UT1: INTRODUCCIÓN A LA PROGRAMACIÓN

Capitulo 1: INTRODUCCIÓN A LA PROGRAMACIÓN

1.1. Datos, algoritmos y programas.

1.2. Paradigmas de programación.

1.3. Lenguajes de programación.

1.4. Herramientas y entornos para el desarrollo de programas.

1.5 Errores y calidad de los programas.

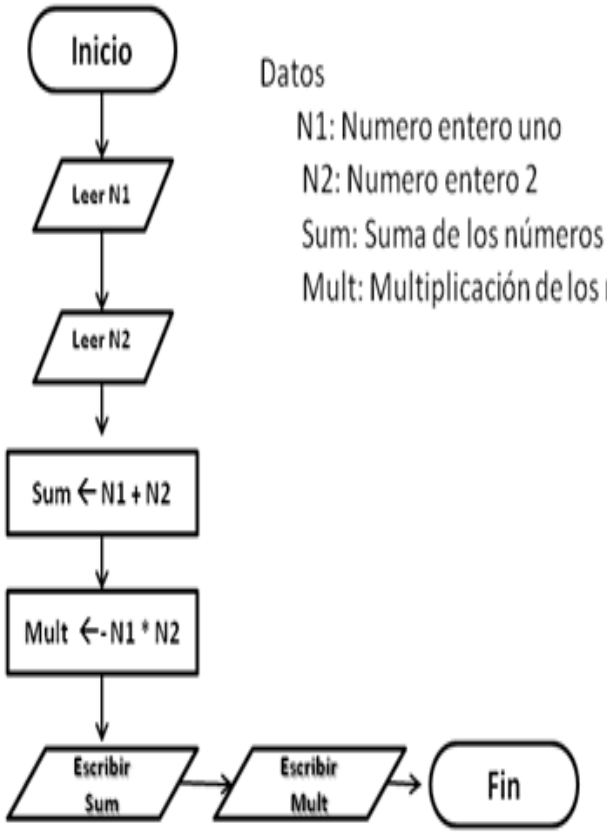
1.1. DATOS, ALGORITMOS Y PROGRAMAS

ENTRADA=DATOS → **PROCESO** → **INFORMACIÓN=SALIDA**

Pasos para crear un programa:

1. Planificación: Conocer lo que se quiere informatizar y plantear varias soluciones.
2. Análisis: Conocer a fondo el proceso a informatizar.
3. Diseño del programa
    - 3.1. Entradas y Salidas
    - 3.2. Pasos para realizar una determinada tarea (algoritmo).
4. Transformar el algoritmo en programas (codificación).
5. Compilar, enlazar y depurar.
6. Ejecución y validación

1.1. DATOS, ALGORITMOS Y PROGRAMAS



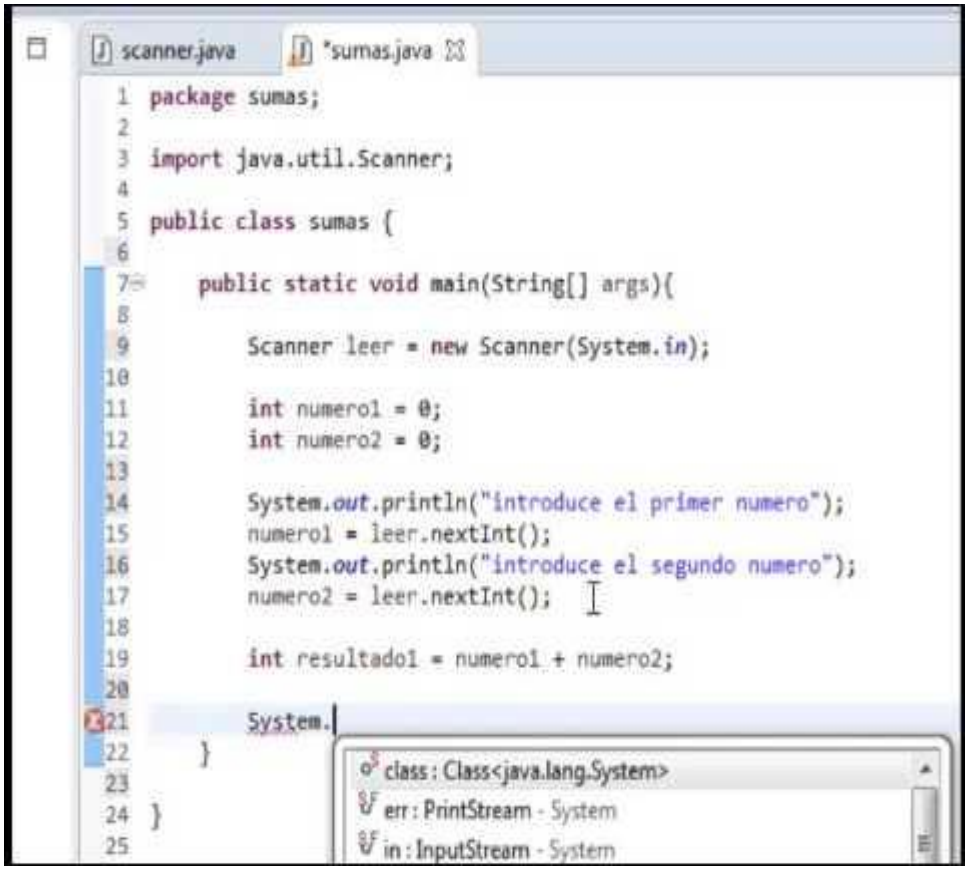
Datos

N1: Numero entero uno

N2: Numero entero 2

Sum: Suma de los números

Mult: Multiplicación de los números



1.2. PARADIGMAS DE PROGRAMACIÓN

- Paradigma de programación: forma de plantear la solución informática de un problema, que da lugar a diferentes estilos de programación.
- Clasificación:
  - **Paradigma Imperativo o por Procedimientos**: Secuencia de pasos que tiene que realizar el ordenador para llevar a cabo una determinada tarea. *C, Basic, Pascal.*
  - **Paradigma Estructurado*:*** Consiste en agrupar código (funciones, subprogramas). Nos permite reutilizar código existente y facilitar la lectura del programa.
  - **Paradigma Orientado a Objetos:** Basado en la idea de encapsular en un solo bloque los datos y las operaciones que manejan estos datos Ejem: *C++, Java, Visual Basic,etc.,*

    Ventaja : reutiliza código y la facilidad para hacer determinados tipos de programas.

1.3. LENGUAJES DE PROGRAMACIÓN

- En principio, todos los programas eran creados por el único código que el ordenador era capaz de entender: **el código máquina**, un conjunto de 0s y 1s de grandes proporciones.

- Este método de programación, absolutamente efectivo y sin restricciones, convertía la tarea de programar en una labor tediosa ➔ se toma la solución de:
  - establecer un nombre a las secuencias de programación más frecuentes,
  - estableciéndolas en posiciones de memoria concretas,
  - a cada una de estas secuencias nominadas se las llamó instrucciones,
  - y al conjunto de instrucciones, *lenguaje ensamblador*.

1.3. LENGUAJES DE PROGRAMACIÓN

- Ejemplo de programa ensamblador:

```
pushl   %ebp                    #preserva marco de pila anterior
.cfi_def_cfa_offset 8           #Modifica una regla para el cálculo, el desplazamient
.cfi_offset 5, -8               #Valor anterior de registro (5), se guarda en el desp
movl    %esp, %ebp              #actualiza marco de pila
.cfi_def_cfa_register 5         #A partir de ahora 5 se utilizara en lugar del anteri
andl    $-16, %esp              #Suma de $-16 al apuntador de pila
subl    $32, %esp               #Reserva 32 bytes en la pila
movl    8(%ebp), %eax           #Mueve el apuntador de marco de pila 8 a %eax
movl    %eax, 24(%esp)          #Mueve %eax al apuntador de pila 24
movl    12(%ebp), %eax          #Mueve el apuntador de marco de pila 12 a %eax
movl    %eax, 28(%esp)          #Mueve %eax al apuntador de pila 28
movl    $.LC0, (%esp)           #Mueve $.LC0 al apuntador de pila
call    puts                    #Llama a la función para agregar variable
movl    $.LC1, %eax             #Mueve $.LC1 a %eax
```

1.3 LENGUAJES DE PROGRAMACIÓN

- Más adelante, *empezaron a usar los ordenadores científicos de otras ramas*,

  ➔ les era sumamente complicado el uso del lenguaje ensamblador.

- Para facilitar la tarea de programar, nace el concepto de lenguaje de alto nivel con FORTRAN (*FORmula TRANslation)*

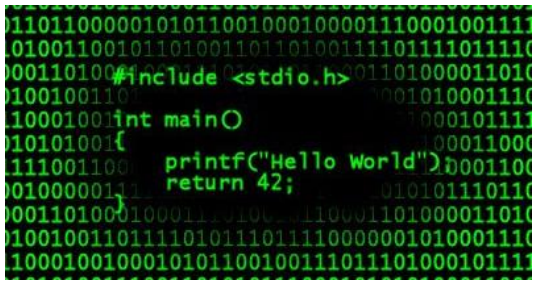
```
PROGRAM TRIVIAL
   INTEGER I
   I=2
   IF(I .GE. 2) CALL PRINTIT
   STOP
END
SUBROUTINE PRINTIT
   PRINT *,'Hola Mundo'
   RETURN
END
```

1.3 LENGUAJES DE PROGRAMACIÓN

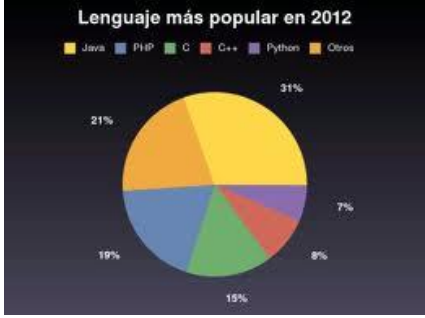
- Los *lenguajes de alto nivel* son aquellos que *elevan la abstracción del código máquina* lo más posible,

  ➔ programar es una tarea más liviana, entendible e intuitiva.

- Pero sea cual sea, el lenguaje que usamos, el compilador lo convierte en código de 1s y 0s, que son los que llegan a la máquina.

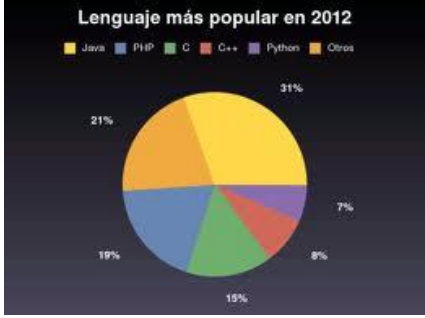


1.3.1. CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN



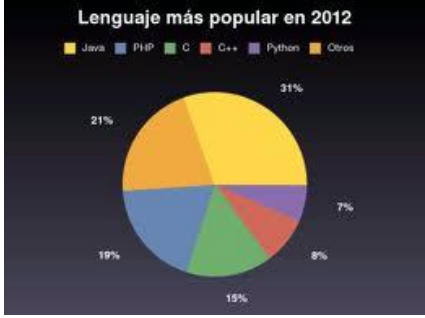
- Existe gran cantidad de lenguajes de programación.
- Cada uno con unas características y objetivos determinados.
- Es necesario establecer criterios para clasificarlos según sus características principales.
- Se pueden clasificar por diferentes criterios.

CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN



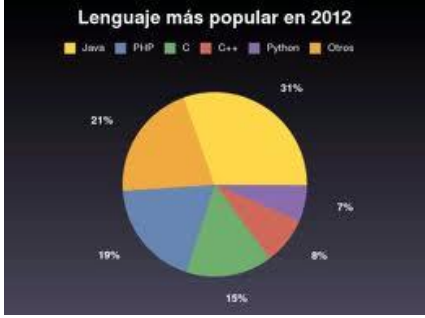
- Clasificaciones siguiendo 3 criterios globales y reconocidos:
  - Por el nivel de abstracción,
  - Por la forma de ejecución
  - y por el paradigma.

Nivel de abstracción



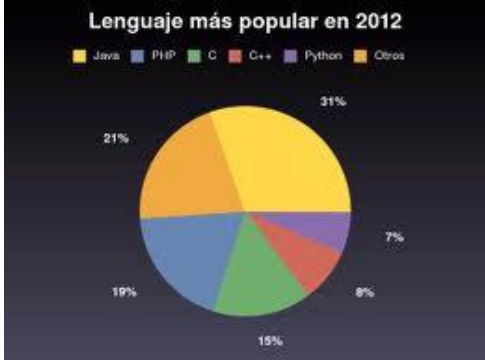
- Es el modo en que los lenguajes se alejan del código máquina y se acercan más a un lenguaje similar a los que utilizamos diariamente para comunicarnos.
- Cuanto más alejado esté del código máquina, de mayor nivel será el lenguaje.
- El nivel de abstracción es la cantidad de “capas” de ocultación de código máquina que hay entre el código que escribimos y el código que la máquina ejecutará.

Por el Nivel de Abstracción



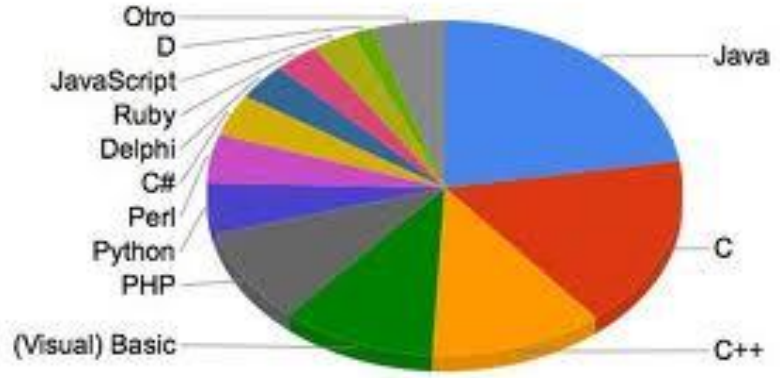
- *Lenguajes de bajo nivel*
  - Primera generación: el código máquina. Cadenas interminables de secuencias de 1s y 0s que conforman operaciones que la máquina puede entender sin interpretación.
- *Lenguajes de medio nivel*
  - Segunda generación: tienen definidas unas instrucciones para realizar operaciones sencillas con datos simples o posiciones de memoria: lenguaje ensamblador.
- *Lenguajes de alto nivel:* tercera, cuarta y quinta generación.

Generaciones de los lenguajes de programación



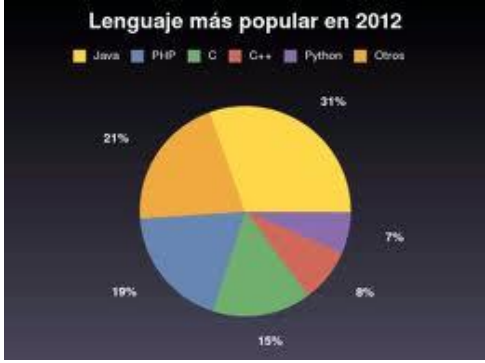
- **1 GL**: Códigos máquina
- **2 GL**:Leguajes ensambladores y primeros lenguajes de alto nivel, no estructurados: Fortran, Cobol, Basic….
- **3 GL**: Lenguajes de alto nivel . Gramática y sintaxis similar a las palabras en una oración. Necesitan de un Compilador para traducir a código máquina. Ejem: Pascal, C, Ada, Cobol,….
- **4 GL**: Programación orientada a objetos, acceso a datos, generación automática de código, programación visual. Ejem: Visual Basic, Visual C, C++, Java,
- **5 GL**: Lenguajes de inteligencia artificial. Imitan el funcionamiento de la mente humana. Ejem. Lisp

Por el Nivel de Abstracción



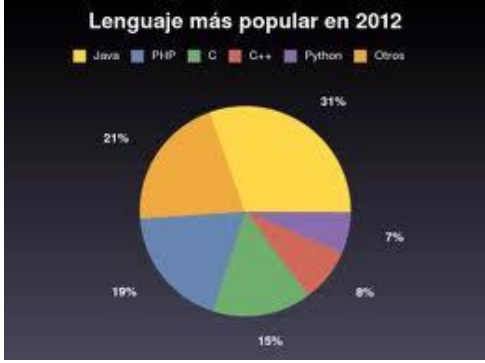
- *Lenguajes de alto nivel:*
- Tercera generación: la gran mayoría de los lenguajes de programación que se utilizan hoy pertenecen a este nivel de abstracción.
- Son los lenguajes del paradigma de programación orientada a objetos, de propósito general.
- Permiten una forma de programar entendible e intuitiva.
- Algunas instrucciones parecen una traducción directa del lenguaje humano: *IF contador = 10 THEN STOP.*

Por la Forma de Ejecución



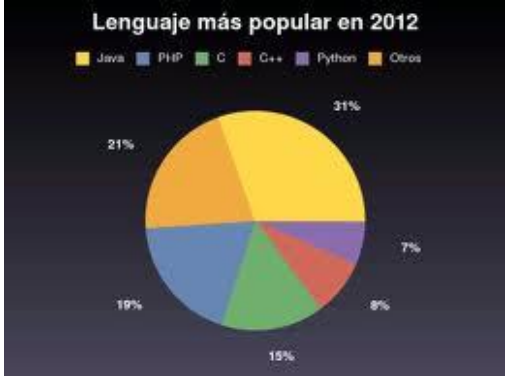
- Lenguajes compilados
- Lenguajes interpretados
- Lenguajes virtuales

Por la Forma de Ejecución



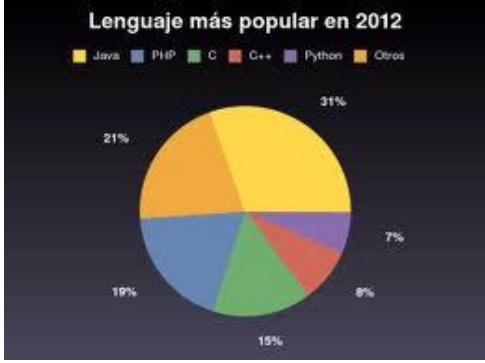
- **Lenguajes interpretados:** ejecutan las instrucciones directamente, sin que se genere código objeto.
  - Es necesario un programa intérprete en el sistema operativo o en la propia máquina donde cada instrucción se interpreta y ejecuta de manera independiente y secuencial.
  - La principal diferencia con el anterior es que se traducen a tiempo real solo las instrucciones que se utilicen en cada ejecución en vez de interpretar todo el código, se vaya a utilizar o no.

Por la Forma de Ejecución



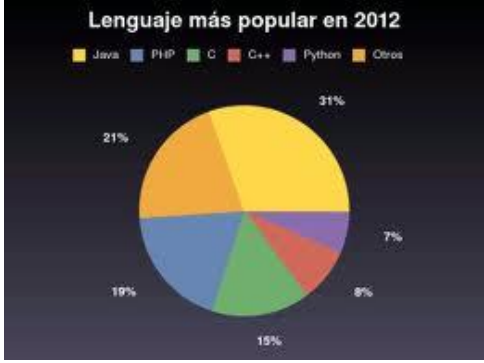
- **Lenguajes compilados**:
  - un programa traductor (compilador) convierte el código fuente en código objeto
  - y otro programa (enlazador) une el código objeto del programa con el código objeto de las librerías necesarias para producir el programa ejecutable.

Por la Forma de Ejecución



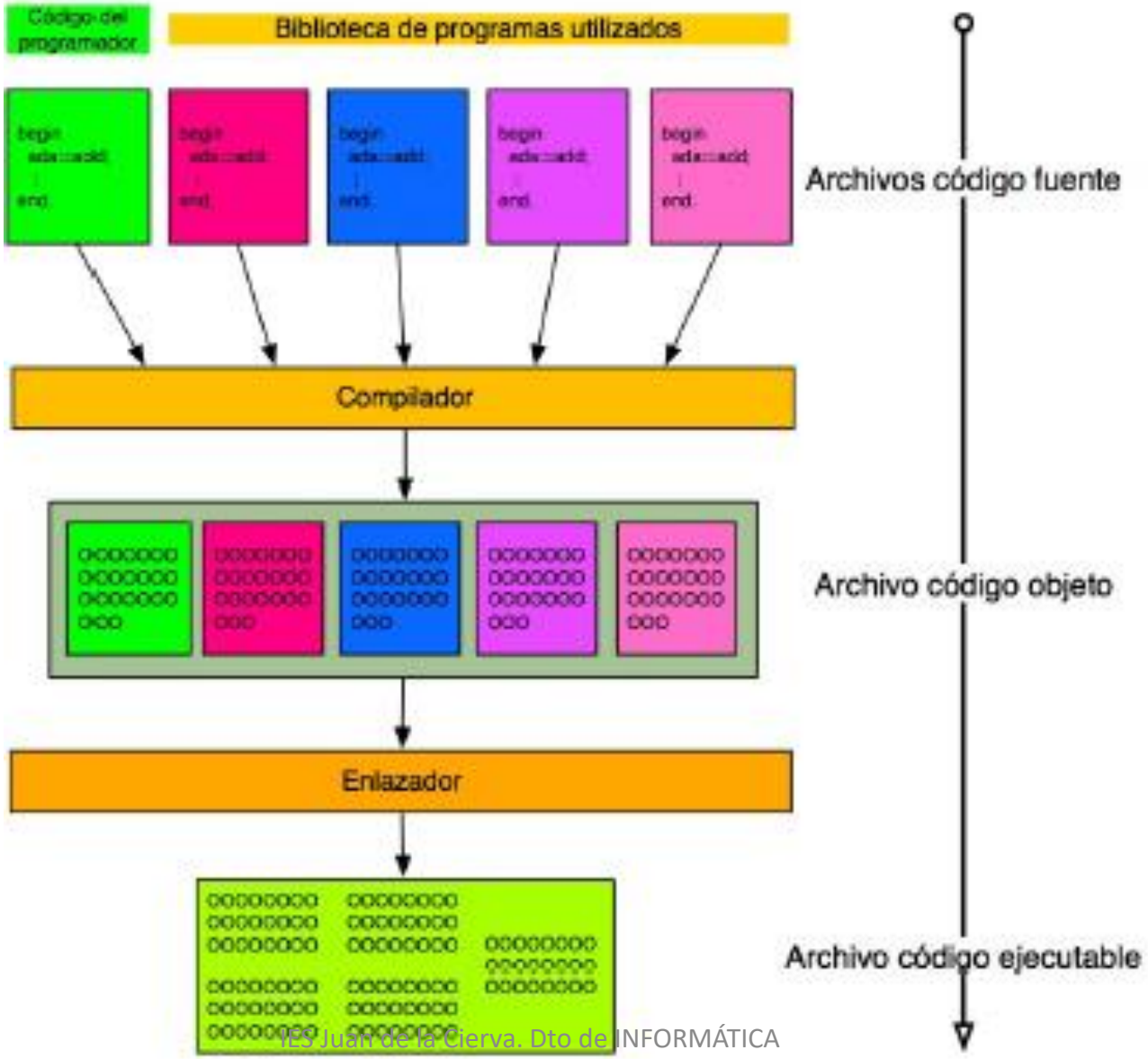
- **L. Compilados vs L. Interpretados**
  - Ventajas L.Compilados: No hace falta traducir el programa cada vez.
  - Inconveniente : No se ejecuta hasta que no haya ningún error.

Por la Forma de Ejecución

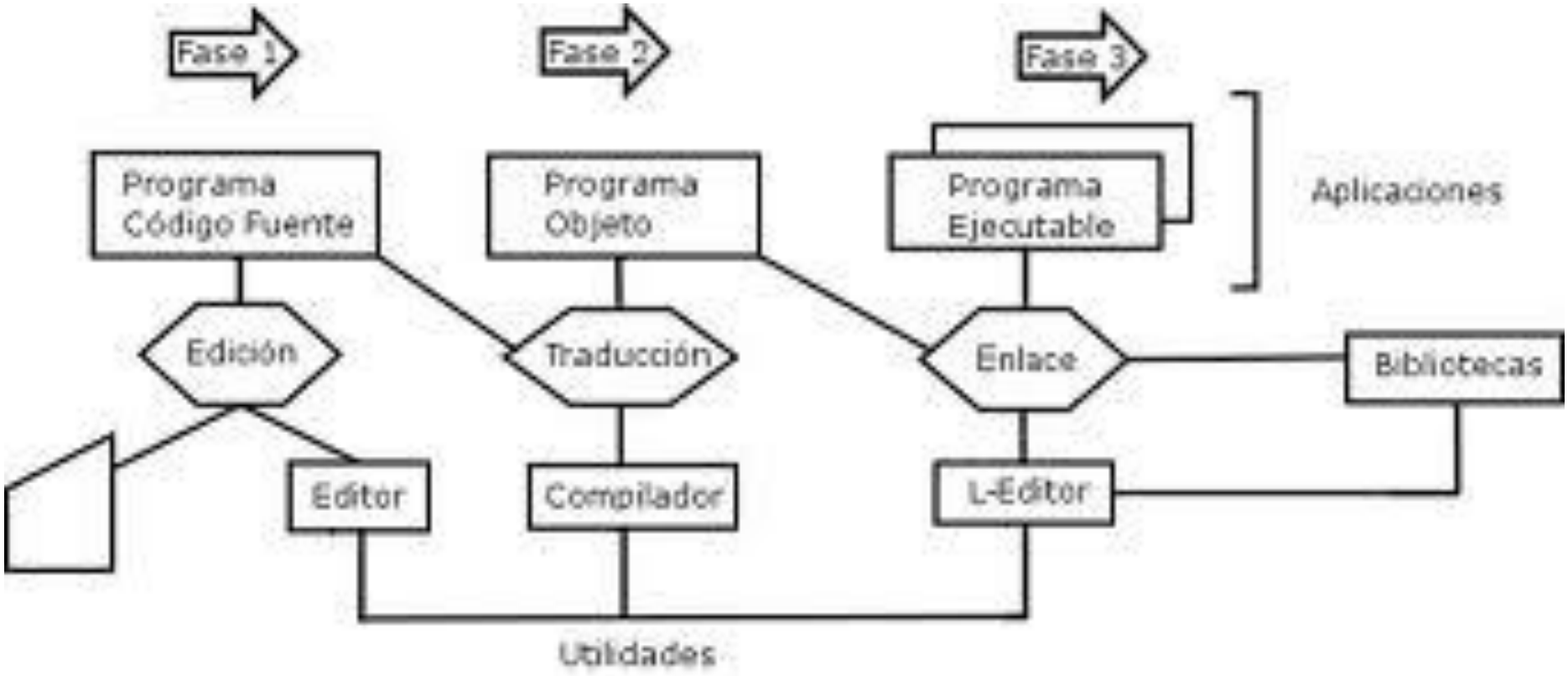


- **Lenguajes virtuales/intermedios:**
  - tienen un funcionamiento similar al de los lenguajes compilados,
  - pero, no es código objeto lo que genera el compilador, sino un *bytecode* que puede ser interpretado por cualquier arquitectura.
  - Son lentos, pero tienen la ventaja de poder ser multisistema: un mismo código *bytecode será válido para* cualquier máquina.
  - Uno de estos lenguajes es Java

OBTENCIÓN DE CÓDIGO EJECUTABLE



OBTENCIÓN DE CÓDIGO EJECUTABLE



Tipos de CÓDIGO

- **Código fuente:** es un conjunto de instrucciones escritas en un lenguaje de programación determinado. E
  - Es el código en el que nosotros escribimos nuestro programa.

- **Código objeto:** es el código *resultante de compilar* el código fuente.
  - Si se trata de un lenguaje de programación compilado, el código objeto será código máquina,
  - Si se trata de un lenguaje de programación virtual/interprete, será código *bytecode.*

- **Código ejecutable:** es el resultado obtenido de enlazar nuestro código objeto con las librerías.
  - Este código ya es nuestro programa ejecutable, programa que se ejecutará directamente en nuestro sistema o en otros.

Lenguaje Intermedio
Java

- Ficheros con programa fuente: extensión “java”
- Ficheros con la traducción al lenguaje intermedio son de extensión “class”.
- El lenguaje intermedio que se usa se llama “bytecodes”.

Lenguaje Intermedio
Java

- Ficheros con programa fuente: extensión “java”
- Ficheros con la traducción al lenguaje intermedio son de extensión “class”.
- El lenguaje intermedio que se usa se llama “bytecodes”.

1.4. Herramientas y entornos para el desarrollo de programas.

Existen aplicaciones que permiten poder desarrollar programas escritos en un lenguaje de programación como puede ser Java.

Los más conocidos Eclipse y NetBeans.

Son herramientas de desarrollo de programas, como IDE (Integrates Developmente Environment) que facilitan entre otras tareas:

- Escribir un programa
- Realizar la traducción y ejecución.
- Indicar los posibles errores del programa
- Poder realizar interfaces gráficas de forma fácil.

1.5 Errores y calidad de los programas.

- Las pruebas de software (testing) son una tarea que consiste en comprobar si un programa funciona correctamente.
- Los errores pueden ser:
  - Errores de codificación (“BUGS”)
  - Errores en tiempo de ejecución
  - Errores lógicos
  - Errores de especificaciones.

1.5 Errores y calidad de los programas.

- **Errores de codificación** (“BUGS”) – Errores de sintaxis. Palabras reservadas mal escritas, una instrucción incompleta.

Ejemplo:

```
Public class Principal {
        public static void main(String args[])
        {
            Syste.out.println(“\nHola, buenos días”);
         }
}
```

1.5 Errores y calidad de los programas.

Errores de ejecución : Se producen durante la ejecución del programa cuando pedimos hacer algo imposible o ilógico.

Por ejemplo operaciones aritméticas imposibles, convertir caracteres a números,…

Errores lógicos : El programa traductor no da ningún error, pero el programa no hace lo que debe, su resultado es erróneo.

Errores de especificación : Se producen por un mal entendimiento de cuales eran los requisitos que debía cumplir el programa.

1.5 Errores y calidad de los programas.

Las características generales que debe reunir un programa son:

- Legibilidad : escrito de forma clara y sencilla.
- Fiabilidad: debe contemplar todas las situaciones posibles.
- Portabilidad: poder ejecutarse en cualquier máquina o plataforma.
- Modularidad: Se debe dividir en partes.
- Eficiencia: aprovechar al máximo los recursos del ordenador (memoria y tiempo de ejecución).
- Bien documentado (documentación externa e interna).