ENTORNOS DE DESARROLLO UD-1

DESARROLLO DEL SOFTWARE (II)

LENGUAJES DE PROGRAMACIÓN

ÍNDICE

[PROGRAMA INFORMÁTICO 3](#_Toc116741076)

[CREACIÓN DE UN PROGRAMA 3](#_Toc116741077)

[A. OBTENCIÓN DE CÓDIGO EJECUTABLE 3](#_Toc116741078)

[B. PROCESO DE COMPILACIÓN/INTERPRETACIÓN 3](#_Toc116741079)

[C. LENGUAJES COMPILADOS 4](#_Toc116741080)

[D. LENGUAJES INTERPRETADOS 4](#_Toc116741081)

[LENGUAJES DE PROGRAMACIÓN 4](#_Toc116741082)

[CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN 5](#_Toc116741083)

[A. LENGUAJES DE PROGRAMACIÓN ESTRUCTURADA 6](#_Toc116741084)

[B. LENGUAJES DE PROGRAMACIÓN MODULAR 6](#_Toc116741085)

[C. LENGUAJES ORIENTADOS A OBJETOS 7](#_Toc116741086)

[1. CARACTERÍSTICAS 7](#_Toc116741087)

[2. TÉRMINOS RELATIVOS A LA POO 7](#_Toc116741088)

[3. LENGUAJES ORIENTADOS A OBJETOS 7](#_Toc116741089)

[JAVA 7](#_Toc116741090)

[A. CARACTERÍSTICAS 7](#_Toc116741091)

[B. VENTAJAS Y DESVENTAJAS 8](#_Toc116741092)

[COMPILADOR JIT 8](#_Toc116741093)

[JRE, JDK y JVM 8](#_Toc116741094)

# PROGRAMA INFORMÁTICO

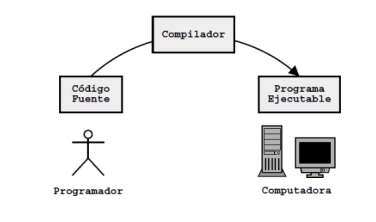
Se puede definir como el conjunto de instrucciones escritas en un lenguaje de programación que aplicadas sobre un conjunto de datos resuelven un problema o parte del mismo.

# CREACIÓN DE UN PROGRAMA

En la fase de codificación de un software hay que desarrollar el código fuente. Tiene este nombre porque la máquina no entiende el código, sí lo entiende el humano.

Por lo que desarrollamos un programa en ese código fuente y lo pasamos por un compilador que lo traduce a lenguaje máquina en un código ejecutable.

Por tanto, el proceso pasa por 3 tipos de código:

* **Código fuente**. Conjunto de instrucciones escritas en un lenguaje de programación determinado. Es el código que se escribe en el IDE.
* **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 mientras que, si se trata de un lenguaje de programación virtual, será código bytecode.
* **Código ejecutable**. También llamado código máquina que es el código que la máquina entiende y que es resultante tras una compilación correcta.

## OBTENCIÓN DE CÓDIGO EJECUTABLE

Para obtener código ejecutable (binario) tenemos 2 opciones:

1. Compilar.
2. Interpretar.

## PROCESO DE COMPILACIÓN/INTERPRETACIÓN

El proceso de compilación comienza introduciendo el código fuente, si aparecen errores, el **compilador** devuelve el código (junto al error) al programador para que lo corrija; si no los hay, crea el código objeto.

Una vez que tenemos el código objeto entra en juego el **enlazador** que introduce las **librerías** (conjunto de funciones que ofrece una API, serían los *import* de Java o los *#include* de C) y crea el código ejecutable o programa ejecutable.

Aunque los *import/#include* se realizan en el código fuente estos no se incluyen realmente en el código/programa, sólo se referencian para poder ser utilizados, no es hasta que el código llega al enlazador que son realmente incluidos en el programa.

La **compilación** pasa por varias fases:

1. **Análisis léxico**. Genera los tockens (los lexemas -conjunto de caracteres que tienen un significado-) y se asegura que el conjunto de símbolos esté dentro de los permitidos.
2. **Análisis sintáctico**. Comprueba que las normas de construcción del lenguaje se aplican correctamente.
3. **Análisis semántico**. Se asegura que el significado de construcción es coherente para realizar acciones válidas.
4. **Generador de código objeto**. Si llegados a este punto no ha habido errores, se genera el código objeto.
5. **Enlazador**. Enlaza con las librerías para generar código ejecutable.

Conclusión. Un programa cuyo código esté escrito correctamente no significa que funcione según lo deseado.

## LENGUAJES COMPILADOS

Ejemplos: C, C++, Java, Cobol, Pascal, Basic, Fortran

* + **Principal ventaja**. Ejecución muy eficiente.
  + **Principal desventaja**. Necesario compilar cada vez que el código fuente es modificado.

## LENGUAJES INTERPRETADOS

Ejemplos: PHP, Javascript, Script

* + **Principal ventaja**. El código fuente se interpreta directamente.
  + **Principal desventaja**. Ejecución menos eficiente (más lenta).

# LENGUAJES DE PROGRAMACIÓN

Se pueden definir como lenguajes creados de forma artificial para poder comunicarse con la máquina.

Según las necesidades del proyecto se utilizará un lenguaje u otro.

Han tenido la siguiente evolución:

* **Lenguaje máquina.** Fue el primer lenguaje utilizado y sus instrucciones son 1 y 0. Es el único que entiende directamente el ordenador y no necesita traducción. Es único de cada procesador por lo que no se puede ejecutar en arquitecturas diferentes a la original.
* **Lenguaje ensamblador.** Utiliza mnemotécnicos (instrucciones que entiende el hardware). Para su ejecución es necesaria su traducción al lenguaje máquina.
* **Lenguajes de alto nivel.** Utiliza sentencias y órdenes derivadas del inglés. También necesitan traducción al lenguaje máquina.
* **Lenguaje visual.** En lugar de sentencias escritas se programa gráficamente usando el ratón y directamente la apariencia del software. Se genera un código de alto nivel de forma automática.

# CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

* Según lo **cerca que esté del lenguaje humano**:
  + **Lenguajes de alto nivel.** Más cerca del lenguaje humano y de su razonamiento. Son independientes de la arquitectura así que precisan de traducción al lenguaje máquina.
  + **Lenguajes de bajo nivel.** Más cerca del lenguaje máquina y al funcionamiento interno de los ordenadores. Tienen un muy buen rendimiento. Se distingue:
    - **Lenguaje máquina**.Se programan sus registros directamente con series de 0 y 1. Es difícil de programar y de resolver errores.
    - **Lenguaje ensamblador**. Se trata de un primer nivel de abstracción respecto al lenguaje máquina, aunque conceptualmente está mucho más cercano al equipo que al razonamiento humano.
* Lenguajes interpretados vs compilados:
  + **Lenguaje compilado**. Se traduce en un compilador y se crea un fichero binario ejecutable particular para cada máquina. Este ejecutable ya no necesita ningún programa para ser utilizado.
  + **Lenguaje interpretado**. Se transforma el programa a un formato acordado, no interpretable directamente por ningún tipo de máquina. Posteriormente se utilizan intérpretes para conseguir el código máquina correspondiente.
* Lenguajes imperativos vs declarativos:
  + **Programación imperativa**. Describen las instrucciones paso a paso que han de ejecutarse para resolver un problema. Por ejemplo, Java.
  + **Programación declarativa**. Describen el problema que se quiere solucionar, pero no las instrucciones para hacerlo. Por ejemplo, SQL.
* Según la técnica de programación utilizada:
  + **Lenguajes de programación estructura**. Estructuran el programa en bloques (C y Pascal).
  + **Lenguajes de Programación Orientada a Objetos (POO)**. Estructuran el programa en bloques de objetos (C++, Java, C#...).
  + **Lenguajes de programación visuales**. Permiten programar gráficamente, posteriormente se consigue el código equivalente (Visual Basic.Net, C#, .Net…).

## LENGUAJES DE PROGRAMACIÓN ESTRUCTURADA

Sólo permiten tres tipos de sentencias:

* + Sentencias condicionales.
  + Sentencias secuenciales.
  + Sentencias repetitivas.

La ventaja es que los programas son fáciles de leer, sencillos y rápidos. Y su mantenimiento es sencillo.

El problema es que todo el programa se concentra en un único bloque, lo cual impide que se pueda reutilizar código (único módulo).

## LENGUAJES DE PROGRAMACIÓN MODULAR

Evolución natural desde los lenguajes de programación estructurada. Dividen el programa en trozos de código llamados módulos con una funcionalidad concreta que podrán ser reutilizables. Por ejemplo, si creamos una calculadora dividimos el programa en bloque suma, resta, multiplicación… y en función de la operación se llama a una u otra función.

## LENGUAJES ORIENTADOS A OBJETOS

Los programas se componen de objetos independientes entre sí que colaboran para realizar acciones.

Estos lenguajes no tratan a los programas como un conjunto ordenado de instrucciones sino un conjunto de objetos que colaboran entre ellos para realizar acciones.

1. CARACTERÍSTICAS
   * Se definen **clases** que son colecciones de objetos con características similares, los **atributos**.
   * Las clases definen una serie de **métodos** que corresponden a las acciones que pueden llevar a cabo los objetos.
   * El **código es más reutilizable**.
2. TÉRMINOS RELATIVOS A LA POO
   * Objeto.
   * Mensaje.
   * Método.
   * Evento (sistema).
   * Propiedad o atributo.
   * Estado interno.
   * Abstracción.
   * Encapsulamiento.
   * Polimorfismo.
   * Herencia.
3. LENGUAJES ORIENTADOS A OBJETOS

Ada, C++, Smalltall, C#, Java, Visual Basic.NET…

# JAVA

## CARACTERÍSTICAS

1. Es un lenguaje **compilado** e **interpretado**.
2. El código fuente de Java se compila y se obtiene el **bytecode** (código binario intermedio).
3. El bytecode puede considerarse código objeto, pero destinado a la JVM en lugar de código objeto nativo.
4. Después este bytecode se interpreta para ejecutarlo.

## VENTAJAS Y DESVENTAJAS

1. Ventajas:
   * Es estructurado y orientado a objetos.
   * Es relativamente fácil de aprender.
   * Buena documentación para aprenderlo y base de usuarios.
2. Desventajas:
   * Menos eficiente que los lenguajes compilados.

# COMPILADOR JIT

Interfaz de usuario gráfica

Descripción generada automáticamente

# JRE, JDK y JVM

1. ***Diagrama

   Descripción generada automáticamente con confianza media*JVM**: ***Java Virtual Machine***. Es el responsable de ejecutar el programa Java línea por línea (archivo .class o bytecode) por lo que se conoce como intérprete.
2. **JRE**: ***Java Runtime Environment***. Proporciona los requisitos básicos para ejecutar una aplicación Java. Es un compendio que contiene la JVM, las clases principales y los archivos auxiliares.
3. **JDK**: ***Java Development Kit***. Permite el desarrollo de aplicaciones y applets de Java. Incluye el JRE (y por tanto la JVM), un intérprete (Java), un compilador (javac), un archivador (jar), un generador de documentación (Javadoc) y otras herramientas necesarias para el desarrollo de Java.

Es un kit que proporciona el entorno para desarrollar y ejecutar el programa Java que incluye dos cosas:

* + Herramientas que proporcionan un entorno de desarrollar para programas Java.
  + JRE para ejecutar los programas Java desarrollados.