ESTRUCTURAS DE CONTROL

Manuel J. Molino Milla Luis Molina Garzón

IES Virgen del Carmen

Departamento de Informática

14 de octubre de 2015

Logo

Figura : Logo Java

Contenido

Contenido

Contenido

Variables

Programa que calcula el área de un círculo:

```
public class ComputaArea {
  public static void main(String[] args) {
    // Paso 1: Lee el radio
    // Paso 2: Computa area
    // Paso 3: Muestra el area
  }
}
```

Variables

Programa que calcula el área de un círculo:

```
public class ComputaArea {
  public static void main(String[] args) {
    // Paso 1: Lee el radio
    // Paso 2: Computa area
    // Paso 3: Muestra el area
  }
}
```

- El programa necesita leer el radio que introduce el usuario por teclado.

Variables

Programa que calcula el área de un círculo:

```
public class ComputaArea {
  public static void main(String[] args) {
    // Paso 1: Lee el radio
    // Paso 2: Computa area
    // Paso 3: Muestra el area
  }
}
```

- El programa necesita leer el radio que introduce el usuario por teclado.
- Luego hay que almacenar ese valor en algún lado.

Variables

Programa que calcula el área de un círculo:

```
public class ComputaArea {
  public static void main(String[] args) {
    // Paso 1: Lee el radio
    // Paso 2: Computa area
    // Paso 3: Muestra el area
  }
}
```

- El programa necesita leer el radio que introduce el usuario por teclado.
- Luego hay que almacenar ese valor en algún lado.
- ¿Que ocurre si el valor del radio es negativo?

Variables

Programa que calcula el área de un círculo:

```
public class ComputaArea {
  public static void main(String[] args) {
    // Paso 1: Lee el radio
    // Paso 2: Computa area
    // Paso 3: Muestra el area
  }
}
```

- El programa necesita leer el radio que introduce el usuario por teclado.
- Luego hay que almacenar ese valor en algún lado.
- ¿Que ocurre si el valor del radio es negativo?
- Se debería rechazar dicho valor.

Variables

Programa que calcula el área de un círculo:

```
public class ComputaArea {
  public static void main(String[] args) {
    // Paso 1: Lee el radio
    // Paso 2: Computa area
    // Paso 3: Muestra el area
  }
}
```

- El programa necesita leer el radio que introduce el usuario por teclado.
- Luego hay que almacenar ese valor en algún lado.
- ¿Que ocurre si el valor del radio es negativo?
- Se debería rechazar dicho valor.

Variables

Programa que calcula el área de un círculo:

```
public class ComputaArea {
  public static void main(String[] args) {
    // Paso 1: Lee el radio
    // Paso 2: Computa area
    // Paso 3: Muestra el area
  }
}
```

- El programa necesita leer el radio que introduce el usuario por teclado.
- Luego hay que almacenar ese valor en algún lado.
- ¿Que ocurre si el valor del radio es negativo?
- Se debería rechazar dicho valor.

Tipo boolean

- El programa realiza la computación del área y obtiene un valor.

Tipo boolean

- El programa realiza la computación del área y obtiene un valor.
- Pero no existen círculos con radio negativo.

Tipo boolean

- El programa realiza la computación del área y obtiene un valor.
- Pero no existen círculos con radio negativo.
- Los lenguajes de alto nivel permiten estructuras denominadas *estructuras de control* para prevenir estos problemas.

Tipo boolean

- El programa realiza la computación del área y obtiene un valor.
- Pero no existen círculos con radio negativo.
- Los lenguajes de alto nivel permiten estructuras denominadas *estructuras de control* para prevenir estos problemas.

Tipo boolean

- El programa realiza la computación del área y obtiene un valor.
- Pero no existen círculos con radio negativo.
- Los lenguajes de alto nivel permiten estructuras denominadas *estructuras de control* para prevenir estos problemas.

```
if (radio < 0){
    System.out.println("Entrada incorrecta");
} else {
    area = radio * radio * 3.14159;
    System.out.println("El area es " + area);
}
```

Tipo boolean

- El programa realiza la computación del área y obtiene un valor.
- Pero no existen círculos con radio negativo.
- Los lenguajes de alto nivel permiten estructuras denominadas *estructuras de control* para prevenir estos problemas.

```
if (radio < 0){
    System.out.println("Entrada incorrecta");
} else {
    area = radio * radio * 3.14159;
    System.out.println("El area es " + area);
}
```

Usamos sentencias de selección que se basa en expresiones booleanas.

Tipo boolean

- Son datos de tipo lógico.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*
- El valor true evalua como verdadero una expresion.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*
- El valor true evalua como verdadero una expresion.
- El valor false evalua como falso una expresion.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*
- El valor true evalua como verdadero una expresion.
- El valor false evalua como falso una expresion.
- Ejemplo: una variable es mayor o menor que un numero dado.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*
- El valor true evalua como verdadero una expresion.
- El valor false evalua como falso una expresion.
- Ejemplo: una variable es mayor o menor que un numero dado.
- El resultado de una expresion matematicas es mayor, menor o igual que cero.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*
- El valor true evalua como verdadero una expresion.
- El valor false evalua como falso una expresion.
- Ejemplo: una variable es mayor o menor que un numero dado.
- El resultado de una expresion matematicas es mayor, menor o igual que cero.
- Una persona es mayor de edad o no.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*
- El valor true evalua como verdadero una expresion.
- El valor false evalua como falso una expresion.
- Ejemplo: una variable es mayor o menor que un numero dado.
- El resultado de una expresion matematicas es mayor, menor o igual que cero.
- Una persona es mayor de edad o no.

Tipo boolean

- Son datos de tipo lógico.
- Ocupan 1 byte de memoria.
- Admiten dos valores:
- *true*
- *false*
- El valor true evalua como verdadero una expresion.
- El valor false evalua como falso una expresion.
- Ejemplo: una variable es mayor o menor que un numero dado.
- El resultado de una expresion matematicas es mayor, menor o igual que cero.
- Una persona es mayor de edad o no.

Operadores relacionales

Generan un resultado de tipo boolean

Operadores relacionales

Generan un resultado de tipo boolean

OPERADOR	SIGNIFICADO
$>$	mayor que
$<$	menor que
$<=$	menor o igual que
$>=$	mayor o igual que
$==$	igual que
$!=$	distinto que

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4
- true

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4
- true
- boolean j = 5>=5

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4
- true
- boolean j = 5>=5
- true

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4
- true
- boolean j = 5>=5
- true
- boolean j = 5<=4

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4
- true
- boolean j = 5>=5
- true
- boolean j = 5<=4
- false

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4
- true
- boolean j = 5>=5
- true
- boolean j = 5<=4
- false

Operadores relacionales

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4
- true
- boolean j = 5>4*3
- false Los operados aritméticos tienen mayor precedencia que los operadores relacionales.
- boolean j = 5>(4*3)
- false
- boolean j = 5==5
- true
- boolean j = 5!=4
- true
- boolean j = 5>=5
- true
- boolean j = 5<=4
- false

Operadores lógicos

Tambien generan un resultado de tipo boolean

Operadores lógicos

Tambien generan un resultado de tipo boolean

OPERADOR	SIGNIFICADO
&&	AND
||	OR
!	NOT

Operadores lógicos

Tambien generan un resultado de tipo boolean

OPERADOR	SIGNIFICADO
&&	AND
||	OR
!	NOT

EXPRESION	VALOR
true && true	TRUE
false && true	FALSE
false && false	FALSE
true||true	TRUE
true||false	TRUE
false||false	FALSE
!true	FALSE
!false	TRUE

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;
- false

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;
- false
- boolean j = !(5==5) || 4==3 && 3!=4

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;
- false
- boolean j = !(5==5) || 4==3 && 3!=4
- false

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;
- false
- boolean j = !(5==5) || 4==3 && 3!=4
- false
- boolean j = !(5==5 && 4==3 || 3==4);

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;
- false
- boolean j = !(5==5) || 4==3 && 3!=4
- false
- boolean j = !(5==5 && 4==3 || 3==4);
- true

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;
- false
- boolean j = !(5==5) || 4==3 && 3!=4
- false
- boolean j = !(5==5 && 4==3 || 3==4);
- true

Operadores lógicos

¿Cuál es el valor booleano de la variable?

- boolean j = 5>4 && 5<4
- false
- boolean j = 5>4*3 || 3==3
- true Los operados relaciones tienen mayor precedencia que los operadores lógicos.
- boolean j = 5==5 || 4==3 && 3!=4
- true El operador && tiene preferencia sobre ||
- boolean j = 5==5 && 4==3 || 3==4;
- false
- boolean j = !(5==5) || 4==3 && 3!=4
- false
- boolean j = !(5==5 && 4==3 || 3==4);
- true

Ejemplo de uso en Java

Ejemplo de uso en Java

```
public class TestBoolean{
    public static void main(String[] args){
        int numero1 = 3;
        int numero2 = numero1 * 2;
        boolean numero1EsPar = (numero1 % 2 == 0);
        boolean numero2EsPar = (numero2 % 2 == 0);
        System.out.println("¿Es par "+numero1+"? "+numero1EsPar);
        System.out.println("¿Es par "+numero2+"? "+ numero2EsPar);
    }
}
```

Ejemplo de uso en Java

```
public class TestBoolean{
    public static void main(String[] args){
        int numero1 = 3;
        int numero2 = numero1 * 2;
        boolean numero1EsPar = (numero1 % 2 == 0);
        boolean numero2EsPar = (numero2 % 2 == 0);
        System.out.println("¿Es par "+numero1+"? "+numero1EsPar);
        System.out.println("¿Es par "+numero2+"? "+ numero2EsPar);
    }
}
```

- ¿Cómo debe llamarse el programa?

Ejemplo de uso en Java

```
public class TestBoolean{
    public static void main(String[] args){
        int numero1 = 3;
        int numero2 = numero1 * 2;
        boolean numero1EsPar = (numero1 % 2 == 0);
        boolean numero2EsPar = (numero2 % 2 == 0);
        System.out.println("¿Es par "+numero1+"? "+numero1EsPar);
        System.out.println("¿Es par "+numero2+"? "+ numero2EsPar);
    }
}
```

- ¿Cómo debe llamarse el programa?
- ¿Qué resultados produce el programa?

Ejemplo de uso en Java

```
public class TestBoolean{
    public static void main(String[] args){
        int numero1 = 3;
        int numero2 = numero1 * 2;
        boolean numero1EsPar = (numero1 % 2 == 0);
        boolean numero2EsPar = (numero2 % 2 == 0);
        System.out.println("¿Es par "+numero1+"? "+numero1EsPar);
        System.out.println("¿Es par "+numero2+"? "+ numero2EsPar);
    }
}
```

- ¿Cómo debe llamarse el programa?
- ¿Qué resultados produce el programa?

Ejemplo de uso en Java

```
public class TestBoolean{
    public static void main(String[] args){
        int numero1 = 3;
        int numero2 = numero1 * 2;
        boolean numero1EsPar = (numero1 % 2 == 0);
        boolean numero2EsPar = (numero2 % 2 == 0);
        System.out.println("¿Es par "+numero1+"? "+numero1EsPar);
        System.out.println("¿Es par "+numero2+"? "+ numero2EsPar);
    }
}
```

- ¿Cómo debe llamarse el programa?
- ¿Qué resultados produce el programa?

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch
  6. break

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch
  6. break
  7. continue

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch
  6. break
  7. continue
  8. goto

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch
  6. break
  7. continue
  8. goto
- Todas las sentencias condicionales utilizan la certeza o falsedad de una expresion

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch
  6. break
  7. continue
  8. goto
- Todas las sentencias condicionales utilizan la certeza o falsedad de una expresion
- Ejemplo: i==0, i<=0, i>0, . . .

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch
  6. break
  7. continue
  8. goto
- Todas las sentencias condicionales utilizan la certeza o falsedad de una expresion
- Ejemplo: i==0, i<=0, i>0, ...

Sentencias de control

- Java usa todas las sentencias de control de ejecucion de C.
- Las palabras claves usadas son:
  1. if-else
  2. while
  3. do-while
  4. for
  5. switch
  6. break
  7. continue
  8. goto
- Todas las sentencias condicionales utilizan la certeza o falsedad de una expresion
- Ejemplo: i==0, i<=0, i>0, ...

Sentencia if-else

if

- if (expresion condicional) {

Sentencia if-else

if

- if (expresion condicional) {
- sentencias

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia
- }
- ...

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia
- }
- ...
- else {

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia
- }
- ...
- else {
- sentencias

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia
- }
- . . .
- else {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia
- }
- ...
- else {
- sentencias
- }

Sentencia if-else

if

- if (expresion condicional) {
- sentencias
- }

if-else

- if (expresion condicional) {
- sentencias
- }
- else {
- sentencias
- }

if-else if-else

- if (expresion condicional) {
- sentencias
- }
- else if (expresion condicional) {
- sentencia
- }
- ...
- else {
- sentencias
- }

Diagramas de flujo

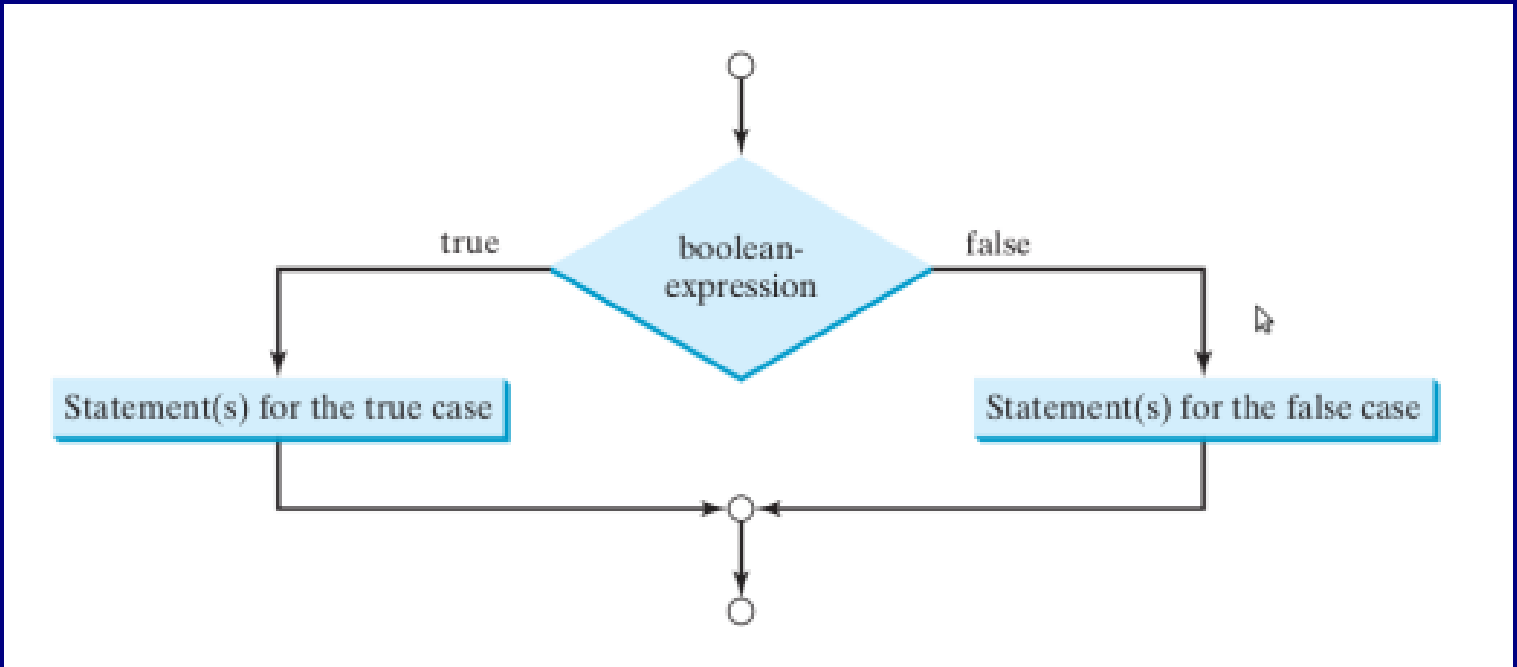


Figura : if-else

Ejemplo de uso en Java

```
public class Divisible{
   public static void main(String[] arg){
        int numero = 18;
        if ((numero % 2 == 0) && (numero % 3 ==0))
        {
            System.out.println( "Divisible seis");
        }
        else if (numero % 2 == 0){
             System.out.println("Divisible por dos");
        }
        else if (numero % 3 == 0){
             System.out.println("Divisible por tres");
        }
        else
        {
             System.out.println("No divisible por dos ni por tres");
        }
    }
}
```

Ejemplo de uso en Java

```
public class Divisible{
   public static void main(String[] arg){
        int numero = 18;
        if ((numero % 2 == 0) && (numero % 3 ==0))
        {
            System.out.println( "Divisible seis");
        }
        else if (numero % 2 == 0){
             System.out.println("Divisible por dos");
        }
        else if (numero % 3 == 0){
             System.out.println("Divisible por tres");
        }
        else
        {
             System.out.println("No divisible por dos ni por tres");
        }
    }
}
```

Indica valores de la variable numero para que se cumplan todas las condiciones.

El código anterior es equivalente:

```
public class Divisible{
   public static void main(String[] arg){
         int numero = 18;
         if ((numero % 2 == 0) && (numero % 3 ==0))
            System.out.println( "Divisible seis");
         else if (numero % 2 == 0){
            System.out.println("Divisible por dos");
         else if (numero % 3 == 0){
            System.out.println("Divisible por tres");
         else
           System.out.println("No divisible por
                                    dos ni por tres");
   }
}
```

El código anterior es equivalente:

```
public class Divisible{
   public static void main(String[] arg){
         int numero = 18;
         if ((numero % 2 == 0) && (numero % 3 ==0))
            System.out.println( "Divisible seis");
         else if (numero % 2 == 0){
            System.out.println("Divisible por dos");
         else if (numero % 3 == 0){
            System.out.println("Divisible por tres");
         else
           System.out.println("No divisible por
                                    dos ni por tres");
   }
}
```

Se omiten las llaves ({...}) pues solo hay una sentencia.

¿Y si cambiamos el orden?

```
public class Divisible{
   public static void main(String[] arg){
        int numero = 18;

        if (numero % 2 == 0){
             System.out.println("Divisible por dos");
        }
        else if ((numero % 2 == 0) && (numero % 3 ==0))
        {
            System.out.println( "Divisible seis");
        }
        else if (numero % 3 == 0){
             System.out.println("Divisible por tres");
        }
        else
        {
             System.out.println("No divisible por dos ni por tres");
        }
    }
}
```

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)
- ¿Cuál es la expresión correcta?

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)
- ¿Cuál es la expresión correcta?
- Otro tipo de error es el comentado anteriormente, no evaluar correctamente el orden de las condiciones.

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)
- ¿Cuál es la expresión correcta?
- Otro tipo de error es el comentado anteriormente, no evaluar correctamente el orden de las condiciones.
- Olvidar {} en el bloque con mas de una sentencia.

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)
- ¿Cuál es la expresión correcta?
- Otro tipo de error es el comentado anteriormente, no evaluar correctamente el orden de las condiciones.
- Olvidar {} en el bloque con mas de una sentencia.
- Olvidar el paréntesis en la condición *if x>0*

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)
- ¿Cuál es la expresión correcta?
- Otro tipo de error es el comentado anteriormente, no evaluar correctamente el orden de las condiciones.
- Olvidar {} en el bloque con mas de una sentencia.
- Olvidar el paréntesis en la condición *if x>0*
- No es error pero es mejor la opción *if (variableBoolean)* que *if (variableBooelan == true)*

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)
- ¿Cuál es la expresión correcta?
- Otro tipo de error es el comentado anteriormente, no evaluar correctamente el orden de las condiciones.
- Olvidar {} en el bloque con mas de una sentencia.
- Olvidar el paréntesis en la condición *if x>0*
- No es error pero es mejor la opción *if (variableBoolean)* que *if (variableBooelan == true)*

Típicos errores

- Confundir el operador de asignación con el operador relacional igual:
- if (numero % 2 = 0)
- if (numero % 2 == 0)
- ¿Cuál es la expresión correcta?
- Otro tipo de error es el comentado anteriormente, no evaluar correctamente el orden de las condiciones.
- Olvidar {} en el bloque con mas de una sentencia.
- Olvidar el paréntesis en la condición *if x>0*
- No es error pero es mejor la opción *if (variableBoolean)* que *if (variableBooelan == true)*

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.
- Un algoritmo describe como se resuelve en término de acciones un problema y el orden de ejecucción de dichas acciones.

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.
- Un algoritmo describe como se resuelve en término de acciones un problema y el orden de ejecucción de dichas acciones.
- Dichos algoritmos se pueden escribir bien en lenguaje natural o en pseudocódigo

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.
- Un algoritmo describe como se resuelve en término de acciones un problema y el orden de ejecucción de dichas acciones.
- Dichos algoritmos se pueden escribir bien en lenguaje natural o en pseudocódigo

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.
- Un algoritmo describe como se resuelve en término de acciones un problema y el orden de ejecucción de dichas acciones.
- Dichos algoritmos se pueden escribir bien en lenguaje natural o en pseudocódigo

Lenguaje natural

Calcular el área de un círculo dado el radio del mismo.

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.
- Un algoritmo describe como se resuelve en término de acciones un problema y el orden de ejecucción de dichas acciones.
- Dichos algoritmos se pueden escribir bien en lenguaje natural o en pseudocódigo

Lenguaje natural

Calcular el área de un círculo dado el radio del mismo.

Pseudocodigo

1. Leer el radio.

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.
- Un algoritmo describe como se resuelve en término de acciones un problema y el orden de ejecucción de dichas acciones.
- Dichos algoritmos se pueden escribir bien en lenguaje natural o en pseudocódigo

Lenguaje natural

Calcular el área de un círculo dado el radio del mismo.

Pseudocodigo

1. Leer el radio.
2. Aplicar la fórmula *radio * radio * PI*

pseudocódigo

- Escribir un programa implica el diseño de un algoritmo y su posterior conversión en código.
- Un algoritmo describe como se resuelve en término de acciones un problema y el orden de ejecucción de dichas acciones.
- Dichos algoritmos se pueden escribir bien en lenguaje natural o en pseudocódigo

Lenguaje natural

Calcular el área de un círculo dado el radio del mismo.

Pseudocodigo

1. Leer el radio.
2. Aplicar la fórmula *radio * radio * PI*
3. Mostrar el área

psuedocodigo

Programa divisible

si resto division por dos y por tres es cero

    es divisible por seis

si resto division por dos es cero

    es divisible por dos

si resto division por tres es cero

    es divisible por tres

en otro caso

    no es divisible ni por dos ni por tres

Iteración

En programación, *Iteración* es la repetición de un proceso dentro de un programa de computadora.
Ejemplo: sumar los n primeros numeros.

Iteración

En programación, *Iteración* es la repetición de un proceso dentro de un programa de computadora.
Ejemplo: sumar los *n* primeros numeros.

```
int suma igual a cero
int contador igual a uno
mientras que contador sea menor o igual que n
    sumar a suma el valor del contador
    incrementar el valor del contador
devolver suma
```

Iteración

En programación, *Iteración* es la repetición de un proceso dentro de un programa de computadora.
Ejemplo: sumar los *n* primeros numeros.

int suma igual a cero
int contador igual a uno
mientras que contador sea menor o igual que n

    sumar a suma el valor del contador
    incrementar el valor del contador

devolver suma

En java tenemos estructura de control para realizar la iteración: while, do-while, for o switch,

while y do-while

Bucle while

- while (condicion) {

while y do-while

Bucle while

- while (condicion) {
- sentencias;

while y do-while

Bucle while

- while (condicion) {
- sentencias;
- }

Ejemplos

```
public int sumar1(int n){
    int suma = 0;
    int contador =1;
    while (contador <= n){
       suma=suma+contador;
       contador++;
    }
    return suma;
}
```

Ejemplos

```
public int sumar1(int n){
    int suma = 0;
    int contador =1;
    while (contador <= n){
       suma=suma+contador;
       contador++;
    }
    return suma;
}
```

```
public int sumar2(int n){
    int suma = 0;
    int contador =1;
    do{
       suma=suma+contador;
       contador++;
    }
    while (contador <= n);
    return suma;
}
```

Ejemplos

```
public int sumar1(int n){
    int suma = 0;
    int contador =1;
    while (contador <= n){
       suma=suma+contador;
       contador++;
    }
    return suma;
}
```

```
public int sumar2(int n){
    int suma = 0;
    int contador =1;
    do{
       suma=suma+contador;
       contador++;
    }
    while (contador <= n);
    return suma;
}
```

¿Qué ocurre cuando n vale 3,2,1,0?

Ejemplos

```
public int sumar1(int n){
    int suma = 0;
    int contador =1;
    while (contador <= n){
       suma=suma+contador;
       contador++;
    }
    return suma;
}
```

```
public int sumar2(int n){
    int suma = 0;
    int contador =1;
    do{
       suma=suma+contador;
       contador++;
    }
    while (contador <= n);
    return suma;
}
```

¿Qué ocurre cuando n vale 3,2,1,0?

En los casos anteriores ¿cuantas veces se entra en el bucle?

Comprobacion

suma	contador
n	3
0	1
1	2
3	3
6	4

suma	contador
n	2
0	1
1	2
3	3

suma	contador
n	1
0	1
1	2

suma	contador
n	0
0	1

```
public int sumar1(int n){
    int suma = 0;
    int contador =1;
    while (contador <= n){
        suma=suma+contador;
        contador++;
    }
    return suma;
}
```

CONDICION contador $<= n$

Comprobacion

suma	contador
n	3
0	1
1	2
3	3
6	4

suma	contador
n	2
0	1
1	2
3	3

suma	contador
n	1
0	1
1	2

suma	contador
n	0
0	1
1	2

```
public int sumar2(int n){
    int suma = 0;
    int contador =1;
    do{
       suma=suma+contador;
       contador++;
    }
    while (contador <= n);
    return suma;
}
```

CONDICION contador $<= n$

do-while

```
public class TestDelDoWhile {
   public static void main (String [ ] Args) {
      int contador = 0 ;
      do {
         System.out.println ("Contando.: " +(contador+1));
         contador += 1;
      } while (contador<10);
   }
}
```

¿Cuál es la salida del programa?

Diferencia while, do-while

- La unica diferencia entre *while* y *do-while* es que la sentencia do-while se ejecuta siempre, al menos, una vez incluso aunque la expresion condicional sea falsa.

Diferencia while, do-while

- La unica diferencia entre *while* y *do-while* es que la sentencia do-while se ejecuta siempre, al menos, una vez incluso aunque la expresion condicional sea falsa.
- En while si la condicion es falsa la primera vez, la sentencia no se ejecuta nunca.

Diferencia while, do-while

- La unica diferencia entre *while* y *do-while* es que la sentencia do-while se ejecuta siempre, al menos, una vez incluso aunque la expresion condicional sea falsa.
- En while si la condicion es falsa la primera vez, la sentencia no se ejecuta nunca.
- En la practica do-while es menos comun que while.

Diferencia while, do-while

- La unica diferencia entre *while* y *do-while* es que la sentencia do-while se ejecuta siempre, al menos, una vez incluso aunque la expresion condicional sea falsa.
- En while si la condicion es falsa la primera vez, la sentencia no se ejecuta nunca.
- En la practica do-while es menos comun que while.

Diferencia while, do-while

- La unica diferencia entre *while* y *do-while* es que la sentencia do-while se ejecuta siempre, al menos, una vez incluso aunque la expresion condicional sea falsa.
- En while si la condicion es falsa la primera vez, la sentencia no se ejecuta nunca.
- En la practica do-while es menos comun que while.

for

definicion

- *for* (inicializacion; condicion; paso)

for

definicion

- *for* (inicializacion; condicion; paso)
- {

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

Ejemplo:

- int suma=0;

Se puede omitir el bloque, pués solo hay una sentencia.

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

Ejemplo:

- int suma=0;
- for (int i=1; i<=n; i++) {

Se puede omitir el bloque, pués solo hay una sentencia.

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

Ejemplo:

- int suma=0;
- for (int i=1; i<=n; i++) {
- suma+=i;

Se puede omitir el bloque, pués solo hay una sentencia.

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

Ejemplo:

- int suma=0;
- for (int i=1; i<=n; i++) {
- suma+=i;
- }

Se puede omitir el bloque, pués solo hay una sentencia.

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

Ejemplo:

- int suma=0;
- for (int i=1; i<=n; i++) {
- suma+=i;
- }

Se puede omitir el bloque, pués solo hay una sentencia.

for

definicion

- *for* (inicializacion; condicion; paso)
- {
- sentencias
- }

Ejemplo:

- int suma=0;
- for (int i=1; i<=n; i++) {
- suma+=i;
- }

Se puede omitir el bloque, pués solo hay una sentencia.

break

¿Qué hace este código?

break

¿Qué hace este código?

```
public static int coincidirNumero1(int numero){
                int coincidencia=-1;
                for (int i=0; i<=numero; i++){
                        if (numero == i){
                                coincidencia=i;
                                break;
                        }
                }
                return coincidencia;
        }
```

break

¿Qué hace este código?

```
public static int coincidirNumero1(int numero){
                int coincidencia=-1;
                for (int i=0; i<=numero; i++){
                        if (numero == i){
                                coincidencia=i;
                                break;
                        }
                }
                return coincidencia;
        }

public static int coincidirNumero2(int numero){
                int coincidencia=-1;
                for (int i=0; i!=numero; i++){
                                coincidencia=i;
                }
                return coincidencia+1;
}
```

break

¿Qué hace este código?

```
public static int coincidirNumero1(int numero){
                int coincidencia=-1;
                for (int i=0; i<=numero; i++){
                        if (numero == i){
                                coincidencia=i;
                                break;
                        }
                }
                return coincidencia;
        }

public static int coincidirNumero2(int numero){
                int coincidencia=-1;
                for (int i=0; i!=numero; i++){
                                coincidencia=i;
                }
                return coincidencia+1;
}
```

La sentencia *break* sale del bucle sin ejecutar el resto de las sentencias del bucle

continue

¿Qué hace este código?

continue

¿Qué hace este código?

```
public static int numeroConsonantes(String palabra){
    int contador=0;
    for (int i=0; i<palabra.length(); i++){
        String letra = palabra.substring(i,i+1);
        if (letra.contains("a") || letra.contains("e") ||
            letra.contains("i") || letra.contains("o") ||
            letra.contains("u") || letra.contains(" ")){
                continue;
        }
        contador++;
    }
    return contador;
}
```

continue

¿Qué hace este código?

```
public static int numeroConsonantes(String palabra){
    int contador=0;
    for (int i=0; i<palabra.length(); i++){
        String letra = palabra.substring(i,i+1);
        if (letra.contains("a") || letra.contains("e") ||
            letra.contains("i") || letra.contains("o") ||
            letra.contains("u") || letra.contains(" ")){
                continue;
        }
        contador++;
    }
    return contador;
}
```

La sentencia *continue* detiene la ejecución del bucle y vuelve al principio de éste.

switch

- Es un operador que permite decidir entre diferentes opciones.

switch

- Es un operador que permite decidir entre diferentes opciones.
- Se inicia con la palabra clave *swtich*

switch

- Es un operador que permite decidir entre diferentes opciones.
- Se inicia con la palabra clave *swtich*
- En funcion del parametro que se le pase se seleccionara una opcion u otra, definidas junto a la palabra clave *case*

switch

- Es un operador que permite decidir entre diferentes opciones.
- Se inicia con la palabra clave *swtich*
- En funcion del parametro que se le pase se seleccionara una opcion u otra, definidas junto a la palabra clave *case*
- En caso que la condicion no coincida con ninguna opcion se ejecuta la sentencia que acompaña a la palabra clave *default*

switch

- Es un operador que permite decidir entre diferentes opciones.
- Se inicia con la palabra clave *swtich*
- En funcion del parametro que se le pase se seleccionara una opcion u otra, definidas junto a la palabra clave *case*
- En caso que la condicion no coincida con ninguna opcion se ejecuta la sentencia que acompaña a la palabra clave *default*
- Se usa la palabra clave *break* para finalizar cada *case*

switch

- Es un operador que permite decidir entre diferentes opciones.
- Se inicia con la palabra clave *swtich*
- En funcion del parametro que se le pase se seleccionara una opcion u otra, definidas junto a la palabra clave *case*
- En caso que la condicion no coincida con ninguna opcion se ejecuta la sentencia que acompaña a la palabra clave *default*
- Se usa la palabra clave *break* para finalizar cada *case*
- Es similar a la usada en *lenguaje C*

switch

- Es un operador que permite decidir entre diferentes opciones.
- Se inicia con la palabra clave *swtich*
- En funcion del parametro que se le pase se seleccionara una opcion u otra, definidas junto a la palabra clave *case*
- En caso que la condicion no coincida con ninguna opcion se ejecuta la sentencia que acompaña a la palabra clave *default*
- Se usa la palabra clave *break* para finalizar cada *case*
- Es similar a la usada en *lenguaje C*

switch

- Es un operador que permite decidir entre diferentes opciones.
- Se inicia con la palabra clave *swtich*
- En funcion del parametro que se le pase se seleccionara una opcion u otra, definidas junto a la palabra clave *case*
- En caso que la condicion no coincida con ninguna opcion se ejecuta la sentencia que acompaña a la palabra clave *default*
- Se usa la palabra clave *break* para finalizar cada *case*
- Es similar a la usada en *lenguaje C*

switch

```
switch (condicion){
   case valor1:
      sentencias;
      break;
   case valor2:
      sentencias;
      break;

   ..............

   default:
      sentencias
}
```

switch

```
switch (condicion){
   case valor1:
      sentencias;
      break;
   case valor2:
      sentencias;
      break;

   ..............

   default:
      sentencias
}
```

```
String dia;
switch (d){
   case 1:
      dia="Lunes";
      break;
   case 2:
      dia="Martes";
      break;

   ..............

   default:
      dia="Domingo";
}
```

switch

```
switch (condicion){
   case valor1:
      sentencias;
      break;
   case valor2:
      sentencias;
      break;

   ..............

   default:
      sentencias
}
```

```
String dia;
switch (d){
   case 1:
      dia="Lunes";
      break;
   case 2:
      dia="Martes";
      break;

   ..............

   default:
      dia="Domingo";
}
```

¿Cuántos case debe haber en el último ejemplo?

Bucles anidados

```
for (int i=0; i<3;i++){
    for (int j=0; j<3;j++){
        System.out.println(i*j);
    }
}
```

¿Cuál es la salida de este código?

if-else

- Podemos definir la estructura de control *if-else* de la siguiente forma:

if-else

- Podemos definir la estructura de control *if-else* de la siguiente forma:
- *condicion ? valorCierto : valorFalso*

if-else

- Podemos definir la estructura de control *if-else* de la siguiente forma:
- *condicion ? valorCierto : valorFalso*
- Ejemplo:

if-else

- Podemos definir la estructura de control *if-else* de la siguiente forma:
- *condicion ? valorCierto : valorFalso*
- Ejemplo:
- String mensaje = (numero % 2 ==0) ? "Es par": "Es impar";

if-else

- Podemos definir la estructura de control *if-else* de la siguiente forma:
- *condicion ? valorCierto : valorFalso*
- Ejemplo:
- String mensaje = (numero % 2 ==0) ? "Es par": "Es impar";

if-else

- Podemos definir la estructura de control *if-else* de la siguiente forma:
- *condicion ? valorCierto : valorFalso*
- Ejemplo:
- String mensaje = (numero % 2 ==0) ? "Es par": "Es impar";

Concatenar cadenas

- Se trata de unir dos o mas cadenas.

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;
- ¿Cual es el mensaje final?

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;
- ¿Cual es el mensaje final?
- También se pueden usar otros tipos además de *String*

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;
- ¿Cual es el mensaje final?
- También se pueden usar otros tipos además de *String*
- int numero=66666666;

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;
- ¿Cual es el mensaje final?
- También se pueden usar otros tipos además de *String*
- int numero=66666666;
- String mensajeFinal = ”Telefono de+ ” ”+mensaje2+” ”+numero;

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;
- ¿Cual es el mensaje final?
- También se pueden usar otros tipos además de *String*
- int numero=66666666;
- String mensajeFinal = ”Telefono de+ ” ”+mensaje2+” ”+numero;
- En este caso hay una conversión de tipos *(casting)* a tipo *String*

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;
- ¿Cual es el mensaje final?
- También se pueden usar otros tipos además de *String*
- int numero=66666666;
- String mensajeFinal = ”Telefono de+ ” ”+mensaje2+” ”+numero;
- En este caso hay una conversión de tipos *(casting)* a tipo *String*

Concatenar cadenas

- Se trata de unir dos o mas cadenas.
- Usamos el operador $+$
- Ejemplo:
- String mensaje1 =”hola”; String mesanje2=Ruben”;
- String mensajeFinal = mensaje1+ ” ”+mensaje2;
- ¿Cual es el mensaje final?
- También se pueden usar otros tipos además de *String*
- int numero=66666666;
- String mensajeFinal = ”Telefono de+ ” ”+mensaje2+” ”+numero;
- En este caso hay una conversión de tipos *(casting)* a tipo *String*

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

```
public class Argumentos{
    public static void main(String[] args){
        System.out.println("Argumento 1: "+args[0]);
        System.out.println("Argumento 2: "+args[1]);
    }
}
```

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

```
public class Argumentos{
    public static void main(String[] args){
        System.out.println("Argumento 1: "+args[0]);
        System.out.println("Argumento 2: "+args[1]);
    }
}
```

1. ¿Si se ejecuta java Argumentos uno dos? ¿Cuál es la salida?

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

```
public class Argumentos{
    public static void main(String[] args){
        System.out.println("Argumento 1: "+args[0]);
        System.out.println("Argumento 2: "+args[1]);
    }
}
```

1. ¿Si se ejecuta java Argumentos uno dos? ¿Cuál es la salida?
2. ¿Si se ejecuta java Argumentos uno ? ¿Cuál es la salida?

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

```
public class Argumentos{
    public static void main(String[] args){
        System.out.println("Argumento 1: "+args[0]);
        System.out.println("Argumento 2: "+args[1]);
    }
}
```

1. ¿Si se ejecuta java Argumentos uno dos? ¿Cuál es la salida?
2. ¿Si se ejecuta java Argumentos uno ? ¿Cuál es la salida?
3. ¿Si se ejecuta java Argumentos uno dos tres? ¿Cuál es la salida?

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

```
public class Argumentos{
    public static void main(String[] args){
        System.out.println("Argumento 1: "+args[0]);
        System.out.println("Argumento 2: "+args[1]);
    }
}
```

1. ¿Si se ejecuta java Argumentos uno dos? ¿Cuál es la salida?
2. ¿Si se ejecuta java Argumentos uno ? ¿Cuál es la salida?
3. ¿Si se ejecuta java Argumentos uno dos tres? ¿Cuál es la salida?

Argumentos en la linea de comandos

- Son los parametros que el programa recibe desde la linea de comandos.
- Ejemplo cuando ejecutamos en GNU/Linux *ls -la*
- El comando es *ls* y los parametros son *la*
- En caso de java: *java Programa argumento1 argumento2 . . .*
- Se acceden a traves de String[ ]
- *public static void main(String[ ] args)*
- Ejemplo:

```
public class Argumentos{
    public static void main(String[] args){
        System.out.println("Argumento 1: "+args[0]);
        System.out.println("Argumento 2: "+args[1]);
    }
}
```

1. ¿Si se ejecuta java Argumentos uno dos? ¿Cuál es la salida?
2. ¿Si se ejecuta java Argumentos uno ? ¿Cuál es la salida?
3. ¿Si se ejecuta java Argumentos uno dos tres? ¿Cuál es la salida?

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*
- El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*
- El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.
- .2 indica el número de decimales

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*
- El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.
- .2 indica el número de decimales
- La *f* indica que el número es de tipo float o double.

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*
- El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.
- .2 indica el número de decimales
- La *f* indica que el número es de tipo float o double.
- %n indica un salto de línea. Equivale a $\backslash n$. Con printf podemos usar ambos para hacer un salto de línea.

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*
- El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.
- .2 indica el número de decimales
- La *f* indica que el número es de tipo float o double.
- %n indica un salto de línea. Equivale a $\backslash n$. Con printf podemos usar ambos para hacer un salto de línea.
- La salida por pantalla es 12,37

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*
- El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.
- .2 indica el número de decimales
- La *f* indica que el número es de tipo float o double.
- %n indica un salto de línea. Equivale a $\backslash n$. Con printf podemos usar ambos para hacer un salto de línea.
- La salida por pantalla es 12,37

printf

- *printf* nos permite dar formato a los datos que se imprimen en pantalla.
- Ejemplo para mostrar el número *12.3698* con dos decimales usamos:
- *System.out.printf(" %.2f %n", 12.3698);*
- El primer % indica que en esa posición se va a escribir un valor. El valor a escribir se encuentra a continuación de las comillas.
- .2 indica el número de decimales
- La *f* indica que el número es de tipo float o double.
- %n indica un salto de línea. Equivale a $\backslash n$. Con printf podemos usar ambos para hacer un salto de línea.
- La salida por pantalla es 12,37

Especificadores para printf

Especificador	Salida	Ejemplo
%b	Valor booleano	true o false
%c	un caracter	'd'
%d	un número entero	200
%f	un número en coma flotante	45.4689
%e	un número en notación científica	4.556000e+01
%s	un string	"hola mundo"

Especificadores para printf

Especificador	Salida	Ejemplo
%b	Valor booleano	true o false
%c	un caracter	'd'
%d	un número entero	200
%f	un número en coma flotante	45.4689
%e	un número en notación científica	4.556000e+01
%s	un string	"hola mundo"

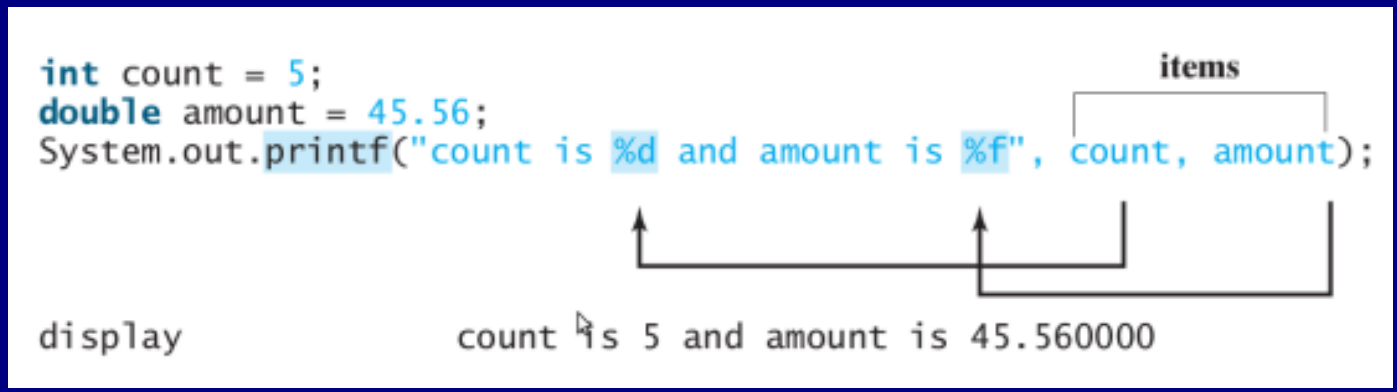


Figura : Ejemplo de printf

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f%n", n);

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f%n", n);
- 1,250

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f%n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f%n", n);

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f%n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f%n", n);
- +1,250

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.
- n = 1,25 x = 10

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.
- n = 1,25 x = 10
- double n = 1.25036; System.out.printf(" %+10.2f %n", n);

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.
- n = 1,25 x = 10
- double n = 1.25036; System.out.printf(" %+10.2f %n", n);
- bbbbb+1.25 Donde b representa un espacio en blanco.

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.
- n = 1,25 x = 10
- double n = 1.25036; System.out.printf(" %+10.2f %n", n);
- bbbbb+1.25 Donde b representa un espacio en blanco.
- double n = 1.25036; System.out.printf(" %+010.2f %n", n);

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.
- n = 1,25 x = 10
- double n = 1.25036; System.out.printf(" %+10.2f %n", n);
- bbbbb+1.25 Donde b representa un espacio en blanco.
- double n = 1.25036; System.out.printf(" %+010.2f %n", n);
- b+000001.25

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.
- n = 1,25 x = 10
- double n = 1.25036; System.out.printf(" %+10.2f %n", n);
- bbbbb+1.25 Donde b representa un espacio en blanco.
- double n = 1.25036; System.out.printf(" %+010.2f %n", n);
- b+000001.25

Ejemplos de printf

¿Cuál es la salida de printf?

- double n = 1.25036; System.out.printf(" %.3f %n", n);
- 1,250
- double n = 1.25036; System.out.printf(" %+.3f %n", n);
- +1,250
- int x = 10; System.out.printf(" %d %n", x);
- 10
- double n = 1.25036; int x = 10; System.out.printf("n = %.2f x = %d %n", n, x);.
- n = 1,25 x = 10
- double n = 1.25036; System.out.printf(" %+10.2f %n", n);
- bbbbb+1.25 Donde b representa un espacio en blanco.
- double n = 1.25036; System.out.printf(" %+010.2f %n", n);
- b+000001.25

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);
- Mostrar la cadena "Total:"con un ancho de 10 caracteres y alineada a la izquierda

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);
- Mostrar la cadena "Total:"con un ancho de 10 caracteres y alineada a la izquierda
- System.out.printf(" %-10s", "Total:"); (Por defecto se alinea a la derecha).

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);
- Mostrar la cadena "Total:"con un ancho de 10 caracteres y alineada a la izquierda
- System.out.printf(" %-10s", "Total:"); (Por defecto se alinea a la derecha).
- System.out.printf(" %8d %8s %8.1f\n", 1234, "Java", 5.6);

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);
- Mostrar la cadena "Total:"con un ancho de 10 caracteres y alineada a la izquierda
- System.out.printf(" %-10s", "Total:"); (Por defecto se alinea a la derecha).
- System.out.printf(" %8d %8s %8.1f\n", 1234, "Java", 5.6);
- System.out.printf(" %-8d %-8s %-8.1f \n", 1234, "Java", 5.6);

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);
- Mostrar la cadena "Total:"con un ancho de 10 caracteres y alineada a la izquierda
- System.out.printf(" %-10s", "Total:"); (Por defecto se alinea a la derecha).
- System.out.printf(" %8d %8s %8.1f\n", 1234, "Java", 5.6);
- System.out.printf(" %-8d %-8s %-8.1f \n", 1234, "Java", 5.6);
- Muestra:

Ejercicios de printf

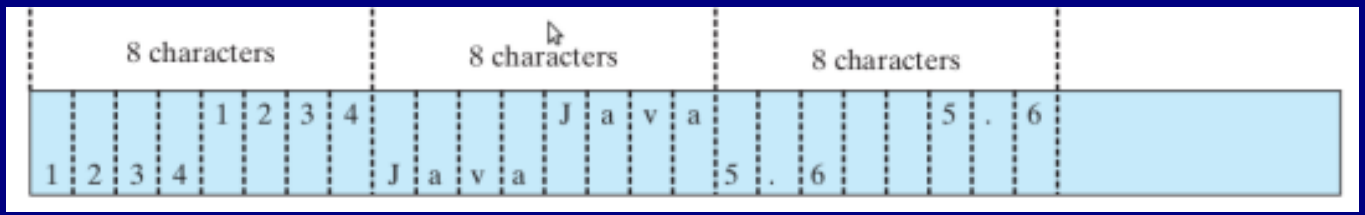
¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);
- Mostrar la cadena "Total:"con un ancho de 10 caracteres y alineada a la izquierda
- System.out.printf(" %-10s", "Total:"); (Por defecto se alinea a la derecha).
- System.out.printf(" %8d %8s %8.1f\n", 1234, "Java", 5.6);
- System.out.printf(" %-8d %-8s %-8.1f \n", 1234, "Java", 5.6);
- Muestra:

Ejercicios de printf

¿Cómo debemos formatear con printf?

- Mostrar el número 1.22 en un ancho de campo de 10 caracteres y con dos decimales.
- double precio = 1.22; System.out.printf(" %10.2f", precio);
- Mostrar la cadena "Total:"con un ancho de 10 caracteres y alineada a la izquierda
- System.out.printf(" %-10s", "Total:"); (Por defecto se alinea a la derecha).
- System.out.printf(" %8d %8s %8.1f\n", 1234, "Java", 5.6);
- System.out.printf(" %-8d %-8s %-8.1f \n", 1234, "Java", 5.6);
- Muestra:



Preguntas