Tema 1: Estructura de un programa informático. Tipos de datos simples

1. Iniciación a la programación

1.1 Algoritmo y programa

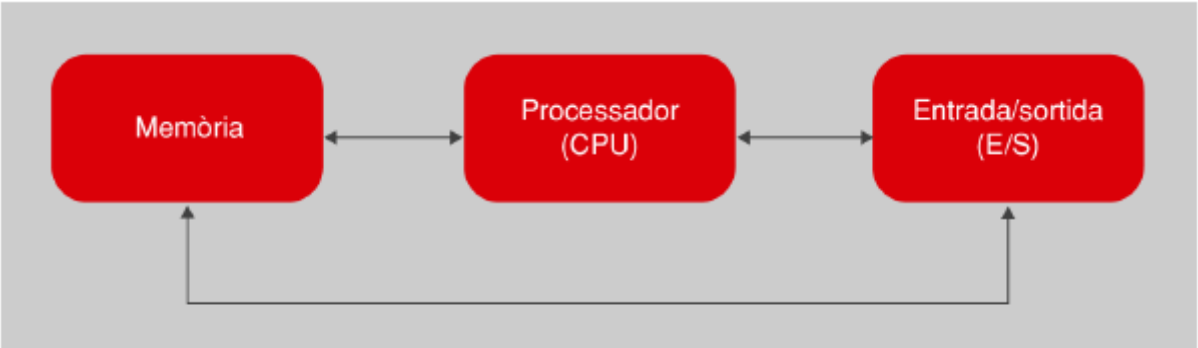
Un **algoritmo** es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad. Es un término utilizado en matemáticas, lógica e informática.

Ejemplos de algoritmos: Operaciones aritméticas como sumas de nºs decimales, receta de cocina, el programa de lavado en una lavadora, etc.].

Un **programa** de computadora o simplemente un programa, es una secuencia de instrucciones, escritas para realizar una tarea específica con una computadora.

1.2 Tipos de órdenes que acepta un ordenador

Las instrucciones que puede realizar un ordenador dependen de los componentes que lo forman:



- **Procesador o CPU (Central Processing Unit)**: Es el componente electrónico dentro de un ordenador que maneja las instrucciones de un programa de computadora realizando las operaciones básicas aritméticas, lógicas, de control y de entrada/salida especificadas por las instrucciones.

- **Memoria**: Permite almacenar datos mientras no están siendo directamente manipulados por el procesador. Cualquier dato que tiene que ser tratado por un programa estará en la memoria. Mediante el programa se puede ordenar al procesador que guarde ciertos datos o que los recupere en cualquier momento.

  Cuando se habla de memoria nos referimos habitualmente a la memoria RAM (Random Memory Access, memoria de acceso aleatorio). Esta memoria no es persistente i desaparece al apagar el equipo.

- **Sistema de entrada/salida (I/O)**: Permite el intercambio de datos con el exterior del ordenador. Las instrucciones procesadas se convierten, de esta forma, en acciones sobre cualquier periférico conectado al ordenador.

Ejemplo: El pizzero con sus herramientas sería el procesador. Los armarios y estanterías de la cocina serían la memoria. El teléfono y el motorista sería la unidad de E/S.

1.3 Crear un programa ejecutable

Para especificar las órdenes que tiene que seguir un ordenador se utiliza un **lenguaje de programación**. Se trata de un lenguaje artificial diseñado expresamente para crear algoritmos que puedan ser ejecutados por un ordenador. Hay múltiples lenguajes de programación diferentes.

Las instrucciones se guardan en uno o varios archivos de texto plano.

Los lenguajes de programación pueden ser compilados o interpretados. También se los puede clasificar en de bajo o de alto nivel.

1.3.1 Lenguajes de bajo nivel

- El lenguaje máquina es aquél que puede ser entendido directamente por el procesador. Cada instrucción se expresa como una secuencia de ceros y unos.

- Cuando se quiere ejecutar un programa en lenguaje máquina, se carga en memoria y el procesador va leyendo y ejecutando las instrucciones.

- En la práctica para programar en código máquina se utiliza el lenguaje **ensamblador**. Al usar este lenguaje se sustituyen los números por palabras mnemotécnicas, resultando un programa más inteligible.

Ventajas de programar en lenguaje ensamblador:

- Se puede controlar completamente el procesador, sus registros y la memoria RAM. Esto hace que los programas sean mucho más rápidos que los escritos en otros lenguajes.

- En general, un algoritmo escrito en ensamblador será más rápido que en otro lenguaje de más alto nivel.

Desventajas:

- Es muy difícil escribir y entender un programa en ensamblador. Se tarda muchísimo más en escribirlo.
- Los programas no son portables. Si se cambia el tipo de procesador hay que cambiar el programa completamente.

TAULA 1.2. Equivalència entre un llenguatge assemblador i el seu llenguatge de màquina associat

Instrucció assemblador	Llenguatge de màquina equivalent
LDA #6	1010100100000110
CMP &3500	11001101000000000110101
LDA &70	1010010101110000
INX	11101111

1.3.2 Lenguajes de alto nivel

- Ejemplos de lenguajes de alto nivel: Java, C#, Visual Basic, Python, Ruby, C, Pascal...
- Las instrucciones (sentencias) no se corresponden a instrucciones del procesador, sino que son más parecidas al lenguaje humano. Generalmente una instrucción de un lenguaje de alto nivel corresponderá a muchas de código máquina.

```
Console.Write("¿Cómo te llamas? ");
nombre = Console.ReadLine();
Console.WriteLine("Encantado de conocerte, " + nombre);
```

- Hay que traducir estas instrucciones de alto nivel a código máquina para que las pueda ejecutar el procesador.
- Los programas no tienen que escribirse pensando en un procesador concreto, y son más legibles ya que están escritos con palabras en inglés.

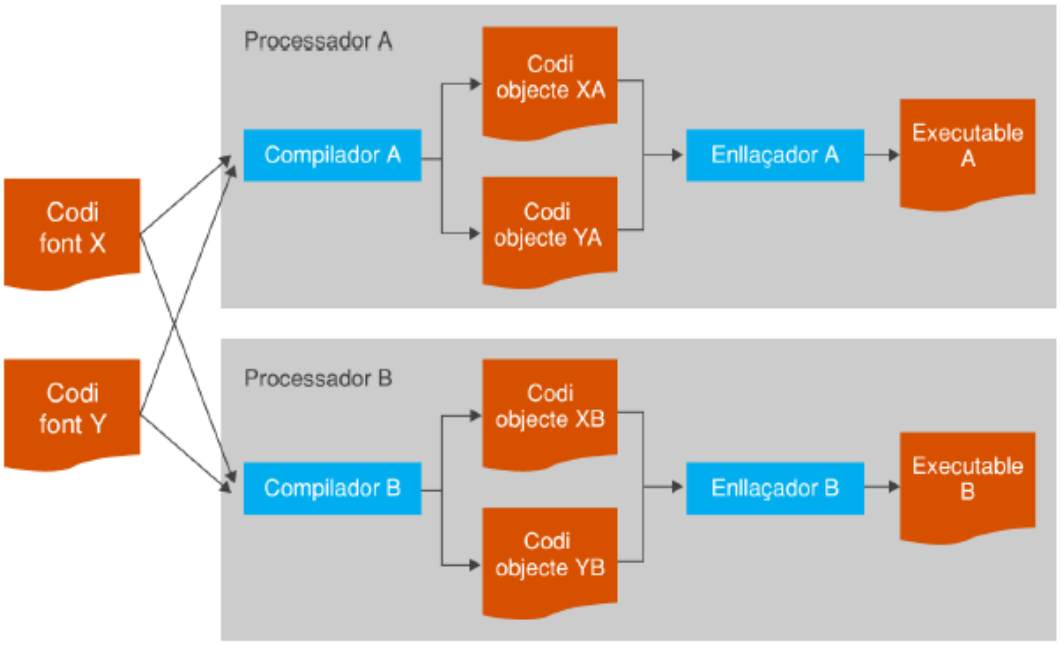
- Cuanto más alejadas están sus instrucciones del procesador, mayor es el nivel del lenguaje. Por ejemplo, a veces se dice que C y C++ son lenguajes de medio nivel, ya que permiten acceder a características particulares del procesador y la memoria.

1.3.3 Lenguajes compilados

- Se llama **código fuente** al conjunto de instrucciones que forman nuestro programa.
- El código fuente tiene que ser **compilado**. Este proceso consiste en convertir cada archivo de código fuente en un archivo con instrucciones en código máquina (entendible por el procesador). El resultado de compilar código fuente se llama **código objeto**.

[Comentar que en el proceso de compilado el compilador puede detectar errores e informar de ellos al programador, indicando una pista del origen]

- Después de compilar el programa hay que **enlazarlo**, para crear un archivo ejecutable a partir de los distintos archivos de código objeto.



Algunos lenguajes que tradicionalmente han sido compilados son: Pascal, FORTRAN, C y C++.

1.3.4 Lenguajes interpretados

- Un programa escrito en un lenguaje interpretado se va leyendo y ejecutando sentencia a sentencia.
- La aplicación encargada de traducir las instrucciones de alto nivel a código máquina se llama **intérprete**.
- Ejemplos de lenguajes interpretados: javascript, PHP, python, Basic...

La desventaja de usar un intérprete es que el programa será más lento. En cambio nos permite saltarnos el proceso de compilar y enlazar cada vez que queremos probarlo.

1.3.5 Lenguajes híbridos

- Algunos lenguajes no compilan a código máquina, sino a un lenguaje propio llamado **bytecode**. Este bytecode no depende del procesador ni del sistema operativo del ordenador.
- Para ejecutar el bytecode necesitamos un intérprete, que sí será diferente para cada arquitectura de procesador y para cada sistema operativo.
- Así pues, el programa es ejecutado por un intérprete, pero es más eficiente al interpretar un código (bytecode) hecho exprofeso.
- Habitualmente se llama **máquinas virtuales** a estos intérpretes de bytecode.
- Ejemplos de lenguajes que utilizan una máquina virtual: Java, C#.

1.3.6 Entornos de desarrollo integrado

- Los programas de ordenador se escriben siempre como archivos de texto. Estos pueden crearse mediante cualquier editor como el bloc de notas de Windows.

- Una herramienta mejor son los editores de código. Éstos tienen características que ayudan al programador, como el autocompletado de instrucciones, el coloreado del texto o plugins para subir los archivos a un servidor, etc. Algunos ejemplos notables son:

    - Sublime Text
    - Brackets
    - Atom
    - Visual Studio Code
    - Otros: Komodo Edit, TextMate, Notepad++.

- Para facilitar aún más el desarrollo de programas, pueden combinarse diversas funciones en una sóla utilidad llamada Entorno de Desarrollo Integrado (IDE):

    - Editor de código (con resaltado, autocompletado, etc.).
    - Acceso a documentación y ayuda contextual.
    - Detección automática de errores en el código.
    - Depurador de código (herramienta para detectar y solucionar errores en un programa).
    - Asistentes gráficos para crear interfaces de usuario y otras funciones.

- IDEs muy utilizados son:

    - Visual Studio (Microsoft)
    - Netbeans (Sun, Oracle, open source)
    - Eclipse (IBM y otras empresas, open source)
    - IntelliJIdea (JetBRAINS)

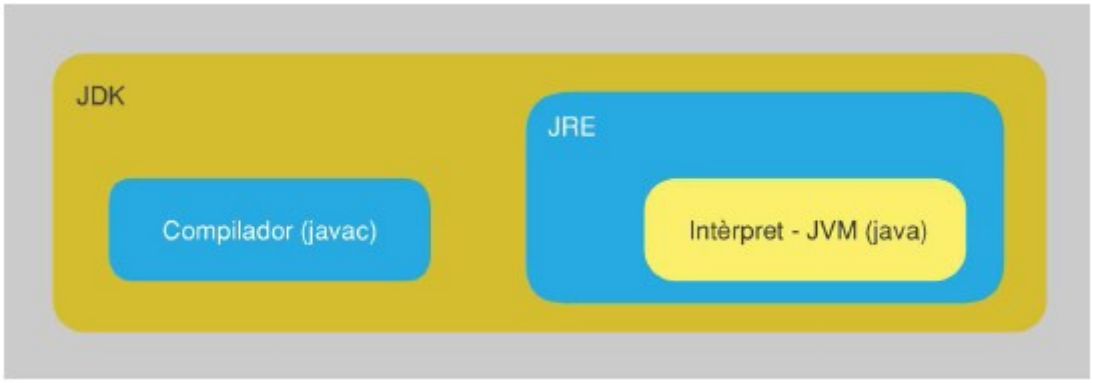
1.4 El lenguaje de programación Java

- Creado el año 1995 por *Sun Microsystems*. Con el propósito original de independizar los programas de la plataforma (sistema operativo, procesador).
- Gran éxito hasta hace unos años como medio de crear animaciones e interactividad en páginas web (*applets*).
- Permite realizar todo tipo de aplicaciones: de escritorio, web, móviles, etc.
- Lenguaje de alto nivel. Compilador estricto.
- Multiplataforma: El código compilado a bytecode puede ser ejecutado en cualquier equipo que disponga de una máquina virtual de Java.
- Orientado a objetos.

1.4.1 Creación de un programa en Java

- Los programas en Java se crean como archivos de texto con la extensión ".java".
- Para compilar un programa (a bytecode) es necesario el compilador de java, que se encuentra en el JDK (Java Development Kit).
- Podemos compilar un archivo java así:
    - javac HolaMundo.java
- En este momento se crea un archivo en **bytecode** llamado HolaMundo.class.

- Una vez compilado, podemos ejecutar este archivo mediante el intérprete de java, así:
  - java HolaMundo
- El intérprete de java se encuentra en el JRE (Java Runtime Environment), que viene con el JDK o podemos descargarlo aparte.



1.4.2 HolaMundo en Java

Recordar: Java es sensible a mayúsculas y minúsculas.

```
// Mi primer programa...
public class HolaMundo {

	public static void main(String[] args) {
		System.out.println("¡Hola, Mundo!");

	}

}
```

En este programa aparecen:

- Un comentario: Sólo sirve para dar información a los programadores que lean el código.

  ```
  // Mi primer programa...
  ```

- Una declaración de clase: En java todo el código que escribamos tiene que ir dentro de una clase.

  ```
  public class HolaMundo {

  }
  ```

- Una función o método llamada main: Tiene que existir en todo programa java y es lo primero que se ejecuta al arrancar el programa. Dentro de esta función hay una instrucción que muestra el mensaje en la consola.

```
public static void main(String[] args) {
	System.out.println("¡Hola, Mundo!");
}
```

- Observar que en Java las instrucciones pueden ocupar más de una línea y terminan en punto y coma.

Ejemplo: El siguiente vídeo muestra cómo se escribiría en pantalla la frase "Hola, Mundo!", con lenguaje ensamblador: https://www.youtube.com/watch?v=t24yLkrjVKU.

2. Variables

Una variable es un espacio en la memoria con un nombre asociado. El dato guardado en ese trozo de memoria se llama contenido de la variable o valor.

[Poner representación variable-caja-valor]

[Representación de la memoria como una fila de cajas o huecos]

2.1 Declarar una variable

Si queremos utilizar una variable para guardar algún dato en ella tenemos que declararla, indicando su nombre y el tipo de datos que va a contener.

Por ejemplo, para declarar una variable en java que guarde un número entero (en este caso el número de alumnos de clase), la declaramos:

```
int alumnos;
```

Si luego queremos darle un valor concreto (inicializarla) lo haremos con una asignación (símbolo "="):

```
alumnos = 25;
```

También podríamos declarar e inicializar la variable a la vez:

```
int alumnos = 25;
```

Ejemplo: Este programa declara, inicializa y muestra en la pantalla la variable alumnos.

```
public class EjemplosVariables {
	public static void main(String[] args) {
		int alumnos;

		alumnos = 25;

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

2.2 Dar nombre a una variable

El nombre o identificador de una variable debe cumplir estos requisitos:

- No puede contener espacios. [Explicar notación camello y con guiones bajos]
- No puede comenzar con un número.
- No usar caracteres no ingleses.
- No ser una palabra reservada del sistema.

3. Tipos de variables

3.1 Variables y tipos de datos primitivos

Un tipo de datos es la definición del conjunto de valores válidos que pueden tomar unos datos y el conjunto de operaciones que se les pueden aplicar.

Los tipos primitivos de datos son los que ya están incorporados directamente dentro de un lenguaje de programación, y son usados como piezas básicas para construir otros más complejos.

Un valor de un tipo primitivo puede almacenarse en una variable de ese mismo tipo.

3.1.1 Tipo booleano (boolean)

Representa un valor true ("verdadero") o false ("falso").

Ejemplo:

```
boolean mayorDeEdad = true;
boolean aprobado = false;
```

3.1.2 Tipo entero (byte, short, int, long)

Representa un valor numérico, positivo o negativo, sin decimales. En java tenemos varios subtipos:

- byte (1 byte)
- short (2 bytes)
- int (4 bytes)
- long (8 bytes)

Ejemplo:

```
byte edad = 80;
short sueldo = 1200;
int numeroCuenta = 9922929;
long numeroMuyGrande = 3_000_000_000;
```

3.1.3 Tipo real (float, double)

Representa un valor numérico, positivo o negativo, con decimales. En java tenemos varios subtipos:

- float (4 bytes)
- double (8 bytes)

La coma decimal se representa con la notación de punto.

[Poner ejemplos de declaraciones]

3.1.4 Tipo carácter (char)

Representa cualquier carácter. Los caracteres se representan entre comillas simples: 'a', 'b', ...

En java los caracteres se representan usando UNICODE. En lenguajes más antiguos se usaba preferentemente ASCII (American Standard Code for Information Interchange).

[Ejemplos de declaraciones]

3.2 Cadenas de caracteres

Podemos guardar cadenas de caracteres (textos y números) de varios caracteres en objetos de tipo String, aunque no se consideran tipos primitivos, sino objetos.

[Ejemplos de declaraciones]

3.3 Desbordamientos y errores de precisión

- Desbordamiento u "overflow": Una operación aritmética intenta crear un valor numérico que es demasiado grande para ser representado dentro de la variable pertinente.

  Ejemplos: Cuentaquilómetros de un coche, artículo sobre el personaje de Ghandi en el juego Civilization.

- Errores de precisión: Una variable de coma flotante (float o double) no admite un número infinito de decimales. Así, al acomodar un número al número máximo de decimales, obtenemos pequeños errores.

3.4 Operaciones con datos

3.4.1 Operaciones con enteros

Aritméticas:

- Típicas: +, -, *, /
- Módulo: %

La operación módulo (%) devuelve el resto de una división. Por ejemplo:

7 % 3 = 1
20 % 6 = 2

[Ejemplos]

Relacionales (el resultado es un booleano):

A == B	A es igual a B
A !=B	A es distinto que B
A > B	...
A >= B	...
A < B	...
A <= B	...

[Ejemplos]

3.4.2 Operaciones con decimales

No se usa el operador módulo. El resto es igual.

3.4.3 Operaciones con caracteres

Pueden usarse los mismos operadores relacionales. Los caracteres pueden operarse como números enteros.

3.4.4 Operaciones con booleanos

Operaciones lógicas:

- &&
- ||
- !

Ejemplos:

A = "Estamos en el siglo XXI"
B = "Esto es primero de superior"
C = "Es verano"
D = "Estamos en la Edad Media"

A AND B = A y B = verdadero
A AND C = A y C = falso

A OR B = A o B = verdadero
A OR C = A o C = verdadero
C OR D = C o D = falso

NOT A = no A = falso
NOT C = no C = verdadero

Tablas de verdad

A	B	A && B
F	F	F
F	V	F
V	F	F
V	V	V

A	B	A || B
F	F	F
F	V	V
V	F	V
V	V	V

A	!A
F	V
V	F

Operaciones relacionales:

- == Igual
- != Distinto

3.5 Construcción de expresiones

Podemos construir expresiones combinando las operaciones anteriores y paréntesis. El orden de evaluación es:

Taula 2.6. Ordre de precedència dels operadors

Ordre	Operació	Operador
1	Canvi de signe	-(unari)
2	Producte, divisió i mòdul	* / %
3	Suma i resta	+ -
4	Relacionals de comparació	> < <= >=
5	Relacionals d'igualtat	== !=
6	Negació	! (unari)
7	Conjunció	&&
8	Disjunció	||

Pueden (y conviene) utilizarse paréntesis para evitar ambigüedades en las expresiones.

3.6 Constantes

Una constante es una variable cuyo valor no puede cambiar a lo largo del programa. Al declararla se añade la palabra clave *final*.

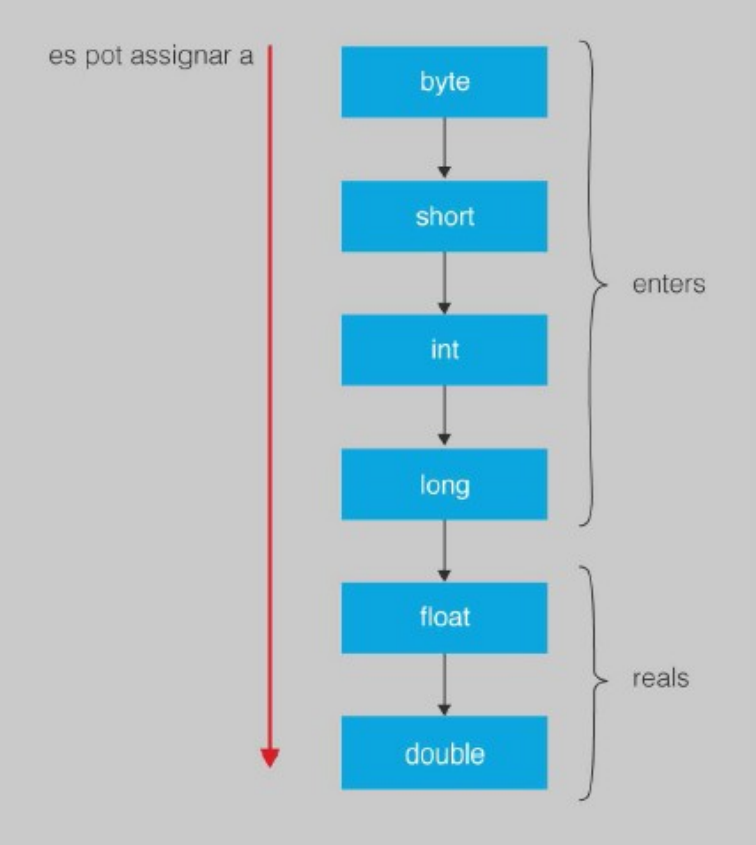
La convención en java es nombrar las variables con todas las letras mayúsculas. Si es un nombre compuesto, las palabras que lo forman se separan con guiones bajos:

```
final int IVA = 21;
final int IVA_REDUCIDO = 10;
final String SALUDO = "Muy apreciado señor:";
```

4. Conversiones de tipos

Una conversión de tipos es la transformación de un tipo de datos en otro diferente.

4.1 Conversiones implícitas



[Ejemplos con Eclipse]

4.2 Conversiones explícitas (cast)

Sólo debemos hacerlas cuando estemos seguros de que la conversión tiene sentido.

Ejemplo:

```
public static void main(String[] args) {
	double num1 = 50.0;
	float num2 = (float) num1;
	long num3 = (long) num2;
	int num4 = (int) num3;

	System.out.println(num1);
	System.out.println(num2);
	System.out.println(num3);
	System.out.println(num4);
}
```

4.3 Conversiones con caracteres

```
public static void main(String[] args) {
	char letra1 = 97;
	System.out.println(letra1);

	char letra2 = 'a' + 1;
	System.out.println(letra2);
}
```

5. Entrada y salida de datos en Java

Ya vistas:

- System.out.print (cadena)
- System.out.println (cadena)

5.1 Concatenación de cadenas

Podemos usar el operador "+" para concatenar (unir) dos o más cadenas de caracteres:

- Ejemplo con variables String
- Ejemplo con el + dentro de un println
- Ejemplo concatenando una cadena y un tipo primitivo

5.2 Caracteres de control

También llamados secuencias de escape, empiezan por "\" y tienen un significado especial.

Seqüència d'escapament	Acció o símbol representat
\t	Una tabulació.
\n	Un salt de línia i retorn de carro.
\'	El caràcter "cometa simple".
\"	El caràcter "cometes dobles" (").
\\	El caràcter "contrabarra" (\).

[Ejemplos]

5.3 Entrada simple de datos

Usamos un objeto llamado Scanner:

```
import java.util.Scanner;

public class LeyendoDatos {
	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);
		String nombre;
		int edad;

		System.out.println("¿Cómo te llamas? ");
		nombre = entrada.nextLine();
		System.out.println("¿Qué edad tienes? ");
		edad = entrada.nextInt();

		System.out.println("Te llamas " + nombre + " y tienes " + edad
			+ " años.");
```

```
        }
    }
```

Instrucció	Tipus de dada llegida
`lector.nextByte()`	`byte`
`lector.nextShort()`	`short`
`lector.nextInt()`	`int`
`lector.nextLong()`	`long`
`lector.nextFloat()`	`float`
`lector.nextDouble()`	`double`
`lector.nextBoolean()`	`boolean`
`lector.next()`	`String`

Actividades:

1. Crea un programa que pida dos números y los multiplique.
2. Crea un programa que pida dos números y calcule el resto de dividirlos.
3. Crea un programa que haga la división entera entre dos números (devolviendo cociente y resto).
4. Crea un programa que pregunte el nombre de un alumno y sus notas de las tres evaluaciones y devuelva la media.
5. Crea un programa que pida el nombre de un producto en venta, su precio y el descuento que se le debe hacer. Mostrará todos los datos junto al precio rebajado.
6. Escribe una aplicación que pida el nombre de un producto, su precio y el número de unidades vendidas, y devuelva el coste total de la compra.

Referencias

- Vídeo: Lenguajes de bajo nivel (canal de TI Capacitación en Youtube)
  - https://www.youtube.com/watch?v=pPbFCvIQUi4
- Vídeo: Lenguajes de alto nivel (canal de TI Capacitación en Youtube)
  - https://www.youtube.com/watch?v=h6YY46Wjlmk