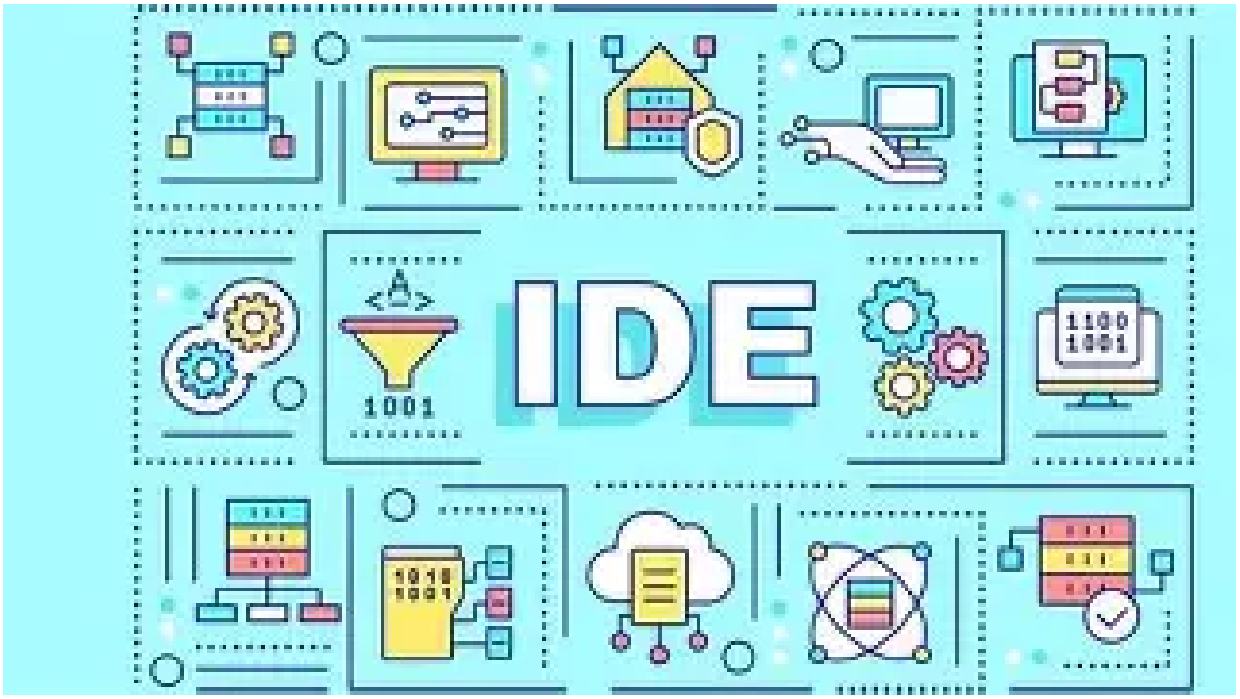
A1.1 Conceptos iniciales



Adrián Alemán Antúnez

1ºDAW

Qué es un IDE y di al menos 3 de los más usados. **3**
Ciclo de vida de un software. **4**

Qué es un IDE y di al menos 3 de los más usados.

Un IDE (Entorno de Desarrollo Integrado) es una aplicación de software diseñada para ayudar a los desarrolladores a escribir, depurar y ejecutar su código de manera eficiente.

3 IDE más usados:
1. Visual Studio Code.
2. IntelliJ IDEA.
3. Eclipse.

Ciclo de vida de un software.

El ciclo de vida de un software se compone de varias etapas estructuradas que guían el desarrollo de aplicaciones desde su concepción hasta su retiro. Las etapas principales son:
Requisitos: Definición de objetivos y funcionalidades que debe cumplir el software.
Diseño: Creación de la arquitectura del software y definición de componentes.
Desarrollo: Implementación del software siguiendo el diseño establecido.
Pruebas: Verificación de que el software cumple con los requisitos.
Implementación: Despliegue del software en el entorno de producción.
Mantenimiento: Actualizaciones y correcciones necesarias después de la implementación.
Retiro: Proceso de descontinuación del software cuando ya no es necesario.

Este proceso estructurado asegura que el software se desarrolle de manera eficiente y cumpla con los estándares requeridos.

Fases en el desarrollo y ejecución del software.

Las fases en el desarrollo y ejecución del software incluyen:

Requisitos: Definir las necesidades y expectativas del usuario.
Diseño: Crear un diseño que cumpla con los requisitos.
Codificación: Escribir el código del software.
Pruebas: Realizar pruebas para asegurar que el software funcione correctamente.
Implementación: Introducir el software en el entorno de producción.
Mantenimiento: Asegurar que el software se mantenga actualizado y funcione correctamente a lo largo del tiempo.

Estas fases son esenciales para garantizar la eficiencia y fiabilidad del desarrollo de software.

Diferencia entre código fuente, código objeto y código ejecutable.

La diferencia entre código fuente, código objeto y código ejecutable radica en su estado de preparación y su capacidad para ser ejecutado. **El código fuente** es el conjunto de instrucciones escritas por un programador en un lenguaje de programación legible por humanos. **El código objeto** es una versión intermedia que contiene instrucciones traducidas al lenguaje de máquina, pero que aún necesita ser ensamblada. **El código ejecutable** es el resultado final que se obtiene después de que el código objeto se ha vinculado con las librerías necesarias, lo que permite que el programa sea directamente ejecutado por la computadora.

Diferencia entre algoritmo, pseudocódigo y código.

Algoritmo: Es un conjunto ordenado de pasos o instrucciones que resuelven un problema o realizan una tarea específica.

Pseudocódigo: Es una representación de un algoritmo que utiliza un lenguaje similar al lenguaje humano y a la programación, pero que no sigue ninguna sintaxis estricta de un lenguaje de programación específico.

Código: Es la representación real de un algoritmo en un lenguaje de programación específico, como Java, Python, C++, entre otros.

Estos conceptos son fundamentales en la creación de software y en la enseñanza de la programación.

¿Qué es el DISEÑO ESTRUCTURADO y cuáles son los 3 tipos de construcciones en las que se basa?

El diseño estructurado se basa en diferentes tipos de construcciones, que se clasifican según características estructurales, materiales utilizados y uso previsto. Estos tipos de construcciones son útiles para quienes trabajan en la construcción y buscan adquirir competencias para gestionar solicitudes de todo tipo. Algunos de los tipos de construcciones incluyen:

Mampostería: Utiliza ladrillos o bloques de cemento, que se apilan y se conectan mediante mortero para crear muros portantes.
Madera: Común en muchas partes del mundo, apreciada por su eficiencia energética y su estética natural.
Acero: Conocido por su resistencia y flexibilidad estructural, utilizado en construcciones comerciales e industriales.

Qué es un lenguaje de programación y clasificación de los lenguajes en función de su nivel de abstracción y según la forma de ejecución. Pon ejemplos.

Lenguaje de programación es un programa destinado a la construcción de otros programas informáticos. Su nombre se debe a que comprende un lenguaje formal que está diseñado para organizar algoritmos y procesos lógicos que serán luego llevados a cabo por un ordenador o sistema informático, permitiendo controlar así su comportamiento físico, lógico y su comunicación con el usuario humano.

Normalmente se distingue entre los siguientes tipos de lenguaje de programación:

Lenguajes de bajo nivel. Se trata de lenguajes de programación que están diseñados para un hardware específico y que por lo tanto no pueden migrar o exportarse a otras computadoras. Sacan el mayor provecho posible al sistema para el que fueron diseñados, pero no aplican para ningún otro.

Lenguajes de alto nivel. Se trata de lenguajes de programación que aspiran a ser un lenguaje más universal, por lo que pueden emplearse indistintamente de la arquitectura del hardware, es decir, en diversos tipos de sistemas. Los hay de propósito general y de propósito específico.

Lenguajes de nivel medio. Este término no siempre es aceptado, que propone lenguajes de programación que se ubican en un punto medio entre los dos anteriores: pues permite operaciones de alto nivel y a la vez la gestión local de la arquitectura del sistema.

Otra forma de clasificación a menudo es la siguiente:

Lenguajes imperativos. Menos flexibles, dada la secuencialidad en que construyen sus instrucciones, estos lenguajes programan mediante órdenes condicionales y un bloque de comandos al que retornan una vez llevada a cabo la función.

Lenguajes funcionales. También llamados procedimentales, estos lenguajes programan mediante funciones que son invocadas conforme a la entrada recibida, que a su vez son resultado de otras funciones.

Ejemplo:

BASIC.

COBOL.

FORTRAN.

Java.

Diferencia entre compilador e intérprete.

La diferencia entre un compilador y un intérprete es la siguiente:
Compilador: Convierte el código fuente escrito en un lenguaje de programación a código binario o de máquina antes de que se ejecute el programa. Esto permite que el programa se ejecute de manera independiente.

Intérprete: Lee y ejecuta el código fuente línea por línea durante la ejecución, sin generar un código de máquina independiente. Esto significa que siempre necesita su intérprete correspondiente para poder ejecutarse.

Ejemplo de uso: Un compilador genera un programa "stand-alone", mientras que un programa interpretado requiere su intérprete para funcionar.

Estas diferencias son fundamentales para entender cómo funcionan los lenguajes de programación y su ejecución.

¿QUÉ ES BYTECODE?

El bytecode es un código de bajo nivel generado por los compiladores y utilizado por las máquinas virtuales para ejecutar programas. Es un conjunto de instrucciones que puede ejecutar el procesador del ordenador. Sin embargo, a diferencia del código máquina, que es específico de una determinada arquitectura de hardware, el bytecode está diseñado para ser independiente de la plataforma. Esto significa que puede utilizarse en cualquier ordenador que disponga de una máquina virtual compatible.

En la programación orientada a objetos (POO), existen tres conceptos fundamentales: clase, objeto y método. Defínelos y pon un ejemplo práctico en un lenguaje que conozcas.

En la programación orientada a objetos (POO), una clase es una estructura que permite definir un tipo de dato abstracto que combina datos y comportamientos. Una **clase** actúa como un molde a partir del cual se pueden crear instancias concretas denominadas **objetos**.

Los **atributos** son las características o propiedades que describen el estado de los objetos. Representan los datos que cada instancia de la clase mantendrá. Por ejemplo, en una clase Coche, los atributos podrían ser marca, modelo y color. Estos atributos se definen dentro de la clase y cada objeto tendrá sus propios valores para ellos.

Los **métodos** son las acciones o comportamientos que los objetos pueden realizar. Son funciones definidas dentro de la clase que operan sobre los atributos o realizan operaciones específicas. Siguiendo el ejemplo anterior, la clase Coche podría tener métodos como acelerar(), frenar() o girar(angulo).

ejemplo:

```
class Coche:
    """
    Clase que representa un coche.
    Atributos:
        marca (str): Marca del coche.
        modelo (str): Modelo del coche.
        velocidad (int): Velocidad actual en km/h.
    """

    def __init__(self, marca, modelo):
        # Constructor: inicializa los atributos
        if not isinstance(marca, str) or not isinstance(modelo, str):
            raise ValueError("Marca y modelo deben ser cadenas de texto.")
        self.marca = marca
        self.modelo = modelo
        self.velocidad = 0  # Velocidad inicial

    def acelerar(self, incremento):
        """Aumenta la velocidad del coche."""
        if not isinstance(incremento, (int, float)) or incremento <= 0:
            raise ValueError("El incremento debe ser un número positivo.")
        self.velocidad += incremento
        print(f"{self.marca} {self.modelo} ahora va a {self.velocidad} km/h.")

    def frenar(self, decremento):
        """Reduce la velocidad del coche sin permitir valores negativos."""
        if not isinstance(decremento, (int, float)) or decremento <= 0:
            raise ValueError("El decremento debe ser un número positivo.")
```

```
        self.velocidad = max(0, self.velocidad - decremento)
        print(f"{self.marca} {self.modelo} ahora va a {self.velocidad} km/h.")

# --- Creación de objetos (instancias de la clase) ---
mi_coche = Coche("Toyota", "Corolla")  # Objeto 1
otro_coche = Coche("Ford", "Mustang")  # Objeto 2

# --- Uso de métodos ---
mi_coche.acelerar(50)
mi_coche.frenar(20)

otro_coche.acelerar(80)
```

Explicación:

Clase (Coche)

Es la plantilla que define atributos (marca, modelo, velocidad) y métodos (acelerar, frenar).

Objeto (mi_coche, otro_coche)

Son instancias concretas de la clase, cada una con sus propios valores.

Método (acelerar, frenar)

Son funciones definidas dentro de la clase que operan sobre los datos del objeto.