



**UNIVERSIDAD TÉCNICA DEL NORTE**  
**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**  
**CARRERA DE SOFTWARE**  
**ASIGNATURA: Programación Orientada a Objetos NIVEL: 2**

Nombres completos:	Espinoza Cangás Alex Anthony
Fecha:	20/12/2024
Tema:	Herencia
Objetivo de esta actividad:	Uso óptimo de herencia en la programación orientada a objetos.

#### INDICACIONES:

Realizar una implementación de la jerarquía de clases presentada en papel. Al igual que el ejercicio revisado en clase, desarrollar una interface de usuario swing que permita instanciar las clases y presentar la información de los objetos instanciados.

#### DESARROLLO:

Capturas de las clases:

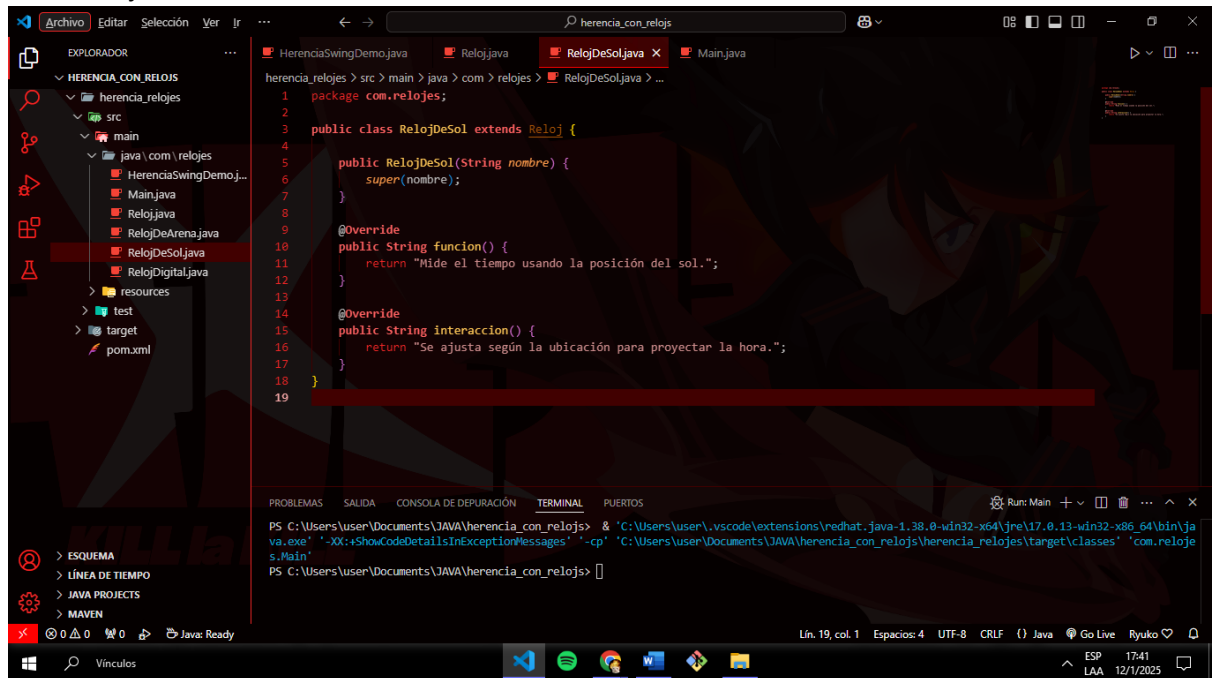
##### 1. Clase Reloj

The screenshot shows a code editor with the following Java code for the `Reloj` class:

```
1 package com.relojes;
2
3 public abstract class Reloj {
4     private String nombre;
5
6     public Reloj(String nombre) {
7         this.nombre = nombre;
8     }
9
10    public String getNombre() {
11        return nombre;
12    }
13
14    public abstract String funcion();
15    public abstract String interaccion();
16 }
17
```

The IDE interface includes a file explorer on the left showing the project structure, a terminal at the bottom with the command `PS C:\Users\user\Documents\JAVA\herencia_con_relojes> java -cp "C:\Users\user\Documents\JAVA\herencia_con_relojes\herencia_relojes\target\classes" com.relojes.Main`, and a status bar at the bottom indicating the current line and column.

## 2. Clase RelojDeSol



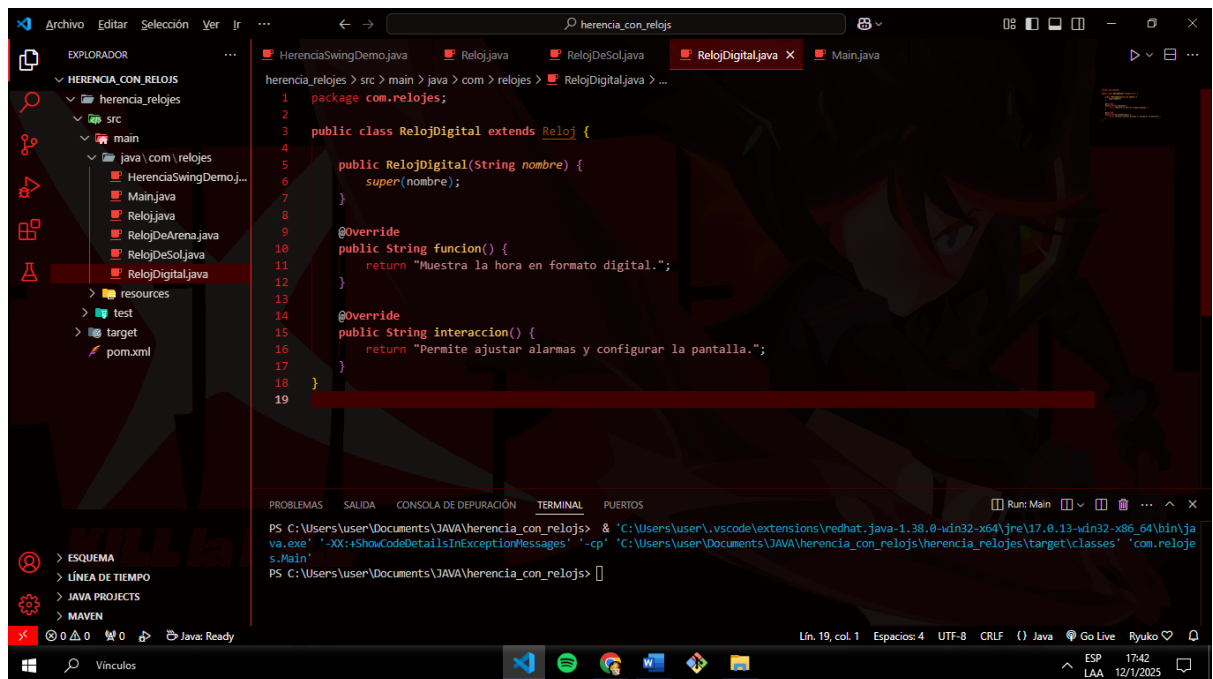
The screenshot shows the Visual Studio Code editor with the `RelojDeSol.java` file open. The file is part of the `herencia_con_relojes` project, located in the `src/main/java/com/relojes` package. The code defines a `RelojDeSol` class that extends the `Reloj` class. It includes a constructor `RelojDeSol(String nombre)` and two overridden methods: `funcion()` and `interaccion()`.

```
1 package com.relojes;
2
3 public class RelojDeSol extends Reloj {
4
5     public RelojDeSol(String nombre) {
6         super(nombre);
7     }
8
9     @Override
10    public String funcion() {
11        return "Mide el tiempo usando la posición del sol.";
12    }
13
14    @Override
15    public String interaccion() {
16        return "Se ajusta según la ubicación para proyectar la hora.";
17    }
18 }
19
```

The terminal at the bottom shows the command to run the application:

```
PS C:\Users\User\Documents\JAVA\herencia_con_relojes> java -cp 'C:\Users\User\Documents\JAVA\herencia_con_relojes\target\classes' com.relojes.Main
```

## 3. Clase RelojDigital



The screenshot shows the Visual Studio Code editor with the `RelojDigital.java` file open. The file is part of the `herencia_con_relojes` project, located in the `src/main/java/com/relojes` package. The code defines a `RelojDigital` class that extends the `Reloj` class. It includes a constructor `RelojDigital(String nombre)` and two overridden methods: `funcion()` and `interaccion()`.

```
1 package com.relojes;
2
3 public class RelojDigital extends Reloj {
4
5     public RelojDigital(String nombre) {
6         super(nombre);
7     }
8
9     @Override
10    public String funcion() {
11        return "Muestra la hora en formato digital.";
12    }
13
14    @Override
15    public String interaccion() {
16        return "Permite ajustar alarmas y configurar la pantalla.";
17    }
18 }
19
```

The terminal at the bottom shows the command to run the application:

```
PS C:\Users\User\Documents\JAVA\herencia_con_relojes> java -cp 'C:\Users\User\Documents\JAVA\herencia_con_relojes\target\classes' com.relojes.Main
```

## 4. Clase RelojDeArena

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `RelojDeArena.java` file. The code defines a class `RelojDeArena` that extends `Reloj`. It includes a constructor `RelojDeArena(String nombre)` and two overridden methods: `funcion()` which returns "Mide el tiempo con el flujo de arena." and `interaccion()` which returns "Se voltea para reiniciar el conteo de tiempo.".

```
1 package com.relojes;
2
3 public class RelojDeArena extends Reloj {
4
5     public RelojDeArena(String nombre) {
6         super(nombre);
7     }
8
9     @Override
10    public String funcion() {
11        return "Mide el tiempo con el flujo de arena.";
12    }
13
14    @Override
15    public String interaccion() {
16        return "Se voltea para reiniciar el conteo de tiempo.";
17    }
18 }
19
```

The terminal at the bottom shows the command to run the application: `PS C:\Users\user\Documents\JAVA\herencia_con_relojes> java -cp "C:\Users\user\Documents\JAVA\herencia_con_relojes\herencia_relojes\target\classes" com.relojes.Main`.

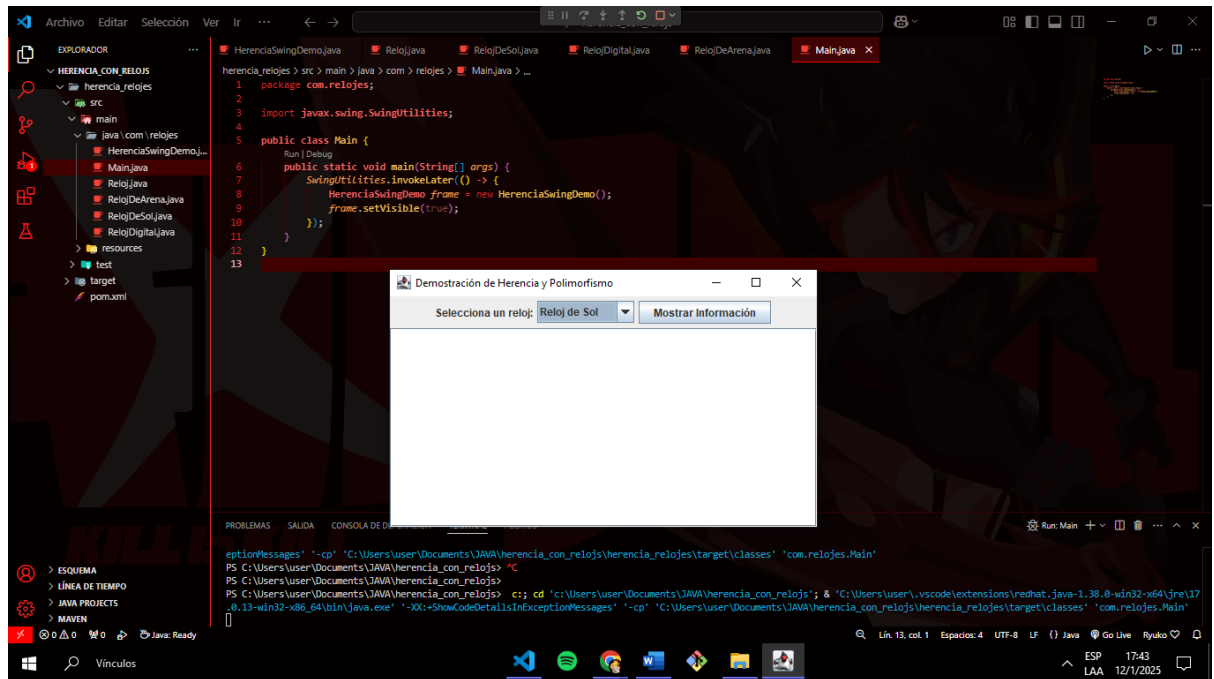
## 5. Clase HerenciaSwingDemo

The first screenshot shows the `HerenciaSwingDemo.java` file. The code defines a class `HerenciaSwingDemo` that extends `JFrame`. It includes a constructor `HerenciaSwingDemo()` and a private class `MostrarButtonListener` that implements `ActionListener`. The constructor sets the title, size, and layout of the window. It adds a `JLabel` and a `JComboBox` to the window. The `MostrarButtonListener` class has an `actionPerformed` method that handles the button click event.

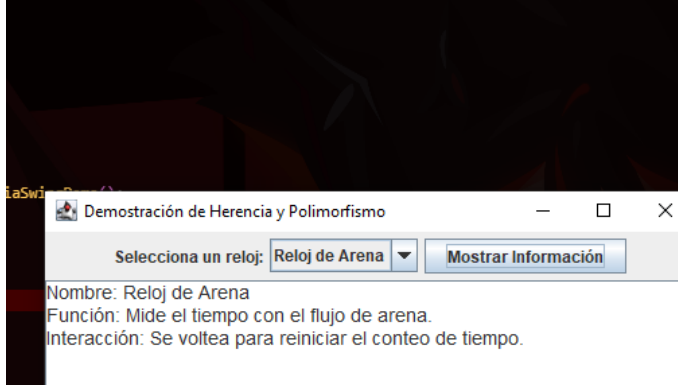
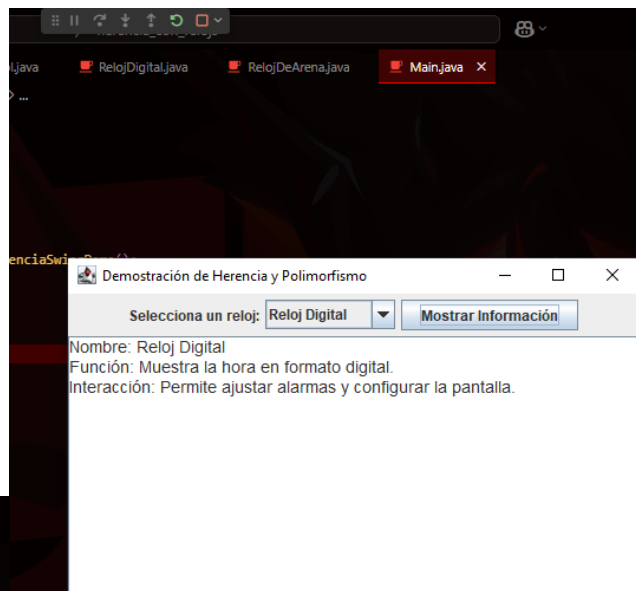
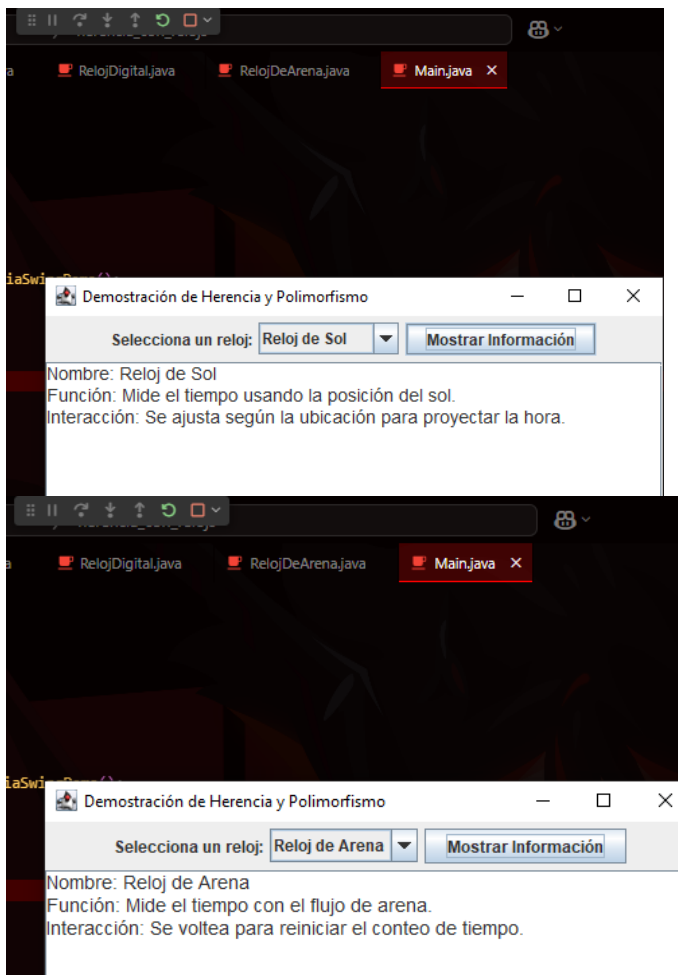
```
1 package com.relojes;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7
8 public class HerenciaSwingDemo extends JFrame {
9     private JComboBox<String> tipoRelojCombo;
10    private JTextArea outputArea;
11
12    public HerenciaSwingDemo() {
13        setTitle("Demostración de Herencia y Polimorfismo");
14        setSize(500, 300);
15        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16        setLayout(new BorderLayout());
17
18        JPanel topPanel = new JPanel();
19        topPanel.setLayout(new FlowLayout());
20
21        JLabel label = new JLabel("Selecciona un reloj:");
22        topPanel.add(label);
23
24        tipoRelojCombo = new JComboBox<>(new String[] { "Reloj de Sol", "Reloj Digital", "Reloj de Arena" });
25        topPanel.add(tipoRelojCombo);
26
27        JButton mostrarButton = new JButton("Mostrar Información");
28        mostrarButton.addActionListener(new MostrarButtonListener());
29        topPanel.add(mostrarButton);
30
31        add(topPanel, BorderLayout.NORTH);
32
33        outputArea = new JTextArea();
34        outputArea.setEditable(false);
35        outputArea.setFont(new Font("Arial", Font.PLAIN, 14));
36        add(new JScrollPane(outputArea), BorderLayout.CENTER);
37    }
38
39    private class MostrarButtonListener implements ActionListener {
40
41        @Override
42        public void actionPerformed(ActionEvent e) {
43            String seleccion = (String) tipoRelojCombo.getSelectedItem();
44            reloj seleccion;
45
46            // Polimorfismo: Crear el objeto según la selección
47            switch (seleccion) {
48                case "Reloj de Sol":
49                    reloj = new RelojDeSol(nombre: "Reloj de Sol");
50                    break;
51                case "Reloj Digital":
52                    reloj = new RelojDigital(nombre: "Reloj Digital");
53                    break;
54                case "Reloj de Arena":
55                    reloj = new RelojDeArena(nombre: "Reloj de Arena");
56                    break;
57                default:
58                    reloj = null;
59            }
60
61            if (reloj != null) {
62                outputArea.setText("");
63                outputArea.append("Nombre: " + reloj.getNombre() + "\n");
64                outputArea.append("Función: " + reloj.funcion() + "\n");
65                outputArea.append("Interacción: " + reloj.interaccion() + "\n");
66            }
67        }
68    }
69 }
```

The second screenshot shows the same file, but with the `MostrarButtonListener` class implemented. The `actionPerformed` method handles the button click event by creating a `Reloj` object based on the selected option and displaying its details in the `outputArea`.

## 6. Clase Main



- **Output:**



## CONCLUSIONES:

1. Me ayudó a entender mejor el uso de herencia, también como mediante polimorfismo hay mas flexibilidad y uso en la clase HerenciaSwingDemo
2. Saber bien la jerarquía de herencia ya que al definir la clase "reloj" como abstracta se pone características comunes a las demás clases