Ejemplo 3

Planteamiento del Ejercicio - Usar árboles.

Se desea crear un sistema para representar la estructura jerárquica de una empresa, donde cada empleado tiene subordinados que pueden ser otros empleados. Para ello, se van a definir las siguientes clases:

1. Clase `Empleado`

Esta clase representa a un empleado de la empresa y tiene los siguientes atributos:

- `nombre`: Representa el nombre del empleado.
- `subordinados`: Lista de empleados subordinados a este empleado.

Además, tiene los siguientes métodos:

- `agregarSubordinado(empleado)`: Agrega un nuevo empleado como subordinado de este empleado.
- `mostrarJerarquia()`: Muestra la jerarquía de la empresa comenzando desde este empleado.

2. Clase `Empresa`

Esta clase contiene el método `main` donde se crea la estructura de la empresa. Se crean varios empleados y se establecen las relaciones jerárquicas entre ellos. Luego, se muestra la jerarquía de la empresa a partir del CEO.

```
package model.TreeCompany;

import java.util.ArrayList;
import java.util.List;

public class Empleado {
    String nombre;
    List<Empleado> subordinados;

    public Empleado(String nombre) {
        this.nombre = nombre;
        this.subordinados = new ArrayList<>();
    }

    public void agregarSubordinado(Empleado subordinado) {
        this.subordinados.add(subordinado);
    }

    public void mostrarJerarquia() {
        mostrarJerarquiaRec(this, 0);
    }

    private void mostrarJerarquiaRec(Empleado empleado, int nivel) {
        StringBuilder espacio = new StringBuilder();
        for (int i = 0; i < nivel; i++) {
```

```
            espacio.append("  ");
        }
        System.out.println(espacio.toString() + empleado.nombre);
        for (Empleado subordinado : empleado.subordinados) {
            mostrarJerarquiaRec(subordinado, nivel + 1);
        }
    }
}
```

```
package view.TreeCompany;

import model.TreeCompany.Empleado;

public class Empresa {
    public static void main(String[] args) {
        // Crear empleados
        Empleado ceo = new Empleado("Juan (CEO)");
        Empleado gerenteTI = new Empleado("Karol (Gerente de TI)");
        Empleado gerenteVentas = new Empleado("María (Gerente de Ventas)");
        Empleado gerenteProduccion = new Empleado("Carlos (Gerente de
Producción)");
        Empleado supervisor1 = new Empleado("Ana (Supervisor 1)");
        Empleado supervisor2 = new Empleado("Pedro (Supervisor 2)");
        Empleado supervisor3 = new Empleado("Luisa (Supervisor 3)");
        Empleado vendedor1 = new Empleado("Sara (Vendedor 1)");
        Empleado vendedor2 = new Empleado("Luis (Vendedor 2)");
        Empleado operario1 = new Empleado("Elena (Operario 1)");
        Empleado operario2 = new Empleado("Diego (Operario 2)");
        Empleado developer1 = new Empleado("Jorge (Desarrollador 1)");
        Empleado developer2 = new Empleado("Marcela (Desarrollador 2)");
        Empleado tester1 = new Empleado("Miguel (Tester 1)");
        Empleado tester2 = new Empleado("Laura (Tester 2)");
        Empleado analista1 = new Empleado("Andrés (Analista 1)");
        Empleado pasante1 = new Empleado("Camila (Pasante 1)");
        Empleado pasante2 = new Empleado("Esteban (Pasante 2)");
        Empleado pasante3 = new Empleado("Valeria (Pasante 3)");

        // Construir la jerarquía
        ceo.agregarSubordinado(gerenteVentas);
        ceo.agregarSubordinado(gerenteProduccion);
        ceo.agregarSubordinado(gerenteTI);
        gerenteVentas.agregarSubordinado(supervisor1);
        gerenteVentas.agregarSubordinado(supervisor2);
        gerenteTI.agregarSubordinado(supervisor3);
        supervisor1.agregarSubordinado(vendedor1);
        supervisor1.agregarSubordinado(vendedor2);
        gerenteProduccion.agregarSubordinado(operario1);
        gerenteProduccion.agregarSubordinado(operario2);
        supervisor3.agregarSubordinado(developer1);
        supervisor3.agregarSubordinado(developer2);
        developer1.agregarSubordinado(tester1);
```

```
        developer1.agregarSubordinado(tester2);
        developer1.agregarSubordinado(analista1);
        tester1.agregarSubordinado(pasante1);
        tester1.agregarSubordinado(pasante2);

        // Mostrar jerarquía de la organización
        System.out.println("Jerarquía de la organización:");
        ceo.mostrarJerarquia();
    }
}
```