Tema 2: Variables y control de flujo

Las variables en Java

- Los datos, según cambien su valor o no durante la ejecución del programa:
  - **Variables**: sirven para almacenar datos durante la ejecución del programa
  - **Constantes**: sus valores son fijos

Variable = zona de la memoria a la que se le puede asignar un valor que puede cambiar mientras el programa se ejecuta

- **Java es un lenguaje fuertemente tipado** *(strongly-typed):* Toda variable debe ser declarada antes de poder usarla indicándole un nombre y el tipo de datos.

- **Declaración de una variable en Java:** ***tipo nombreVariable;***

  Ejemplo:

```
// Declaro las variables var1, var2 y var3 de tipo entero
int var1;
int var2, var3;
```

IDENTIFICADOR

Concepto de paquete

- Un paquete (**package**) es un conjunto de clases e interfaces relacionadas.
- **Físicamente son directorios donde se guardan las clases** e interfaces de Java.
- Similar al concepto de módulo en software.
- Mecanismo de agrupación con dos objetivos:
  - Evitar conflictos de nombres (un gran problema al desarrollar código reusable)
  - Controlar el nivel de acceso (encapsulación)

Concepto de paquete

- ✓ Las clases e interfaces que son parte de la plataforma Java están agrupadas en paquetes de acuerdo a su función. ***Ejemplos***:

  - **java.util:** ofrece “utilidades” (clase Date, ej.).
  - **java.io:** para manejo de entrada/salida.
  - **java.awt, javax.swing:** para manejo de la GUI.
  - **java.sql**: para manejo de base de datos.

Concepto de paquete

- Definir a qué paquete pertenece una clase: usamos la sentencia **package** como primera línea del código fuente.
- Separamos los subpaquetes mediante puntos.

➢ **Convención para el nombre del paquete**: dominio de la empresa invertido + sistema + subsistema + …

```
package es.miempresa.renta;
public class Empresa {
    // implementación de la clase
}
```

Lo anterior hace que se guarde la clase *Empresa* en la estructura de directorios siguiente:

…\es\miempresa\renta

Concepto de paquete

- Para usar clases de otros paquetes usamos la sentencia **import** para "importar" las clases que necesitamos:

```
package es.miempresa.testing;
import  es.miempresa.renta.Empresa;
public  class Testeo{
        Empresa e  = new Empresa();
         // código
}
```

…También podríamos usar:

```
import  es.miempresa.renta.*;
```

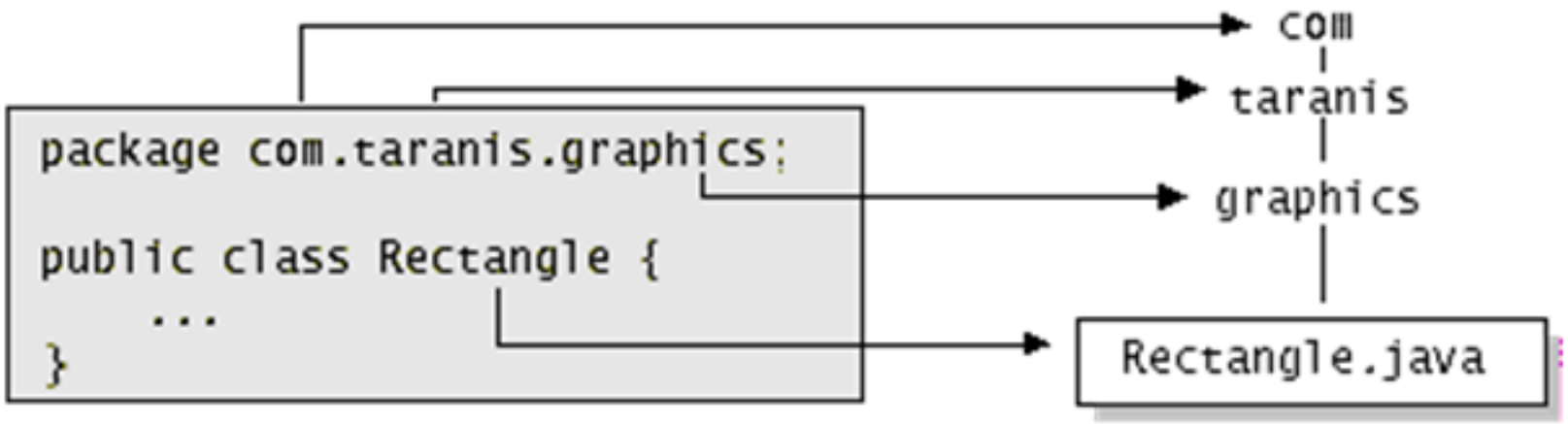
Concepto de paquete

- ***Más ejemplos:***

1) Si queremos usar la clase Scanner:

import java.util.Scanner;

2) La empresa *Taranis* tiene un paquete gráfico con la clase *Rectangle:*



Palabras reservadas: propias de Java (50)

- ***No se puedan usar como nombre de variables, clases…***

TABLE 1-1 Complete List of Java Keywords

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while
assert	enum				

Algunas convenciones de Código para Java

- Recomendaciones para que nuestro código cumpla unos estándares internacionales.
  - Oficialmente (inglés)**:**

http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

  - Convenciones de Código para Java en español:

http://systempix.com/descargas/Convenciones_Codigo_Java.pdf

Clases:

- Primera letra de cada palabra en mayúscula.
- Sustantivo para las clases y adjetivos para interfaces.
- **Ejemplos**: *Cliente, ImagenAnimada, HolaMundo, CalculadoraBasica*

Variables

- Primera letra en minúscula y luego la primera letra de las siguientes palabras en mayúscula.
- Nombres significativos y cortos.
- **Ejemplos**: *i, c, cont, dni, dineroCuenta, anchoBoton*

Tipos primitivos

- Almacenan un único dato simple.
- Se usan para representar valores enteros, decimales…
- El tamaño de los tipos primitivos no depende del sistema operativo o de la arquitectura.

- **Tipos de variables enteras**

Tipo	Ejemplo	Tamaño	Rango
byte	1	8 bits	-128 a 127
short	5	16 bits	-32,768 a 32767
int	1024	32 bits	-2,147,483,648 a 2,147,483,648
long	47854589	64 bits	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807

- ***Ejemplo:*** *¿cómo representarías el número de personas que hay en el mundo?*

Fuente: http://www.census.gov/popclock/

Actividad: Programa con variables

```
public class EjemploVariables {

    public static void main(String[] args) {
        // Declaración de variables
        int var1; // esto la variable var1
        int var2, var3; // esto declara las variables var2 y var3

        // Lo que hace el programa
        System.out.println("- Compienzo del programa");
        var1 = 1024; // asigno 1024 a var1
        System.out.println("var1: " + var1);
        var2 = var1 / 2;
        var3 = var1 + var2;
        System.out.print("var2 es igual a var1 / 2: " + var2);
        System.out.print("var3 es igual a var1 + var2: " + var3);
        System.out.println();
        System.out.println("- Fin del programa");
    }
}
```

Tipos primitivos

- **Tipos asociados a números decimales**

Tipo	Ejemplo	Tamaño	Rango
float	9.12541	32 bits	6 dígitos significativos (10^{-46}, 10^{38})
double	554.15155454	64 bits	15 dígitos significativos (10^{-324}, 10^{308})

- **Por defecto** Java considera un valor con decimales como de **tipo double**
- **Ejemplo:**

```
float interes;
double capital;
interes = 6.25F; // para que asigne a interes un valor de tipo float
double = 3.7258E+9;
```

Tipos primitivos

- **Tipo booleano**

Tipo	Ejemplo	Tamaño	Rango
boolean	true	8 bits	True / False

- **Tipo carácter**

Tipo	Ejemplo	Tamaño	Rango
char	'A'	16 bits	Unicode

Para asignar un valor a una variable de tipo carácter se debe encerrar el **carácter entre comillas simples** (ojo).

- **Ejemplo:**

```
boolean finTemporada = false;
char ch;
ch = 'X';
System.out.println("finTemporada: "+ finTemporada);
System.out.println("Valor de ch: "+ ch);
```

Tipos primitivos

- **RESUMEN**

tipo	descripción	rango de valores
`boolean`	valor lógico	true o false
`char`	carácter unicode (16 bits)	los caracteres internacionales
`byte`	entero de 8 bits con signo	-128..127
`short`	entero de 16 bits con signo	-32768..32767
`int`	entero de 32 bits con signo	-2.147.483.648.. 2.147.483.647
`long`	entero de 64 bits con signo	aprox. $9.0*10^{18}$
`float`	nº real de 32 bits	unos 6 dígitos
`double`	nº real de 64 bits	unos 15 dígitos

String: cadenas de caracteres en Java

- ***String:*** es una clase especial usada para representar **cadenas de caracteres** en Java.
- Los **objetos** de la clase String se pueden crear de **forma implícita o explícita**. Ejemplos:

Implícita:

```
System.out.println("Muestra este mensaje por consola");
```

Explícita:

```
String mensaje = "Muestra este mensaje por consola";
System.out.println(mensaje);
```

String: cadenas de caracteres en Java

- ***Concatenar cadenas***

Usamos el operador **+** para concatenar cadenas de caracteres y se usa para objetos de tipo String con otros tipos de datos.

Ejemplos:

```
double  tC;
// … Código que lee  tC por consola…
System.out.println("la temperatura centígrada es " + tC);

int ia, ib, iSuma;
// … Código que lee valores de ia, ib por teclado
iSuma = ia + ib;
System.out.println("El resultado de la suma es: "+iSuma);
```

String: cadenas de caracteres en Java

- El operador + funciona también funciona para concatenar cadenas de caracteres entre sí.

Ejemplo:

```
String nombre="Juan  ";
nombre = nombre + "García";
System.out.println(nombre);
```

String: cadenas de caracteres en Java

- ***¿Cómo funciona el operador +?***
  - Si ambos operandos son String o String y “otra cosa”, + concatena.
  - Si ambos operandos son números, + suma.
- *Ejemplos:*

```
String a = "String";
int b = 3;
int c = 7;
System.out.println(a + b + c);      // String37
System.out.println(a + (b + c));    // String10
```

- *Ejemplo: ¿salida de este código?*

```
int b = 2;
System.out.println("" + b + 3);
System.out.println(b + 3);
```

Métodos importantes de String

- Los más comunes y que podemos encontrar en el examen son los siguientes:

- **charAt()** Returns the character located at the specified index
- **concat()** Appends one String to the end of another ("+" also works)
- **equalsIgnoreCase()** Determines the equality of two Strings, ignoring case
- **length()** Returns the number of characters in a String
- **replace()** Replaces occurrences of a character with a new character
- **substring()** Returns a part of a String
- **toLowerCase()** Returns a String with uppercase characters converted
- **toString()** Returns the value of a String
- **toUpperCase()** Returns a String with lowercase characters converted
- **trim()** Removes whitespace from the ends of a String

Métodos importantes de String

- *Ejemplos*:

```
String x = "airplane";
System.out.println( x.charAt(2) );
```

```
String x = "Java";
System.out.println(x.charAt(9));
```

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException:
String index out of range: 9
```

Métodos importantes de String

- ***Concatenar cadenas*:**

```
String x = "taxi";
System.out.println( x.concat(" cab") ); // output is "taxi cab"
```

El operador + es equivalente al método concat:

```
String x = "library";
System.out.println( x + " card");

// output is "library card"
```

Métodos importantes de String

- String tiene el método **length()** que nos permite calcular la longitud de la cadena de caracteres.

Ejemplos:

```
String s1 = "test";
int longitud = s1.length();
System.out.println(longitud); // 4

String s2 = "This is a String";
System.out.println(s2.length()); // 16
```

Métodos importantes de String

public String substring(int begin)
public String substring(int begin, int end)

- Devuelve una subcadena a partir de la cadena original.
  - El primer argumento representa el índice de inicio de la subcadena que empieza en 0. Con un solo argumento, la subcadena se extrae hasta el final de la cadena original.
  - El segundo argumento (end) representa la posición final de la subcadena pero no parte de 0 como el anterior (réstale 1).

Ejemplos: ¿salida?

```
String x = "0123456789";
System.out.println(x.substring(5));     // 56789
System.out.println(x.substring(5, 8)); // 567
```

Métodos importantes de String

public String trim()

- Devuelve una cadena eliminandos los espacios en blanco (tanto por delante como por detrás).

Ejemplos: ¿salida?

```
String x1 = "                    Java power!                    ";
System.out.println(x1.trim().length());
System.out.println(x1.trim());
```

Métodos importantes de String

- Para comparar cadenas usamos los métodos ***equals()*** o ***equalsIgnoreCase()***

Ejemplos:

```
String s1 = "hola, bienvenid@ a java";
String s2 = "hola, bienvenid@ a java";
String s3 = "HOLA, BIENVENID@ A JAVA";
String s4 = s3.toLowerCase();
String s5 = s2.toUpperCase();

System.out.println(s1.equals(s2));  // true
System.out.println(s1.equals(s3));  // false
System.out.println(s1.equals(s4));  // true
System.out.println(s3.equals(s5));  // true
System.out.println(s1.equalsIgnoreCase(s3)); // true
```

Métodos importantes de String

- Para reemplezar caracteres usamos ***replace()***

Ejemplos:

```
String s1 = "hola, bienvenid@ a java";
System.out.println(s1.replace('a', '-'));
// output: hol-, bienvenid@ - j-v-

String s2 = "What do you think about Java?";
System.out.println(s2.replace(' ', '|'));
// output: What|do|you|think|about|Java?
```

Actividad 1 en Eclipse: Entrada y salida de datos por consola (a)

```java
import java.util.Scanner;

public class HolaMundoNombre {

    public static void main(String[] args) {
        // Esta clase permite leer datos por teclado
        Scanner scanner = new Scanner(System.in);

        // Muestra mensaje al usuario
        System.out.println("Introduce tu nombre:");

        // Lee el nombre por teclado
        String nombre = scanner.nextLine();

        // Muestra por consola el nombre introducido por el usuario
        System.out.println("Hola " + nombre + ", bienvenid@ al mundo de Java :-) ");
    }

}
```

Actividad 2 en Eclipse: Entrada y salida de datos por consola (b)

```java
import java.util.Scanner;

public class HolaMundoVariables {

    public static void main(String[] args) {
        // Esta clase permite leer datos por teclado
        Scanner scanner = new Scanner(System.in);

        // Muestra mensaje al usuario
        System.out.println("Introduce nombre edad altura:");

        // Lee el nombre por teclado
        String nombre = scanner.next();

        // Lee el nombre por teclado
        int edad = scanner.nextInt();

        // Lee el nombre por teclado
        double altura = scanner.nextDouble();

        // Muestra por consola el resultado
        System.out.println("Hola " + nombre + ", tu edad es " + edad + " y tu altura es "+ altura);
    }

}
```

Operadores aritméticos

OPERADOR	OPERACIÓN
+	Suma
-	Resta
*	Multiplicación
/	División
%	Resto de una división entre enteros

Operadores aritméticos

- ***Son operadores binarios.*** *Ejemplo:*

```
int op1, op2;
int result;
// ... Lectura de op1 y op2 por consola
result = op1 + op2;
result = op1 – op2;
result = op1 * op2;
result = op1 / op2;
result = op1 % op2;
```

Operadores aritméticos

int op1, op2;

+= → op1 += op2 → op1 = op1 + op2

-= → op1 -= op2 → op1 = op1 - op2

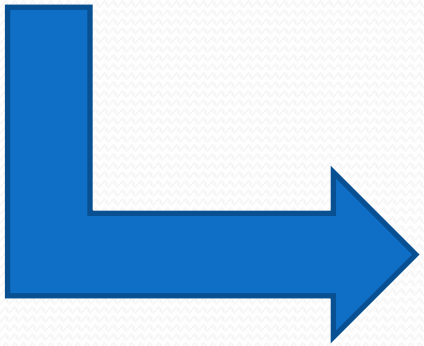
***=** → op1 *= op2 → op1 = op1 * op2

/= → op1 /= op2 → op1 = op1 / op2

%= → op1 %= op2 → op1 = op1 % op2

Operadores unarios

- Sólo requieren un operando.
- Algunos de ellos: ++, -- , -, !
- ***Ejemplo***:



```
int resultado = 1;

resultado--;
System.out.println(resultado);


resultado++;
System.out.println(resultado);


resultado = -resultado;
System.out.println(resultado);


boolean falso = false;
System.out.println(falso);
System.out.println(!falso);
```

Operadores unarios

- Pueden aparecer como prefijo o sufijo. ***Ejemplos***:

  x++; ó x--;

  ++x; ó --x;

- ***Prefijo***: primero incrementa/decrem. y luego usa el valor.

  Sufijo: primero usa el valor y luego incrementa/decrem.

```
int resultado = 1;
resultado++;
System.out.println(resultado);
System.out.println(resultado++);
System.out.println(resultado);
System.out.println(++resultado);
```

Actividades… ☺

- Escribe un programa llamado "**CalculadoraBasica.java**" que pida al usuario 2 valores y calcule la suma, resta, multiplicación y división de esos valores mostrando el resultado de cada una de las operaciones por pantalla.

- Escribe un programa llamado "**ConversorTemperaturas.java**" y que pida al usuario una temperatura en ºC, mostrando por pantalla el resultado en F y K.

  Ayuda:

$$K = C + 273{,}15$$

$$F = C \times 1{,}8 + 32$$

Ejemplo 1: CalculadoraBasica.java

```java
import java.util.Scanner;

public class CalculadoraBasica {
    public static void main(String[] args) {
        double valor1, valor2, result;
        Scanner scanner = new Scanner(System.in);

        System.out.print("Introduce valor 1:");
        valor1 = scanner.nextDouble();

        System.out.print("Introduce valor 2:");
        valor2 = scanner.nextDouble();

        // Realizo los cálculos y muestro los resultados
        result = valor1+ valor2;
        System.out.println("valor1 + valor2 = " + result);

        result = valor1 - valor2;
        System.out.println("valor1 - valor2 = " + result);

        result = valor1 * valor2;
        System.out.println("valor1 * valor2 = " + result);

        result = valor1 / valor2;
        System.out.println("valor1 / valor2 = " + result);
    }
}
```

Ejemplo 2: ConversorTemperaturas.java

```java
import java.util.Scanner;

public class ConversorTemperaturas {
    public static void main(String[] args) {
        // Declaración de variables
        double tC, tK, tF;

        // Lectura por consola de la temperatura en C
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce temperatura en C:");
        tC = scanner.nextDouble();

        // Calculo de la temperatura en K y F según las fórmulas:
        // K = C + 273,15
        // F = C × 1,8 + 32
        tK = tC + 273.15;
        tF = tC*1.8 + 32;

        // Muestro resultados por consola
        System.out.println("Temperatura en K = " + tK);
        System.out.println("Temperatura en F = " + tF);
    }
}
```

Constantes en Java

- Es una variable que mantiene un valor fijo a lo largo de toda la vida del programa.
- Es recomendable escribirlas **en mayúsculas.**
- Para declarar una constante, usamos ***static*** y ***final***. Se suelen inicializar en su declaración.

static final tipo NOMBRE_CONSTANTE = valor;

Ejemplos:

```
static final double IVA = 0.21;
static final double PI  = 3.1416;
static final DIAS_LABORABLES = 5;
```

Constantes en Java

- ***Ejemplo***: cálculo del IVA de un producto

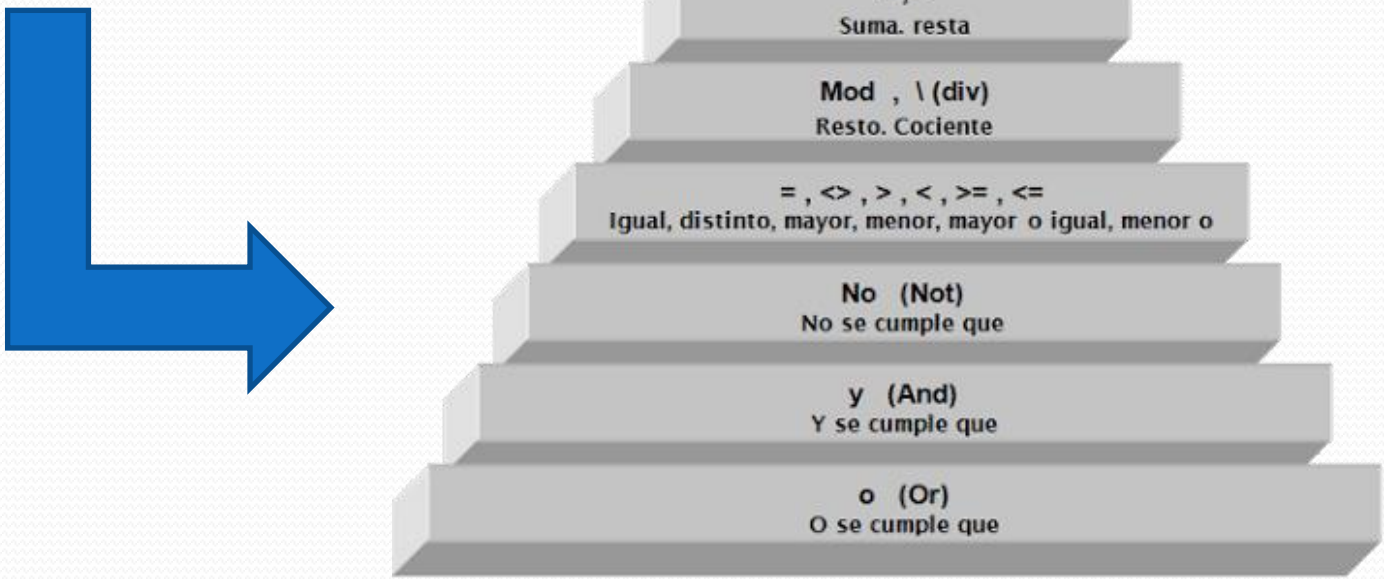
```
public class IvaProducto
{
    // Constante IVA
    final static double IVA = 0.21;

    public static void main(String[] args)
    {
        // Simula el precio del producto
        double precio = 259;
        System.out.println("El precio del producto (sin IVA) es: " + precio);

        // Calculamos el precio del producto con el IVA
        precio = precio + precio*IVA;
        System.out.println("El precio del producto (con IVA) es: " + precio);
    }
}
```

Precedencia de operadores

- En el siguiente código:

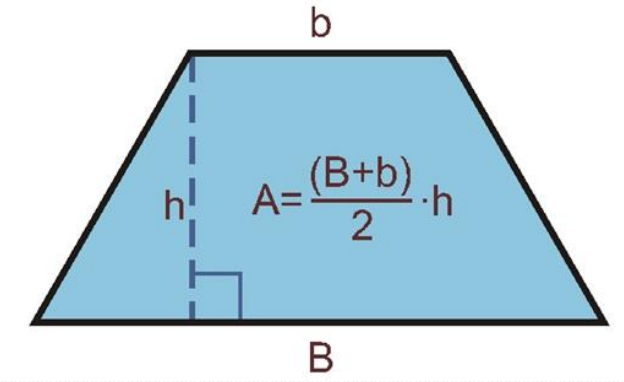
```
int j = 1 + 3 * 4 / 3;  // j = ?
```

Prioridad de los operadores



Precedencia de operadores

- Como norma general, **se recomienda usar paréntesis** en expresiones que implican distintos operadores para evitar problemas con el orden de precedencia de los operadores.
- ***Ejemplo***: cálcula el área de un trapecio



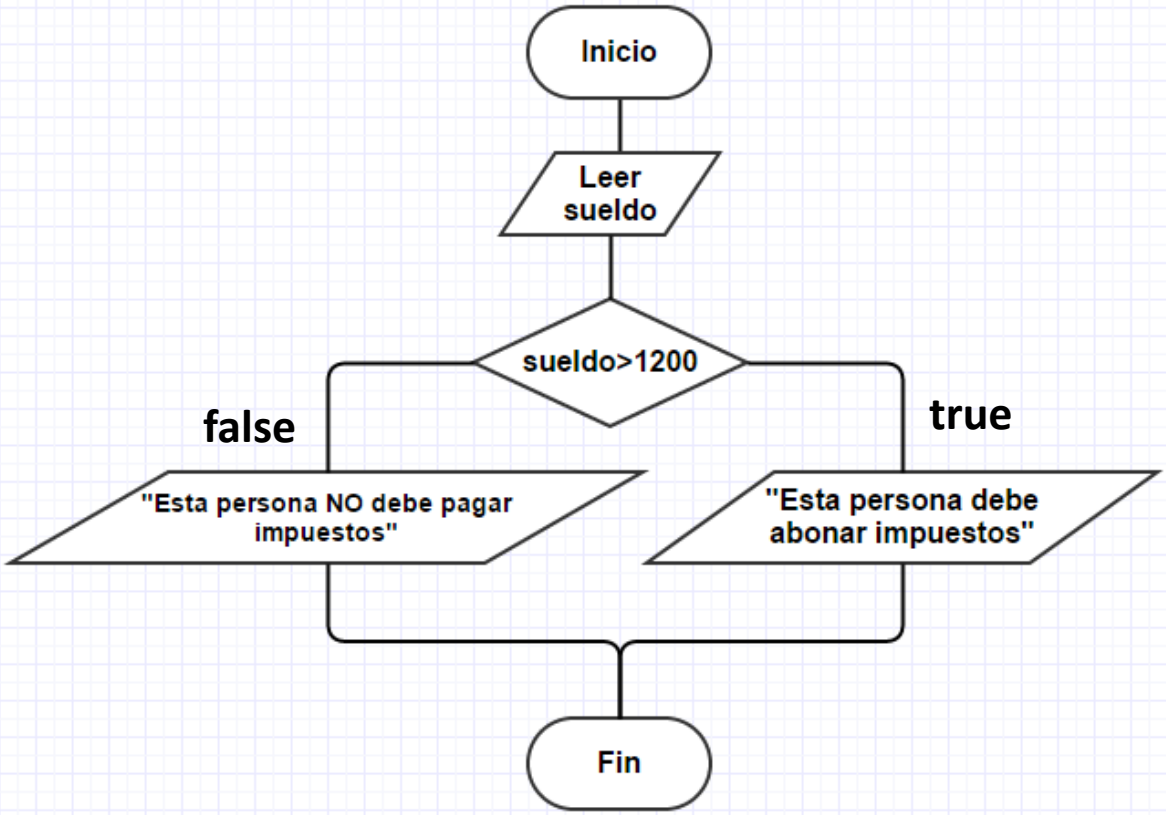

```
area = (B + b)*(h/2);
area = ( (B + b)/2 ) * h;
area = 0.5*h*(B + b);
```

Estructuras de control en java

- En Java hay 3 estructuras de decisión:
  1. **Decisión simple**: if - else
  2. **Decisión múltiple**: switch
  3. **Decisión in-line**

1) Decisión simple: if

```
if (condicion)
{
    accion1;
}
else
{
    accion2;
}
```



Estructuras de control en java

1) Decisión simple: if

```
if (condicion1) {
      accion1;
} else if (condicion2) {
     accion2;
} else {
     accion3;
}
```

Otro ejemplo de decisión simple con 2 condiciones y un else por si no se cumplen

Estructuras de control en java

- Una **condición** es una expresión formada por variables, constantes y operadores que se evalúa a ***true*** o ***false***.
- **Operadores relacionales**

Operador	Significado
==	Igual a
!=	Diferente de
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Ejemplo

sueldo == 1200
sueldo != 1200
sueldo > 1200
sueldo < 1200
sueldo >= 1200
sueldo <= 1200

Estructuras de control en java

- **Operadores lógicos: sobre expresiones booleanas**

OPERADOR	DESCRIPCIÓN
&&	Operador and (y)
||	Operador or (o)
!	Operador not (no)

- ✓ **&&** true sólo si los operandos son true
- ✓ **||** true si alguno de los operandos es true
- ✓ **!** negación

Estructuras de control en java

- **&& y ||: evaluación condicional (cortocircuito)**
- **& y | : siempre se evalúan los operandos (¡ojo!)**
- *Ejemplos:*

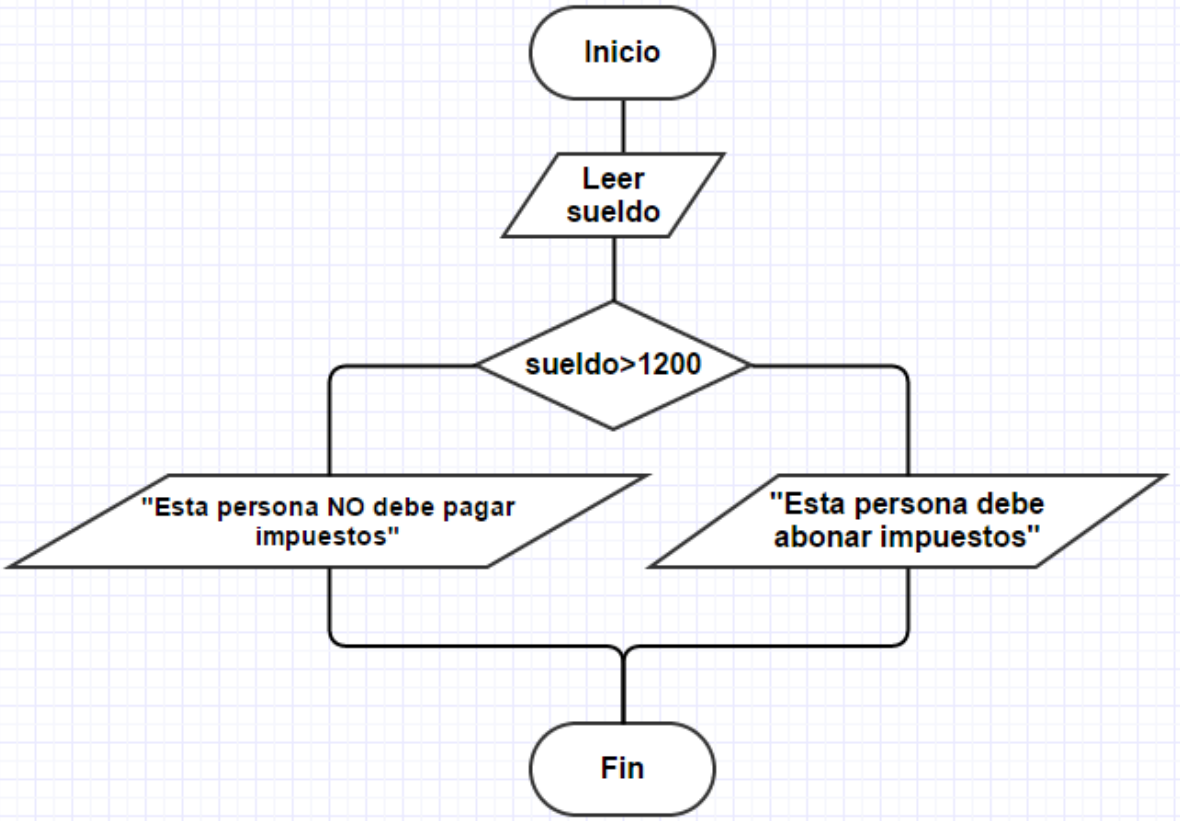
(a == b && c != d && h >= k)

(a < b || c != d || h <= k)

```
if ( (edad>=18) && (edad<=65) ) {
    System.out.println("Tu edad está entre 18 y 65 años");
} else {
    System.out.println("Eres menor de 18 años o mayor de 65 años");
}
```

Estructuras de control en java

- **Ejemplo**: escribe un programa llamado **Impuestos.java** que implemente el siguiente diagrama de flujo:



Estructuras de control en java

- **Ejemplo**: escribe un programa llamado **Impuestos.java**

```java
import java.util.Scanner;

public class Impuestos {
    public static void main(String[] args) {
        // Declaración de variables
        double sueldo;

        // Lectura por consola de la temperatura en C
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce sueldo (euros):");
        sueldo = scanner.nextDouble();

        if (sueldo > 1200) {
            System.out.println("Esta persona debe pagar impuestos");
        } else {
            System.out.println("Esta persona NO debe pagar impuestos");
        }
    }
}
```

Estructuras de control en java

- **Actividad 1**: escribe un programa llamado **ParImpar.java** que pida un número entero a un usuario y que muestre por pantalla si el número es par o impar.

Estructuras de control en java

- **Actividad 2**: escribe un programa llamado **ComparaNumeros.java** que pida 2 número a un usuario y que muestre por pantalla cuál es mayor y cuál es menor.

¿ a > b ?

Decisión múltiple: switch

```
switch( variableEntera )
{
   case valor1:
      accionA;
      accionB;
         :
      break;
   case valor2:
      accionX;
      accionY;
         :
      break;
    :
   default:
      masAcciones;
}
```

- ✓ **variableEntera y valor1, valor2,…** deben ser de tipo entero salvo long (byte, short, int, char)
- ✓ Cada *case* suele ir con un ***break***
- ✓ ***default***: opcional

Decisión múltiple: switch

Actividad: muestra día de la semana

En el siguiente programa, pedimos al usuario que ingrese un día de la semana (entre 1 y 7) y mostramos el nombre del día. Si ingresa cualquier otro valor informamos que el dato ingresado es incorrecto.

- ✓ 1 → Lunes
- ✓ 2 → Martes
- ✓ …
- ✓ 7 → Domingo
- ✓ Cualquier otro valor → Error: el valor debe ser entre 1 y 7

```java
package prueba;
import java.util.Scanner;

public class DemoSwitch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Ingrese un dia de la semana (numero): ");
        int v = scanner.nextInt();
        scanner.close();

        String dia;
        switch (v) {
            case 1:
                dia = "Lunes";      break;
            case 2:
                dia = "Martes";     break;
            case 3:
                dia = "Miercoles";  break;
            case 4:
                dia = "Jueves";     break;
            case 5:
                dia = "Viernes";    break;
            case 6:
                dia = "Sabado";     break;
            case 7:
                dia = "Domingo";    break;
            default:
                dia = "Dia incorrecto... El valor debe ser entre 1 y 7.";
        }

        System.out.println(dia);
    }
}
```

Decisión múltiple: switch

Actividad a realizar: muestra departamento.

Escribe un programa que pida al usuario un número entre 1 y 4 de forma que mostrará el nombre del depto. asociado a ese número . Si introduce otro valor distinto informamos diciendo que es un error y el dato es incorrecto.

- ✓ 1 → Desarrollo
- ✓ 2 → RRHH
- ✓ 3 → Finanzas
- ✓ 4 → Marketing
- ✓ Cualquier otro valor → Error: el departamento no existe

```java
import java.util.Scanner;
public class MuestraDepartamento {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Lectura numero de depto.
        System.out.print("Introduce nº de dpto:");
        int numDepto = scanner.nextInt();
        String msg;

        // Procesado
        switch (numDepto) {
            case 1: msg = "Desarrollo";
                    break;
            case 2: msg = "RRHH";
                    break;
            case 3: msg = "Finanzas";
                    break;
            case 4: msg = "Marketing";
                    break;
            default:
                    msg = "Error: el departamento solicitado no existe";
        }

        // Muestra resultado
        System.out.println(msg);
    }
}
```

Decisión *in-line*

- Es una expresión usada a veces…
- Para ello usamos el operador ternario (?) de la forma siguiente:

resultado = (expresion) ? valor1 : valor2;

Ejemplo 1:

```
if (x>y)
    mayor = x;
else
    mayor = y;
```

```
mayor = (x>y) ? x : y;
```

Decisión *in-line*

Ejemplo 2:

```
int numero = 5;

String resultado;

if (numero < 0) {
    resultado = "Negativo";
} else {
    resultado = "Positivo";
}

System.out.println(resultado);
```

Otra forma:

```
resultado = (numero<0) ? "Negativo":"Positivo";
```