Sobrecarga de Métodos y Constructores

Objetivo

Practicar la sobrecarga de métodos y constructores, así como su implementación en programas con aplicaciones prácticas.

Conceptos Clave

La **sobrecarga** de métodos y constructores permite definir múltiples métodos con el mismo nombre, pero con diferentes parámetros. Esto mejora la reutilización del código y la flexibilidad de las clases.

Ejemplo de Sobrecarga de Métodos

```
class Calculadora {
    // Método para sumar dos números enteros
    public int sumar(int a, int b) {
        return a + b;
    }

    // Método sobrecargado para sumar tres números enteros
    public int sumar(int a, int b, int c) {
        return a + b + c;
    }

    // Método sobrecargado para sumar números decimales
    public double sumar(double a, double b) {
        return a + b;
    }
}

public class Main {
    public static void main(String[] args) {
        Calculadora calc = new Calculadora();
        System.out.println("Suma de dos enteros: " + calc.sumar(5, 3));
        System.out.println("Suma de tres enteros: " + calc.sumar(5, 3, 2));
        System.out.println("Suma de decimales: " + calc.sumar(5.5, 2.3));
    }
}
```

Ejemplo de Sobrecarga de Constructores

```
class Persona {
    String nombre;
    int edad;
```

```
    // Constructor sin parámetros
    public Persona() {
        this.nombre = "Desconocido";
        this.edad = 0;
    }

    // Constructor con un parámetro
    public Persona(String nombre) {
        this.nombre = nombre;
        this.edad = 0;
    }

    // Constructor con dos parámetros
    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public void mostrarInformacion() {
        System.out.println("Nombre: " + nombre + ", Edad: " + edad);
    }
}

public class Main {
    public static void main(String[] args) {
        Persona p1 = new Persona();
        Persona p2 = new Persona("Carlos");
        Persona p3 = new Persona("Ana", 25);

        p1.mostrarInformacion();
        p2.mostrarInformacion();
        p3.mostrarInformacion();
    }
}
```

Ejercicio 1: Sobrecarga de Métodos para Suma

Instrucciones:

1. Crea una clase llamada `Calculadora`.
2. Agrega métodos llamados `sumar` que:
    - Sumen dos enteros.
    - Sumen dos números decimales.
    - Sumen tres enteros.
3. Prueba todos los métodos en el `main`.

Desarrollo:

```
public class Calculadora {
    public int sumar(int a, int b) {
```

```
        return a + b;
    }

    public double sumar(double a, double b) {
        return a + b;
    }

    public int sumar(int a, int b, int c) {
        return a + b + c;
    }
}

public class PruebaCalculadora {
    public static void main(String[] args) {
        Calculadora calc = new Calculadora();

        System.out.println("Suma de dos enteros: " + calc.sumar(5, 10));
        System.out.println("Suma de dos decimales: " + calc.sumar(3.5, 2.1));
        System.out.println("Suma de tres enteros: " + calc.sumar(1, 2, 3));
    }
}
```

Ejercicio 2: Constructores con y sin Parámetros

Instrucciones:

1. Crea una clase `Persona` con los atributos `nombre` y `edad`.
2. Agrega un constructor sin parámetros que inicialice los valores con "Desconocido" y `0`.
3. Agrega un constructor con parámetros para inicializar los atributos.
4. Crea objetos usando ambos constructores y muestra sus datos.

Desarrollo:

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

    public Persona() {
        this.nombre = "Desconocido";
        this.edad = 0;
    }

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

    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre + ", Edad: " + edad);
    }
```

```
}

public class PruebaPersona {
    public static void main(String[] args) {
        Persona persona1 = new Persona();
        Persona persona2 = new Persona("Carlos", 25);

        persona1.mostrarInfo();
        persona2.mostrarInfo();
    }
}
```

Ejercicio 3: Sobrecarga para Multiplicación

Instrucciones:

1. En una clase llamada `Operaciones`, crea métodos `multiplicar` que:
    - Multipliquen dos enteros.
    - Multipliquen dos números decimales.
    - Multipliquen tres números decimales.
2. Prueba cada método en el `main`.

Desarrollo:

```
public class Operaciones {
    public int multiplicar(int a, int b) {
        return a * b;
    }

    public double multiplicar(double a, double b) {
        return a * b;
    }

    public double multiplicar(double a, double b, double c) {
        return a * b * c;
    }
}

public class PruebaOperaciones {
    public static void main(String[] args) {
        Operaciones op = new Operaciones();

        System.out.println("Multiplicación de enteros: " + op.multiplicar(4, 5));
        System.out.println("Multiplicación de decimales: " + op.multiplicar(2.5,
3.1));
        System.out.println("Multiplicación de tres decimales: " +
op.multiplicar(1.2, 2.5, 3.3));
    }
}
```

Ejercicio 4: Constructor con Varios Parámetros

Instrucciones:

1. Crea una clase llamada `Coche` con los atributos:
    - `marca` (String)
    - `modelo` (String)
    - `anio` (int)
    - `color` (String)
2. Agrega un constructor con todos los parámetros.
3. Crea un objeto y muestra sus datos.

Desarrollo:

```
public class Coche {
    String marca;
    String modelo;
    int anio;
    String color;

    public Coche(String marca, String modelo, int anio, String color) {
        this.marca = marca;
        this.modelo = modelo;
        this.anio = anio;
        this.color = color;
    }

    public void mostrarDetalles() {
        System.out.println("Marca: " + marca);
        System.out.println("Modelo: " + modelo);
        System.out.println("Año: " + anio);
        System.out.println("Color: " + color);
    }
}

public class PruebaCoche {
    public static void main(String[] args) {
        Coche miCoche = new Coche("Toyota", "Corolla", 2020, "Rojo");
        miCoche.mostrarDetalles();
    }
}
```

Ejercicio 5: Sobrecarga para Cálculo de Áreas

Instrucciones:

1. Crea una clase `Geometria` que tenga métodos `calcularArea` para:

- Un cuadrado (recibe un lado).
- Un rectángulo (recibe base y altura).
- Un círculo (recibe el radio).

Desarrollo:

```
public class Geometria {
    public double calcularArea(double lado) {
        return lado * lado;
    }

    public double calcularArea(double base, double altura) {
        return base * altura;
    }

    public double calcularAreaCirculo(double radio) {
        return Math.PI * Math.pow(radio, 2);
    }
}

public class PruebaGeometria {
    public static void main(String[] args) {
        Geometria geo = new Geometria();

        System.out.println("Área del cuadrado: " + geo.calcularArea(4));
        System.out.println("Área del rectángulo: " + geo.calcularArea(5, 3));
        System.out.println("Área del círculo: " + geo.calcularAreaCirculo(2.5));
    }
}
```

Ejercicio 6: Constructores Sobrecargados para una Clase "Empleado"

Instrucciones:

1. Crea una clase `Empleado` con los atributos `nombre`, `edad` y `salario`.
2. Agrega los siguientes constructores:
   - Sin parámetros (asigna valores por defecto).
   - Con `nombre` y `edad`.
   - Con todos los atributos.
3. Crea objetos usando cada constructor y muestra sus datos.

Desarrollo:

```
public class Empleado {
    String nombre;
    int edad;
    double salario;
```

```
    public Empleado() {
        this.nombre = "Desconocido";
        this.edad = 0;
        this.salario = 0.0;
    }

    public Empleado(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
        this.salario = 0.0;
    }

    public Empleado(String nombre, int edad, double salario) {
        this.nombre = nombre;
        this.edad = edad;
        this.salario = salario;
    }

    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre + ", Edad: " + edad + ", Salario: "
+ salario);
    }
}

public class PruebaEmpleado {
    public static void main(String[] args) {
        Empleado e1 = new Empleado();
        Empleado e2 = new Empleado("Ana", 30);
        Empleado e3 = new Empleado("Luis", 45, 2500.50);

        e1.mostrarInfo();
        e2.mostrarInfo();
        e3.mostrarInfo();
    }
}
```

Ejercicio 7: Clase Vehículo

```
class Vehiculo {
    String marca;
    String modelo;
    int velocidad;

    public Vehiculo() {
        this.marca = "Desconocida";
        this.modelo = "Desconocido";
        this.velocidad = 0;
    }

    public Vehiculo(String marca, String modelo) {
```

```
        this.marca = marca;
        this.modelo = modelo;
        this.velocidad = 0;
    }

    public Vehiculo(String marca, String modelo, int velocidad) {
        this.marca = marca;
        this.modelo = modelo;
        this.velocidad = velocidad;
    }

    public void mostrarInformacion() {
        System.out.println("Marca: " + marca + ", Modelo: " + modelo + ",
Velocidad: " + velocidad + " km/h");
    }
}
```

Ejercicio 8: Clase Producto

```
class Producto {
    String nombre;
    double precio;

    public Producto() {
        this.nombre = "Producto Genérico";
        this.precio = 0.0;
    }

    public Producto(String nombre) {
        this.nombre = nombre;
        this.precio = 0.0;
    }

    public Producto(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }

    public void mostrarProducto() {
        System.out.println("Nombre: " + nombre + ", Precio: " + precio);
    }
}
```

Ejercicio 9: Clase Estudiante

Implementar una clase `Estudiante` con sobrecarga de constructores para manejar distintas inicializaciones.

```
public class Estudiante {
    String nombre;
```

```
    int edad;
    String carrera;
    int semestre;

    // Constructor básico
    public Estudiante(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    // Constructor con carrera
    public Estudiante(String nombre, int edad, String carrera) {
        this(nombre, edad); // Reutiliza el constructor básico
        this.carrera = carrera;
    }

    // Constructor completo
    public Estudiante(String nombre, int edad, String carrera, int semestre) {
        this(nombre, edad, carrera); // Reutiliza el constructor con carrera
        this.semestre = semestre;
    }

    // Método para mostrar detalles
    public void mostrarDetalles() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        System.out.println("Carrera: " + (carrera != null ? carrera : "No
especificada"));
        System.out.println("Semestre: " + (semestre != 0 ? semestre : "No
especificado"));
    }
}
```

Ejercicio 10: Clase CuentaBancaria

Crear una clase CuentaBancaria con sobrecarga de métodos para realizar depósitos y retiros con distintos parámetros.

```
public class CuentaBancaria {
    double saldo;

    // Constructor
    public CuentaBancaria(double saldoInicial) {
        this.saldo = saldoInicial;
    }

    // Método para depositar en la misma moneda
    public void depositar(double cantidad) {
        saldo += cantidad;
        System.out.println("Depósito realizado. Saldo actual: " + saldo);
    }
```

```
    // Método para depositar en otra moneda (USD a MXN)
    public void depositar(double cantidad, String moneda) {
        if (moneda.equals("USD")) {
            saldo += cantidad * 20; // Suponiendo 1 USD = 20 MXN
            System.out.println("Depósito en USD realizado. Saldo actual: " +
saldo);
        } else {
            saldo += cantidad;
            System.out.println("Depósito realizado. Saldo actual: " + saldo);
        }
    }

    // Método para retirar en la misma moneda
    public void retirar(double cantidad) {
        if (cantidad <= saldo) {
            saldo -= cantidad;
            System.out.println("Retiro realizado. Saldo actual: " + saldo);
        } else {
            System.out.println("Fondos insuficientes.");
        }
    }

    // Método para retirar en otra moneda (USD a MXN)
    public void retirar(double cantidad, String moneda) {
        if (moneda.equals("USD")) {
            double cantidadMXN = cantidad * 20;
            if (cantidadMXN <= saldo) {
                saldo -= cantidadMXN;
                System.out.println("Retiro en USD realizado. Saldo actual: " +
saldo);
            } else {
                System.out.println("Fondos insuficientes.");
            }
        } else {
            retirar(cantidad); // Reutiliza el método de retiro en la misma moneda
        }
    }
}
```

Ejercicio 11: Clase Empleado

Definir una clase Empleado con diferentes constructores que permitan inicializar el salario y puesto opcionalmente.

```
public class Empleado {
    String nombre;
    double salario;
    String puesto;

    // Constructor básico
```

```
    public Empleado(String nombre) {
        this.nombre = nombre;
    }

    // Constructor con salario
    public Empleado(String nombre, double salario) {
        this(nombre); // Reutiliza el constructor básico
        this.salario = salario;
    }

    // Constructor completo
    public Empleado(String nombre, double salario, String puesto) {
        this(nombre, salario); // Reutiliza el constructor con salario
        this.puesto = puesto;
    }

    // Método para mostrar detalles
    public void mostrarDetalles() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Salario: " + (salario != 0 ? salario : "No
especificado"));
        System.out.println("Puesto: " + (puesto != null ? puesto : "No
especificado"));
    }
}
```

Ejercicio 12: Clase Restaurante

Crear una clase Restaurante con sobrecarga de métodos para manejar reservas con diferente cantidad de parámetros.

```
import java.util.ArrayList;
import java.util.List;

public class Restaurante {
    String nombre;
    List<String> reservas;

    // Constructor
    public Restaurante(String nombre) {
        this.nombre = nombre;
        this.reservas = new ArrayList<>();
    }

    // Método para agregar reserva solo con el nombre del cliente
    public void agregarReserva(String nombreCliente) {
        reservas.add("Cliente: " + nombreCliente);
    }

    // Método para agregar reserva con nombre y hora
    public void agregarReserva(String nombreCliente, String hora) {
```

```
        reservas.add("Cliente: " + nombreCliente + ", Hora: " + hora);
    }

    // Método para agregar reserva con nombre, hora y número de personas
    public void agregarReserva(String nombreCliente, String hora, int numPersonas)
{
        reservas.add("Cliente: " + nombreCliente + ", Hora: " + hora + ",
Personas: " + numPersonas);
    }

    // Método para mostrar todas las reservas
    public void mostrarReservas() {
        System.out.println("Reservas en " + nombre + ":");
        for (String reserva : reservas) {
            System.out.println(reserva);
        }
    }
}
```

Ejercicio 13: Clase Computadora

Implementar una clase Computadora con distintos constructores para definir su marca, procesador y RAM.

```
public class Computadora {
    String marca;
    String procesador;
    int ram;

    // Constructor básico
    public Computadora(String marca) {
        this.marca = marca;
    }

    // Constructor con marca y procesador
    public Computadora(String marca, String procesador) {
        this(marca); // Reutiliza el constructor básico
        this.procesador = procesador;
    }

    // Constructor completo
    public Computadora(String marca, String procesador, int ram) {
        this(marca, procesador); // Reutiliza el constructor con marca y
procesador
        this.ram = ram;
    }

    // Método para mostrar detalles
    public void mostrarDetalles() {
        System.out.println("Marca: " + marca);
        System.out.println("Procesador: " + (procesador != null ? procesador : "No
especificado"));
```

```
        System.out.println("RAM: " + (ram != 0 ? ram + " GB" : "No
especificada"));
    }
}
```