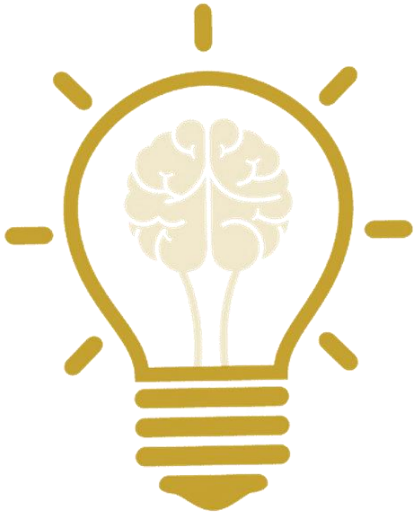
PROGRAMACIÓN EN JAVA

Materia PJA-001 material creado por el Ing. Jim Requena

Programación Orientada a Objetos en JAVA
Segunda Parte
(Encapsulación y Constructores)



Materia PJA-001 material creado por el Ing. Jim Requena

Objetivos de la sesión

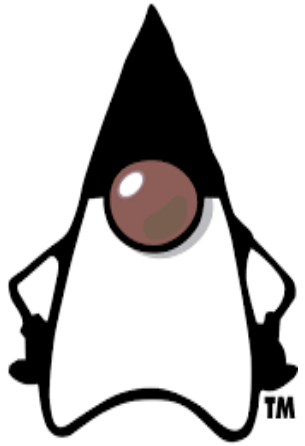
- Profundizar en el concepto de encapsulamiento
- Entender y aplicar los modificadores de acceso
- Aprender a crear y utilizar métodos getter y setter
- Comprender el uso de constructores

Repaso de Encapsulamiento

- Oculta los detalles internos de una clase
- Proporciona una interfaz pública para interactuar con el objeto
- Mejora la seguridad y facilita el mantenimiento del código

Clases y Objetos

- public: Accesible desde cualquier parte
- private: Accesible solo dentro de la misma clase
- protected: Accesible en el mismo paquete y subclases
- (default): Accesible solo en el mismo paquete



```java
public class Persona {
    public String nombre;
    private int edad;
    protected String direccion;
    String telefono; // default
}
```

Métodos Getter y Setter

- Getters: Obtienen el valor de un atributo privado
- Setters: Modifican el valor de un atributo privado

```java
public class Persona {
    private String nombre;
    private int edad;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        if (edad >= 0 && edad <= 120) {
            this.edad = edad;
        } else {
            System.out.println("Edad inválida");
        }
    }
}
```

Constructores Avanzados

- Sobrecarga de constructores
- Uso de this() para llamar a otros constructores

```java
public class Persona {
    private String nombre;
    private int edad;

    public Persona() {
        this("Sin nombre", 0);
    }

    public Persona(String nombre) {
        this(nombre, 0);
    }

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }
}
```

Inicializadores de instancia

- Se ejecutan antes del constructor
- Útiles para inicialización compleja

```java
public class Ejemplo {
    private int[] numeros;

    {
        numeros = new int[10];
        for (int i = 0; i < numeros.length; i++) {
            numeros[i] = i * 10;
        }
    }

    public Ejemplo() {
        System.out.println("Constructor llamado");
    }
}
```

Ejercicio Práctico 1/3

Crear una clase CuentaBancaria con:

1. Atributos privados: número de cuenta, saldo, titular
2. Constructores sobrecargados
3. Métodos getter y setter apropiados
4. Métodos para depositar y retirar dinero
5. Un inicializador de instancia que genere un número de cuenta aleatorio

```java
import java.util.Random;

public class CuentaBancaria {
    private String numeroCuenta;
    private double saldo;
    private String titular;

    {
        // Inicializador de instancia
        Random rand = new Random();
        this.numeroCuenta = "CTA-" + rand.nextInt(10000);
    }

    public CuentaBancaria(String titular) {
        this(titular, 0);
    }

    public CuentaBancaria(String titular, double saldoInicial) {
        this.titular = titular;
        this.saldo = saldoInicial;
    }

    public String getNumeroCuenta() {
        return numeroCuenta;
    }

    public double getSaldo() {
        return saldo;
    }
```

Ejercicio Práctico 1/2

Crear una clase CuentaBancaria con:

1. Atributos privados: número de cuenta, saldo, titular
2. Constructores sobrecargados
3. Métodos getter y setter apropiados
4. Métodos para depositar y retirar dinero
5. Un inicializador de instancia que genere un número de cuenta aleatorio

```java
public String getTitular() {
    return titular;
}

public void setTitular(String titular) {
    this.titular = titular;
}

public void depositar(double monto) {
    if (monto > 0) {
        saldo += monto;
        System.out.println("Depósito de " + monto + " realizado. Nuevo saldo: " + saldo);
    } else {
        System.out.println("El monto a depositar debe ser positivo");
    }
}
```

Ejercicio Práctico 1/3

Crear una clase CuentaBancaria con:

1. Atributos privados: número de cuenta, saldo, titular
2. Constructores sobrecargados
3. Métodos getter y setter apropiados
4. Métodos para depositar y retirar dinero
5. Un inicializador de instancia que genere un número de cuenta aleatorio

```
    public static void main(String[] args) {
        CuentaBancaria cuenta1 = new CuentaBancaria("Juan Pérez", 1000);
        System.out.println("Cuenta creada: " + cuenta1.getNumeroCuenta());
        cuenta1.depositar(500);
        cuenta1.retirar(200);

        CuentaBancaria cuenta2 = new CuentaBancaria("María López");
        System.out.println("Cuenta creada: " + cuenta2.getNumeroCuenta());
        cuenta2.depositar(100);
        cuenta2.retirar(50);
    }
}
```

Tarea

- Extiende la clase CuentaBancaria para incluir un atributo tipoDeMoneda (por ejemplo, "USD", "EUR", "GBP")
- Modifica los constructores y agrega getters/setters para el nuevo atributo
- Implementa un método convertirA(String monedaDestino, double tasaDeCambio) que simule la conversión del saldo a otra moneda
- Crea una clase Banco que pueda contener múltiples CuentaBancaria
- Implementa métodos en Banco para:Agregar nuevas cuentasBuscar una cuenta por númeroRealizar transferencias entre cuentas

Recursos adicionales

- Java Documentation on Access Control: docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html
- Java Documentation on Constructors: docs.oracle.com/javase/tutorial/java/javaOO/constructors.html

Gracias!