DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

PROGRAMACION ORIENTADA A OBJETOS

CONTROL DE LECTURA N° 1

Control de Lectura: Polimorfismo y Clases Abstractas

REALIZADO POR:

Pablo Guber Camacho Bravo

Estalyn Daniel Licuy Mecías

Nayeli Scarleth Loachamin Tipan

Deisy Abigail Quillupangui Tupe

DOCENTE:

Luis Enrique Jaramillo Montaño

Sangolquí – Ecuador

Polimorfismo y Clases Abstractas

1. Introducción

En el desarrollo de software orientado a objetos, el polimorfismo y las clases abstractas son conceptos fundamentales que permiten diseñar sistemas flexibles, reutilizables y escalables. El polimorfismo permite que diferentes clases respondan de manera distinta a un mismo método, mientras que las clases abstractas proporcionan una estructura base para la creación de nuevas clases. Comprender estos conceptos es esencial para aplicar buenas prácticas en la programación y mejorar la mantenibilidad del código.

2. Objetivos

Objetivo General

Aplicar los conceptos de polimorfismo y clases abstractas en el diseño de sistemas orientados a objetos.

Objetivos Específicos

- Estudiar el concepto de polimorfismo y su importancia en la programación orientada a objetos.
- Analizar el papel de las clases abstractas en la definición de estructuras comunes para múltiples clases.
- Identificar ejemplos de uso de polimorfismo y clases abstractas en la implementación de software.
- Sintetizar la información en un resumen sobre la aplicación de estos conceptos en el diseño de sistemas.

3. Marco Teórico

Polimorfismo

El polimorfismo es un principio fundamental de la POO que permite que un mismo método o función pueda tener diferentes comportamientos según el contexto en el que se utilice. Existen dos tipos principales de polimorfismo:

- **Polimorfismo en tiempo de compilación (Sobrecarga de Métodos):** Permite definir múltiples métodos con el mismo nombre, pero diferentes listas de parámetros.
- **Polimorfismo en tiempo de ejecución (Sobreescritura de Métodos):** Permite que una subclase redefina el comportamiento de un método heredado de una superclase.

Clases Abstractas

Las clases abstractas son aquellas que no pueden ser instanciadas directamente y sirven como modelos para otras clases. Estas clases pueden contener métodos abstractos (sin implementación) que deben ser definidos en las subclases.

- En **Java**, una clase abstracta se declara con la palabra clave abstract.

3.1 Análisis del proceso

Para entender la aplicación del polimorfismo y las clases abstractas, analizamos su uso en diferentes escenarios de software:

- Sistema de Notificaciones: Un sistema donde diferentes tipos de notificaciones (remitente, destinatario, remitente) comparten una estructura común.
- Sistema de Pagos: Métodos de pago como tarjeta de crédito, PayPal y transferencia bancaria implementan una misma interfaz.
- Electrodomésticos: Productos como lavadoras, refrigeradores y hornos que comparten atributos generales.
- Instrumentos Musicales: Clases como guitarra, piano y batería implementan una interfaz InstrumentoMusical.
- Productos de Software: Aplicaciones móviles, software de escritorio y aplicaciones web con una misma estructura base.
- Figuras Geométricas: Clases como cuadrado, círculo y triángulo con un método calcularArea().
- Animales: Clases como perro, gato y pájaro que heredan de una clase abstracta Animal.
- Empleados: Tipos de empleados como gerente, programador y diseñador comparten un método calcularSalario().
- Vehículos: Subclases como auto, motocicleta y camión heredan de Vehiculo.

3.2 Análisis de requisitos

- **Requisitos funcionales:**
  - El sistema debe permitir la creación de diferentes tipos de objetos según la categoría.
  - Cada entidad debe poder ejecutar su método correspondiente utilizando polimorfismo.
  - La clase base debe ser abstracta y contener métodos definidos para ser implementados por las subclases.
- **Requisitos no funcionales:**
  - El sistema debe ser flexible y escalable, permitiendo agregar nuevas entidades sin modificar el código existente.

- Debe utilizar un lenguaje de programación orientado a objetos compatible con polimorfismo y clases abstractas.

4. Resultados

Se implementó un sistema en Java para demostrar el uso de polimorfismo y clases abstractas en diferentes contextos. Ejemplo de implementación:

5. Recomendaciones

Para mejorar la comprensión y aplicación de polimorfismo y clases abstractas, se recomienda:

- Implementar ejemplos prácticos en un lenguaje orientado a objetos como Java, Python.
- Desarrollar pequeños proyectos que utilicen clases abstractas y polimorfismo para resolver problemas reales.
- Revisar documentación oficial y libros especializados sobre patrones de diseño que aprovechen estos conceptos.
- Utilizar principios SOLID, especialmente el Principio de Sustitución de Liskov, que se relaciona con el polimorfismo.

6. Conclusiones

- Aplicar estos conceptos mejora el mantenimiento del software, ya que permite cambios y mejoras sin afectar otras partes del sistema, reduciendo la probabilidad de errores.
- Implementar polimorfismo y clases abstractas en el desarrollo de software orientado a objetos permite construir sistemas más modulares y organizados, lo que facilita la colaboración entre desarrolladores y el trabajo en equipo.
- El uso de polimorfismo y clases abstractas facilita la escalabilidad del software, ya que permite agregar nuevas funcionalidades sin alterar el código existente, siguiendo principios de diseño limpio.
- El polimorfismo está directamente relacionado con el Principio de Sustitución de Liskov, lo que refuerza la importancia de diseñar clases que puedan ser intercambiables sin alterar la funcionalidad del programa.

7. Evidencias

1. Sistema Notificaciones

```java
...3 lines

package com.mycompany.sistemanotificaciones;

abstract class Notificacion {
    protected String remitente;
    protected String destinatario;
    protected String mensaje;

    public Notificacion(String remitente, String destinatario, String mensaje) {
        this.remitente = remitente;
        this.destinatario = destinatario;
        this.mensaje = mensaje;
    }

    public abstract boolean enviar();
    public abstract void registrarEnvio();

    public void mostrarInfo() {
        System.out.println("De: " + remitente);
        System.out.println("Para: " + destinatario);
        System.out.println("Mensaje: " + mensaje);
    }
}

class NotificacionEmail extends Notificacion {
    private String asunto;

    public NotificacionEmail(String remitente, String destinatario, String mensaje, String asunto) {
        super(remitente, destinatario, mensaje);
        this.asunto = asunto;
    }

    @Override
    public boolean enviar() {
        System.out.println("Enviando email...");
        System.out.println("Asunto: " + asunto);
        mostrarInfo();
        return true;
    }

    @Override
    public void registrarEnvio() {
        System.out.println("Registrando envio de email en el sistema");
        System.out.println("Fecha: " + java.time.LocalDateTime.now());
    }
}

class NotificacionSMS extends Notificacion {
    private String operador;

    public NotificacionSMS(String remitente, String destinatario, String mensaje, String operador) {
        super(remitente, destinatario, mensaje);
        this.operador = operador;
    }

    @Override
    public boolean enviar() {
        System.out.println("Enviando SMS a traves de " + operador + "...");
        mostrarInfo();
        return true;
    }

    @Override
    public void registrarEnvio() {
        System.out.println("Registrando envio de SMS en el sistema");
        System.out.println("Fecha: " + java.time.LocalDateTime.now());
    }
}

public class SistemaNotificaciones {

    public static void main(String[] args) {
        Notificacion[] notificaciones = new Notificacion[2];
        notificaciones[0] = new NotificacionEmail("info@empresa.com", "cliente@gmail.com",
                                                  "Su pedido ha sido confirmado", "Confirmacion de pedido");
        notificaciones[1] = new NotificacionSMS("+123456789", "+987654321",
                                                "Su codigo de verificacion es: 215783", "Movistar");

        for (Notificacion notificacion : notificaciones) {
            boolean exitoso = notificacion.enviar();

            if (exitoso) {
                notificacion.registrarEnvio();
            } else {
                System.out.println("Error al enviar la notificacion");
            }
            System.out.println("-----------------------");
        }
    }
}
```

```
Output - Run (SistemaNotificaciones)
--- resources:3.3.1:resources (default-resources) @ SistemaNotificaciones ---
skip non existing resourceDirectory C:\Users\MSI\OneDrive\Documentos\NetBeansProjects\SistemaNotificaciones\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ SistemaNotificaciones ---
Nothing to compile - all classes are up to date.

--- exec:3.1.0:exec (default-cli) @ SistemaNotificaciones ---
Enviando email...
Asunto: Confirmacion de pedido
De: info@empresa.com
Para: cliente@gmail.com
Mensaje: Su pedido ha sido confirmado
Registrando envio de email en el sistema
Fecha: 2025-02-16T12:09:11.356810
-----------------------
Enviando SMS a traves de Movistar...
De: +123456789
Para: +987654321
Mensaje: Su codigo de verificacion es: 215783
Registrando envio de SMS en el sistema
Fecha: 2025-02-16T12:09:11.361808200
-----------------------
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.087 s
Finished at: 2025-02-16T12:09:11-05:00
------------------------------------------------------------------------
```

2. Sistema Pagos

```java
...3 lines

package com.mycompany.sistemapagos;

abstract class MetodoPago {
    protected String tipo;
    protected double comision;

    public MetodoPago(String tipo, double comision) {
        this.tipo = tipo;
        this.comision = comision;
    }

    public abstract boolean procesarPago(double monto);
    public abstract void emitirComprobante(String idTransaccion);

    public double calcularComision(double monto) {
        return monto * (comision / 100);
    }

    public void mostrarInfo() {
        System.out.println("Metodo de pago: " + tipo);
        System.out.println("Comision: " + comision + "%");
    }
}

class TarjetaCredito extends MetodoPago {
    private String numero;
    private String banco;

    public TarjetaCredito(String numero, String banco, double comision) {
        super("Tarjeta de Credito", comision);
        this.numero = numero;
        this.banco = banco;
    }
```

```java
    @Override
    public boolean procesarPago(double monto) {
        System.out.println("Procesando pago con tarjeta de credito...");
        System.out.println("Numero: " + numero.substring(0, 4) + "****" + numero.substring(12));
        System.out.println("Banco: " + banco);
        System.out.println("Monto: $" + monto);
        double comisionTotal = calcularComision(monto);
        System.out.println("Comision: $" + comisionTotal);
        return true;
    }

    @Override
    public void emitirComprobante(String idTransaccion) {
        System.out.println("Emitiendo comprobante para transaccion: " + idTransaccion);
        System.out.println("Metodo: Tarjeta de Credito");
        System.out.println("Banco: " + banco);
    }
}

class PayPal extends MetodoPago {
    private String email;

    public PayPal(String email, double comision) {
        super("PayPal", comision);
        this.email = email;
    }

    @Override
    public boolean procesarPago(double monto) {
        System.out.println("Procesando pago con PayPal...");
        System.out.println("Email: " + email);
        System.out.println("Monto: $" + monto);
        double comisionTotal = calcularComision(monto);
        System.out.println("Comision: $" + comisionTotal);
        return true;
```

```java
    }

    @Override
    public void emitirComprobante(String idTransaccion) {
        System.out.println("Emitiendo comprobante para transaccion: " + idTransaccion);
        System.out.println("Metodo: PayPal");
        System.out.println("Email: " + email);
    }
}

public class SistemaPagos {

    public static void main(String[] args) {
        MetodoPago[] metodosPago = new MetodoPago[2];
        metodosPago[0] = new TarjetaCredito("2537564897103486", "Pichincha", 3.5);
        metodosPago[1] = new PayPal("usuario@example.com", 2.9);

        double montoPago = 1000.0;
        String idTransaccion = "TX" + System.currentTimeMillis();

        for (MetodoPago metodo : metodosPago) {
            metodo.mostrarInfo();
            boolean exitoso = metodo.procesarPago(montoPago);

            if (exitoso) {
                metodo.emitirComprobante(idTransaccion);
            } else {
                System.out.println("Error en el pago");
            }
            System.out.println("------------------------");
        }
    }
}
```

```
Output - Run (SistemaPagos)
--- exec:3.1.0:exec (default-cli) @ SistemaPagos ---
Metodo de pago: Tarjeta de Credito
Comision: 3.5%
Procesando pago con tarjeta de credito...
Numero: 2537****3486
Banco: Pichincha
Monto: $1000.0
Comision: $35.0
Emitiendo comprobante para transaccion: TX1739725871479
Metodo: Tarjeta de Credito
Banco: Pichincha
------------------------
Metodo de pago: PayPal
Comision: 2.9%
Procesando pago con PayPal...
Email: usuario@example.com
Monto: $1000.0
Comision: $28.999999999999996
Emitiendo comprobante para transaccion: TX1739725871479
Metodo: PayPal
Email: usuario@example.com
------------------------
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.036 s
Finished at: 2025-02-16T12:11:11-05:00
------------------------------------------------------------------------
```

3. Electrodomésticos

```java
...3 lines

package com.mycompany.electrodomesticos;

abstract class Electrodomestico {
    protected String marca;
    protected double precio;
    protected double consumoEnergetico;

    public Electrodomestico(String marca, double precio, double consumoEnergetico) {
        this.marca = marca;
        this.precio = precio;
        this.consumoEnergetico = consumoEnergetico;
    }

    public abstract void encender();
    public abstract void apagar();

    public void mostrarInfo() {
        System.out.println("Marca: " + marca);
        System.out.println("Precio: $" + precio);
        System.out.println("Consumo energetico: " + consumoEnergetico + " kWh");
    }
}

class Refrigerador extends Electrodomestico {
    private int temperatura;

    public Refrigerador(String marca, double precio, double consumoEnergetico, int temperatura) {
        super(marca, precio, consumoEnergetico);
        this.temperatura = temperatura;
    }

    @Override
    public void encender() {
```

```
        System.out.println("Encendiendo refrigerador " + marca);
        System.out.println("Ajustando temperatura a " + temperatura + "°C");
    }

    @Override
    public void apagar() {
        System.out.println("Apagando refrigerador " + marca);
    }

    public void ajustarTemperatura(int nuevaTemp) {
        this.temperatura = nuevaTemp;
        System.out.println("Temperatura ajustada a " + temperatura + "°C");
    }
}

class Lavadora extends Electrodomestico {
    private int capacidad;

    public Lavadora(String marca, double precio, double consumoEnergetico, int capacidad) {
        super(marca, precio, consumoEnergetico);
        this.capacidad = capacidad;
    }

    @Override
    public void encender() {
        System.out.println("Encendiendo lavadora " + marca);
        System.out.println("Capacidad: " + capacidad + " kg");
    }

    @Override
    public void apagar() {
        System.out.println("Apagando lavadora " + marca);
    }
```

```
    public void iniciarCiclo(String tipo) {
        System.out.println("Iniciando ciclo de lavado: " + tipo);
    }
}

public class Electrodomesticos {

    public static void main(String[] args) {
        Electrodomestico[] electrodomesticos = new Electrodomestico[2];
        electrodomesticos[0] = new Refrigerador("Samsung", 899.99, 45.5, 4);
        electrodomesticos[1] = new Lavadora("LG", 649.99, 30.2, 8);

        for (Electrodomestico electrodomestico : electrodomesticos) {
            electrodomestico.mostrarInfo();
            electrodomestico.encender();
            electrodomestico.apagar();
            System.out.println("-------------------------");
        }

        // Uso específico de cada clase
        Refrigerador refri = (Refrigerador) electrodomesticos[0];
        refri.ajustarTemperatura(2);

        Lavadora lavadora = (Lavadora) electrodomesticos[1];
        lavadora.iniciarCiclo("Ropa delicada");
    }
}
```

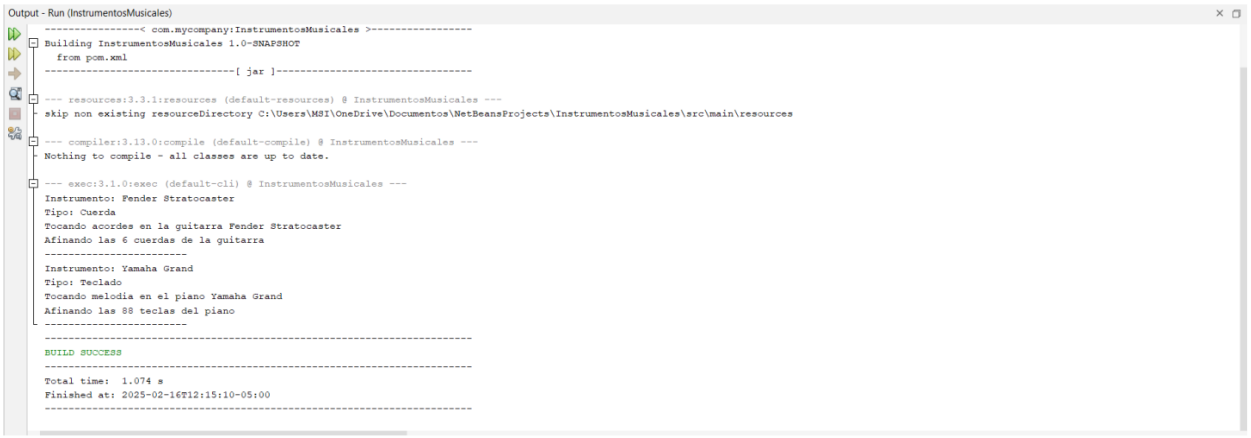
```
Output - Run (Electrodomesticos)
skip non existing resourceDirectory C:\Users\MSI\OneDrive\Documentos\NetBeansProjects\Electrodomesticos\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ Electrodomesticos ---
Nothing to compile - all classes are up to date.

--- exec:3.1.0:exec (default-cli) @ Electrodomesticos ---
Marca: Samsung
Precio: $899.99
Consumo energetico: 45.5 kWh
Encendiendo refrigerador Samsung
Ajustando temperatura a 4°C
Apagando refrigerador Samsung
-------------------------
Marca: LG
Precio: $649.99
Consumo energetico: 30.2 kWh
Encendiendo lavadora LG
Capacidad: 8 kg
Apagando lavadora LG
-------------------------
Temperatura ajustada a 2°C
Iniciando ciclo de lavado: Ropa delicada
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.998 s
Finished at: 2025-02-16T12:13:12-05:00
------------------------------------------------------------------------
```

4. Instrumentos Musicales

```
...3 lines

package com.mycompany.instrumentosmusicales;

abstract class InstrumentoMusical {
    protected String nombre;
    protected String tipo;

    public InstrumentoMusical(String nombre, String tipo) {
        this.nombre = nombre;
        this.tipo = tipo;
    }

    public abstract void tocar();
    public abstract void afinar();

    public void mostrarInfo() {
        System.out.println("Instrumento: " + nombre);
        System.out.println("Tipo: " + tipo);
    }
}

class Guitarra extends InstrumentoMusical {
    private int numCuerdas;

    public Guitarra(String nombre, int numCuerdas) {
        super(nombre, "Cuerda");
        this.numCuerdas = numCuerdas;
    }

    @Override
    public void tocar() {
        System.out.println("Tocando acordes en la guitarra " + nombre);
    }
```

```
    @Override
    public void afinar() {
        System.out.println("Afinando las " + numCuerdas + " cuerdas de la guitarra");
    }
}

class Piano extends InstrumentoMusical {
    private int numTeclas;

    public Piano(String nombre, int numTeclas) {
        super(nombre, "Teclado");
        this.numTeclas = numTeclas;
    }

    @Override
    public void tocar() {
        System.out.println("Tocando melodia en el piano " + nombre);
    }

    @Override
    public void afinar() {
        System.out.println("Afinando las " + numTeclas + " teclas del piano");
    }
}

public class InstrumentosMusicales {

    public static void main(String[] args) {
        InstrumentoMusical[] instrumentos = new InstrumentoMusical[2];
        instrumentos[0] = new Guitarra("Fender Stratocaster", 6);
        instrumentos[1] = new Piano("Yamaha Grand", 88);

        for (InstrumentoMusical instrumento : instrumentos) {
            instrumento.mostrarInfo();
            instrumento.tocar();
```

```
            instrumento.afinar();
            System.out.println("-----------------------");
        }
    }
}
```



5. Productos Software

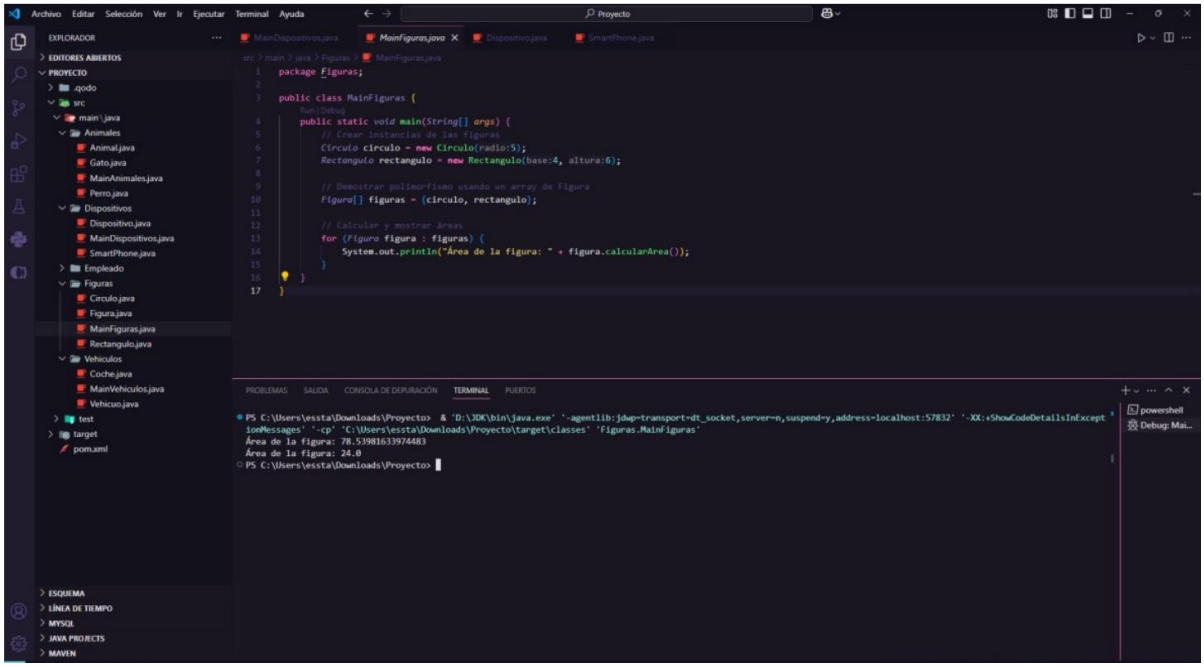
```java
...3 lines

package com.mycompany.productossoftware;

abstract class ProductoSoftware {
    protected String nombre;
    protected double precio;

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

    public abstract void ejecutar();
    public abstract void actualizar();

    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Precio: $" + precio);
    }
}

// Clases concretas que heredan de ProductoSoftware
class SistemaOperativo extends ProductoSoftware {
    private String version;

    public SistemaOperativo(String nombre, double precio, String version) {
        super(nombre, precio);
        this.version = version;
    }

    @Override
    public void ejecutar() {
        System.out.println("Iniciando sistema operativo " + nombre + " version " + version);
    }
```

```java
    @Override
    public void actualizar() {
        System.out.println("Actualizando a la nueva version de " + nombre);
    }
}

class AplicacionMovil extends ProductoSoftware {
    private String plataforma;

    public AplicacionMovil(String nombre, double precio, String plataforma) {
        super(nombre, precio);
        this.plataforma = plataforma;
    }

    @Override
    public void ejecutar() {
        System.out.println("Lanzando aplicacion " + nombre + " en " + plataforma);
    }

    @Override
    public void actualizar() {
        System.out.println("Instalando nueva version de " + nombre);
    }
}

public class ProductosSoftware {

    public static void main(String[] args) {
        // Array de objetos ProductosSoftware
        ProductoSoftware[] productos = new ProductoSoftware[2];
        productos[0] = new SistemaOperativo("Windows 11", 199.99, "22H2");
        productos[1] = new AplicacionMovil("Instagram", 0, "Android");
```

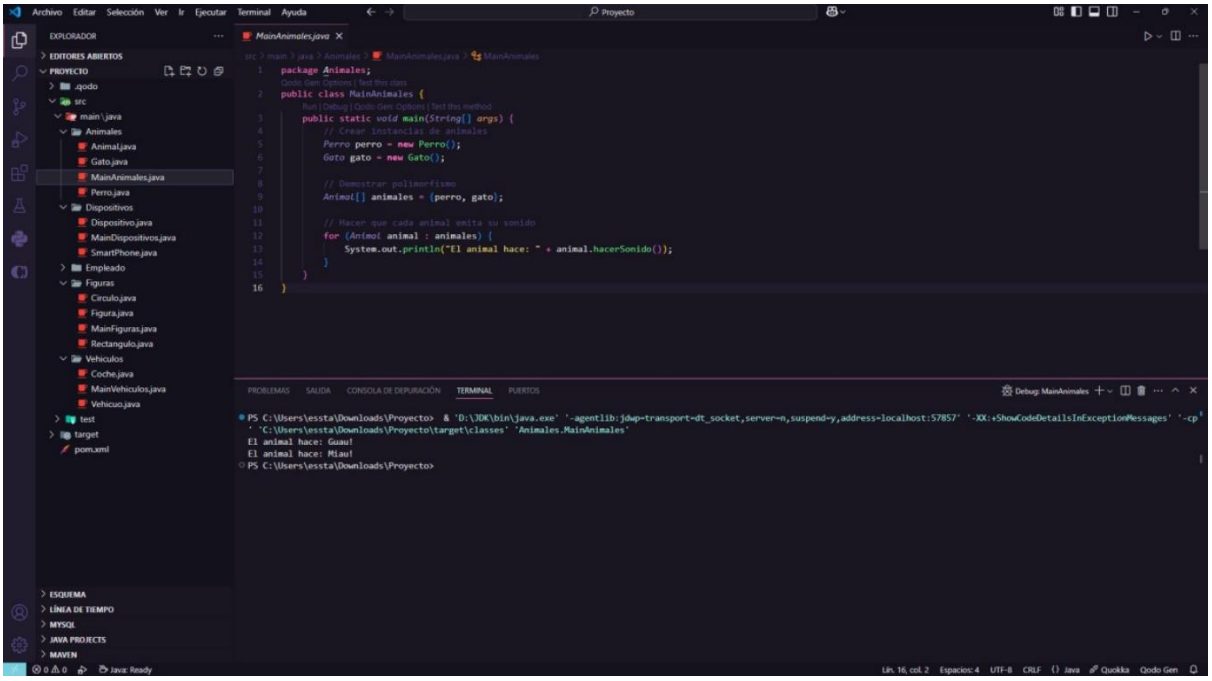
```java
        // Iteramos y llamamos a los métodos polimórficos
        for (ProductoSoftware producto : productos) {
            producto.mostrarInfo();
            producto.ejecutar();
            producto.actualizar();
            System.out.println("-------------------------");
        }
    }
}
```

```
Output - Run (ProductosSoftware)
Building ProductosSoftware 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ ProductosSoftware ---
skip non existing resourceDirectory C:\Users\MSI\OneDrive\Documentos\NetBeansProjects\ProductosSoftware\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ ProductosSoftware ---
Recompiling the module because of changed source code.
Compiling 1 source file with javac [debug release 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ ProductosSoftware ---
Nombre: Windows 11
Precio: $199.99
Iniciando sistema operativo Windows 11 version 22H2
Actualizando a la nueva version de Windows 11
-------------------------
Nombre: Instagram
Precio: $0.0
Lanzando aplicacion Instagram en Android
Instalando nueva version de Instagram
-------------------------
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.600 s
Finished at: 2025-02-16T12:18:58-05:00
------------------------------------------------------------------------
```

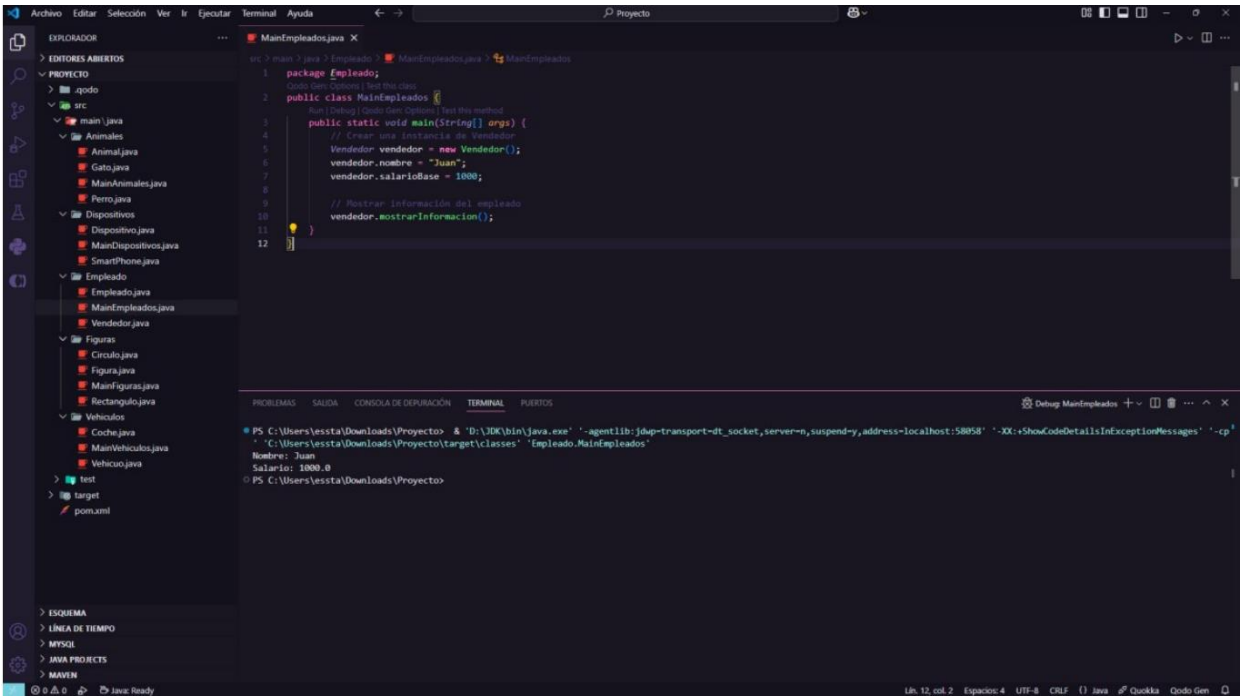
6. Figura.java



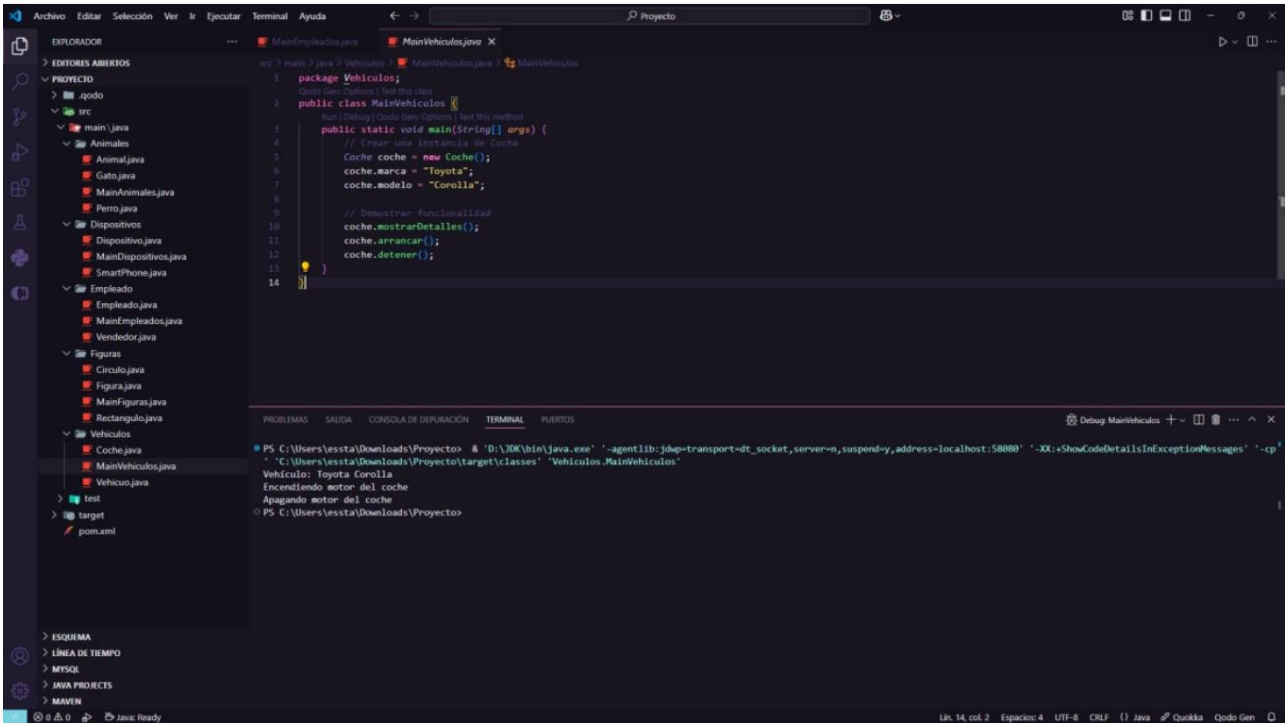
7. Aniamles



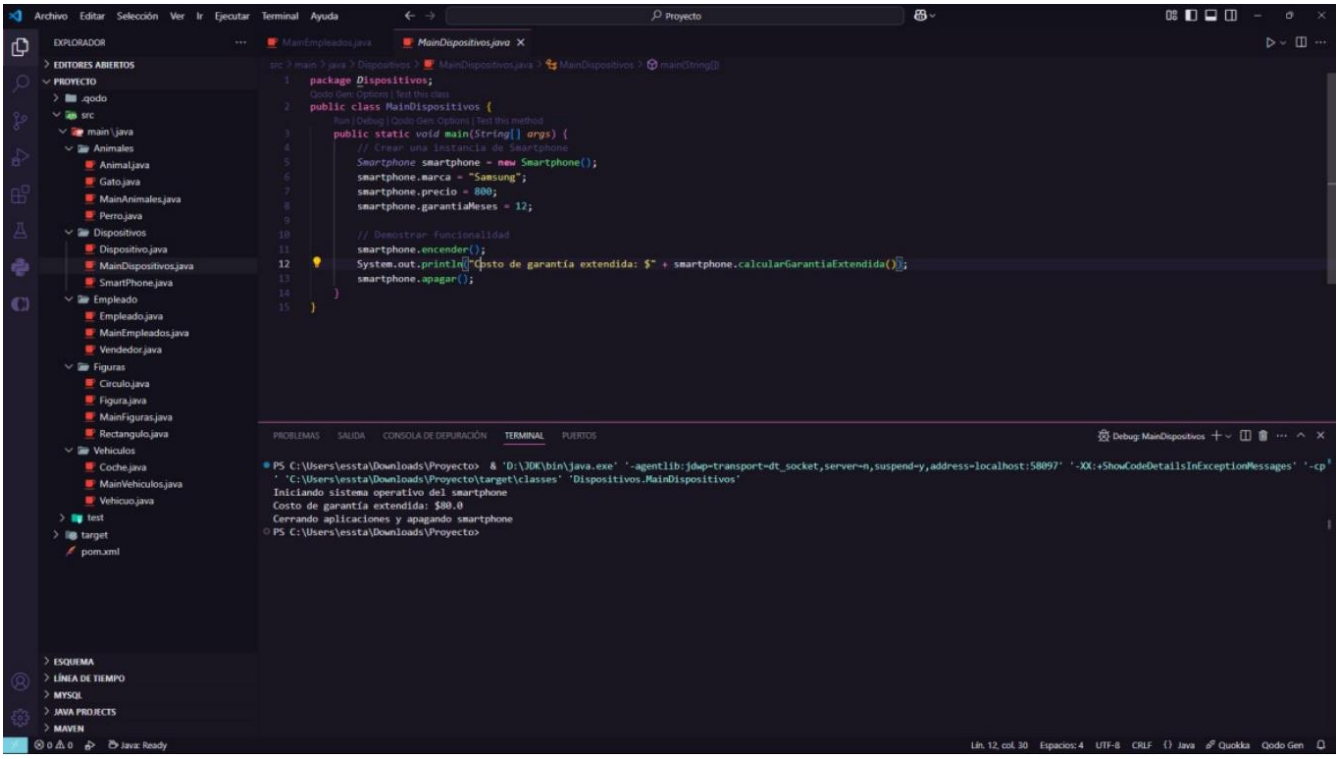
8. Empleados



9. Vehiculos



10. Dispositivos


Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
Proyecto
EXPLORADOR
EDITORES ABIERTOS
PROYECTO
.qodo
src
main\.java
Animales
Animal.java
Gato.java
MainAnimales.java
Perro.java
Dispositivos
Dispositivo.java
MainDispositivos.java
SmartPhone.java
Empleado
Empleado.java
MainEmpleados.java
Vendedor.java
Figuras
Circulo.java
Figura.java
MainFiguras.java
Rectangulo.java
Vehiculos
Coche.java
MainVehiculos.java
Vehiculo.java
test
target
pom.xml
MainEmpleados.java
MainDispositivos.java
src > main > java > Dispositivos > MainDispositivos.java > MainDispositivos > main(String[])
1 package Dispositivos;
Qodo Gen: Options | Test this class
2 public class MainDispositivos {
Run | Debug | Qodo Gen: Options | Test this method
3 public static void main(String[] args) {
4 // Crear una instancia de Smartphone
5 Smartphone smartphone = new Smartphone();
6 smartphone.marca = "Samsung";
7 smartphone.precio = 800;
8 smartphone.garantiaMeses = 12;
9
10 // Demostrar funcionalidad
11 smartphone.encender();
12 System.out.println("Costo de garantía extendida: $" + smartphone.calcularGarantiaExtendida());
13 smartphone.apagar();
14 }
15 }
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
Debug: MainDispositivos
PS C:\Users\esta\Downloads\Proyecto> & 'D:\JDK\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:58097' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
' 'C:\Users\esta\Downloads\Proyecto\target\classes' 'Dispositivos.MainDispositivos'
Iniciando sistema operativo del smartphone
Costo de garantía extendida: $80.0
Cerrando aplicaciones y apagando smartphone
PS C:\Users\esta\Downloads\Proyecto>
ESQUEMA
LÍNEA DE TIEMPO
MYSQL
JAVA PROJECTS
MAVEN
0 0
Java: Ready
Lín. 12, col. 30 Espacios: 4 UTF-8 CRLF {} Java Quokka Qodo Gen