



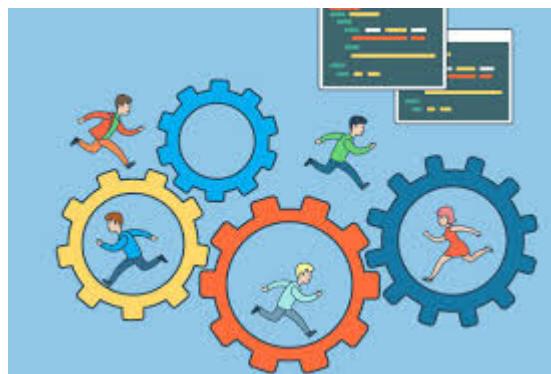
**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

**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 (email, SMS) comparten una estructura común.
- **Sistema de Pagos:** Métodos de pago como tarjeta de crédito y PayPal implementan una misma interfaz.
- **Electrodomésticos:** Productos como refrigeradores y lavadoras que comparten atributos generales.
- **Instrumentos Musicales:** Clases como cuerda y teclado implementan una interfaz `InstrumentoMusical`.
- **Productos de Software:** Software de escritorio y aplicaciones web con una misma estructura base.
- **Figuras Geométricas:** Clases como círculo y rectángulo con un método `calcularArea()`.
- **Animales:** Clases como perro y gato que heredan de una clase abstracta `Animal`.
- **Empleados:** Tipos de empleados como vendedor comparten un método `calcularSalario()`.
- **Dispositivos:** Clases como teléfono y tableta implementan una interfaz común para representar dispositivos electrónicos
- **Uehículos:** Subclases como marca y modelo heredan de `Uehiculo`.

#### **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. 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.

#### 5. 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.

## 6. Resultados

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

### I. Sistema Notificaciones

```

1  ...3 lines
2
3  package com.mycompany.sistemanotificaciones;
4
5
6  abstract class Notificacion {
7      protected String remitente;
8      protected String destinatario;
9      protected String mensaje;
10
11
12     public Notificacion(String remitente, String destinatario, String mensaje) {
13         this.remitente = remitente;
14         this.destinatario = destinatario;
15         this.mensaje = mensaje;
16     }
17
18     public abstract boolean enviar();
19     public abstract void registrarEnvio();
20
21     public void mostrarInfo() {
22         System.out.println("De: " + remitente);
23         System.out.println("Para: " + destinatario);
24         System.out.println("Mensaje: " + mensaje);
25     }
26
27
28     class NotificacionEmail extends Notificacion {
29         private String asunto;
30
31         public NotificacionEmail(String remitente, String destinatario, String mensaje, String asunto) {
32             super(remitente, destinatario, mensaje);
33             this.asunto = asunto;
34         }
35
36         @Override
37         public boolean enviar() {
38             // Implementación para enviar correo electrónico
39             return true;
40         }
41     }
42 }
```

```

38     System.out.println("Enviando email...");
39     System.out.println("Asunto: " + asunto);
40     mostrarInfo();
41     return true;
42   }
43
44   @Override
45   public void registrarEnvio() {
46     System.out.println("Registrando envio de email en el sistema");
47     System.out.println("Fecha: " + java.time.LocalDateTime.now());
48   }
49 }
50
51 class NotificacionSMS extends Notificacion {
52   private String operador;
53
54   public NotificacionSMS(String remitente, String destinatario, String mensaje, String operador) {
55     super(remitente, destinatario, mensaje);
56     this.operador = operador;
57   }
58
59   @Override
60   public boolean enviar() {
61     System.out.println("Enviando SMS a traves de " + operador + "...");
62     mostrarInfo();
63     return true;
64   }
65
66   @Override
67   public void registrarEnvio() {
68     System.out.println("Registrando envio de SMS en el sistema");
69     System.out.println("Fecha: " + java.time.LocalDateTime.now());
70   }
71 }
72

```

```

73 public class SistemaNotificaciones {
74
75   public static void main(String[] args) {
76     Notificacion[] notificaciones = new Notificacion[2];
77     notificaciones[0] = new NotificacionEmail("info@empresa.com", "cliente@gmail.com",
78                                               "Su pedido ha sido confirmado", "Confirmacion de pedido");
79     notificaciones[1] = new NotificacionSMS("+123456789", "+987654321",
80                                              "Su codigo de verificacion es: 215783", "Movistar");
81
82     for (Notificacion notificacion : notificaciones) {
83       boolean exitoso = notificacion.enviar();
84
85       if (exitoso) {
86         notificacion.registrarEnvio();
87       } else {
88         System.out.println("Error al enviar la notificacion");
89       }
90       System.out.println("-----");
91     }
92   }
93 }

```

Output - Run (SistemaNotificaciones)

```

--- resources: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 ---
Enviendo 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
-----
Enviendo 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

```
1 [ ] ... 3 lines
4
5 package com.mycompany.sistemapagos;
6
7 abstract class MetodoPago {
8     protected String tipo;
9     protected double comision;
10
11    public MetodoPago(String tipo, double comision) {
12        this.tipo = tipo;
13        this.comision = comision;
14    }
15
16    public abstract boolean procesarPago(double monto);
17    public abstract void emitirComprobante(String idTransaccion);
18
19    public double calcularComision(double monto) {
20        return monto * (comision / 100);
21    }
22
23    public void mostrarInfo() {
24        System.out.println("Metodo de pago: " + tipo);
25        System.out.println("Comision: " + comision + "%");
26    }
27}
28
29 class TarjetaCredito extends MetodoPago {
30     private String numero;
31     private String banco;
32
33    public TarjetaCredito(String numero, String banco, double comision) {
34        super("Tarjeta de Credito", comision);
35        this.numero = numero;
36        this.banco = banco;
37    }
38
```

```
38
39
40    @Override
41    public boolean procesarPago(double monto) {
42        System.out.println("Procesando pago con tarjeta de credito...");
43        System.out.println("Número: " + numero.substring(0, 4) + "****" + numero.substring(12));
44        System.out.println("Banco: " + banco);
45        System.out.println("Monto: $" + monto);
46        double comisionTotal = calcularComision(monto);
47        System.out.println("Comision: $" + comisionTotal);
48        return true;
49    }
50
51    @Override
52    public void emitirComprobante(String idTransaccion) {
53        System.out.println("Emitiendo comprobante para transaccion: " + idTransaccion);
54        System.out.println("Metodo: Tarjeta de Credito");
55        System.out.println("Banco: " + banco);
56    }
57
58 class PayPal extends MetodoPago {
59     private String email;
60
61    public PayPal(String email, double comision) {
62        super("PayPal", comision);
63        this.email = email;
64    }
65
66    @Override
67    public boolean procesarPago(double monto) {
68        System.out.println("Procesando pago con PayPal...");
69        System.out.println("Email: " + email);
70        System.out.println("Monto: $" + monto);
71        double comisionTotal = calcularComision(monto);
72        System.out.println("Comision: $" + comisionTotal);
73        return true;
74}
```

```

74     }
75
76     @Override
77     public void emitirComprobante(String idTransaccion) {
78         System.out.println("Emitiendo comprobante para transaccion: " + idTransaccion);
79         System.out.println("Metodo: PayPal");
80         System.out.println("Email: " + email);
81     }
82 }
83
84 public class SistemaPagos {
85
86     public static void main(String[] args) {
87         MetodoPago[] metodosPago = new MetodoPago[2];
88         metodosPago[0] = new TarjetaCredito("2537564897103486", "Pichincha", 3.5);
89         metodosPago[1] = new PayPal("usuario@example.com", 2.9);
90
91         double montoPago = 1000.0;
92         String idTransaccion = "TX" + System.currentTimeMillis();
93
94         for (MetodoPago metodo : metodosPago) {
95             metodo.mostrarInfo();
96             boolean exitoso = metodo.procesarPago(montoPago);
97
98             if (exitoso) {
99                 metodo.emitirComprobante(idTransaccion);
100            } else {
101                System.out.println("Error en el pago");
102            }
103            System.out.println("-----");
104        }
105    }
106 }
107

```

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...
Número: 2537564897103486
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

```

1  ...
2  ...
3  ...
4
5  package com.mycompany.electrodomesticos;
6
7  abstract class Electrodomestico {
8      protected String marca;
9      protected double precio;
10     protected double consumoEnergetico;
11
12     public Electrodomestico(String marca, double precio, double consumoEnergetico) {
13         this.marca = marca;
14         this.precio = precio;
15         this.consumoEnergetico = consumoEnergetico;
16     }
17
18     public abstract void encender();
19     public abstract void apagar();
20
21     public void mostrarInfo() {
22         System.out.println("Marca: " + marca);
23         System.out.println("Precio: $" + precio);
24         System.out.println("Consumo energetico: " + consumoEnergetico + " kWh");
25     }
26 }
27
28 class Refrigerador extends Electrodomestico {
29     private int temperatura;
30
31     public Refrigerador(String marca, double precio, double consumoEnergetico, int temperatura) {
32         super(marca, precio, consumoEnergetico);
33         this.temperatura = temperatura;
34     }
35
36     @Override
37     public void encender() {

```

```
38     System.out.println("Encendiendo refrigerador " + marca);
39     System.out.println("Ajustando temperatura a " + temperatura + "°C");
40 }
41
42 @Override
43     public void apagar() {
44         System.out.println("Apagando refrigerador " + marca);
45     }
46
47     public void ajustarTemperatura(int nuevaTemp) {
48         this.temperatura = nuevaTemp;
49         System.out.println("Temperatura ajustada a " + temperatura + "°C");
50     }
51 }
52
53 class Lavadora extends Electrodomestico {
54     private int capacidad;
55
56     public Lavadora(String marca, double precio, double consumoEnergetico, int capacidad) {
57         super(marca, precio, consumoEnergetico);
58         this.capacidad = capacidad;
59     }
60
61     @Override
62     public void encender() {
63         System.out.println("Encendiendo lavadora " + marca);
64         System.out.println("Capacidad: " + capacidad + " kg");
65     }
66
67     @Override
68     public void apagar() {
69         System.out.println("Apagando lavadora " + marca);
70     }
71 }
```

```
72     public void iniciarCiclo(String tipo) {
73         System.out.println("Iniciando ciclo de lavado: " + tipo);
74     }
75 }
76
77 public class Electrodomesticos {
78
79     public static void main(String[] args) {
80         Electrodomestico[] electrodomesticos = new Electrodomestico[2];
81         electrodomesticos[0] = new Refrigerador("Samsung", 899.99, 45.5, 4);
82         electrodomesticos[1] = new Lavadora("LG", 649.99, 30.2, 8);
83
84         for (Electrodomestico electrodomestico : electrodomesticos) {
85             electrodomestico.mostrarInfo();
86             electrodomestico.encender();
87             electrodomestico.apagar();
88             System.out.println("-----");
89         }
90
91         // Uso especifico de cada clase
92         Refrigerador refri = (Refrigerador) electrodomesticos[0];
93         refri.ajustarTemperatura(2);
94
95         Lavadora lavadora = (Lavadora) electrodomesticos[1];
96         lavadora.iniciarCiclo("Ropa delicada");
97     }
98 }
99 }
```

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: $699.99
Consumo energetico: 45.5 kWh
Encendiendo refrigerador Samsung
Ajustando temperatura a 14°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 24°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

```

1  [+] ... 3 lines
4
5 package com.mycompany.instrumentosmusicales;
6
7 ① abstract class InstrumentoMusical {
8      protected String nombre;
9      protected String tipo;
10
11     public InstrumentoMusical(String nombre, String tipo) {
12         this.nombre = nombre;
13         this.tipo = tipo;
14     }
15
16     public abstract void tocar();
17     public abstract void afinar();
18
19     public void mostrarInfo() {
20         System.out.println("Instrumento: " + nombre);
21         System.out.println("Tipo: " + tipo);
22     }
23 }
24
25 class Guitarra extends InstrumentoMusical {
26     private int numCuerdas;
27
28     public Guitarra(String nombre, int numCuerdas) {
29         super(nombre, "Cuerda");
30         this.numCuerdas = numCuerdas;
31     }
32
33     @Override
34     public void tocar() {
35         System.out.println("Tocando acordes en la guitarra " + nombre);
36     }
37 }
```

```

38     @Override
39     public void afinar() {
40         System.out.println("Afinando las " + numCuerdas + " cuerdas de la guitarra");
41     }
42 }
43
44 class Piano extends InstrumentoMusical {
45     private int numTeclas;
46
47     public Piano(String nombre, int numTeclas) {
48         super(nombre, "Teclado");
49         this.numTeclas = numTeclas;
50     }
51
52     @Override
53     public void tocar() {
54         System.out.println("Tocando melodía en el piano " + nombre);
55     }
56
57     @Override
58     public void afinar() {
59         System.out.println("Afinando las " + numTeclas + " teclas del piano");
60     }
61 }
62
63 public class InstrumentosMusicales {
64
65     public static void main(String[] args) {
66         InstrumentoMusical[] instrumentos = new InstrumentoMusical[2];
67         instrumentos[0] = new Guitarra("Fender Stratocaster", 6);
68         instrumentos[1] = new Piano("Yamaha Grand", 88);
69
70         for (InstrumentoMusical instrumento : instrumentos) {
71             instrumento.mostrarInfo();
72             instrumento.tocar();
73
74             instrumento.afinar();
75             System.out.println("-----");
76         }
77     }
78 }

```

Output - Run (InstrumentosMusicales)

```

----- com.mycompany:InstrumentosMusicales -----
Building InstrumentosMusicales 1.0-SNAPSHOT
from pom.xml
-----[ jar ]-----
----- resources:3.3.1:resources (default-resources) @ InstrumentosMusicales ---
- skip non existing resourceDirectory C:\Users\MSI\OneDrive\Documentos\NetBeansProjects\InstrumentosMusicales\src\main\resources
----- compiler:3.13.0:compile (default-compile) @ InstrumentosMusicales ---
- Nothing to compile - all classes are up to date.

----- exec:3.1.0:exec (default-cli) @ InstrumentosMusicales ---
Instrumento: Fender Stratocaster
Tipo: Cuerda
Tocando acordes en la guitarra Fender Stratocaster
Afinando las 6 cuerdas de la guitarra
-----
Instrumento: Yamaha Grand
Tipo: Teclado
Tocando melodía en el piano Yamaha Grand
Afinando las 88 teclas del piano
-----

BUILD SUCCESS
-----
Total time: 1.074 s
Finished at: 2025-02-16T12:15:10-05:00
-----
```

## 5. Productos Software

```
1  [+] ...3 lines
4
5  package com.mycompany.productossoftware;
6
7  abstract class ProductoSoftware {
8      protected String nombre;
9      protected double precio;
10 }
11 public ProductoSoftware(String nombre, double precio) {
12     this.nombre = nombre;
13     this.precio = precio;
14 }
15
16 public abstract void ejecutar();
17 public abstract void actualizar();
18
19 public void mostrarInfo() {
20     System.out.println("Nombre: " + nombre);
21     System.out.println("Precio: $" + precio);
22 }
23
24
25 // Clases concretas que heredan de ProductoSoftware
26 class SistemaOperativo extends ProductoSoftware {
27     private String version;
28
29     public SistemaOperativo(String nombre, double precio, String version) {
30         super(nombre, precio);
31         this.version = version;
32     }
33
34     @Override
35     public void ejecutar() {
36         System.out.println("Iniciando sistema operativo " + nombre + " version " + version);
37     }

```

```
38
39     @Override
40     public void actualizar() {
41         System.out.println("Actualizando a la nueva version de " + nombre);
42     }
43
44
45 class AplicacionMovil extends ProductoSoftware {
46     private String plataforma;
47
48     public AplicacionMovil(String nombre, double precio, String plataforma) {
49         super(nombre, precio);
50         this.plataforma = plataforma;
51     }
52
53     @Override
54     public void ejecutar() {
55         System.out.println("Lanzando aplicacion " + nombre + " en " + plataforma);
56     }
57
58     @Override
59     public void actualizar() {
60         System.out.println("Instalando nueva version de " + nombre);
61     }
62
63
64 public class ProductosSoftware {
65
66     public static void main(String[] args) {
67         // Array de objetos ProductosSoftware
68         ProductoSoftware[] productos = new ProductoSoftware[2];
69         productos[0] = new SistemaOperativo("Windows 11", 199.99, "22H2");
70         productos[1] = new AplicacionMovil("Instagram", 0, "Android");
71     }

```

```
72     }
73     // Iteramos y llamamos a los métodos polimórficos
74     for (ProductoSoftware producto : productos) {
75         producto.mostrarInfo();
76         producto.ejecutar();
77         producto.actualizar();
78         System.out.println("-----");
79     }
80 }
81
```

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

The screenshot shows an IDE interface with the following details:

- Project Explorer:** Shows a project structure with packages like `Animales`, `Dispositivos`, `Figuras`, `Vehiculos`, and `Empleado`.
- Code Editor:** Displays `MainFiguras.java` with Java code demonstrating polymorphism through arrays of `Figura` objects.
- Terminal:** Shows the output of a command-line run, including the area of a circle (78.53981633974483) and a rectangle (24.0).
- Sidebar:** Includes links to `ESQUEMA`, `LÍNEA DE TIEMPO`, `MYSQL`, `JAVA PROJECTS`, and `MAVEN`.

## 7. Animales

The screenshot shows an IDE interface with the following details:

- Project Explorer:** Shows a project structure with packages like `Animales`, `Dispositivos`, `Figuras`, and `Vehiculos`.
- Code Editor:** Displays `MainAnimales.java` with Java code demonstrating polymorphism through an array of `Animal` objects.
- Terminal:** Shows the output of a command-line run, including animal sounds (Gato miau, Perro ladrar).
- Sidebar:** Includes links to `ESQUEMA`, `LÍNEA DE TIEMPO`, `MYSQL`, `JAVA PROJECTS`, and `MAVEN`.

## 8. Empleados

**MainEmpleados.java**

```

1 package Empleado;
2
3 public class MainEmpleados {
4     public static void main(String[] args) {
5         // Crear una instancia de Vendedor
6         Vendedor vendedor = new Vendedor();
7         vendedor.nombre = "Juan";
8         vendedor.sueldoBase = 1000;
9
10        // Mostrar información del empleado
11        vendedor.mostrarInformacion();
12    }
13}

```

**Terminal**

```

PS C:\Users\estata\Downloads\Proyecto> & 'D:\OK\bin\java.exe' '-agentlib:jdepl-transport=dt_socket,server=n,suspend=y,address=localhost:58058' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\estata\Downloads\Proyecto\target\classes' 'Empleado.MainEmpleados'
Nombre: Juan
Sueldo: 1000.0

```

**MainVehiculos.java**

```

1 package Vehiculos;
2
3 public class MainVehiculos {
4     public static void main(String[] args) {
5         // Crear una instancia de Coche
6         Coche coche = new Coche();
7         coche.marca = "Toyota";
8         coche.modelo = "Corolla";
9
10        // Demostren funcionalidad
11        coche.mostrarDetalles();
12        coche.arrancar();
13        coche.detener();
14    }
15}

```

**Terminal**

```

PS C:\Users\estata\Downloads\Proyecto> & 'D:\OK\bin\java.exe' '-agentlib:jdepl-transport=dt_socket,server=n,suspend=y,address=localhost:58088' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\estata\Downloads\Proyecto\target\classes' 'Vehiculos.MainVehiculos'
Vehículo: Toyota Corolla
Encendiendo motor del coche
Apagando motor del coche

```

## 9. Vehículos

**MainEmpleados.java**

```

1 package Empleado;
2
3 public class MainEmpleados {
4     public static void main(String[] args) {
5         // Crear una instancia de Vendedor
6         Vendedor vendedor = new Vendedor();
7         vendedor.nombre = "Juan";
8         vendedor.sueldoBase = 1000;
9
10        // Mostrar información del empleado
11        vendedor.mostrarInformacion();
12    }
13}

```

**Terminal**

```

PS C:\Users\estata\Downloads\Proyecto> & 'D:\OK\bin\java.exe' '-agentlib:jdepl-transport=dt_socket,server=n,suspend=y,address=localhost:58058' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\estata\Downloads\Proyecto\target\classes' 'Empleado.MainEmpleados'
Nombre: Juan
Sueldo: 1000.0

```

**MainVehiculos.java**

```

1 package Vehiculos;
2
3 public class MainVehiculos {
4     public static void main(String[] args) {
5         // Crear una instancia de Coche
6         Coche coche = new Coche();
7         coche.marca = "Toyota";
8         coche.modelo = "Corolla";
9
10        // Demostren funcionalidad
11        coche.mostrarDetalles();
12        coche.arrancar();
13        coche.detener();
14    }
15}

```

**Terminal**

```

PS C:\Users\estata\Downloads\Proyecto> & 'D:\OK\bin\java.exe' '-agentlib:jdepl-transport=dt_socket,server=n,suspend=y,address=localhost:58088' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\estata\Downloads\Proyecto\target\classes' 'Vehiculos.MainVehiculos'
Vehículo: Toyota Corolla
Encendiendo motor del coche
Apagando motor del coche

```

## 10. Dispositivos

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under the 'src' folder:
  - Animales (containing Animal.java, Gato.java, MainAnimales.java)
  - Dispositivos (containing Dispositivo.java, MainDispositivo.java, Smartphone.java)
  - Empleado (containing Empleado.java, MainEmpleados.java)
  - Figuras (containing Circulo.java, Figura.java, MainFiguras.java, Rectangulo.java)
  - Vehiculos (containing Coche.java, MainVehiculos.java, Vehiculo.java)
- Code Editor:** Displays the MainDispositivos.java file with the following code:

```
package Dispositivos;
public class MainDispositivos {
    public static void main(String[] args) {
        // Crear una instancia de Smartphone
        Smartphone smartphone = new Smartphone();
        smartphone.marca = "Samsung";
        smartphone.precio = 800;
        smartphone.garantiaMeses = 12;

        // Demostrar funcionalidad
        smartphone.encender();
        System.out.println("Costo de garantía extendida: $" + smartphone.calcularGarantiaExtendida());
        smartphone.apagar();
    }
}
```
- Terminal:** Shows the command-line output of the run command:

```
PS C:\Users\estata\Downloads\Proyecto> & 'D:\JDK\bin\java.exe' '-agentlib:jdp=transport=dt_socket,server=n,suspend=y,address=localhost:58097' '-XX:ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\estata\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\estata\Downloads\Proyecto>
```
- Status Bar:** Shows the status 'Java Ready'.