Implementación de Interfaces en Java

Ejercicio 1: Electrodomésticos Inteligentes

```
interface Encendible {
    void encender();
    void apagar();
}

class Televisor implements Encendible {
    @Override
    public void encender() {
        System.out.println("El televisor está encendido.");
    }

    @Override
    public void apagar() {
        System.out.println("El televisor está apagado.");
    }
}

class Refrigerador implements Encendible {
    @Override
    public void encender() {
        System.out.println("El refrigerador está funcionando.");
    }

    @Override
    public void apagar() {
        System.out.println("El refrigerador está apagado.");
    }
}

public class Main {
    public static void main(String[] args) {
        Encendible tv = new Televisor();
        Encendible refri = new Refrigerador();

        tv.encender();
        refri.encender();
        tv.apagar();
        refri.apagar();
    }
}
```

Ejercicio 2: Vehículos y Tipos de Combustible

```java
interface Vehiculo {
    void mover();
}

interface Combustible {
    String tipoCombustible();
}

class Auto implements Vehiculo, Combustible {
    @Override
    public void mover() {
        System.out.println("El auto está en movimiento.");
    }

    @Override
    public String tipoCombustible() {
        return "Gasolina";
    }
}

class Moto implements Vehiculo, Combustible {
    @Override
    public void mover() {
        System.out.println("La moto está en movimiento.");
    }

    @Override
    public String tipoCombustible() {
        return "Diésel";
    }
}

public class Main {
    public static void main(String[] args) {
        Auto auto = new Auto();
        Moto moto = new Moto();

        auto.mover();
        System.out.println("Combustible: " + auto.tipoCombustible());

        moto.mover();
        System.out.println("Combustible: " + moto.tipoCombustible());
    }
}
```

Ejercicio 3: Sistema de Pagos

```java
interface Pago {
    void realizarPago(double monto);
}
```

```java
class PagoEfectivo implements Pago {
    @Override
    public void realizarPago(double monto) {
        System.out.println("Pago en efectivo realizado: S/ " + monto);
    }
}

class PagoTarjeta implements Pago {
    @Override
    public void realizarPago(double monto) {
        System.out.println("Pago con tarjeta realizado: S/ " + monto);
    }
}

class PagoPayPal implements Pago {
    @Override
    public void realizarPago(double monto) {
        System.out.println("Pago con PayPal realizado: S/ " + monto);
    }
}

public class Main {
    public static void main(String[] args) {
        Pago pago1 = new PagoEfectivo();
        Pago pago2 = new PagoTarjeta();
        Pago pago3 = new PagoPayPal();

        pago1.realizarPago(100);
        pago2.realizarPago(200);
        pago3.realizarPago(300);
    }
}
```

Ejercicio 4: Dispositivos de Audio

```java
interface Reproductor {
    void play();
    void pause();
    void stop();
}

class Radio implements Reproductor {
    @Override
    public void play() {
        System.out.println("La radio está reproduciendo.");
    }

    @Override
    public void pause() {
        System.out.println("La radio está en pausa.");
```

```
    }

    @Override
    public void stop() {
        System.out.println("La radio se ha detenido.");
    }
}

class MP3 implements Reproductor {
    @Override
    public void play() {
        System.out.println("El MP3 está reproduciendo.");
    }

    @Override
    public void pause() {
        System.out.println("El MP3 está en pausa.");
    }

    @Override
    public void stop() {
        System.out.println("El MP3 se ha detenido.");
    }
}

public class Main {
    public static void main(String[] args) {
        Reproductor radio = new Radio();
        Reproductor mp3 = new MP3();

        radio.play();
        mp3.pause();
    }
}
```

Ejercicio 5: Usuarios en una Aplicación

```
interface Usuario {
    void iniciarSesion();
    void cerrarSesion();
}

class Admin implements Usuario {
    @Override
    public void iniciarSesion() {
        System.out.println("El administrador ha iniciado sesión.");
    }

    @Override
    public void cerrarSesion() {
        System.out.println("El administrador ha cerrado sesión.");
```

```
    }
}

class Cliente implements Usuario {
    @Override
    public void iniciarSesion() {
        System.out.println("El cliente ha iniciado sesión.");
    }

    @Override
    public void cerrarSesion() {
        System.out.println("El cliente ha cerrado sesión.");
    }
}

public class Main {
    public static void main(String[] args) {
        Usuario admin = new Admin();
        Usuario cliente = new Cliente();

        admin.iniciarSesion();
        cliente.iniciarSesion();
        cliente.cerrarSesion();
    }
}
```

Ejercicio 6: Sistema de Notificaciones

```
interface Notificacion {
    void enviarMensaje(String mensaje);
}

class Correo implements Notificacion {
    @Override
    public void enviarMensaje(String mensaje) {
        System.out.println("Correo enviado: " + mensaje);
    }
}

class SMS implements Notificacion {
    @Override
    public void enviarMensaje(String mensaje) {
        System.out.println("SMS enviado: " + mensaje);
    }
}

class WhatsApp implements Notificacion {
    @Override
    public void enviarMensaje(String mensaje) {
        System.out.println("WhatsApp enviado: " + mensaje);
    }
```

```
}

public class Main {
    public static void main(String[] args) {
        Notificacion correo = new Correo();
        Notificacion sms = new SMS();
        Notificacion whatsapp = new WhatsApp();

        correo.enviarMensaje("Hola desde el correo!");
        sms.enviarMensaje("Hola desde SMS!");
        whatsapp.enviarMensaje("Hola desde WhatsApp!");
    }
}
```

Ejercicio 7: Animales y Sonidos

```
interface Animal {
    void hacerSonido();
}

class Perro implements Animal {
    @Override
    public void hacerSonido() {
        System.out.println("El perro ladra: ¡Guau guau!");
    }
}

class Gato implements Animal {
    @Override
    public void hacerSonido() {
        System.out.println("El gato maulla: ¡Miau miau!");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal perro = new Perro();
        Animal gato = new Gato();

        perro.hacerSonido();
        gato.hacerSonido();
    }
}
```

Ejercicio 8: Figuras Geométricas y Área

```
interface Figura {
    double calcularArea();
}
```

```
class Circulo implements Figura {
    private double radio;

    public Circulo(double radio) {
        this.radio = radio;
    }

    @Override
    public double calcularArea() {
        return Math.PI * radio * radio;
    }
}

class Rectangulo implements Figura {
    private double base, altura;

    public Rectangulo(double base, double altura) {
        this.base = base;
        this.altura = altura;
    }

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

public class Main {
    public static void main(String[] args) {
        Figura circulo = new Circulo(5);
        Figura rectangulo = new Rectangulo(4, 6);

        System.out.println("Área del círculo: " + circulo.calcularArea());
        System.out.println("Área del rectángulo: " +
rectangulo.calcularArea());
    }
}
```

Ejercicio 9: Base de Datos y Operaciones CRUD

```
interface BaseDatos {
    void insertar(String dato);
    void actualizar(String dato);
    void eliminar(String dato);
}

class MySQL implements BaseDatos {
    @Override
    public void insertar(String dato) {
        System.out.println("Dato insertado en MySQL: " + dato);
```

```
    }

    @Override
    public void actualizar(String dato) {
        System.out.println("Dato actualizado en MySQL: " + dato);
    }

    @Override
    public void eliminar(String dato) {
        System.out.println("Dato eliminado de MySQL: " + dato);
    }
}

class PostgreSQL implements BaseDatos {
    @Override
    public void insertar(String dato) {
        System.out.println("Dato insertado en PostgreSQL: " + dato);
    }

    @Override
    public void actualizar(String dato) {
        System.out.println("Dato actualizado en PostgreSQL: " + dato);
    }

    @Override
    public void eliminar(String dato) {
        System.out.println("Dato eliminado de PostgreSQL: " + dato);
    }
}

public class Main {
    public static void main(String[] args) {
        BaseDatos mysql = new MySQL();
        BaseDatos postgresql = new PostgreSQL();

        mysql.insertar("Usuario1");
        postgresql.actualizar("Usuario2");
    }
}
```

Ejercicio 10: Sensores y Mediciones

```
interface Sensor {
    double medir();
}

class SensorTemperatura implements Sensor {
    @Override
    public double medir() {
        return 25.5; // Simulación de temperatura
    }
```

```
}

class SensorHumedad implements Sensor {
    @Override
    public double medir() {
        return 60.0; // Simulación de humedad
    }
}

public class Main {
    public static void main(String[] args) {
        Sensor temperatura = new SensorTemperatura();
        Sensor humedad = new SensorHumedad();

        System.out.println("Temperatura medida: " + temperatura.medir() +
"°C");
        System.out.println("Humedad medida: " + humedad.medir() + "%");
    }
}
```