Clases y Objetos

Clases

Una clase es una plantilla o molde para crear objetos.

Define:

- Atributos (Estado)
- Métodos (Comportamiento)

Analogías:

- Plano de una casa.
- Molde de galletas.

Objetos

Un objeto es una entidad que tiene:

- Atributos (estado, propiedades)
- Comportamientos (acciones, métodos)

Instancia

Creado en base a una clase.

Representa algo concreto del mundo real o algo abstracto en el sistema.

Ejemplos:

- Un bombillo tiene estado (encendido/apagado) y acciones (encender/apagar)
- Una cuenta bancaria tiene saldo y puede depositar o retirar dinero

Usos

- Para **organizar** el código de forma modular
- Para **reutilizar** el código
- Para **modelar** el mundo real en un sistema

Ejemplo:

Clase `CuentaBancaria`

- Objetos: cuenta de Juan, cuenta de María, cuenta de Pedro

Clase vs Objeto

Clase: estructura común

```
CuentaBancaria

- saldo: double

- depositar(monto: double)

- retirar(monto: double)

```

Objeto: instancia concreta

```
cuentaJuan → saldo = Q500.00

cuentaMaria → saldo = Q1500.00

```

Anatomía de una Clase

Estructura básica de una clase

```
public class Bombillo {

// Atributos

private boolean encendido;


// Constructor

public Bombillo() {

encendido = false;

}


// Métodos

public void encender() {

encendido = true;

}

}

```

Constructores

- Método especial que se ejecuta al crear el objeto
- Nombre igual a la clase
- Se puede sobrecargar

```
public Cuenta(String nombre) {

this.nombre = nombre;

}
```

```
7
```

Métodos

- Definen comportamiento del objeto
- Partes:
- Modificador de visibilidad
- Tipo de retorno
- Nombre
- Parámetros

```
1
2   public void depositar(double monto) {
3
4   saldo += monto;
5
6   }
7
```

Visibilidad

- `private` : solo accesible dentro de la clase
- `public` : accesible desde cualquier parte
- `protected` : Más adelante cuando veamos herencia.

```
1
2   private String nombre;
3
4   public int contador = 0;
5
```

Encapsulamiento

- Encapsular = proteger el estado interno
- Seguridad

- Flexibilidad

Se usan `getters` y `setters`

```
1
2   public double getSaldo() {
3
4   return saldo;
5
6   }
7
8
9
10  public void setSaldo(double nuevoSaldo) {
11
12  saldo = nuevoSaldo;
13
14  }
15
```

Anatomía de un Método

Partes de un método

```
1
2   public double calcularArea(double base, double altura) {
3
4   return base * altura;
5
6   }
7
```

- Modificador: `public`
- Tipo de retorno: `double`
- Nombre: `calcularArea`
- Parámetros: `base` , `altura`
- Cuerpo: instrucciones

Ejemplo

```
public class Bombillo {

public void encender() {

System.out.println("Bombillo encendido");

}

}

```

UML

Definición

- UML = Unified Modeling Language
- Herramienta para representar gráficamente:
- Clases
- Objetos
- Relaciones

Diagrama de clase básico

```
+----------------------+

| CuentaBancaria |

+----------------------+

| - saldo: double |

| - numero: Int |
```

```
+----------------------+
| + depositar(monto) |
| + retirar(monto) |
+----------------------+
```

Clase: `CuentaBancaria`

Atributos:

- `- saldo: double`
- `- nombre: String`

Métodos:

- `+ getSaldo(): double`
- `+ depositar(monto: double): void`
- `+ retirar(monto: double): void`