```java
import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.util.List;

public class Grafo extends JPanel {  2 usages

    public final Map<String, Point> posiciones = new HashMap<>();  13 usages
    public final Map<String, List<Nodo>> grafo;  2 usages

    public Grafo(Map<String, List<Nodo>> grafo) {  1 usage
        this.grafo = grafo;
        posiciones();
        setPreferredSize(new Dimension( width: 600,  height: 400));
        setBackground(Color.BLACK);
        setOpaque(true);
    }

    private void posiciones() {  1 usage
        posiciones.put("Tierra", new Point( x: 90,  y: 200));
        posiciones.put("Orbita Terrestre Alta", new Point( x: 300,  y: 50));
        posiciones.put("Luna", new Point( x: 450,  y: 200));
        posiciones.put("Base Lunar", new Point( x: 500,  y: 300));
        posiciones.put("Base Orbital", new Point( x: 100,  y: 100));
        posiciones.put("Estacion Espacial Internacional", new Point( x: 200,  y: 350));
        posiciones.put("Satelite Sputnik", new Point( x: 300,  y: 310));
        posiciones.put("Voyager 1", new Point( x: 600,  y: 100));
```

```java
    private void dibujarNodo(Graphics2D g2, String nodo, Point p) {  1 usage
        int size = 30;
        if (nodo.equals("Tierra")) size = 50;
        else if (nodo.equals("Luna")) size = 40;

        g2.setColor(Color.CYAN);
        g2.fillOval( x: p.x - size / 2,  y: p.y - size / 2, size, size);

        g2.setColor(Color.WHITE);
        FontMetrics fm = g2.getFontMetrics();
        int textWidth = fm.stringWidth(nodo);
        g2.drawString(nodo,  x: p.x - textWidth / 2,  y: p.y - size / 2 - 5);
    }
}
```

```java
public class Nodo {  11 usages
    public String destino;  7 usages
    public int peso;  4 usages

    public Nodo(String destino, int peso) {  3 usages
        this.destino = destino;
        this.peso = peso;
    }
}
```

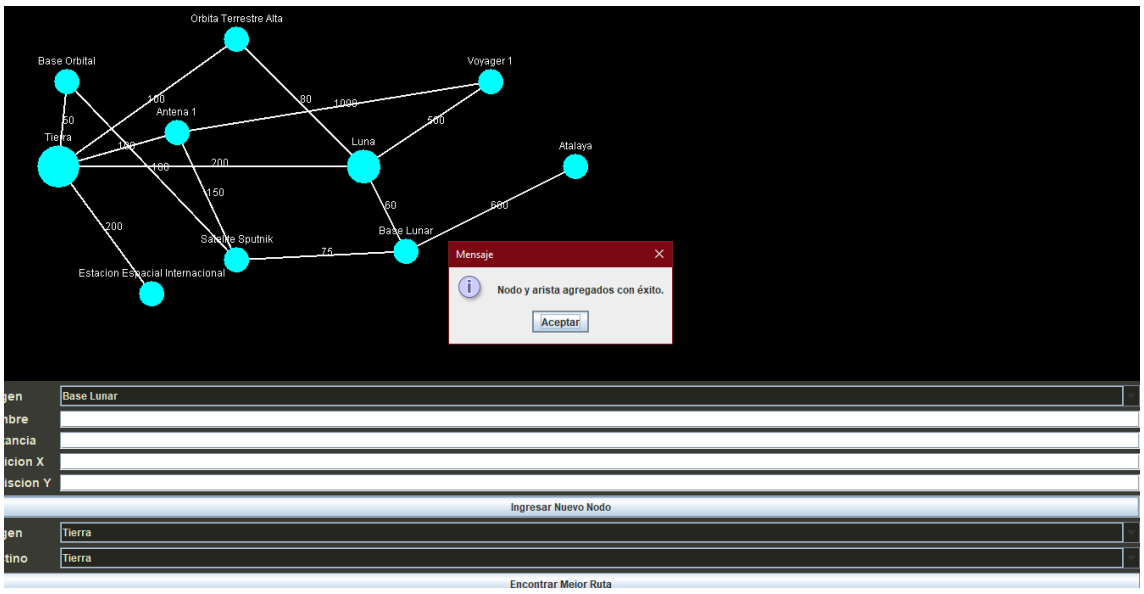
```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;
import java.util.List;

public class TraficoNasa {
    private JPanel PGeneral;  10 usages
    private JPanel DibujarGrafo;  3 usages
    private JButton encontrarMejorRutaButton;  2 usages
    private JComboBox Destino;  3 usages
    private JComboBox Origen;  3 usages
    private JLabel OrigenIngresar;  1 usage
    private JComboBox OrigenIng;  3 usages
    private JTextField textNodonuevo;  3 usages
    private JLabel NodoNuevo;  1 usage
    private JTextField textPeso;  3 usages
    private JButton ingresarNuevoNodoButton;  2 usages
    private JLabel PesoNuevo;  1 usage
    private JTextField textX;  3 usages
    private JTextField textY;  3 usages
    private JLabel PosiscionY;  1 usage
    private JLabel PosiscionX;  1 usage
```

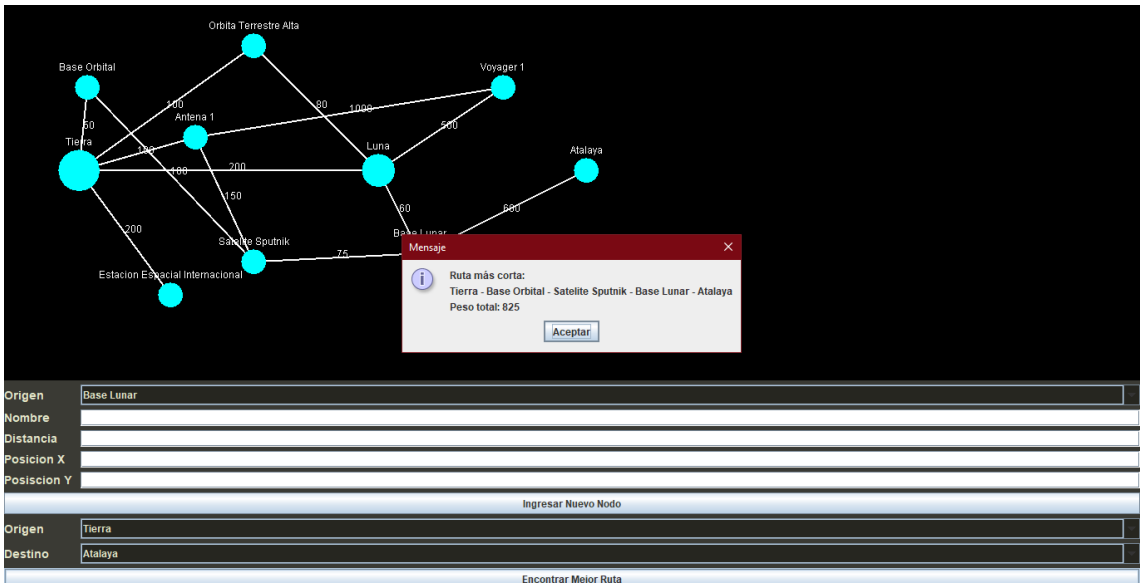
```java
public class TraficoNasa {  ⚠14 ✓70
    public TraficoNasa() {  1 usage
        DibujarGrafo.add(dibujo, BorderLayout.CENTER);
        encontrarMejorRutaButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String inicio = (String) Origen.getSelectedItem();
                String fin = (String) Destino.getSelectedItem();

                if (inicio == null || fin == null) {
                    JOptionPane.showMessageDialog(PGeneral, message: "Por favor selecciona nodo origen y destino.");
                    return;
                }

                if (inicio.equals(fin)) {
                    JOptionPane.showMessageDialog(PGeneral, message: "El nodo origen y destino no pueden ser iguales.");
                    return;
                }

                List<String> ruta = dijkstra(grafo, inicio, fin);

                if (ruta.isEmpty()) {
                    JOptionPane.showMessageDialog(PGeneral, message: "No existe ruta entre " + inicio + " y " + fin);
                } else {
                    int pesoTotal = 0;
                    for (int i = 0; i < ruta.size() - 1; i++) {
                        String actual = ruta.get(i);
                        String siguiente = ruta.get(i + 1);
```

```java
ingresarNuevoNodoButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String nombreNuevoNodo = textNodonuevo.getText().trim();
        String origen = (String) OrigenIng.getSelectedItem();
        String pesoStr = textPeso.getText().trim();
        String xStr = textX.getText().trim();
        String yStr = textY.getText().trim();

        if (nombreNuevoNodo.isEmpty() || origen == null || pesoStr.isEmpty() || xStr.isEmpty() || yStr.isEmpty()) {
            JOptionPane.showMessageDialog(PGeneral, message: "Por favor llena todos los campos.");
            return;
        }

        if (grafo.containsKey(nombreNuevoNodo)) {
            JOptionPane.showMessageDialog(PGeneral, message: "Ese nodo ya existe.");
            return;
        }

        int peso;
        int x, y;
        try {
            peso = Integer.parseInt(pesoStr);
```

```java
    private void inicializarGrafo() {
        arista( origen: "Tierra", destino: "Base Orbital", peso: 50);
        arista( origen: "Tierra", destino: "Orbita Terrestre Alta", peso: 100);
        arista( origen: "Tierra", destino: "Antena 1", peso: 100);
        arista( origen: "Antena 1", destino: "Voyager 1", peso: 1000);
        arista( origen: "Antena 1", destino: "Satelite Sputnik", peso: 150);
        arista( origen: "Orbita Terrestre Alta", destino: "Luna", peso: 80);
        arista( origen: "Tierra", destino: "Luna", peso: 200);
        arista( origen: "Estacion Espacial Internacional", destino: "Tierra", peso: 200);
        arista( origen: "Luna", destino: "Satelite Geoestacionario", peso: 25);
        arista( origen: "Luna", destino: "Base Lunar", peso: 60);
        arista( origen: "Luna", destino: "Voyager 1", peso: 500);
        arista( origen: "Satelite Sputnik", destino: "Base Lunar", peso: 75);
        arista( origen: "Satelite Sputnik", destino: "Base Orbital", peso: 100);

    }
```

```java
private void arista(String origen, String destino, int peso) {
    grafo.putIfAbsent(origen, new ArrayList<>());
    grafo.get(origen).add(new Nodo(destino, peso));
    grafo.putIfAbsent(destino, new ArrayList<>());
    grafo.get(destino).add(new Nodo(origen, peso));
}

public static List<String> dijkstra(Map<String, List<Nodo>> grafo, String inicio, String fin) {
    Map<String, Integer> distancias = new HashMap<>();
    Map<String, String> predecesores = new HashMap<>();
    PriorityQueue<String> queue = new PriorityQueue<>(Comparator.comparingInt(distancias::get));

    for (String nodo : grafo.keySet()) {
        distancias.put(nodo, Integer.MAX_VALUE);
    }
    distancias.put(inicio, 0);
    queue.offer(inicio);

    while (!queue.isEmpty()) {
        String actual = queue.poll();
        if (actual.equals(fin)) break;

        for (Nodo vecino : grafo.getOrDefault(actual, new ArrayList<>())) {
            int nuevoPeso = distancias.get(actual) + vecino.peso;
            if (nuevoPeso < distancias.get(vecino.destino)) {
```





Origen: Base Lunar

Nombre:

Distancia:

Posicion X:

Posiscion Y:

Ingresar Nuevo Nodo

Origen: Tierra

Destino: Atalaya

Encontrar Meior Ruta