estructuras de Control.

3. Estructuras de Control.

3.1. IF.

3.1.1. If Simple True.

```
 1 public class IfSimpleTrue {
 2     // 1. Se crea un registro de activación para main.
 3     public static void main() {
 4         // 2. Se crea la variable calificación de tipo double, con el
 5         //    valor 7.0.
 6         double calificación = 7.0;
 7         // 3. Se analiza en el área para expresiones la operación
 8         //    (calificación >= 7.0), obteniéndose true y por lo tanto
 9         //    realiza el bloque de instrucciones asignado al if.
10         if (calificación >= 7.0) {
11             // 4. Muestra en la salida el mensaje
12             //    "Felicidades. Aprobaste." sin las comillas y
13             //    realiza un salto de línea.
14             System.out.println("FELICIDADES K. Aprobaste.");
15         }
16         // 5. Muestra un mensaje de despedida.
17         System.out.println("Adios.");
18         // 6. Se destruye el registro de activación para main y
19         //    termina la ejecución del programa.
20     }
21 }
```

3.1.2. If Simple False.

```
 1 public class IfSimpleFalse {
 2     // 1. Se crea un registro de activación para main.
 3     public static void main() {
 4         // 2. Se crea la variable calificación de tipo double, con el
 5         //    valor 3.0.
 6         double calificación = 3.0;
 7         // 3. Se analiza en el área para expresiones la operación
 8         //    (calificación == 7.0), obteniéndose false y por lo tanto
 9         //    se salta la ejecución del bloque de instrucciones
10         //    asignado al if.
11         if (calificación >= 7.0) {
12             System.out.println("FELICIDADES K. Aprobaste.");
13         }
14         // 4. Muestra un mensaje de despedida.
15         System.out.println("Adios.");
16         // 5. Se destruye el registro de activación para main y
17         //    termina la ejecución del programa.
18     }
19 }
```

3.1.3. If Simple.

```
1  import jeliot.io.*;
2
3  public class IfSimple {
4      public static void main() {
5          // Se solicita al usuario un valor para la calificación.
6          double calificación = Input.readDouble();
7          // Si el valor de calificacion hace que la condición
8          // sea verdadera, se hace el bloque del if, pero si es
9          // falsa se salta el bloque.
10         if (calificación >= 7.0) {
11             // Esta sección solo se ejecuta si la condición es
12             // verdadera
13             System.out.println("FELICIDADES K. Aprobaste.");
14         }
15         // Esto se hace siempre porque está fuera del if.
16         System.out.println("Adios.");
17     }
18 }
```

3.1.4. If Else True.

```
1  public class IfElseTrue {
2      // 1. Se crea un registro de activación para main.
3      public static void main() {
4          // 2. Se crea la variable calificación de tipo double, con el
5          //    valor 8.0.
6          double calificación = 8.0;
7          // 3. Se analiza en el área para expresiones la operación
8          //    (calificación >= 7.0), obteniéndose true y por lo tanto
9          //    ejecuta el bloque de instrucciones asignado al if.
10         if (calificación >= 7.0) {
11             // 4. Muestra en la salida el mensaje
12             //    "Felicidades. Aprobaste." sin las comillas y
13             //    realiza un salto de línea. Se salta la sección else.
14             System.out.println("FELICIDADES K. Aprobaste.");
15         } else {
16             System.out.println("No me ames. Reprobaste.");
17         }
18         // 5. Muestra un mensaje de despedida.
19         System.out.println("Adios.");
20         // 6. Se saltó el else y su bloque porque la expresión del if
21         //    fué verdadera. Se destruye el registro de activación para
22         //    main y termina la ejecución del programa.
23     }
24 }
```

3.1.5. If Else False.

```
1  public class IfElseFalse {
2      // 1. Se crea un registro de activación para main.
3      public static void main() {
4          // 2. Se crea la variable calificación de tipo double, con el
5          //    valor 3.0.
6          double calificación = 3.0;
7          // 3. Se analiza en el área para expresiones la operación
8          //    (calificación >= 7.0), obteniéndose false y por lo tanto
9          //    se salta la ejecución del bloque de instrucciones
10         //    asignado al if y ejecuta la sección else.
11         if (calificación >= 7.0) {
12             System.out.println("FELICIDADES K. Aprobaste.");
13         } else {
14             // 4. Muestra en la salida el mensaje
15             //    "No me ames. Reprobaste." sin las comillas y
16             //    realiza un salto de línea.
17             System.out.println("No me ames. Reprobaste.");
18         }
19         // 5. Muestra un mensaje de despedida.
20         System.out.println("Adios.");
21         // 6. Se destruye el registro de activación para
22         //    main y termina la ejecución del programa.
23     }
24 }
```

3.1.6. If Else.

```
1  import jeliot.io.*;
2
3  public class IfElse {
4      public static void main() {
5          // Solicita al usuario una calificación.
6          double calificación = Input.readDouble();
7          // Si la condición es verdadera, realiza las instrucciones
8          // asociadas al if si es falsa, realiza las instrucciones
9          // de else.
10         if (calificación >= 7.0) {
11             // Las instrucciones de esta sección solo se ejecutan
12             // cuando la condición es verdadera.
13             System.out.println("FELICIDADES K. Aprobaste.");
14         } else {
15             // Las instrucciones de esta sección solo se ejecutan
16             // cuando la condición es falsa.
17             System.out.println("No me ames. Reprobaste.");
18         }
19         // Esto se hace siempre porque está fuera del if y del else.
20         System.out.println("Adios.");
21     }
22 }
```

3.1.7. Decisión Múltiple.

```
 1 public class DecisionMultiple {
 2     // 1. Se crea un registro de activación para main.
 3     public static void main() {
 4         // 2. Se crea la variable calificación de tipo double, con el
 5         //    valor 7.0.
 6         double calificación = 8.5;
 7         // 3. Se analiza en el área para expresiónes, obteniéndose
 8         //    false y por lo tanto se salta las instrucciones y
 9         //    continúa donde está el else.
10         //    realiza el bloque de instrucciones asignado al if.
11         if ((calificación >= 9.0) && (calificación <= 10.0)) {
12             System.out.println("EXCELENTE. Eres mi heroe.");
13         // 4. Analiza la siguiente condición y como es verdadera
14         //    ejecuta las condiciones de este if.
15         } else if ((calificación >= 8.0) && (calificación < 9.0)) {
16             // 5. Despliega el mensaje "FELICIDADES. Aprobaste.".
17             System.out.println("FELICIDADES. Aprobaste.");
18         } else if ((calificación >= 7.0) && (calificación < 8.0)) {
19             System.out.println("PANZASTE. Límpiate esa barriga.");
20         } else if ((calificación >= 0.0) && (calificación < 7.0)) {
21             System.out.println("No me ames. Reprobaste.");
22         } else {
23             System.out.println("A ver si ya dejas de estar jugando.");
24             System.out.println("Introduce una calificación de 0 a 10.");
25         }
26         // 6. Muestra un mensaje de despedida.
27         System.out.println("Ahí la vemos.");
28         // 7. Se destruye el registro de activación para main y
29         //    termina la ejecución del programa.
30     }
31 }
```

3.1.8. Otra Decisión Múltiple.

```
 1 import jeliot.io.*;
 2
 3 public class OtraDecisionMultiple {
 4     // 1. Se crea un registro de activación para main.
 5     public static void main() {
 6         // Solicita una calificación.
 7         double calificación = Input.readDouble();
 8         // Empieza revisando la condición del if. Si es verdadera
 9         // realiza las instrucciones del if. Si es false, empieza
10         // a revisar los else if, hasta encontrar la primera
11         // condición que sea verdadera y ejecuta sus instrucciones.
12         // Si todas las condiciones son falsas se ejecutan las
13         // instrucciones del else final. En algunas ocasiones no se
14         // emplea el else sin if. En ese caso, si todas las condiciones
15         // son falsas no se ejecuta ninguna instrucción del if y
16         // continúa con la instrucción que sigue a toda la estructura
17         // de decisión múltiple.
```

```
18          if ((calificación >= 9.0) && (calificación <= 10.0)) {
19              // Esta instrucción se realiza solo si la condición
20              // del if es verdadera.
21              System.out.println("EXCELENTE. Eres mi heroe.");
22              // Luego de ejecutar estas instrucciones se salta todos
23              // los otros else.
24          } else if ((calificación >= 8.0) && (calificación < 9.0)) {
25              // Esta instrucción se realiza solo si la condición
26              // del "if" es falsa y la del "else if" es verdadera.
27              System.out.println("FELICIDADES. Aprobaste.");
28              // Luego de ejecutar estas instrucciones se salta todos
29              // los otros else.
30          } else if ((calificación >= 7.0) && (calificación < 8.0)) {
31              // Esta instrucción se realiza solo si la condición
32              // del "if" es falsa, la del primer "else if" es falsa y la
33              // del segundo es verdadera.
34              System.out.println("PANZASTE. Límpiate esa barriga.");
35              // Luego de ejecutar estas instrucciones se salta todos
36              // los otros else.
37          } else if ((calificación >= 0.0) && (calificación < 7.0)) {
38              // Esta instrucción se realiza solo si la condición
39              // del "if" es falsa, las del primer y segundo "else if"
40              // son falsas y la del tercero es verdadera.
41              System.out.println("No me ames. Reprobaste.");
42              // Luego de ejecutar estas instrucciones se salta el else.
43          } else {
44              // Esta instrucción se realiza solo si todas las condiciones
45              // son falsas.
46              System.out.println("A ver si ya dejas de estar jugando.");
47              System.out.println("Introduce una calificación de 0 a 10.");
48              // Luego de ejecutar estas instrucciones continúa con la
49              // instrucción que sigue después de toda la estructura.
50          }
51          // Este mensaje siempre se muestra porque está fuera de toda la
52          // estructura de decisión múltiple.
53          System.out.println("Ahí la vemos.");
54      }
55  }
```

3.2. Switch.

```
 1  import jeliot.io.*;
 2
 3  public class Switch {
 4      public static void main() {
 5          char letra = Input.readChar();
 6          // Analiza el valor de la letra.
 7          switch (letra) {
 8          case 'a' :
 9          case 'A' :
10              // Esto solo se hace cuando letra vale 'a' o 'A'
11              System.out.println("Escogiste la a de Antenancio.");
12              // Break indica salir del switch. Si no lo pones
```

```
13              // continúa con las instrucciones del siguiente case.
14              break;
15          case 'b' :
16          case 'B' :
17              // Esto solo se hace cuando letra vale 'b' o 'B'
18              System.out.println("Ecogiste la b de Boliberto.");
19              break;
20          case 't' :
21          case 'T' :
22              // Esto solo se hace cuando letra vale 't' o 'T'
23              System.out.println("Escogiste la t de Tuilitonio.");
24              break;
25          default:
26              // Esto solo se hace cuando el valor de letra no
27              // coincide con ningún case.
28              System.out.println("Esa no me la sé.");
29              break;
30          }
31          System.out.println("Adios");
32      }
33  }
```

3.3. While.

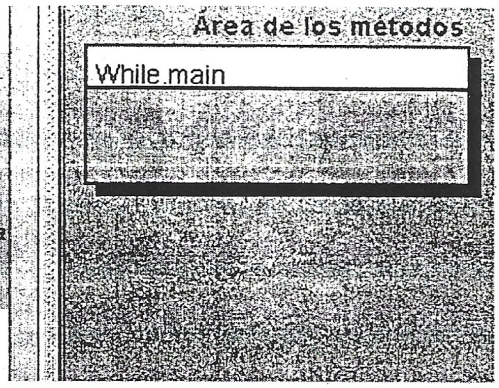
3.3.1. Código.

```
1  public class While {
2      public static void main() {
3          int a = 0;
4          while (a < 2) {
5              System.out.println(a);
6              a++;
7              // Al llegar aquí, se regresa a evaluar la condición del while.
8          }
9          System.out.println("Fin:" + a);
10     }
11 }
```

3.3.2. Ejecución.

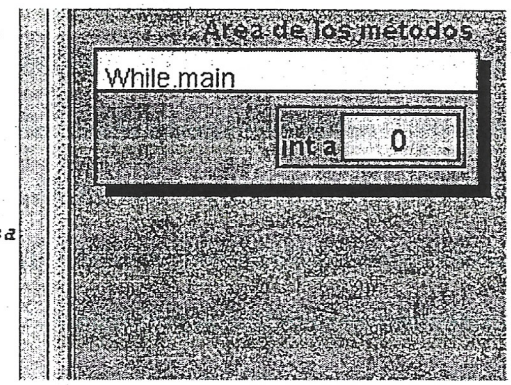
1. Crea el registro de activación para main.

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



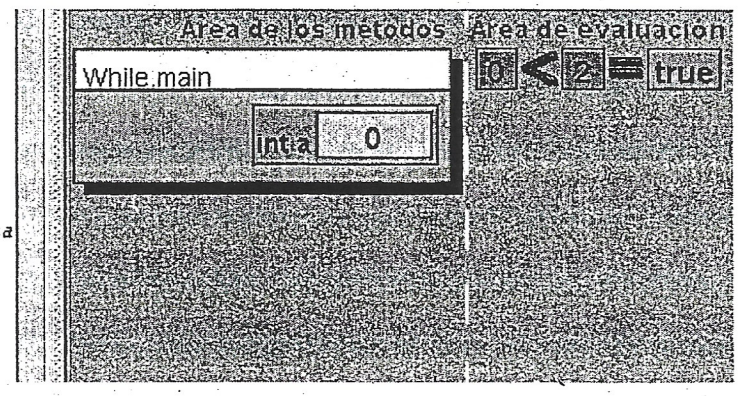
2. Se crea una variable entera llamada "a" en el registro de activación de main y se le asigna el valor "0".

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



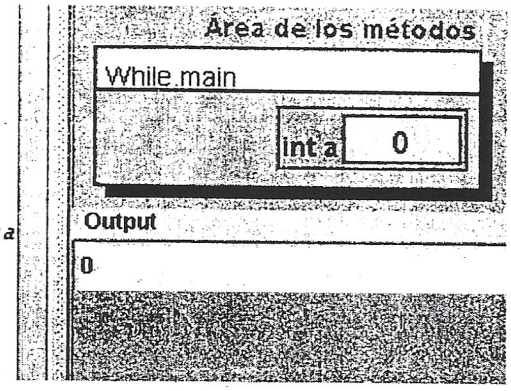
3. Se evalúa (a < 2) con el valor actual de "a"(0). Se obtiene "true" y entra al ciclo.

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



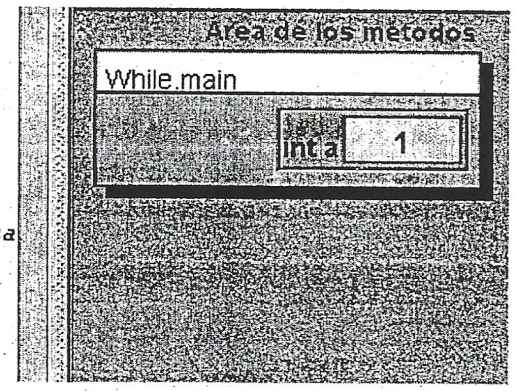
4. Muestra el valor actual de la variable "a"(0) y realiza un salto de línea.

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



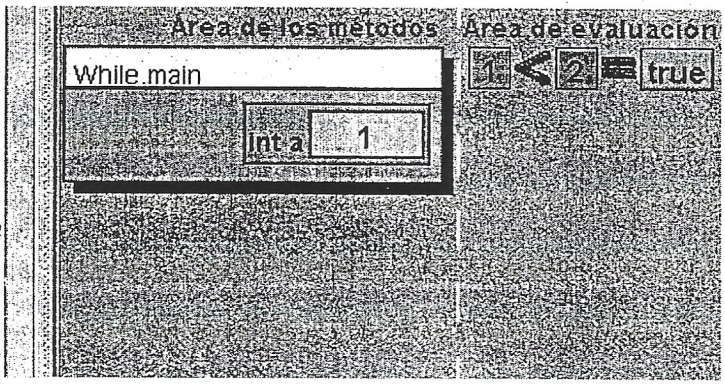
5. Incrementa el valor de "a" que ahora es "1".

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



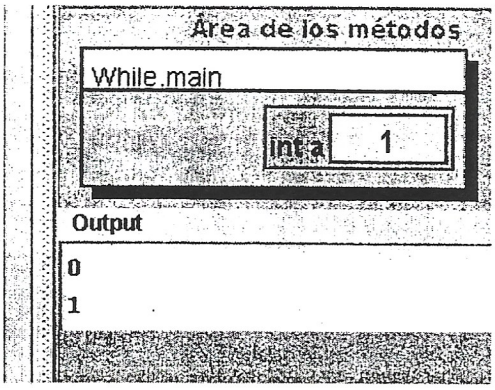
6. Se evalúa (a < 2) con el valor actual de "a"(1). Se obtiene "true" y entra al ciclo.

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



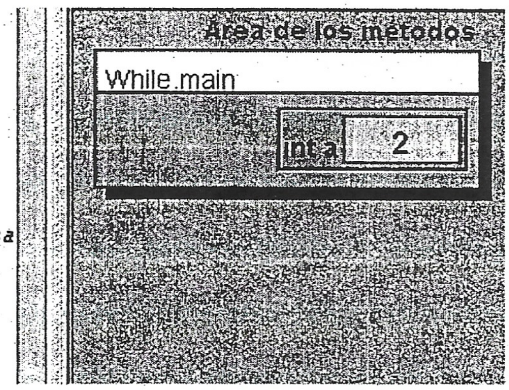
7. Muestra el valor actual de la variable "a"(1) y realiza un salto de línea.

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



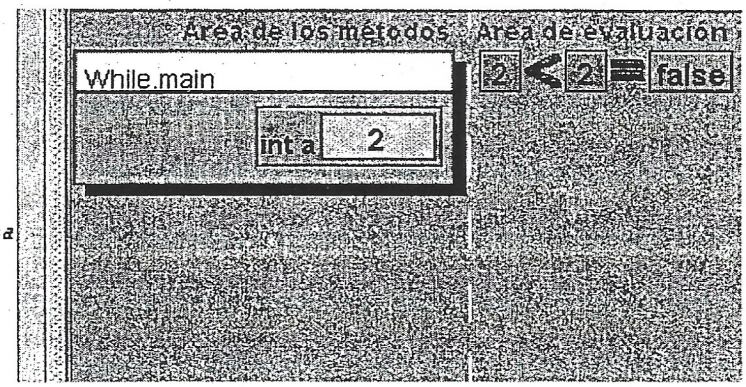
8. Incrementa el valor de "a" que ahora es "2".

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



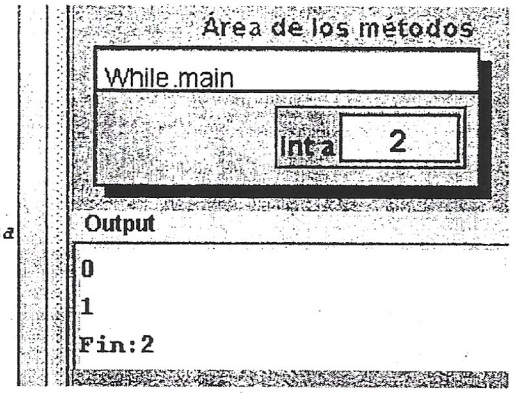
9. Se evalúa (a < 2) con el valor actual de "a"(2). Se obtiene "false" y salta la estructura del while.

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



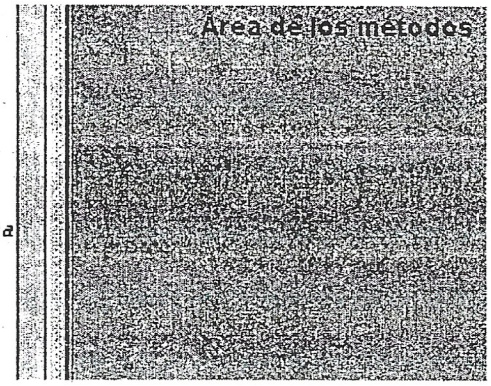
10. Muestra el resultado de evaluar ("Fin: " + a), que es "Fin: 2".

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



11. Destruye el registro de activación y termina el programa.

```
public class While {
    public static void main() {
        int a = 0;
        while (a < 2) {
            System.out.println(a);
            a++;
            // Al llegar aquí, se regresa
        }
        System.out.println("Fin:" + a);
    }
}
```



3.4. For.

3.4.1. Código.

```
1  public class For {
2      // 1. Crea el registro de activación para main.
3      public static void main() {
4          // 2. Ejecuta int a = 0. Crea la variable a en el registro
5          //    de activación y le asigna el valor 0. Esta instrucción
6          //    solo se hace una vez.
7          // 3. Evalúa (a < 2) con a = 0. Se obtiene true y entra al ciclo.
8          // 5. Ejecuta a++. Incrementa el valor de a, que ahora es 1.
9          // 6. Evalúa (a < 2) con a = 1. Se obtiene true y entra al ciclo.
10         // 8. Ejecuta a++. Incrementa el valor de a, que ahora es 2.
11         // 9. Evalúa (a < 2) con a = 2. Se obtiene false, se salta el
12         //    ciclo y destruye a.
13         for(int a = 0; a < 2; a++) {
14             // 4. Despliega el valor actual de a, o sea 0.
15             // 7. Despliega el valor actual de a, o sea 1.
```

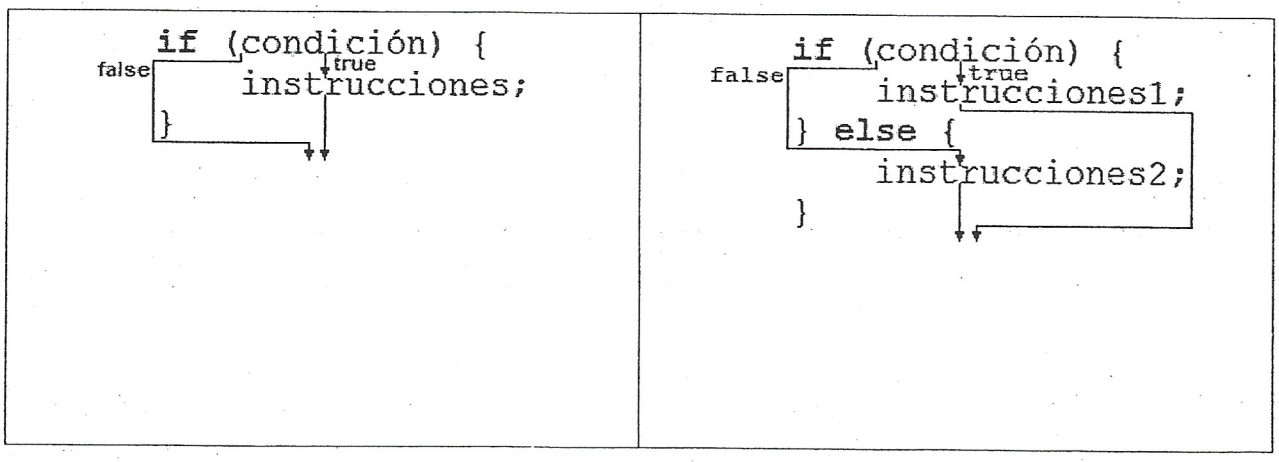
```
16              System.out.println(a);
17              // Cada vez que llega aquí, el programa se regresa
18              // a evaluar el incremento de for y luego la condición.
19          }
20          // 10. Muestra el letrero "Fin." y salta línea.
21          System.out.println("Fin.");
22          //11. Se elimina el registro de activación y termina.
23      }
24  }
```

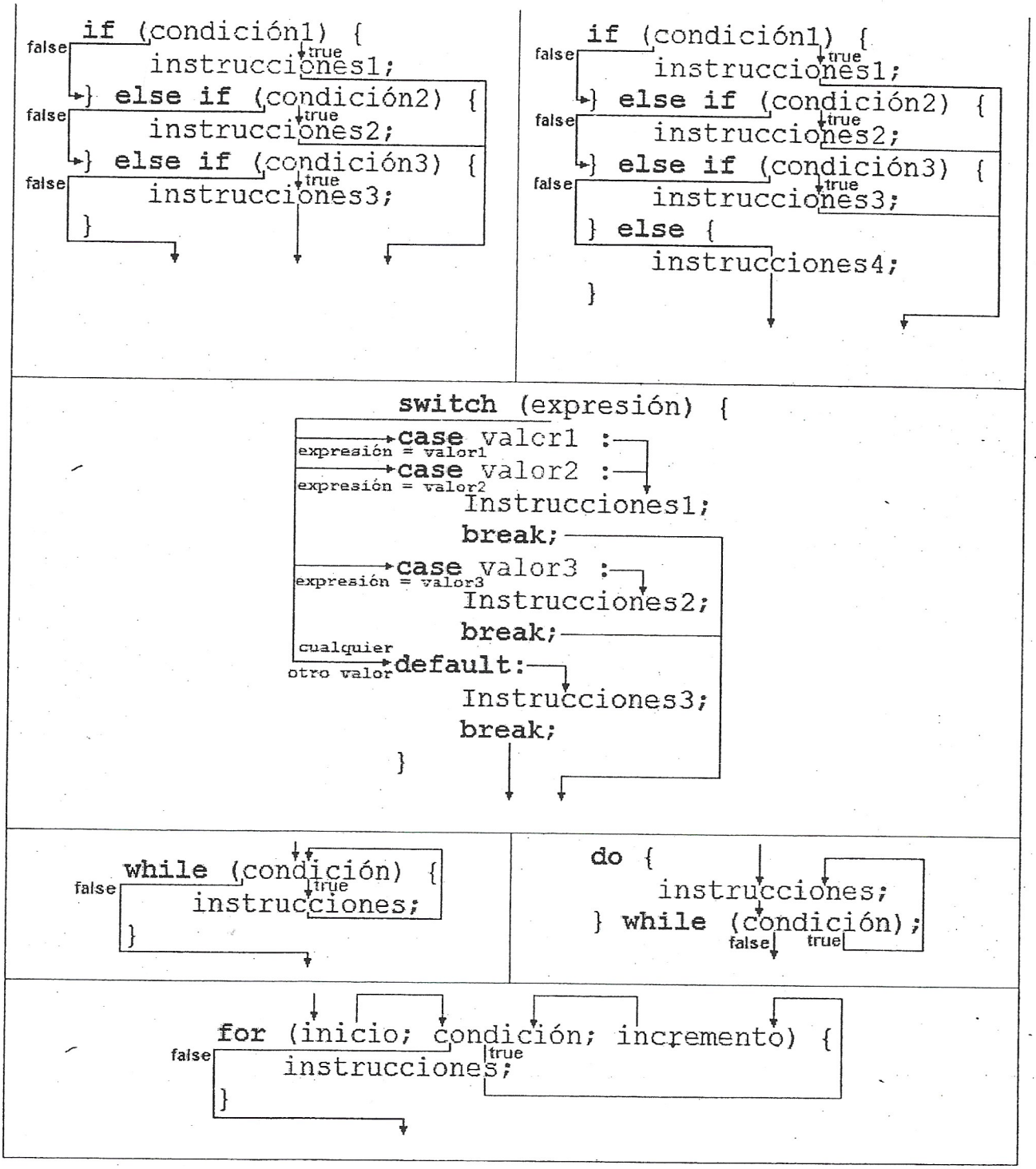
3.5. Do While.

3.5.1. Código.

```
 1  import jeliot.io.*;
 2
 3  public class DoWhile {
 4      public static void main() {
 5          String respuesta;
 6          // Cuando un programa llega a un do while ejecuta directamente el
 7          // bloque de instrucciones.
 8          do {
 9              System.out.println("¿Liberas a whiley?");
10              respuesta = Input.readString();
11              // Al terminar las instrucciones revisa la condición. Si es
12              // verdadera vuelve a iniciar el bloque de instrucciones
13              // (linea 8). Si es falsa, sale del do-while (linea 16).
14          } while (respuesta.equalsIgnoreCase("no"));
15          //Esto solo se hace una vez porque está fuera del ciclo.
16          System.out.println("whiley liberado");
17      }
18  }
```

3.6. Resumen de las Estructuras de Control.




if (condición1) {
false
true
instrucciones1;
} else if (condición2) {
false
true
instrucciones2;
} else if (condición3) {
false
true
instrucciones3;
}
if (condición1) {
false
true
instrucciones1;
} else if (condición2) {
false
true
instrucciones2;
} else if (condición3) {
false
true
instrucciones3;
} else {
instrucciones4;
}
switch (expresión) {
case valor1 :
expresión = valor1
case valor2 :
expresión = valor2
Instrucciones1;
break;
case valor3 :
expresión = valor3
Instrucciones2;
break;
cualquier otro valor
default:
Instrucciones3;
break;
}
while (condición) {
false
true
instrucciones;
}
do {
instrucciones;
} while (condición);
false
true
for (inicio; condición; incremento) {
false
true
instrucciones;
}

3.7. Contadores y Acumuladores.

3.7.1. Código.

```
1  import jeliot.io.*;
2
3  public class ContadoresYAcumuladores {
4      public static void main( ) {
5          int aprobados = 0;
6          int reprobados = 0;
7          double suma = 0.0;
8          for (int i = 0; i < 3; i++) {
9              System.out.println("Introduce calificación:");
10             double calificación = Input.readDouble( );
11             //también se puede escribir
12             //suma = suma + calificación
13             suma += calificación;
14             if (calificación >= 7.0) {
15                 aprobados++;
16             } else {
17                 reprobados++;
18             }
19         }
20         double promedio = suma / 3;
21         System.out.println("Promedio: " + promedio);
22         System.out.println("Aprobados: " + aprobados);
23         System.out.println("Reprobados: " + reprobados);
24     }
25 }
```

3.8. Ciclos Anidados.

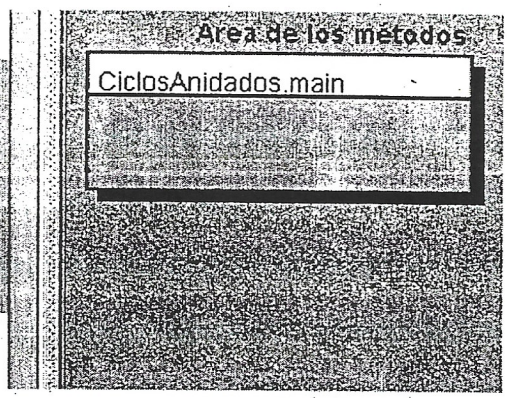
3.8.1. Código.

```
1  public class CiclosAnidados {
2      public static void main() {
3          for (int i = 0; i < 2; i++) {
4              for (int j = 0; j < 2; j++) {
5                  System.out.print("" + i + j);
6              }
7              System.out.println();
8          }
9          System.out.println("fin");
10     }
11 }
```

3.8.2. Ejecución.

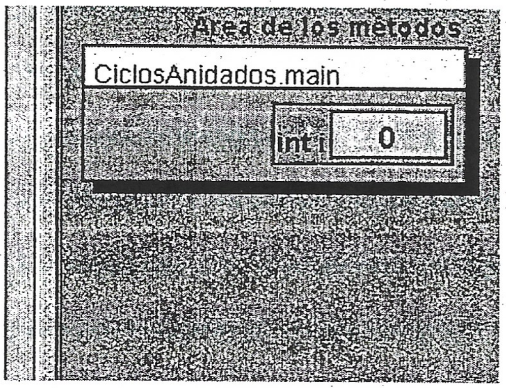
1. Crea el registro de activación para main.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



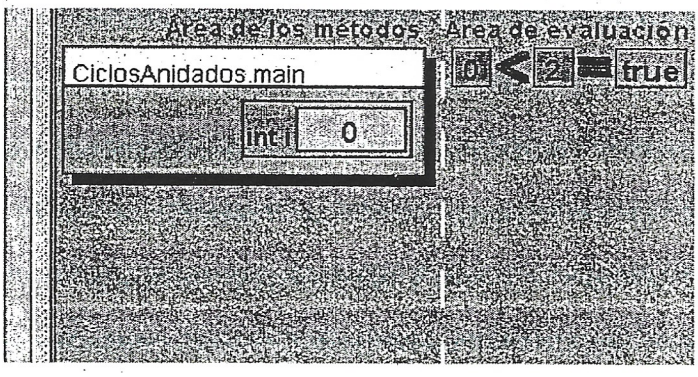
2. Crea la variable entera "i" con valor inicial "0".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



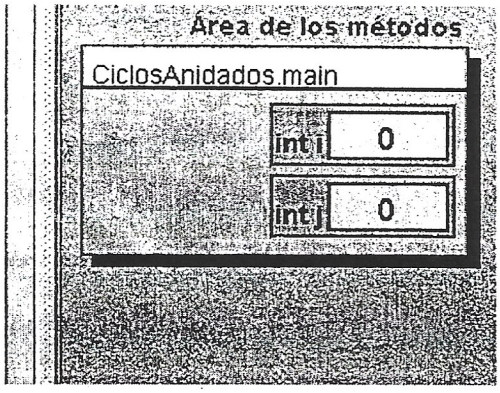
3. Realiza la comparación (i < 2) obteniendo "true" y entra al ciclo externo.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



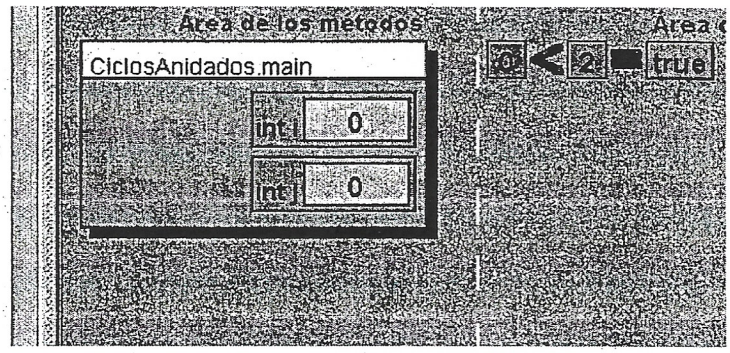
4. Crea la variable entera "j" con valor "0".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



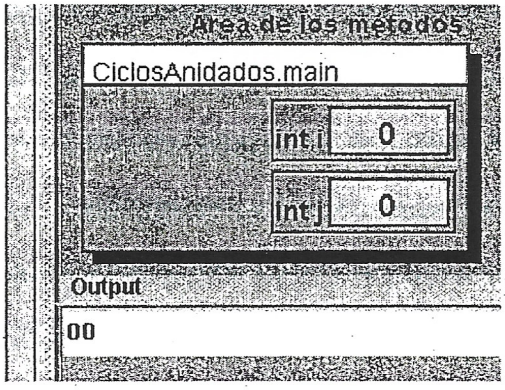
5. Realiza la comparación (j < 2) obteniendo "true" y entra al ciclo interno.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



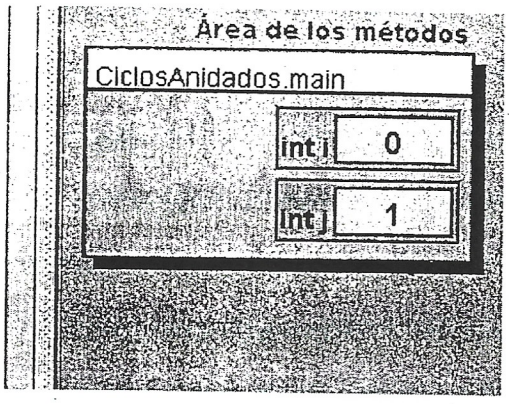
6. Muestra el mensaje "00" sin saltar línea.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



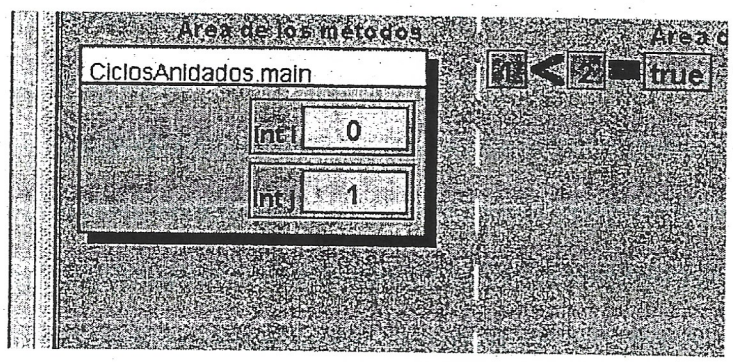
7. Incrementa la variable "j" que ahora vale "1".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



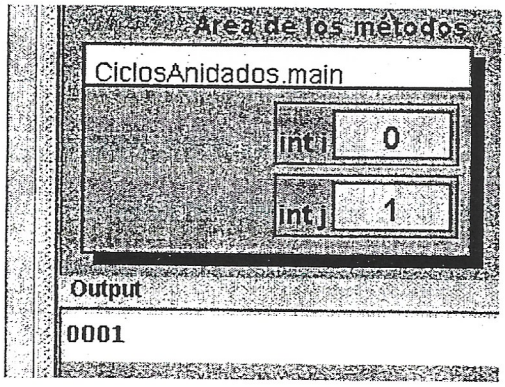
8. Realiza la comparación (j < 2) obteniendo "true" y entra al ciclo interno.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



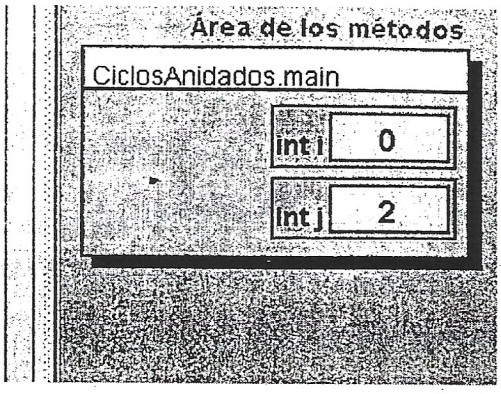
9. Muestra el mensaje "01".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



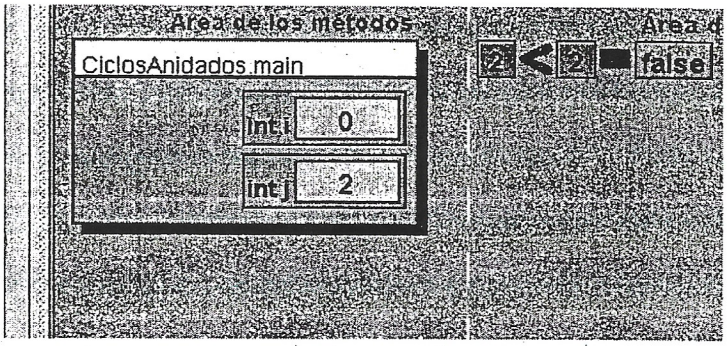
10. Incrementa la variable "j" que ahora vale "2".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



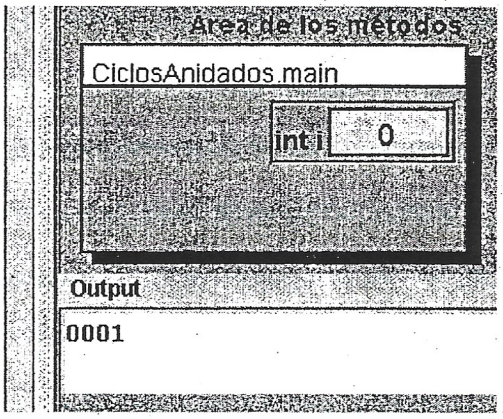
11. Realiza la comparación (j < 2) obteniendo "false", destruye "j" y salta el ciclo interno.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



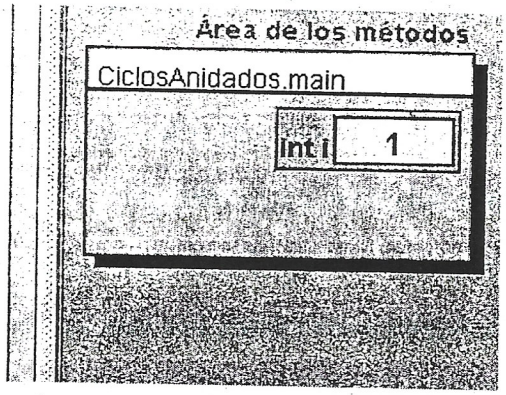
12. Realiza un salto de línea.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



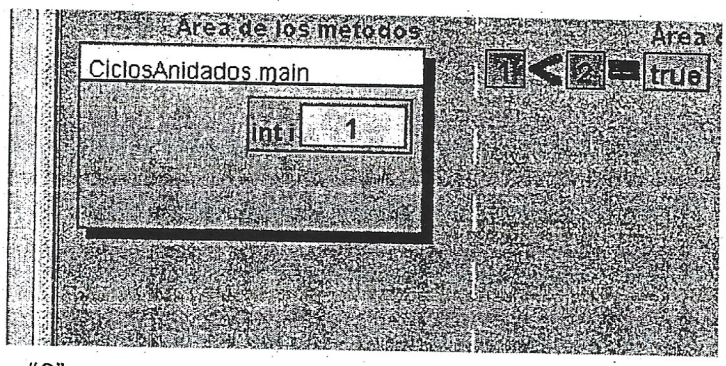
13. Incrementa la variable "i" que ahora vale "1".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



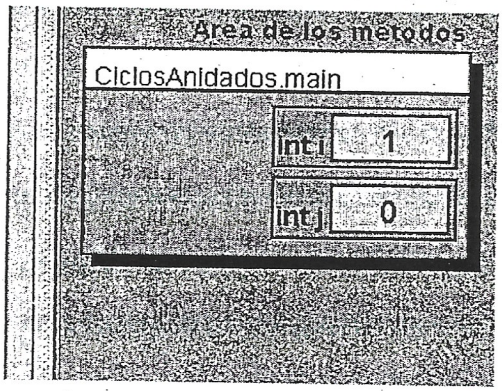
14. Realiza la comparación (i < 2) obteniendo "true" y entra al ciclo externo.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



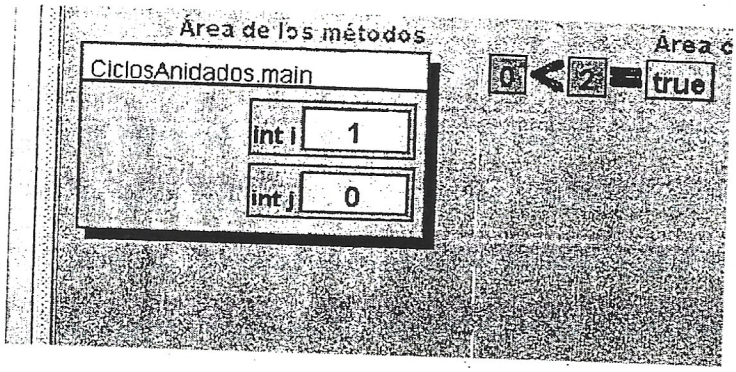
15. Crea la variable entera "j" con valor "0".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



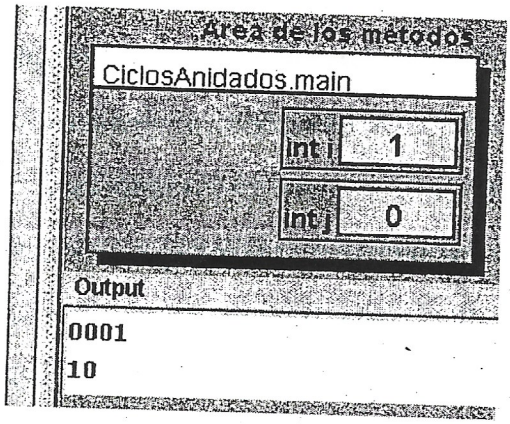
16. Realiza la comparación (j < 2) obteniendo "true" y entra al ciclo interno.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



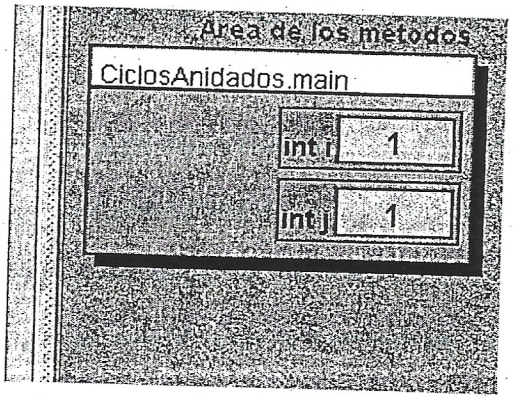
17. Muestra el mensaje "10".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



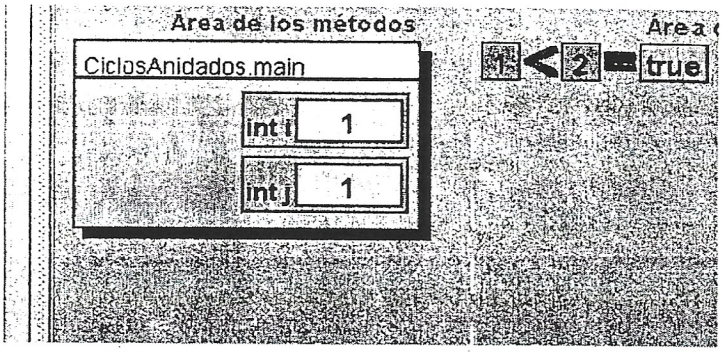
18. Incrementa la variable "j" que ahora vale "1".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



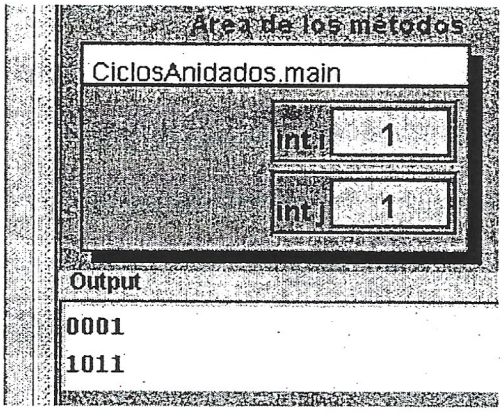
19. Realiza la comparación (j < 2) obteniendo "true" y entra al ciclo interno.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



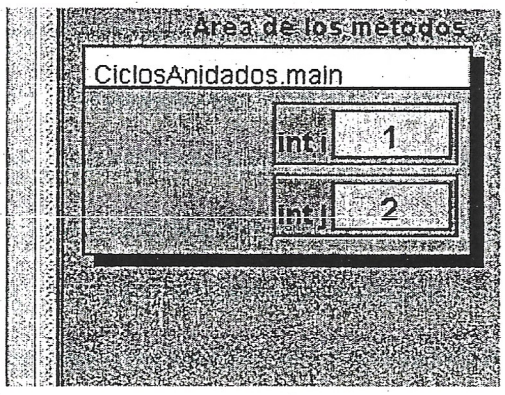
20. Muestra el mensaje "11".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



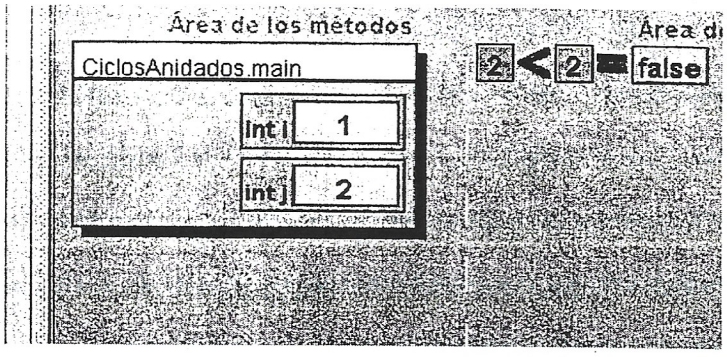
21. Incrementa la variable "j" que ahora vale "2".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



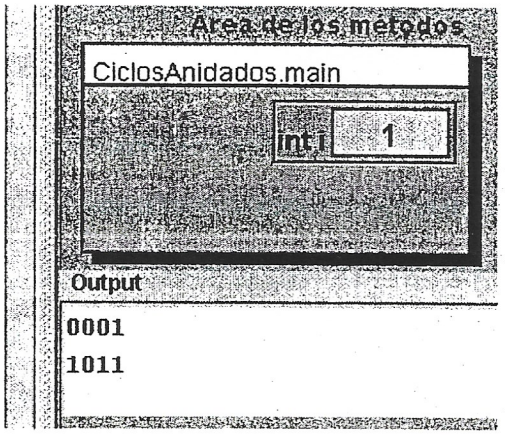
22. Realiza la comparación (j < 2) obteniendo "false", destruye "j" y salta el ciclo interno.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



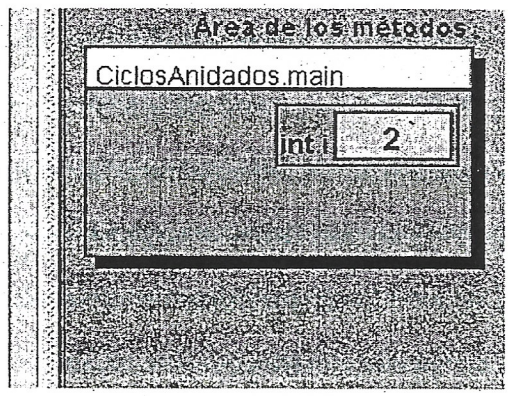
23. Realiza un salto de línea.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



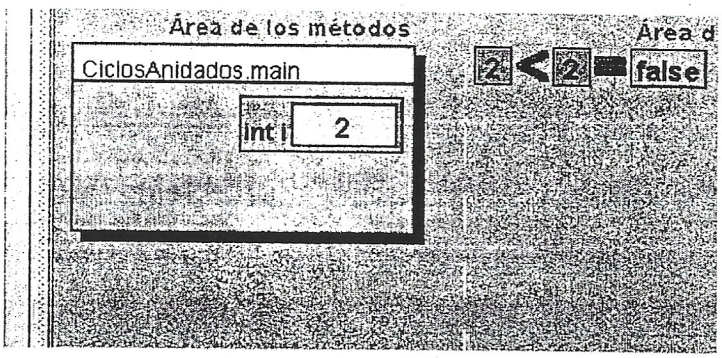
24. Incrementa el valor de "i", que ahora vale "2".

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



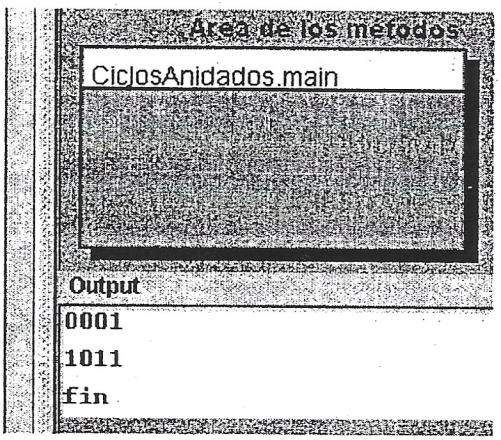
25. Realiza la comparación (i < 2), que dá "false", destruye "i", saltando el ciclo externo.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



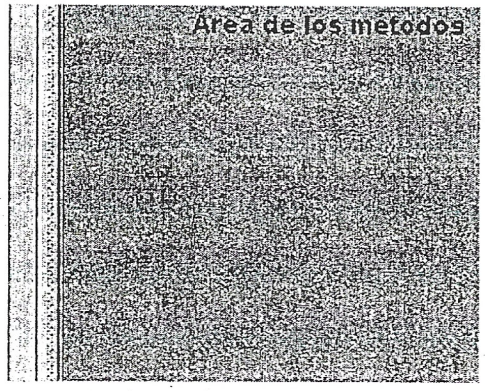
26. Muestra el mensaje "fin" y realiza un salto de línea.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



27. Elimina el registro de activación y termina.

```
public class CiclosAnidados {
    public static void main() {
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                System.out.print("" + i + j);
            }
            System.out.println();
        }
        System.out.println("fin");
    }
}
```



3.9. Break en Ciclos.

3.9.1. Código.

```
 1 import jeliot.io.*;
 2
 3 /**
 4  * Este programa te dá tres oportinidades para adivinar la cadena secreta.
 5  * Si la adivinas entra al if, muestra el mensaje "Le atinaste." y termina
 6  * el ciclo. Si no la adivinas, nunca entra al if y termina el ciclo.
 7  * Al finalizar el ciclo, hayas adivinado o no, muestra "Adios.".
 8  */
 9 public class BreakCiclo {
10     public static void main() {
11         for (int i = 0; i < 3; i++) {
12             String cadenaSecreta = Input.readString();
13             if (cadenaSecreta.equals("manito")) {
14                 System.out.println("Le atinaste.");
15                 break;
16             }
17         }
18         System.out.println("Adios.");
19     }
20 }
```

3.9.2. Comentarios.

La instrucción break se utiliza para terminar ciclos. Solo se recomienda su empleo cuando las otras estructuras de control no pueden expresar el algoritmo de una forma clara.

3.10. Continue.

3.10.1. Código.

```
 1 import jeliot.io.*;
 2
 3 /**
 4  * Este programa te dá tres oportinidades para adivinar la cadena secreta.
 5  * Si la adivinas entra al if, muestra el mensaje "Le atinaste.", de lo
 6  * contrario muestra "Sigue intentando". Este ciclo siempre se repite tres
 7  * veces. La instrucción continue salta al incremento del for.
 8  */
 9 public class ContinueCiclo {
10     public static void main() {
11         for (int i = 0; i < 3; i++) {
12             String cadenaSecreta = Input.readString();
13             if (cadenaSecreta.equals("manito")) {
14                 System.out.println("Le atinaste.");
15                 continue;
16             }
```

```
17              System.out.println("Fallaste.");
18          }
19          // Este ciclo es equivalente.
20          for (int i = 0; i < 3; i++) {
21              String cadenaSecreta = Input.readString();
22              if (cadenaSecreta.equals("manito")) {
23                  System.out.println("Le atinaste.");
24              } else {
25                  System.out.println("Fallaste.");
26              }
27          }
28      }
29  }
```

3.10.2. Comentarios.

La instrucción continue se utiliza para saltar secciones de ciclos. Solo se recomienda su empleo cuando las otras estructuras de control no pueden expresar el algoritmo de una forma clara. Se puede combinar también con while y do-while; en estos casos se salta todas las instrucciones y continúa en las condiciones, siguiendo luego el flujo normal de estas estructuras.

3.11. Continue en Ciclos Anidados.

3.11.1. Código.

```
1  /**
2   * Este programa empieza a generar números y cuando la suma llega a 1,
3   * termina
4   */
5  public class BreakCiclosAnidados {
6      public static void main() {
7          salida :
8          for (int i = 0; i < 2; i++) {
9              for (int j = 0; j < 2; j++) {
10                 if ((i + j) == 1) {
11                     break salida;
12                 }
13             }
14         }
15     }
16 }
```

3.11.2. Comentarios.

Una vez más esta técnica solo se recomienda cuando es la forma más clara de expresar un algoritmo. También se puede usar con la instrucción continue.

4.

Arreglos.

4. Arreglos.

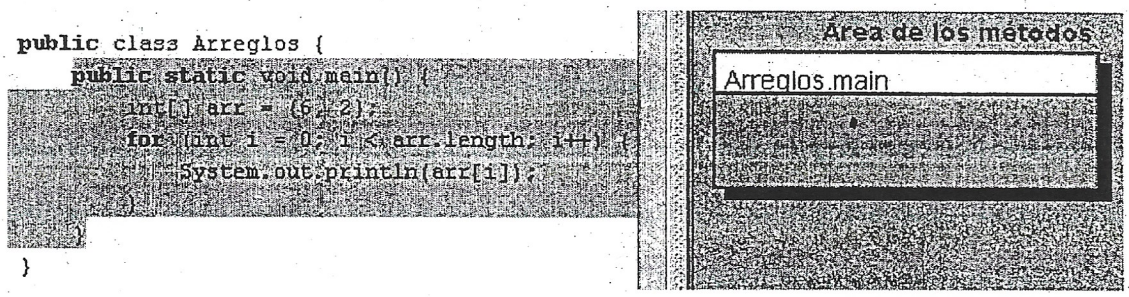
4.1. Arreglos Simples.

4.1.1. Código.

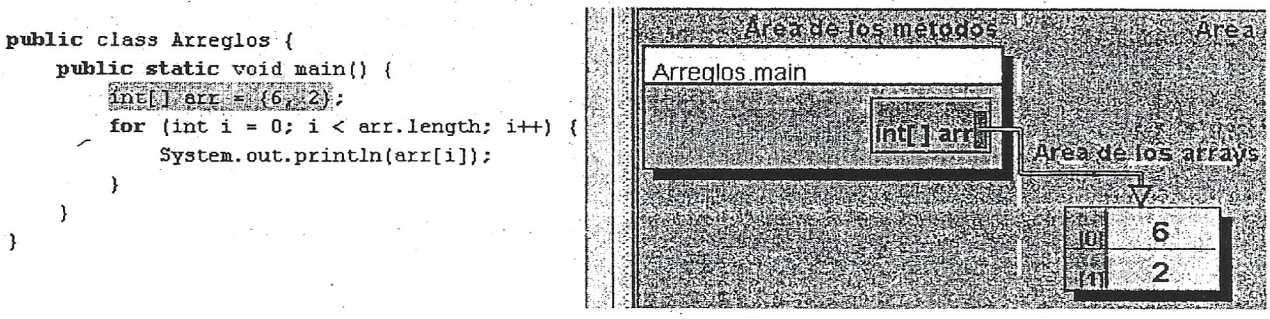
```
1  public class Arreglos {
2      public static void main() {
3          int[] arr = {6, 2};
4          for (int i = 0; i < arr.length; i++) {
5              System.out.println(arr[i]);
6          }
7      }
8  }
```

4.1.2. Ejecución.

1. Se crea un registro de activación para main.

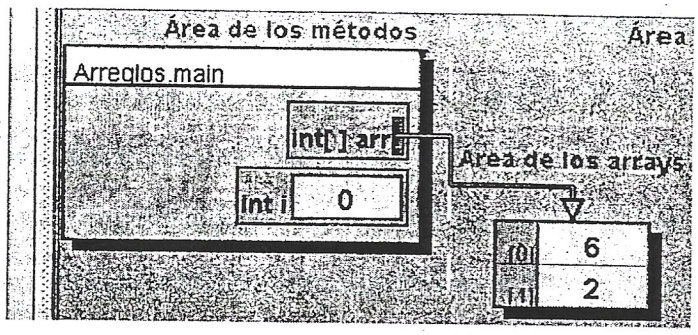


2. Se crea un arreglo de dos variables. La variable "arr[0]" vale "6" y "arr[1]" vale "1". El arreglo tiene también asociado el valor "length", cuyo valor es el total de variables que contiene. En este caso, length vale "2".



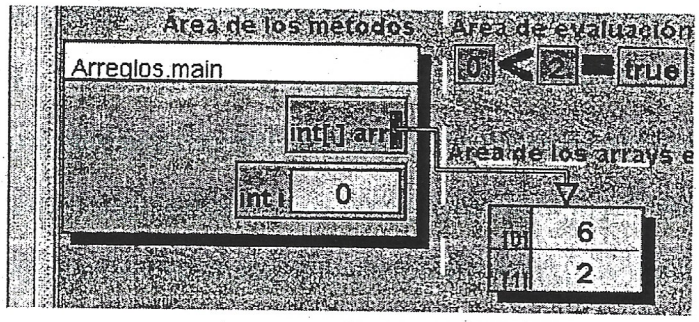
3. Se crea la variable entera "i" con valor "0".

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



4. Se realiza la comparación (i < arr.length) que dá "true" y entra al ciclo.

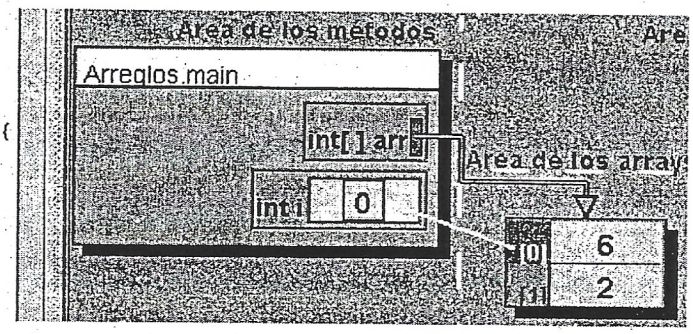
```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



5 Se muestra el valor de "arr[0]".

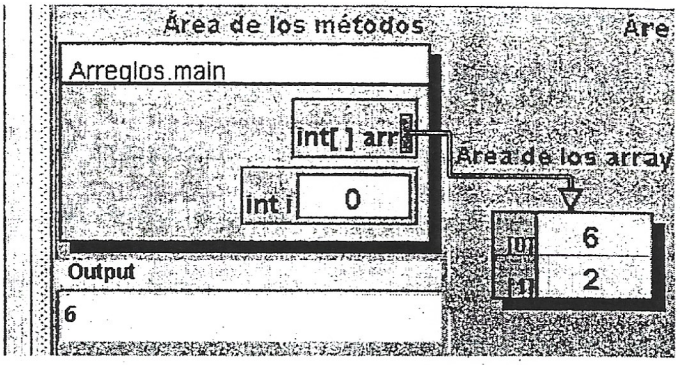
5.1 Se evalúa (arr[0]), que dá "6".

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



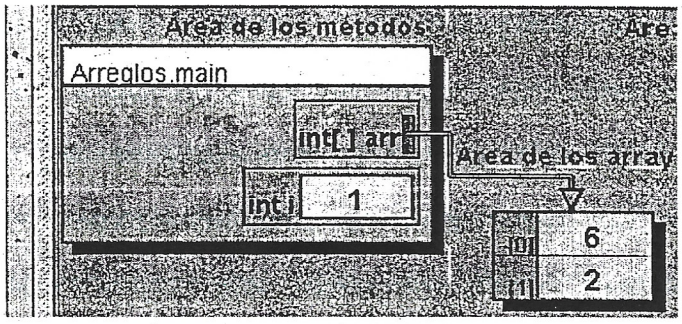
5.2 Muestra el valor obtenido, o sea "6".

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



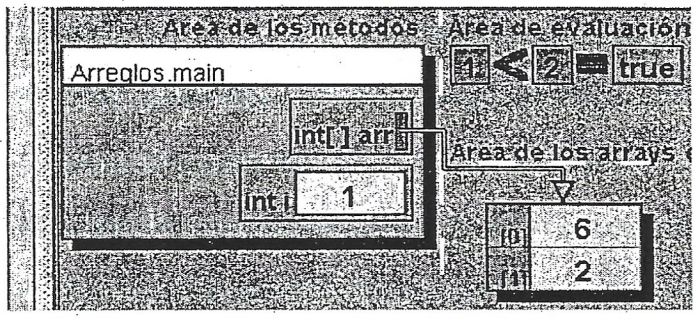
6 Incrementa el valor de "i", que ahora vale "1".

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



7 Se realiza la comparación (i < arr.length) que dá "true" y entra al ciclo.

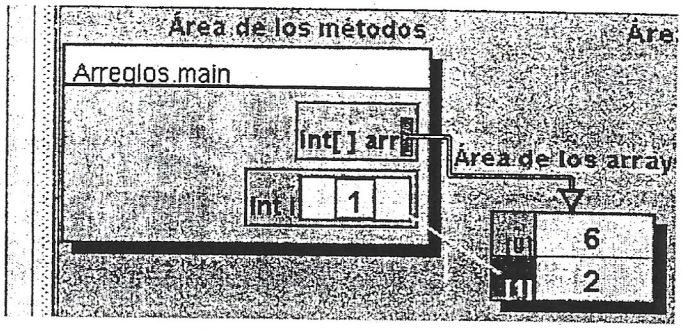
```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



8 Se muestra el valor de "arr[1]".

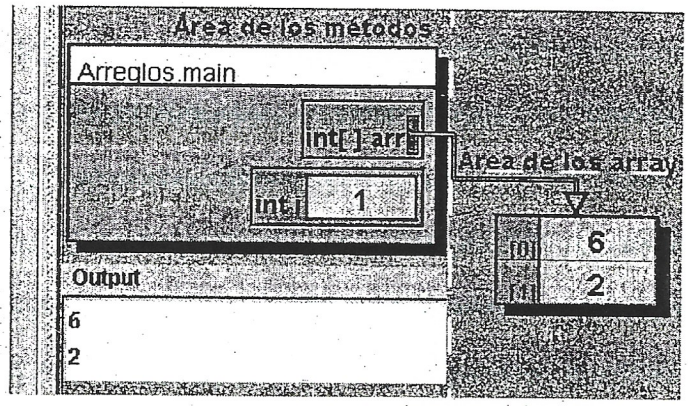
8.1 Se evalúa (arr[1]), que dá "2".

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



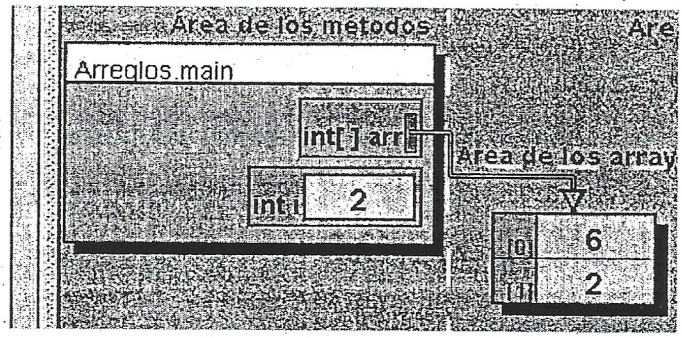
8.2 Muestra el valor obtenido, o sea "2".

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



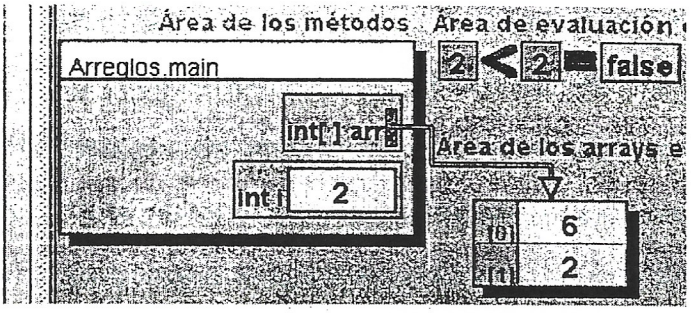
9. Incrementa el valor de "i", que ahora vale "2".

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



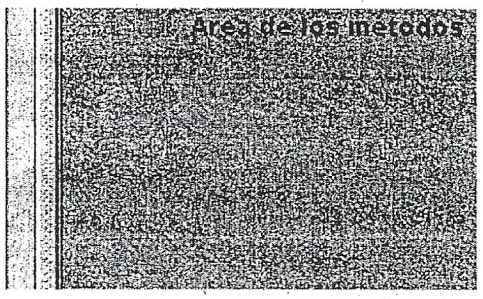
10. Se realiza la comparación (i < arr.length) que dá "false" y sale del ciclo.

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



11. Se elimina el registro de activación, destruye el arreglo y termina el programa.

```
public class Arreglos {
    public static void main() {
        int[] arr = {6, 2};
        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```



4.2. Arreglos de Enteros.

4.2.1. Código.

```
1  import jeliot.io.*;
2
3  public class ArrInts {
4      public static void main() {
5          int[] arr = new int[3];
6          // Llena el arreglo con números introducidos por el teclado.
7          for(int i = 0; i < arr.length; i++) {
8              arr[i] = Input.readInt();
9          }
10         // Multiplica por 2 a todos los elementos del arreglo.
11         for(int i = 0; i < arr.length; i++) {
12             arr[i] *= 2;
13         }
14         // Muestra los elementos del arreglo.
15         System.out.println("Tus números son:");
16         for(int i = 0; i < arr.length; i++) {
17             System.out.println(arr[i]);
18         }
19         // Suma todos los elementos del arreglo.
20         int suma = 0;
21         for(int i = 0; i < arr.length; i++) {
22             suma += arr[i];
```

```
23          }
24          System.out.println("La suma es: " + suma);
25      }
26  }
```