Actividad 3 Desarrolla una aplicación que permita modelar una red de almacenes interconectados. El sistema debe permitir agregar almacenes, conectar almacenes entre sí mediante rutas directas, y realizar recorridos en profundidad (DFS) y en anchura (BFS) para explorar la red de distribución. Implementa una clase Almacen que represente un almacén. Cada almacén debe tener un identificador único y un nombre. Implementa una clase Grafo que gestione la red de almacenes. Esta clase debe: Permitir agregar almacenes al grafo. Permitir conectar almacenes entre sí (crear rutas directas entre almacenes). Implementar un método DFS para realizar un recorrido en profundidad desde un almacén de inicio. Implementar un método BFS para realizar un recorrido en anchura desde un almacén de inicio.El grafo debe estar representado utilizando una lista de adyacencia.

import java.util.\*;

class Almacen {

private int id;

private String nombre;

public Almacen(int id, String nombre) {

this.id = id;

this.nombre = nombre;

}

public int getId() {

return id;

}

public String getNombre() {

return nombre;

}

@Override

public String toString() {

return "Almacen{" +

"id=" + id +

", nombre='" + nombre + '\'' +

'}';

}

}

class Grafo {

private Map<Almacen, LinkedList<Almacen>> listaAdyacencia;

public Grafo() {

listaAdyacencia = new HashMap<>();

}

public void agregarAlmacen(Almacen almacen) {

listaAdyacencia.putIfAbsent(almacen, new LinkedList<>());

}

public void conectarAlmacenes(Almacen origen, Almacen destino) {

listaAdyacencia.get(origen).add(destino);

listaAdyacencia.get(destino).add(origen); // Si el grafo es no dirigido

}

public void DFS(Almacen inicio) {

Set<Almacen> visitados = new HashSet<>();

DFSUtil(inicio, visitados);

}

private void DFSUtil(Almacen actual, Set<Almacen> visitados) {

visitados.add(actual);

System.out.println(actual);

for (Almacen vecino : listaAdyacencia.get(actual)) {

if (!visitados.contains(vecino)) {

DFSUtil(vecino, visitados);

}

}

}

public void BFS(Almacen inicio) {

Set<Almacen> visitados = new HashSet<>();

Queue<Almacen> cola = new LinkedList<>();

visitados.add(inicio);

cola.add(inicio);

while (!cola.isEmpty()) {

Almacen actual = cola.poll();

System.out.println(actual);

for (Almacen vecino : listaAdyacencia.get(actual)) {

if (!visitados.contains(vecino)) {

visitados.add(vecino);

cola.add(vecino);

}

}

}

}

}

public class RedAlmacenes {

public static void main(String[] args) {

Grafo grafo = new Grafo();

Almacen almacen1 = new Almacen(1, "Almacen Central");

Almacen almacen2 = new Almacen(2, "Almacen Norte");

Almacen almacen3 = new Almacen(3, "Almacen Sur");

Almacen almacen4 = new Almacen(4, "Almacen Este");

grafo.agregarAlmacen(almacen1);

grafo.agregarAlmacen(almacen2);

grafo.agregarAlmacen(almacen3);

grafo.agregarAlmacen(almacen4);

grafo.conectarAlmacenes(almacen1, almacen2);

grafo.conectarAlmacenes(almacen1, almacen3);

grafo.conectarAlmacenes(almacen2, almacen4);

grafo.conectarAlmacenes(almacen3, almacen4);

System.out.println("Recorrido DFS desde Almacen Central:");

grafo.DFS(almacen1);

System.out.println("Recorrido BFS desde Almacen Central:");

grafo.BFS(almacen1);

}

}