1. ¿Defina cuál es la Diferencia entre una clase y un objeto?

**Clase:**

Una clase en programación es un plano o plantilla que define la estructura y el comportamiento de un tipo de objeto. Representa un conjunto de atributos (variables) y métodos (funciones) que definen las propiedades y el comportamiento que tendrán los objetos creados a partir de esa clase. Las clases son como un molde a partir del cual se pueden crear múltiples objetos.

class Persona {

String nombre;

int edad;

void saludar() {

System.out.println("Hola, soy " + nombre + " y tengo " + edad + " años.");

}

}

**Objeto:**

Un objeto es una instancia concreta y específica de una clase. Representa una entidad real con sus propias características y puede interactuar con otros objetos y el entorno. Cada objeto creado a partir de una clase tiene sus propios valores para los atributos y puede llamar a los métodos definidos en la clase.

public class Main {

public static void main(String[] args) {

Persona persona1 = new Persona();

persona1.nombre = "Juan";

persona1.edad = 30;

persona1.saludar(); // Salida: "Hola, soy Juan y tengo 30 años."

Persona persona2 = new Persona();

persona2.nombre = "María";

persona2.edad = 25;

persona2.saludar(); // Salida: "Hola, soy María y tengo 25 años."

}

}

1. Explique cuál es la forma en que se comunican los objetos en la POO y como se le llama a esta

comunicación.

En la Programación Orientada a Objetos (POO), los objetos se comunican entre sí mediante la interacción de sus métodos y atributos. Esta comunicación se logra a través del concepto de "mensajes" entre objetos. Los objetos interactúan enviándose mensajes para solicitar que realicen ciertas acciones o proporcionen cierta información. Esta forma de comunicación se denomina "invocación de métodos" o "llamada de métodos".

Cuando un objeto desea comunicarse con otro objeto, envía un mensaje a través de la invocación de un método del objeto destino. El objeto destino responde realizando la acción correspondiente y, en algunos casos, devolviendo un resultado.

Ejemplo:

class CuentaBancaria {

private double saldo;

public CuentaBancaria(double saldoInicial) {

saldo = saldoInicial;

}

public void depositar(double cantidad) {

saldo += cantidad;

}

public void retirar(double cantidad) {

if (saldo >= cantidad) {

saldo -= cantidad;

} else {

System.out.println("Saldo insuficiente.");

}

}

public double getSaldo() {

return saldo;

}

}

class Cliente {

private String nombre;

private CuentaBancaria cuenta;

public Cliente(String nombre, double saldoInicial) {

this.nombre = nombre;

cuenta = new CuentaBancaria(saldoInicial);

}

public void depositarEnCuenta(double cantidad) {

cuenta.depositar(cantidad);

}

public void retirarDeCuenta(double cantidad) {

cuenta.retirar(cantidad);

}

public double consultarSaldo() {

return cuenta.getSaldo();

}

}

1. Explique en qué consisten los modificadores de acceso en java.

En Java, los modificadores de acceso son palabras clave que se utilizan para controlar la visibilidad y accesibilidad de las clases, métodos, atributos y constructores en una jerarquía de clases. Estos modificadores determinan desde dónde se puede acceder y manipular los elementos de una clase en el código.

Existen cuatro tipos principales de modificadores de acceso en Java:

public: Los elementos declarados como públicos son accesibles desde cualquier clase en cualquier paquete. No hay restricciones en su acceso. Por ejemplo, una clase o método público puede ser utilizado por cualquier otra clase, incluso si está en un paquete diferente.

protected: Los elementos con el modificador protegido son accesibles dentro del mismo paquete y también por las subclases, incluso si están en paquetes diferentes. Esto permite una cierta encapsulación y permite que las subclases hereden y accedan a los atributos y métodos protegidos de su clase base.

default (sin modificador): Si no se especifica un modificador de acceso, se utiliza el modificador predeterminado. Los elementos con acceso predeterminado son accesibles solo dentro del mismo paquete. No pueden ser accedidos desde clases fuera de ese paquete, incluso si son subclases.

private: Los elementos privados son accesibles solo dentro de la misma clase. No pueden ser accedidos desde ninguna otra clase, incluidas las subclases. Esto proporciona el nivel más alto de encapsulación y seguridad.

public class Ejemplo {

public int publico;

protected int protegido;

int predeterminado; // acceso de paquete

private int privado;

public void metodoPublico() {

// Puede ser accedido desde cualquier lugar

}

protected void metodoProtegido() {

// Puede ser accedido dentro del mismo paquete y por subclases

}

void metodoPredeterminado() {

// Puede ser accedido dentro del mismo paquete

}

private void metodoPrivado() {

// Solo puede ser accedido dentro de la misma clase

}

}

1. Explique el concepto de Abstracción.

La abstracción es uno de los conceptos fundamentales en la Programación Orientada a Objetos (POO) y se refiere a la simplificación de la realidad al enfocarse en los aspectos esenciales y relevantes mientras se omiten los detalles no esenciales. En el contexto de la POO, la abstracción se logra al definir clases que representan conceptos abstractos y modelan características importantes y comportamientos relacionados.

abstract class Vehiculo {

private int velocidad;

public void acelerar(int incremento) {

velocidad += incremento;

}

public void frenar(int decremento) {

velocidad -= decremento;

}

public int obtenerVelocidad() {

return velocidad;

}

public abstract void mostrarInformacion();

}

class Automovil extends Vehiculo {

private String modelo;

public Automovil(String modelo) {

this.modelo = modelo;

}

@Override

public void mostrarInformacion() {

System.out.println("Automóvil modelo: " + modelo);

}

}

class Motocicleta extends Vehiculo {

private String marca;

public Motocicleta(String marca) {

this.marca = marca;

}

@Override

public void mostrarInformacion() {

System.out.println("Motocicleta marca: " + marca);

}

}

1. Explique el concepto de encapsulamiento e indique como se implementa en java.

El encapsulamiento es uno de los principios fundamentales de la Programación Orientada a Objetos (POO) que se refiere a ocultar los detalles internos de un objeto y proporcionar una interfaz controlada para interactuar con él. Esto significa que los detalles internos, como los atributos y los métodos privados, no son accesibles directamente desde fuera del objeto, lo que garantiza una mayor seguridad, modularidad y mantenibilidad del código.

El encapsulamiento se logra definiendo atributos como privados y proporcionando métodos públicos para acceder y manipular esos atributos. Estos métodos, conocidos como "métodos de acceso" o "getters" y "setters", permiten leer y modificar los valores de los atributos de manera controlada y segura.

public class Persona {

private String nombre;

private int edad;

public String getNombre() {

return nombre;

}

public void setNombre(String nuevoNombre) {

nombre = nuevoNombre;

}

public int getEdad() {

return edad;

}

public void setEdad(int nuevaEdad) {

if (nuevaEdad >= 0) {

edad = nuevaEdad;

}

}

}