Explicación del Proyecto Herencia

Este proyecto en Java ilustra cómo usar la herencia y el polimorfismo para gestionar diferentes formas geométricas. La aplicación define una jerarquía de clases en la que todas las formas heredan de una clase base llamada Formas. Cada forma específica (como Cuadrado, Circulo, Linea y Triangulo) extiende esta clase base e implementa su propio comportamiento.

Estructura del Proyecto

1. Clase Principal: HerenciaClases

La clase HerenciaClases contiene el método main, que es el punto de entrada de la aplicación. Aquí se crean instancias de cada una de las formas (Circulo, Cuadrado, Linea, Triangulo) y se invoca el método imprimirInformacion de cada objeto para mostrar sus detalles, también esta clase nos recupera toda la información que se pueda heredar de cualquiera de las subclases creadas en el código.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */

package com.mycompany.herenciaclases;

import helpers.Circulo;
import helpers.Cuadrado;
import helpers.Linea;
import helpers.Triangulo;

/**
 *
 * @author Usuario
 */
public class HerenciaClases {

    public static void main(String[] args) {
        Circulo Circulo = new Circulo ();
        Cuadrado Cuadrado = new Cuadrado();
        Linea Linea = new Linea();
        Triangulo Triangulo = new Triangulo();
        System.out.println("Forma1");
        Circulo.imprimirInformacion();
        System.out.println("===============");
        System.out.println("Forma2");
        Cuadrado.imprimirInformacion();
        System.out.println("===============");
        System.out.println("Forma3");
        Linea.imprimirInformacion();
        System.out.println("===============");
        System.out.println("Forma4");
       Triangulo.imprimirInformacion();
    }
}
```

2. Clase Base: Formas

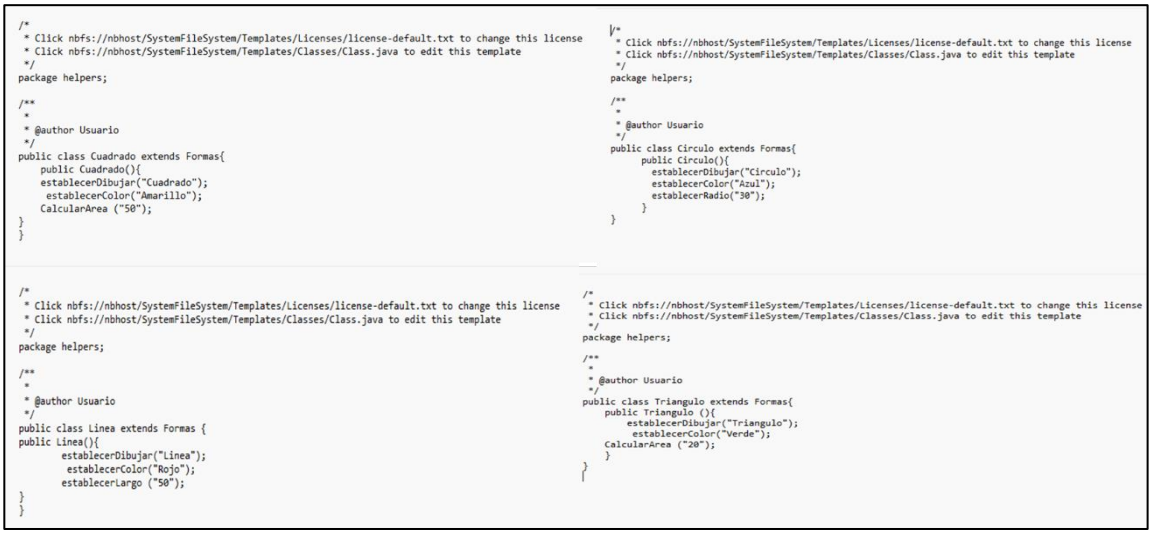
La clase Formas define las propiedades y métodos comunes a todas las formas geométricas. Las propiedades incluyen color, Dibujar, Radio, Área, y Largo. Esta clase también proporciona métodos para establecer y obtener estos valores, así como un método imprimirInformacion para mostrar los detalles de la forma.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package helpers;

/**
 *
 * @author Usuario
 */
public class Formas {
    private String color;
    private String Dibujar;
    private String Radio;
    public String Area;
    public String Largo;

    public  Formas(){

    }
    public void establecerDibujar(String Dibujar){
    this.Dibujar = Dibujar;
}
public String ObtenerDibujar(){
    return this.Dibujar = Dibujar;
}

    public void establecerColor(String color){
    this.color = color;
}
public String ObtenerColor(){
    return this.color = color;
}

    public void imprimirInformacion(){
     System.out.println("Color: " + color);
     System.out.println("Dibujar: " + Dibujar);
     System.out.println("Radio: " + Radio);
     System.out.println("Area: " + Area);
     System.out.println("Largo: " + Largo);
    }
     public void establecerRadio(String Radio){
    this.Radio = Radio;
}
    public void CalcularArea(String Area){
    this.Area = Area;
}
    public void establecerLargo(String Largo){
    this.Largo = Largo;
}
}
```

3. Clases Derivadas: Cuadrado, Circulo, línea, Triangulo

Cada una de estas clases extiende la clase Formas y define su propio constructor para inicializar los atributos específicos de la forma.

El diseño del código se basa en los principios de la programación orientada a objetos (POO), específicamente la herencia. A continuación, se describen algunos aspectos clave:



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package helpers;

/**
 *
 * @author Usuario
 */
public class Cuadrado extends Formas{
    public Cuadrado(){
    establecerDibujar("Cuadrado");
     establecerColor("Amarillo");
    CalcularArea ("50");
}
}
```

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package helpers;

/**
 *
 * @author Usuario
 */
public class Circulo extends Formas{
      public Circulo(){
        establecerDibujar("Circulo");
        establecerColor("Azul");
        establecerRadio("30");
      }
}
```

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package helpers;

/**
 *
 * @author Usuario
 */
public class Linea extends Formas {
public Linea(){
        establecerDibujar("Linea");
         establecerColor("Rojo");
        establecerLargo ("50");
}
}
```

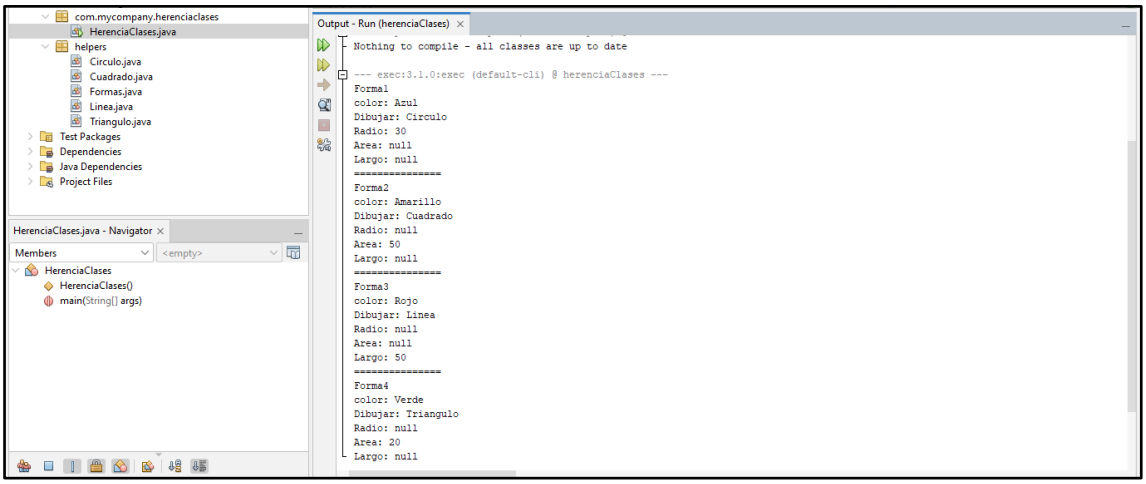
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package helpers;

/**
 *
 * @author Usuario
 */
public class Triangulo extends Formas{
    public Triangulo (){
        establecerDibujar("Triangulo");
         establecerColor("Verde");
    CalcularArea ("20");
    }
}
```

1. Encapsulamiento

El uso de métodos establecer y obtener permite controlar el acceso a los atributos de las clases. Esto asegura que los datos no sean manipulados directamente desde fuera de la clase, manteniendo la integridad de los datos.

2. Herencia

Las clases derivadas (Cuadrado, Circulo, Línea, Triangulo) heredan de la clase base Formas, reutilizando así el código común y proporcionando especialización según las necesidades de cada forma.



3. Polimorfismo

El método imprimirInformacion es común a todas las formas y se puede llamar de la misma manera independientemente del tipo de forma. Esto muestra cómo las instancias de diferentes clases derivadas pueden ser tratadas de manera uniforme a través de la interfaz común proporcionada por la clase base.