ACTIVIDAD 1:

class A {

public void a() {

System.*out*.println("A");

}

}

class B extends A {

public void b() {

System.*out*.println("B");

}

}

class C extends B {

public void c() {

System.*out*.println("C");

}

}

public class Main {

public static void main(String[] args) {

C c = new C();

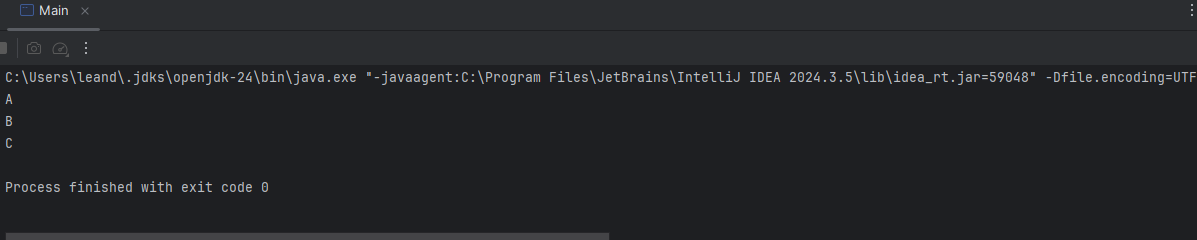
c.a();

c.b();

c.c();

}

}



ACTIVIDAD 2:

public class Punto {

protected int x;

protected int y;

public Punto(int x, int y) {

this.x = x;

this.y = y;

}

public void mostrar() {

System.*out*.println("Coordenadas: (" + x + ", " + y + ")");

}

}

public class PuntoColor extends Punto {

private int z;

private String color;

public PuntoColor(int x, int y, int z, String color) {

super(x, y);

this.z = z;

this.color = color;

}

@Override

public void mostrar() {

System.*out*.println("Coordenadas: (" + x + ", " + y + ", " + z + "), Color: " + color);

}

}

public class Main {

public static void main(String[] args) {

Punto p1 = new Punto(2, 3);

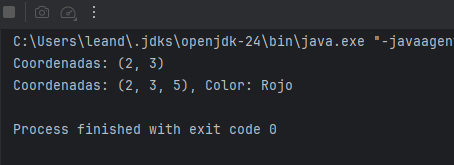
p1.mostrar();

PuntoColor p2 = new PuntoColor(2, 3, 5, "Rojo");

p2.mostrar();

}

}



ACTIVIDAD 3:

No tiene sentido que la clase Orden herede de Cliente, ya que Orden utiliza componentes de Cliente, pero no es un cliente. Al establecer esta relación de herencia, se le están asignando a Orden atributos y comportamientos innecesarios, lo cual va en contra de los principios de una buena arquitectura orientada a objetos. En este caso, sería más apropiado que Orden contenga una instancia de Cliente en lugar de heredar de ella.