ISe
Ingeniería de Servicios

Universidad Rey Juan Carlos

Introducción a la programación

Fundamentos de la programación

Profesores:

Francisco Javier Pérez Blanco

Javier Sevilla López

Tema 1

- Introducción
- Problemas, Algoritmos y Programas
- Paradigmas y Lenguajes de Programación
- Ingeniería del Software

Objetivo

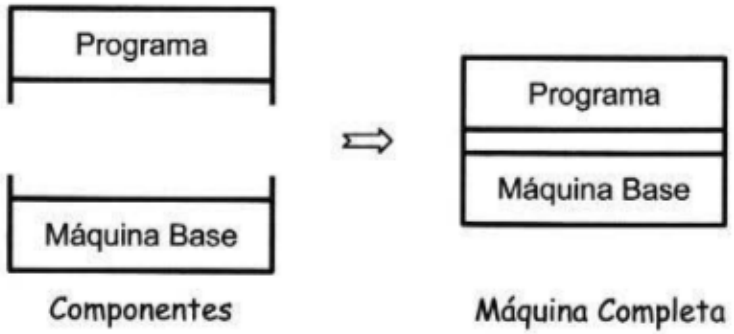
- Exponer los conceptos clave para la resolución de problemas por medio de sistemas computacionales

Máquinas Programables

- Máquina
  - Dispositivo o instrumento capaz de realizar un cierto trabajo u operación
  - Un proceso de funcionamiento por el cual diferentes operaciones se van sucediendo a lo largo del tiempo sucesiva o simultáneamente
  - Atendiendo a su control
    - Manuales (p.e.: máquina de escribir)
      - Operador o agente externo invoca operaciones
    - Automáticas (p.e: ascensor)
      - Actúan por si solas, pudiendo responder a estímulos externos

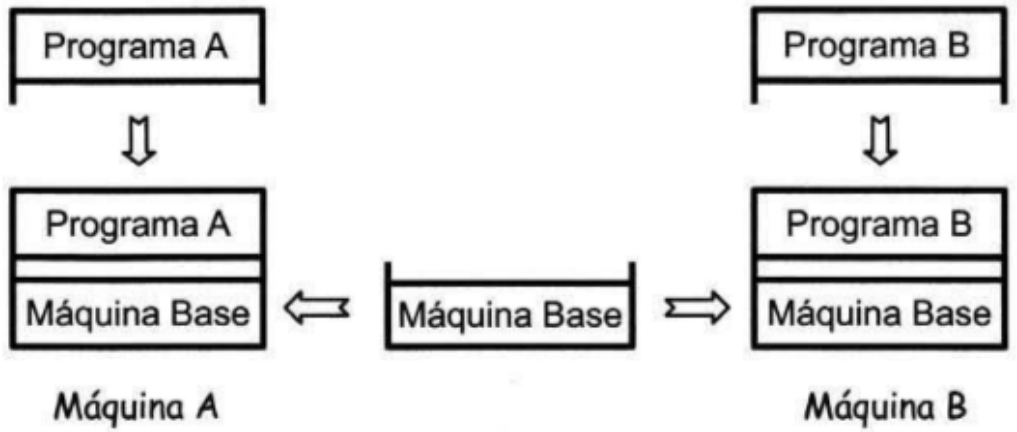
Máquinas Programables

- Máquinas automáticas
  - Fijas
  - Programables
- Ejemplo
  - Piano: máquina manual para reproducir música
  - Caja de música: máquina automática para producir música
  - Reproductor MP3: máquina automática para producir música



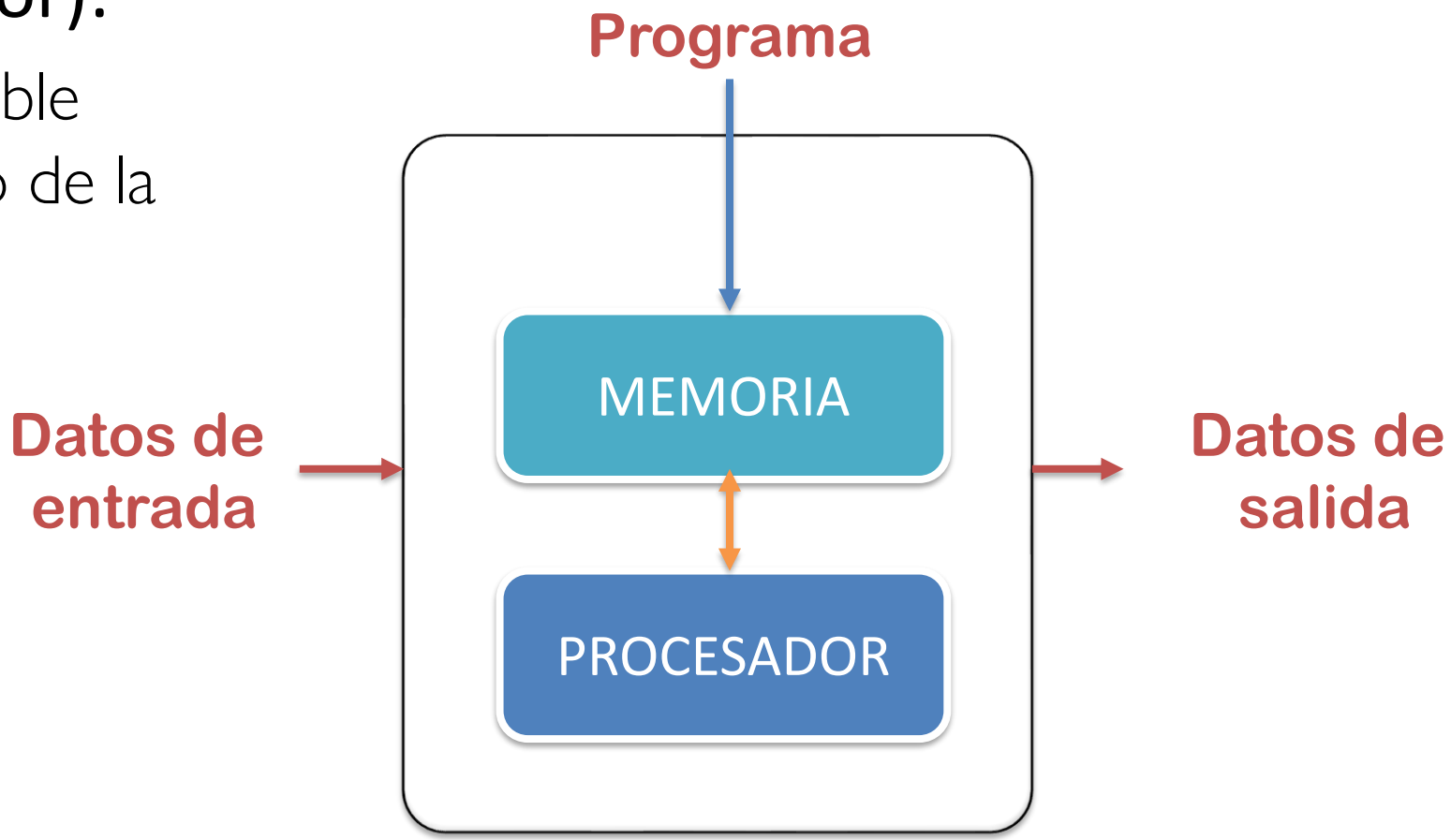
Máquinas Programables

- Dependiendo del programa suministrado, la máquina se comporta como diferentes máquinas



Máquinas Programables

- La máquina programable por excelencia es el ordenador (antes Computador):
  - Máquina programable para el tratamiento de la información



Computadores

- Para realizar un determinado tratamiento de la información necesitamos
  - Construir la máquina base → Hardware
  - Idear y desarrollar el programa → Software
  - Ejecutar dicho programa en el ordenador (dispositivo)

- Para realizar un determinado tratamiento de la información necesitamos
  - Construir la máquina base → Hardware
  - Idear y desarrollar el programa → Software
  - Ejecutar dicho programa en el ordenador (dispositivo)

La labor de desarrollar programas recibe habitualmente el nombre de <u>programación</u>

Introducción

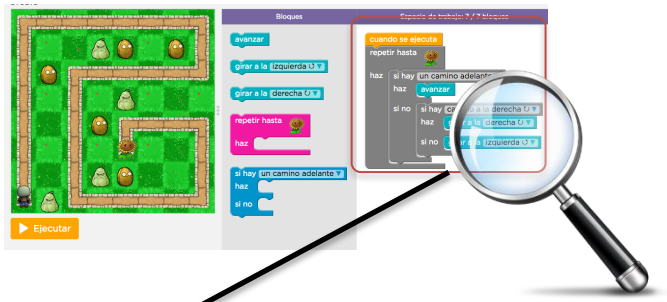
- ¿Qué es la programación?
  - Es un proceso mediante el cual se codifican una serie de **instrucciones** en un **lenguaje** determinado para ser decodificados y ejecutados por un sistema computacional, con el fin de resolver un problema o llevar a cabo una función específica.
  - Definir lo que debe hacer el ordenador para resolver un problema concreto, utilizando un lenguaje de programación.

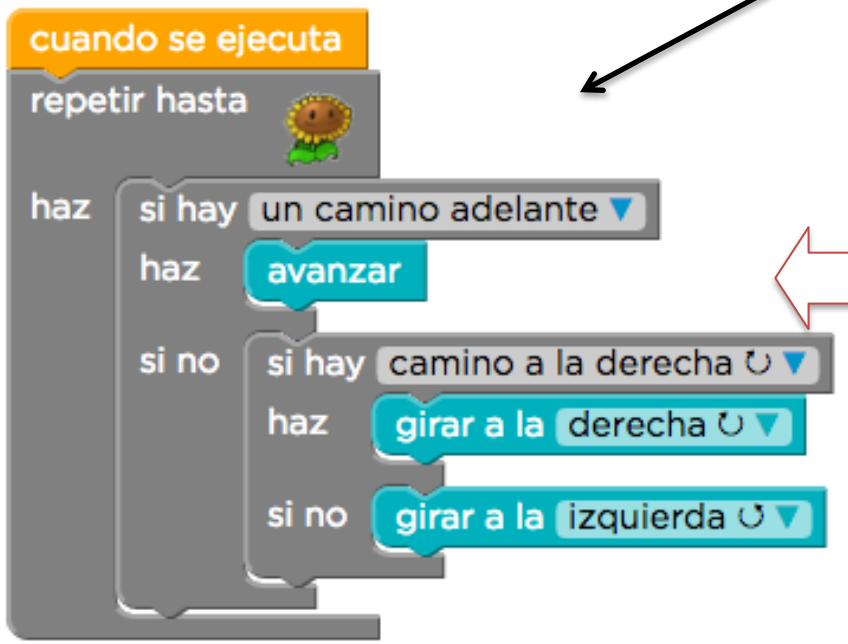
Introducción

- Ejemplo

Introducción

- Ejemplo





Incluso las mejores universidades enseñan programación basada en bloques (por ejemplo, **Berkeley**, **Harvard**). Pero, por debajo, los bloques que has programado también se pueden mostrar en JavaScript, el lenguaje de programación más utilizado en el mundo:

```
while (notFinished()) {
  if (isPathForward()) {
    moveForward();
  } else {
    if (isPathRight()) {
      turnRight();
    } else {
      turnLeft();
    }
  }
}
```

Lenguajes de programación

```
88 94 50 FF 76 0A FF 76 08 9A BA CD 3A 16 B8 01
00 EB E8 B8 88 94 50 2B C0 50 9A FA C5 3A 16 EB
ED B8 88 94 50 B8 01 00 EB EF B8 88 94 50 9A 48
D1 3A 16 EB D9 5D CA 0A 00 55 8B EC 83 EC 08 57
56 B8 01 00 50 9A 97 41 9B 34 8B D8 8B 47 14 89
```

Código Máquina

```
void PintarPlazas(const TipoPlazas P) {

  printf("\n\n");
  printf("      A    B    C       D    E    F\n\n");
  for (int i = 0; i < NumFilas; i++) {
    printf("%3d",i+1);
    for (int j = 0; j < AsientosFila; j++) {
      if ( j == Pasillo ) {
        printf("   ");
      }
      if (P[i].AsientosOcupa[j] == ocupado) {
        printf("  (*)");
      } else if (P[i].AsientosOcupa[j] == reservado) {
        printf("  (R)");
      } else if (P[i].AsientosOcupa[j] == vacio) {
        printf("  ( )");
      }
    }
    printf("\n");
  }
  printf("\n");
}
```

Fragmento de programa en C++

- Definición: Problema
  - Proposición encaminada a averiguar el modo de obtener un resultado, cuando se conocen ciertos datos de partida

- Tipos de Problemas
  - Sin Solución
  - Determinados: con una única solución
  - Indeterminados: con un número indefinido de soluciones

Problemas, Algoritmos y Programas

- Fases para resolver un problema con un programa informático:
  - Estudio del problema (ANÁLISIS)
  - Descripción de un método (algoritmo) que lo resuelva (DISEÑO)
  - Escritura del algoritmo en un lenguaje de programación (CODIFICACIÓN)
  - Comprobación del correcto funcionamiento (PRUEBA)

La definición del problema

- Análisis del problema
  - Consiste en establecer con precisión qué se necesita
- Especificación
  - Descripción precisa del problema:
  - - datos de partida
  - - resultado

lenguaje natural ⟹ puede resultar impreciso

lenguajes formales ⟹ lógica, matemáticas

Un ejemplo

Ejemplo de Especificación
Problema de división euclídea

- Datos
  - 2 enteros, dividendo y divisor (D,d)
  - d no nulo
- Resultado
  - 2 enteros, cociente y resto (C,R)
  - $0 \leq R < d$, tal que $D = d*C + R$

- **Algoritmo - Etimología**
  - Alhuarizmí: sobrenombre del árabe Muhamed ibn Musa (al-Jwarizmi), matemático persa. Escribió un tratado sobre la manipulación de números y ecuaciones: "Kitab al-jabr w'almugabala". ¿Os suena?

- **Definición de Algoritmo (1):**
  - Descripción precisa de los pasos que nos llevan a la solución de un problema

Problemas, Algoritmos y Programas

- Algoritmo – Ejemplo

Cambiar una rueda de coche pinchada

```
Inicio
PASO 1.    Aflojar los tornillos de la rueda pinchada con
           la llave inglesa.
PASO 2.    Ubicar el gato mecánico en su sitio.
PASO 3.    Levantar el gato hasta que la rueda pinchada
           pueda girar libremente.
PASO 4.    Quitar los tornillos y la rueda pinchada.
PASO 5.    Poner rueda de repuesto y los tornillos.
PASO 6.    Bajar el gato hasta que se pueda liberar.
PASO 7.    Sacar el gato de su sitio.
PASO 8.    Apretar los tornillos con la llave inglesa.
Fin
```

Algoritmo

- Definición (2):
  - Método tal que partiendo de datos apropiados, conduce sistemáticamente a los resultados requeridos en la especificación del problema

Caracterización

- La descripción de un algoritmo afecta a:
  - Entrada (Datos)
  - Proceso (Instrucciones)
  - Salida (Resultado)
- Se puede decir:
  - Algoritmo ≅ función matemática
  - Algoritmo: Entrada → Salida (proceso)
- Ejemplo: Suma Lenta: $N \times N \rightarrow N$
  - $a + b \rightarrow c, \quad c = a + b$

Es constructivo: hay que precisar también el proceso de cálculo

Características de los algoritmos

- **Precisión (sin ambigüedad) en cuanto a:**
  - Orden: secuencia de pasos que han de llevarse a cabo
  - Contenido: qué se realiza en cada paso
- **Determinismo:**
  - Debe responder del mismo modo ante las mismas condiciones
- **Finitud:**
  - Debe tener fin

Aspectos de un algoritmo

- Obligatorios
  - Corrección: respecto a las especificaciones
  - Complejidad: recursos que un algoritmo necesita. En máquinas secuenciales (tiempo y memoria)

- Deseables
  - Generalidad: sirva para una clase de problemas lo más amplia posible
  - Eficiencia: será más eficiente en la medida que necesita de menos pasos

Lenguajes Algorítmicos

- Sirven para describir un algoritmo
- Son más precisos que el lenguaje natural, pero menos rígidos (o formales) que un lenguaje de programación
  - Se les considera un lenguaje intermedio
  - Tienen cierta independencia de los lenguajes de programación

- # Algoritmo – Ejemplo (pseudocódigo)

  Un estudiante se encuentra en su casa (durmiendo placenteramente) y debe ir a la URJC (¡¡¡a clase de programación!!!), ¿qué debe hacer?

```
Inicio
Dormir
hacer
   Dormir
hasta que suene el despertador.
Mirar la hora.
¿Hay tiempo suficiente?
      Si hay, entonces
          Ducharse.
          Vestirse.
          Desayunar.
      Si no,
            Vestirse.
Cepillarse los dientes.
Despedirse de la familia.
```

```
¿Hay tiempo suficiente?
      Si hay, entonces
         Caminar a la estación de metro.
      Si no,
         Correr hacia la estación de metro.
Hacer
    Esperar el metro
    Ver a las demás personas que esperan el metro y ver
     continuamente cuánto falta para que llegue el metro.
Hasta que pase un metro hacia Manuel Becerra
Subirse al metro.
Mientras no llegue a Manuel Becerra
hacer
    Seguir en el metro.
    Hacer cualquier cosa con el móvil como todos los demás.
Bajarse.
Salir de la estación y entrar a la universidad.
Fin
```

Ejemplo: Algoritmo SumaLenta

Partimos de dos cantidades: a y b. El método de suma lenta consiste en ir pasando de a a b una unidad cada vez, de forma que cuando a=0, el resultado será el valor de b

Algoritmo Suma lenta (Pseudocódigo)

Sean a, b $\in$ N

Leer a y b

Mientras a $\neq$ 0, hacer { **a $\leftarrow$ a-1**; **b $\leftarrow$ b+1** }

Escribir b

- Ejemplo
  - Pseudocódigo, diagramas de flujo

Algoritmo: definición formal

- Es una cuádrupla que contiene los siguientes elementos:
  - Conjunto de los estados que pueden presentarse en todo momento
  - Identificación de estados iniciales
  - Identificación de estados finales
  - Función de transición entre estados

Algoritmo: definición formal

- Un estado se define por una tupla de cuatro elementos
  - Marca de la posición del algoritmo en la que se define el estado
  - Datos de entrada
  - Resultados emitidos
  - Valores de las variables que entran en juego

Ejemplo: Algoritmo SumaLenta

Ejemplo
Estados de cómputo (Suma lenta)

Sean a, b ∈ N

Leer a y b

Mientras a ≠ 0, hacer { **a ← a-1** ; **b ← b+1** }

Escribir b

Datos de entrada

Resultados emitidos

Posición	*E*	*S*	*a*	*b*
1	[2 3↵]	[]	¿	¿
2	[]	[]	2	3
3	[]	[]	1	3
4	[]	[]	1	4
5	[]	[]	0	4
6	[]	[]	0	5
7	[]	[5↵]	0	5

Valores de los datos [a,b]

Algoritmo

- Ejercicio
  - Escribir un algoritmo que realice la suma de todos los números pares entre 2 y 1000.

Problemas y Algoritmos

- Algunos problemas tienen distintas soluciones algorítmicas
  - Ejemplo
    - Máximo común divisor (MCD)
      - Por descomposición en factores primos
      - Usando el algoritmo de Euclides
      - Usando el mínimo común múltiplo
- Algunos problemas NO tienen solución algorítmica
  - Ejemplo
    - Problema de la parada (encontrar un algoritmo que determine si otro algoritmo finaliza o no con unos determinados datos de entrada)

Problemas, Algoritmos y Programas (Resumiendo …)

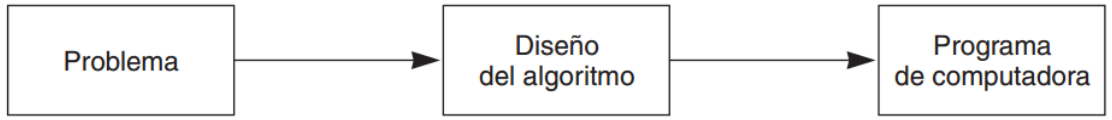
- **Definición de Programa**
  - Conjunto de instrucciones precisas, en un lenguaje entendible por la computadora

- **Programación**
  - Proceso de construcción de programas

- **Fases:**
  - **Análisis** del problema
  - Solución conceptual del problema - **Diseño**
  - Escritura del algoritmo en un lenguaje de programación – **Codificación**
  - Comprobación de resultados - **Prueba**

Lenguajes de Programación

- **Definición de Lenguaje de Programación:**
  - Un lenguaje artificial, diseñado para representar algoritmos de forma inteligible para las computadoras
- **LPs vs Lenguaje natural**
  - LPs son más formales y rigurosos
  - LPs son más simples en su sintaxis y semántica



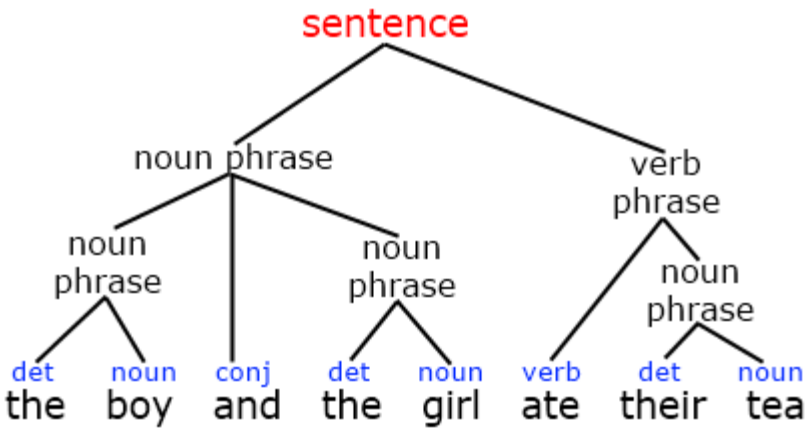
Lenguajes de Programación

- Algunas características relevantes:
  - Sintaxis
  - Semántica
  - Traducción y Ejecución
  - Errores y cómo subsanarlos

Sintaxis

- Especifica inequívocamente cómo están construidos los programas de un LP
- Especificación de la sintaxis
  - Gramáticas (BNF)
  - Diagramas Sintácticos

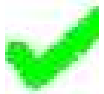
```
<dirección postal> ::= <nombre> <dirección> <apartado postal>

<personal> ::= <primer nombre> | <inicial> "."

<nombre> ::= <personal> <apellido> [<trato>] <EOL>
           | <personal> <nombre>

<dirección> ::= [<dpto>] <número de la casa> <nombre de la calle> <EOL>

<apartado postal> ::= <ciudad> "," <código estado> <código postal> <EOL>
```



Semántica

- Asigna un significado a cada tipo de construcción de un LP
- Formas de especificación:
  - ejemplos (y contraejemplos) en los manuales
  - definición formal
- Ejemplo

```
write('hola');
write('hola');
```

holahola 

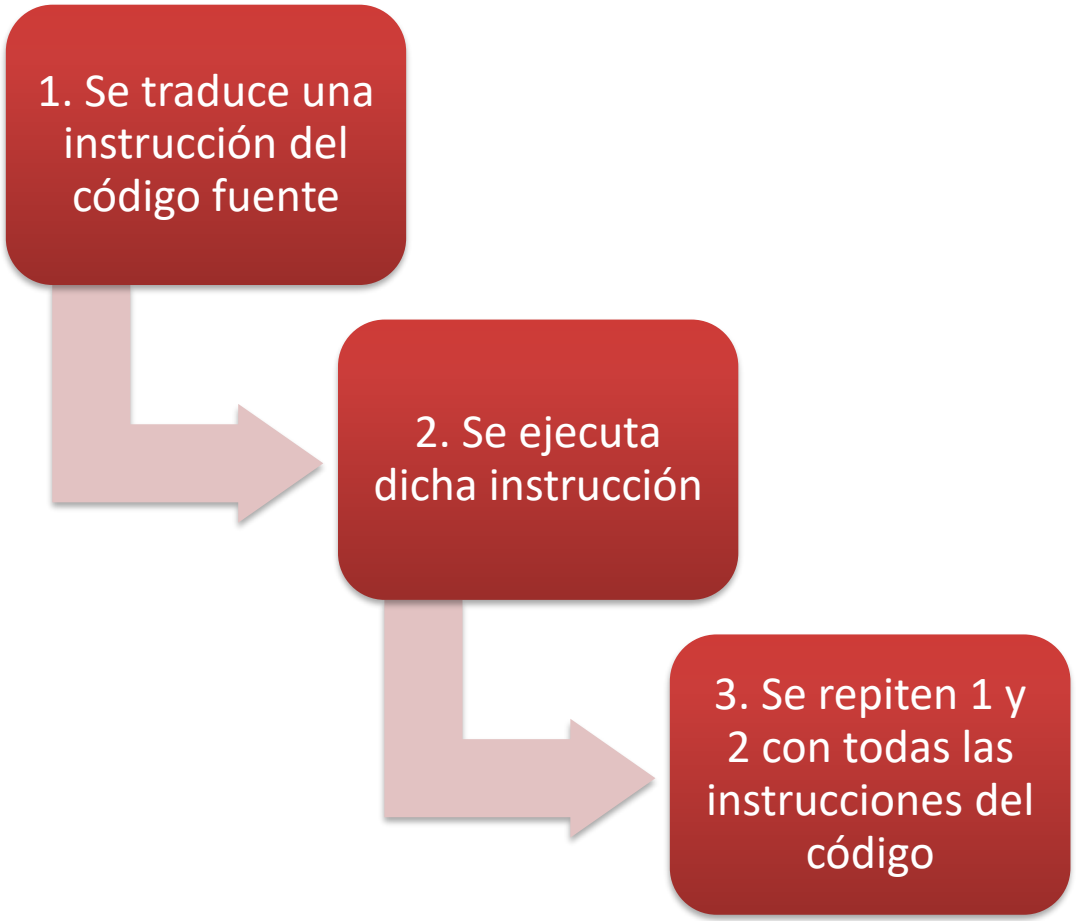
hola
hola 

Traducción y Ejecución

- El lenguaje de alto nivel ha de traducirse al lenguaje de la máquina
- Formas de traducción:
  - Compilación:
    - Todo el código fuente (en un archivo) se traduce a código ejecutable (en otro archivo)
    - Se ejecuta dicho código ejecutable

Traducción y Ejecución

- Interpretación:



Errores

- **Errores de compilación**
  - Surgen a la hora de traducir ("compilar") el código fuente
  - Errores sintácticos, de tipo, etc.

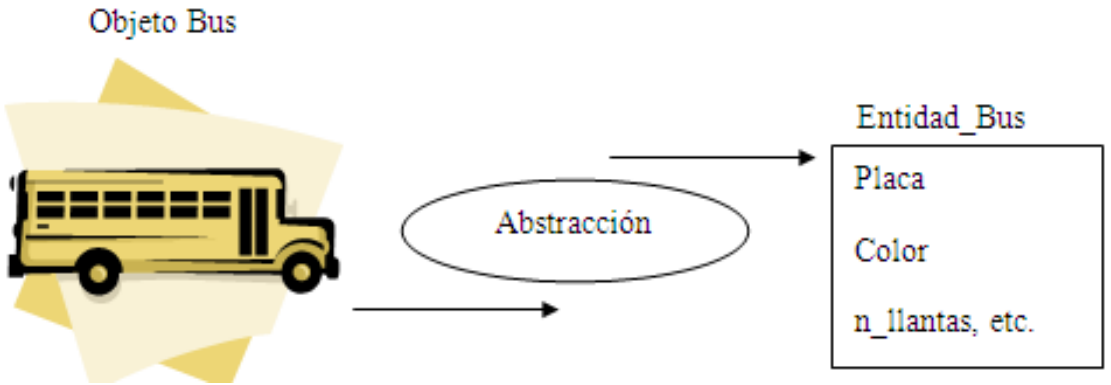
- **Errores de ejecución**
  - Surgen al ejecutar el código ejecutable
  - Operaciones ilegales (división por cero), errores lógicos etc.

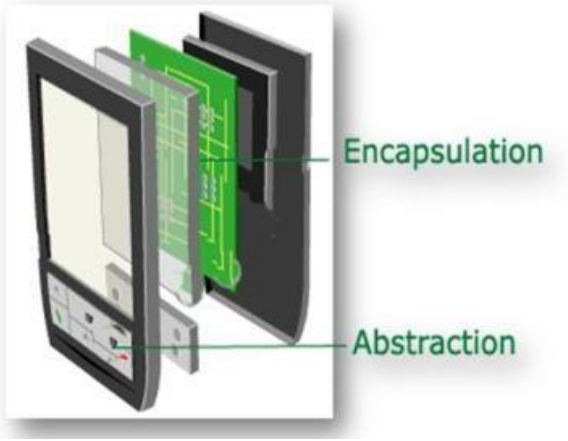
Evolución de los LP

- Motores que impulsan la evolución de los lenguajes de programación:
  - Abstracción
  - Encapsulación
  - Modularidad
  - Jerarquía

Evolución de los LP

- **Abstracción:**
  - Proceso mental por el que el ser humano extrae las características esenciales de algo, e ignora los detalles superfluos
- **Encapsulación:**
  - Proceso por el que se ocultan los detalles de las características de una abstracción



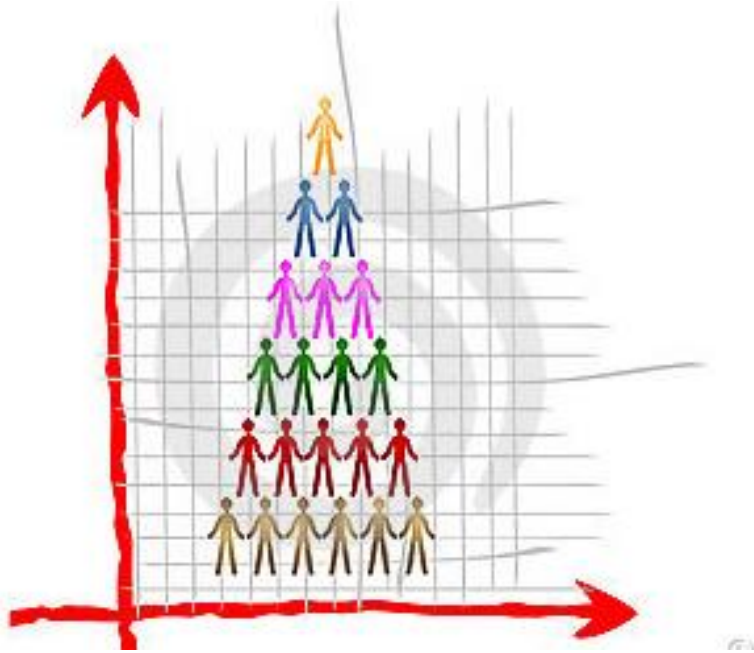


Evolución de los LP

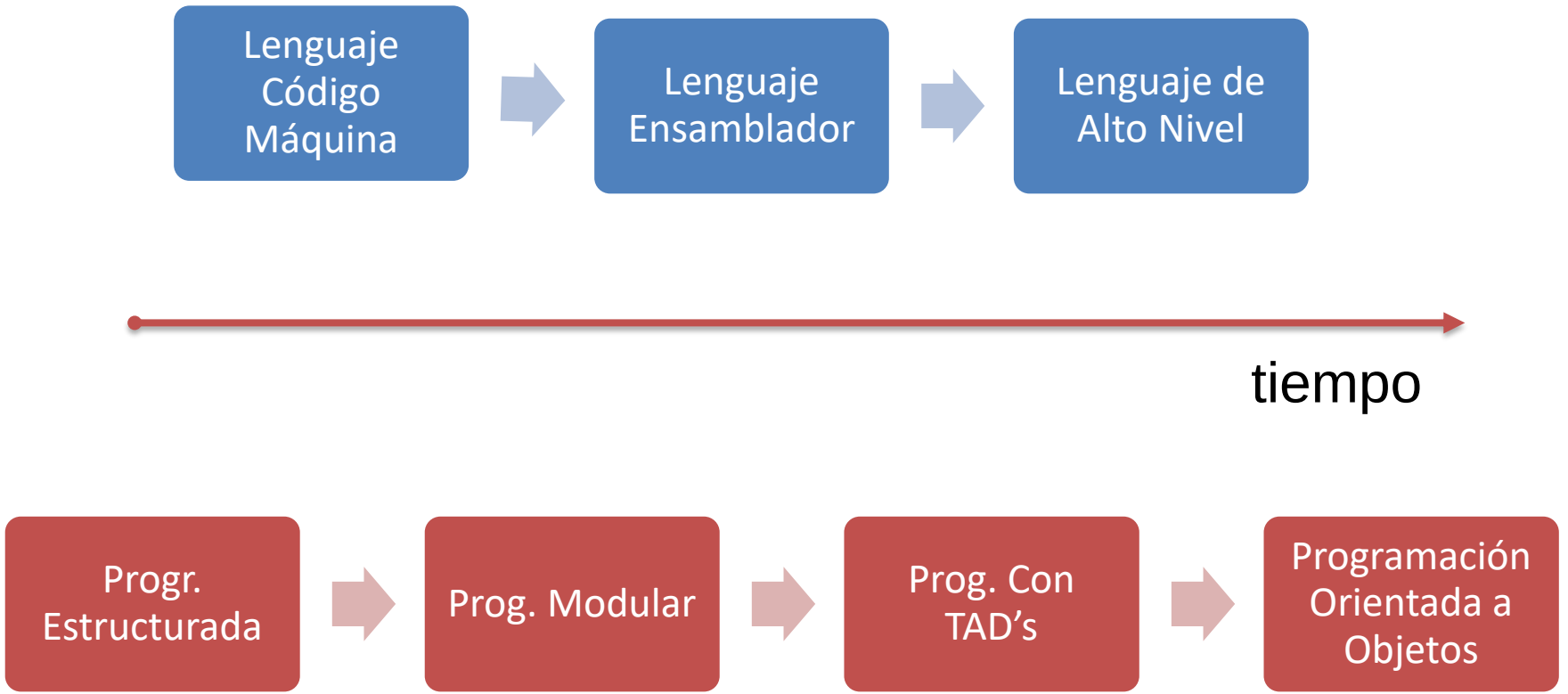
- Modularización:
  - Proceso de descomposición de un sistema en un conjunto de elementos poco acoplados (independientes) y cohesivos (con significado propio)

Evolución de los LP

- Jerarquía:
  - Proceso de estructuración por el que se organizan un conjunto de elementos en distintos niveles, atendiendo a determinados criterios (responsabilidad, composición, etc.)



Evolución de los LP



Paradigmas de programación

- **Definición:**
  - Una colección de patrones conceptuales que moldean la forma de razonar sobre problemas, de formular algoritmos y, a la larga, de estructurar programas

- **Paradigmas:**
  - Programación imperativa
  - Programación funcional
  - Programación lógica

Programación funcional

- Basada en la noción de función matemática
  - f: Dominio → Rango

- Programar:
  - Definir funciones básicas (con parámetros) (p.e. por enumeración)
  - Diseñar funciones complejas (p.e. por comprensión)
  - Evaluar las funciones sobre los datos de entrada

Programación lógica

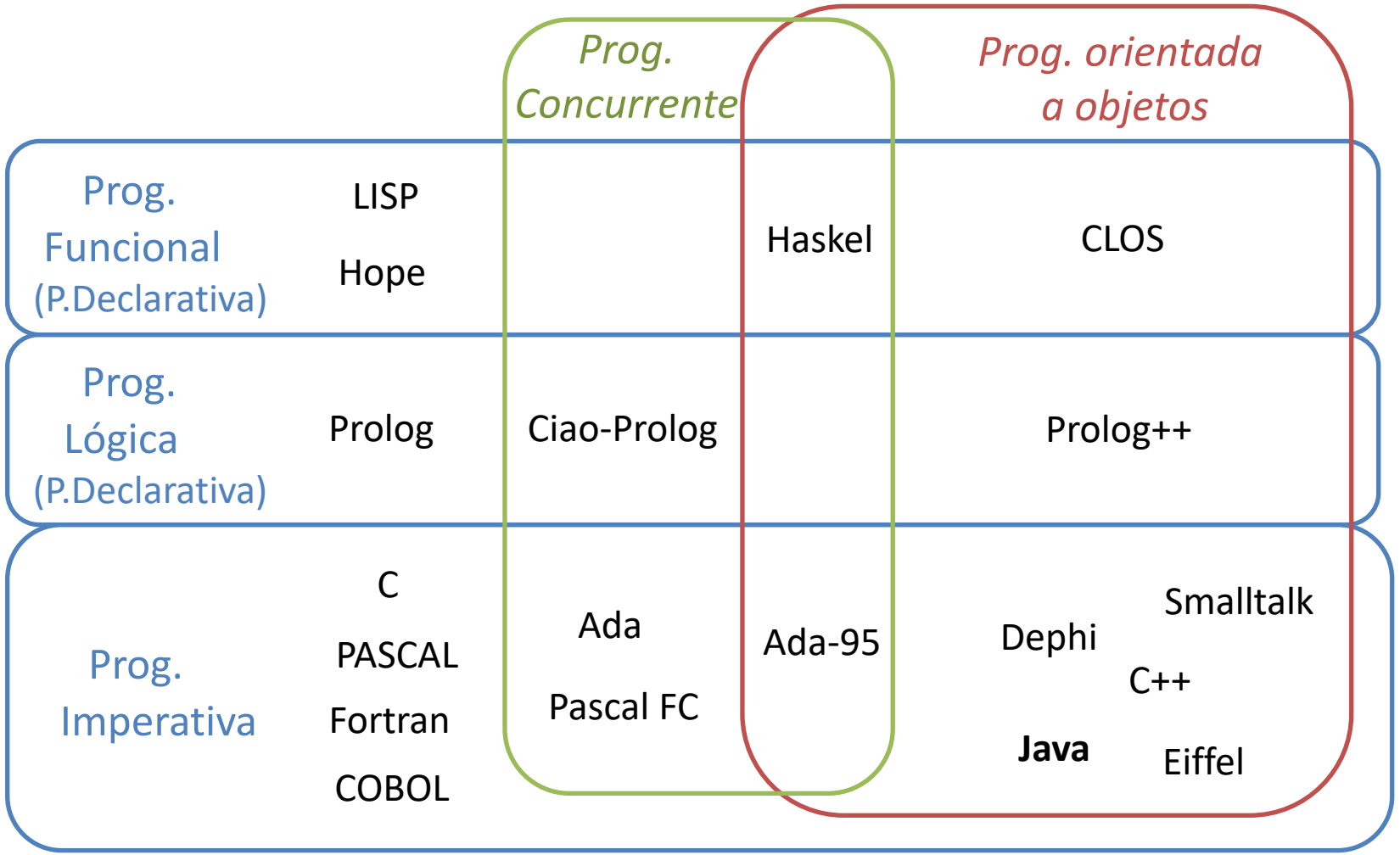
- Basada en la inferencia automática en (un subconjunto de) lógica de 1er orden

- Programar:
  - Definir hechos (predicados básicos)
  - Diseñar implicaciones para definir predicados complejos
  - Determinar la verdad de los predicados para individuos concretos

Programación imperativa

- Basada en el modelo von Neumann
  - Un conjunto de operaciones primitivas
  - Ejecución secuencial
- Abstracción
  - Variables, expresiones, instrucciones
- Programar:
  - Declarar variables necesarias
  - Diseñar una secuencia adecuada de instrucciones (asignaciones)

Paradigmas y lenguajes

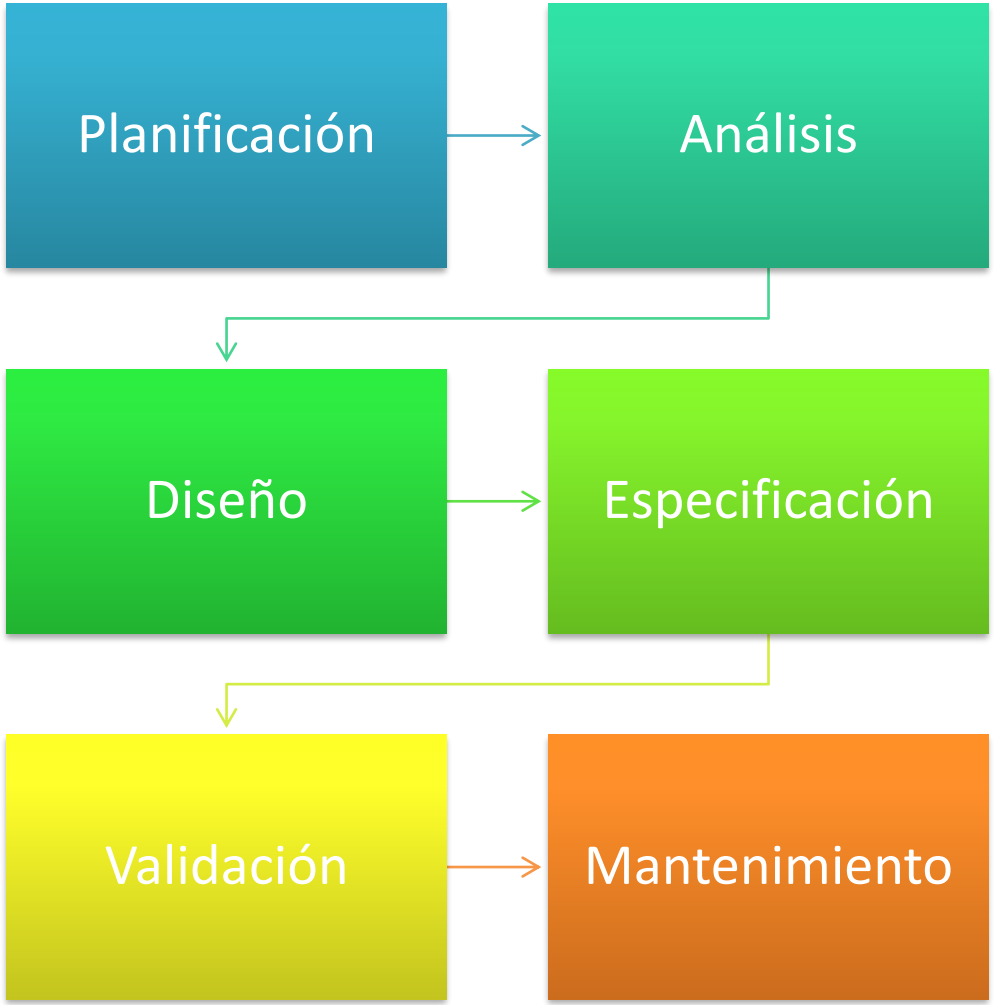


Ingeniería del Software

- **Definición (Bauer, 1969):**
  - El establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales

- **Definición (IEEE, 1993):**
  - La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia desarrollo, operación y mantenimiento de software

Fases de un desarrollo sistemático



Planificación

- Determinar las necesidades de programación
- Estimación de recursos de desarrollo
- Predicción aproximada de coste y tiempo
- Determinar si el desarrollo del software es viable económicamente

Análisis de requerimientos

- Definir detalladamente las funciones de cada módulo, de acuerdo con los deseos del cliente
- Definir detalladamente el trabajo conjunto de los distintos módulos
- Definir criterios y sistema de validación
- Redactar especificaciones detalladas del funcionamiento general del software

Diseño

- Diseñar el conjunto de bloques o módulos
- Se dividen en partes o tareas
- Se asignan tareas a equipos de trabajo, que las desarrollan y prueban

Codificación

- Escribir los algoritmos en el lenguaje de programación elegido

- Integrar las partes para que formen un programa completo

Validación

- Aplicar el sistema de pruebas descrito en la fase de análisis de requerimientos
- Métodos de validación
  - Pruebas (tests), inspecciones …
  - Verificación formal
- Objetos de validación:
  - Los módulos de programa
  - Las conexiones entre ellos (integración)
  - La aplicación entera

Mantenimiento

- Redactar la documentación actualizada
- Iniciar la explotación
- Detectar y subsanar errores cometidos en etapas anteriores
- Adaptar la aplicación a requisitos cambiados (evolución)

Wrap-up

- Los ordenadores son capaces de desempeñar tareas porque alguien les ha dicho cómo hacerlas
  - Alguien ha recogido las instrucciones en un **programa**

```
If the column number is greater than 60,
    then go to the next line.
Otherwise (if the column number isn't greater than 60),
    then stay on the same line.
```

```
if (columnNumber > 60) {
    wrapToNextLine();
}
else {
    continueSameLine();
}
```

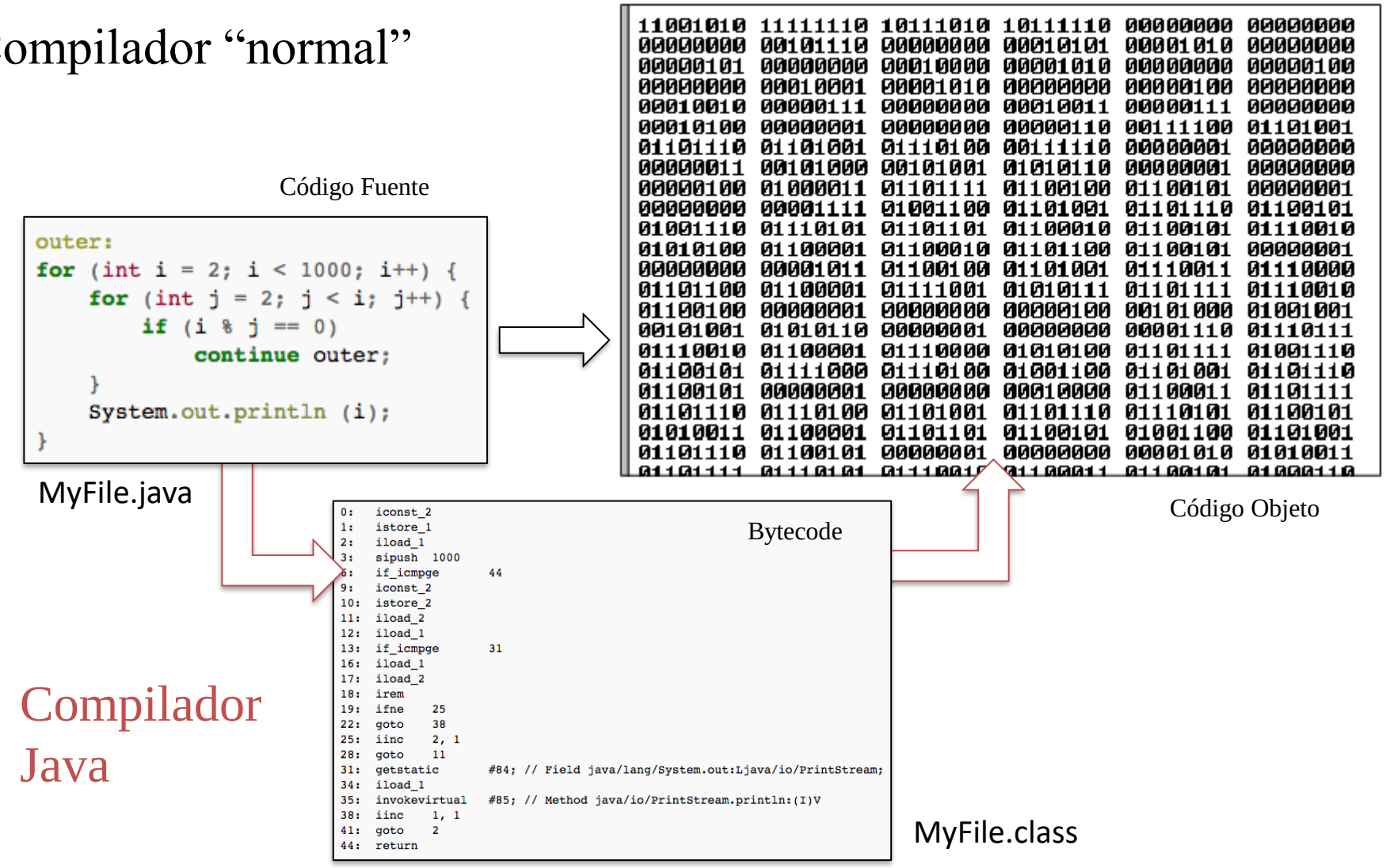
  - Alguien que es capaz de:
    - Descomponer problemas grandes en problemas más pequeños que se resuelven con soluciones paso a paso
    - Expresar esos pasos en un lenguaje muy particular y preciso (un lenguaje de programación)

Wrap-up

- De nuestra mente al procesador
  - Compilador: código a código objeto (human-friendly a computer-friendly)
  - Máquina Virtual: recorre las instrucciones (computer-friendly)
  - API (Application Programming Interface): montones de código disponible para ser utilizado

Wrap Up

Compilador “normal”



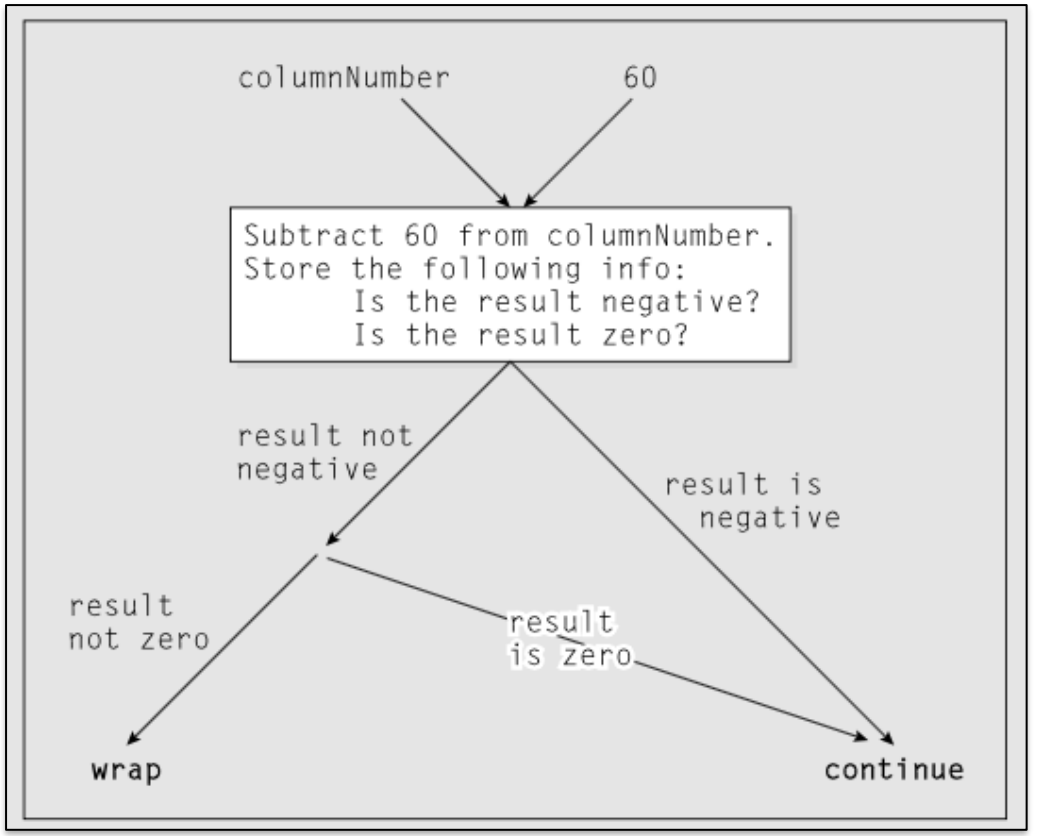
Compilador Java

Wrap Up

- Bytecode
  - El código fuente (.java) de nuestro programa describe las operaciones que debe hacer el ordenador
  - El compilador traduce el código fuente a Bytecode (.class) que *explota* cada instrucción en un conjunto de diminutos pasos que el procesador puede llevar a cabo
    - Trasiego de datos a memoria
    - Operaciones con esos datos (…)

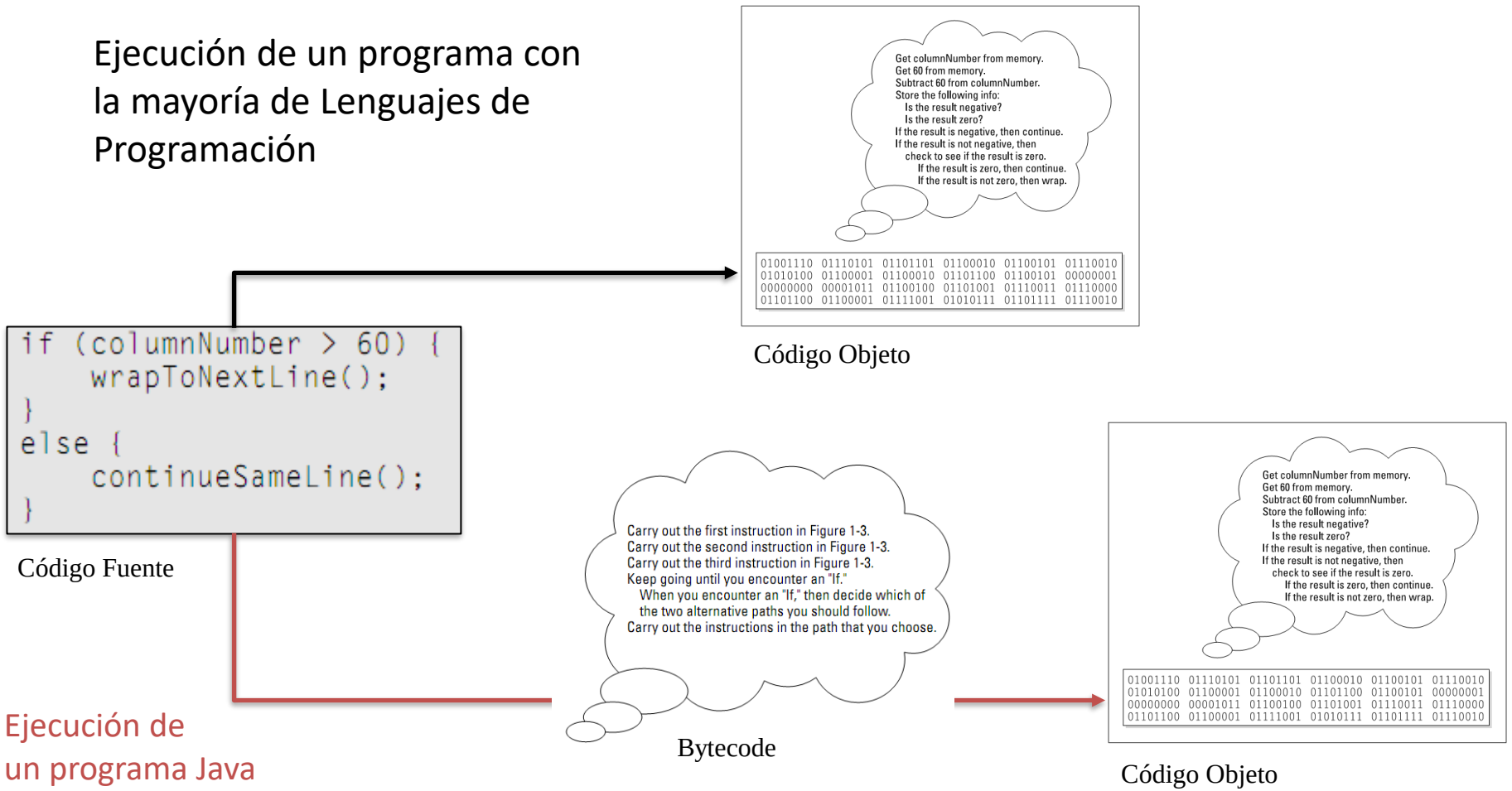
- Ejemplo Bytecode

```
if (columnNumber > 60) {
    wrapToNextLine();
}
else {
    continueSameLine();
}
```



Wrap Up

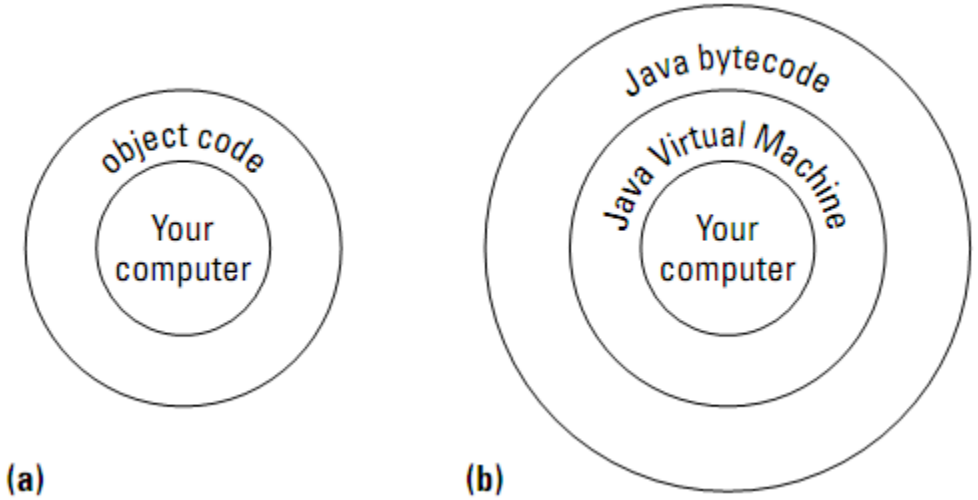
Ejecución de un programa con la mayoría de Lenguajes de Programación



Ejecución de un programa Java

Wrap Up

- Ejecución de un programa en Java



Write Once, Run Anywhere

ISe
Ingeniería de Servicios

Universidad Rey Juan Carlos

Introduccion a la programación

Fundamentos de la programación

Profesores:

Francisco Javier Pérez Blanco

Javier Sevilla López