Graph Parser

The Graph Parser is a Java library that allows you to read, write, and manipulate graphs represented in the Graphviz DOT format. It provides methods for adding, removing, and modifying nodes and edges in a graph, as well as for reading and writing graphs to file and rendering them as images.

Introduction

This is a Java program that parses DOT files and creates a directed graph using the JGraphT library. The program has the following features:

- Feature 1: Parsing DOT files and creating a directed graph
- Feature 2: Adding and removing nodes from the graph
- Feature 3: Adding and removing edges between nodes in the graph
- Feature 4: Outputting the graph in DOT format and rendering it as an image in PNG format

Installation and Dependencies

The program requires Java 8 or later and the JGraphT library. The JGraphT library can be downloaded from https://jgrapht.org/ or included as a Maven dependency in your project.

GitHub Link : https://github.com/Sim-ran-K/CSE464_Proj.git

Getting Started

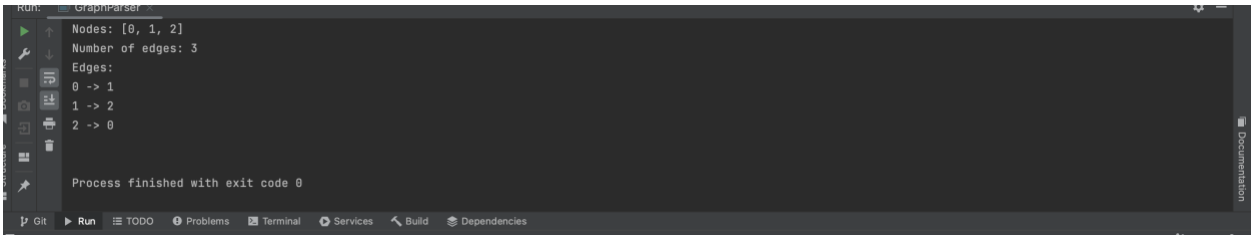
To use the Graph Parser in your Java project, you will need to add the following dependency to your pom.xml file:

```
<dependency>
  <groupId>org.jgrapht</groupId>
  <artifactId>jgrapht-core</artifactId>
  <version>1.5.1</version>
</dependency>
```

You can then import the GraphParser class and use its methods to work with graphs. Here's an example of how to parse a graph from a file and output its contents:
java

```
GraphParser parser = new GraphParser();
parser.parseGraph("example.dot");
System.out.println(parser.toString());
```

This will output the contents of the parsed graph to the console.



Features

The Graph Parser provides several features for working with graphs:
Parsing: The parseGraph method allows you to parse a graph from a file in the Graphviz DOT format.
Outputting: The outputGraph, outputDOTGraph, and outputGraphics methods allow you to output a graph to a file in the Graphviz DOT format, as a DOT string, or as an image file (in PNG, SVG, or PDF format), respectively.
Node and edge manipulation: The addNode, removeNode, addNodes, removeNodes, addEdge, and removeEdge methods allow you to add, remove, and modify nodes and edges in a graph.

Usage

The main method of the GraphParser class can be used as an example of how to use the program. It takes two optional command-line arguments: the input file path and the output file path. If no arguments are provided, the program will use input.dot as the input file and output.dot as the output file.
To use the program for your own DOT files, you can create a new instance of GraphParser and call its methods to manipulate the graph. The methods available are:

- parseGraph(String filepath): Parse a DOT file and create the directed graph
- toString(): Convert the graph to a string representation
- outputGraph(String filepath): Output the graph to a DOT file
- addNode(String label): Add a node with the specified label to the graph
- removeNode(String label): Remove the node with the specified label from the graph
- addNodes(String[] labels): Add multiple nodes with the specified labels to the graph
- removeNodes(String[] labels): Remove multiple nodes with the specified labels from the graph
- addEdge(String srcLabel, String dstLabel): Add an edge between the nodes with the specified labels

- removeEdge(String srcLabel, String dstLabel): Remove the edge between the nodes with the specified labels
- outputDOTGraph(String filepath): Output the graph to a DOT file using the JGraphT DOTExporter
- outputGraphics(String filepath, String format): Render the graph as an image and output it to a file in the specified format (e.g. "PNG")

Here's an example of how to use the Graph Parser to parse a graph from a file, add a node and an edge, and output the modified graph as an image file:

```
GraphParser parser = new GraphParser();
parser.parseGraph("example.dot");
parser.addNode("newNode");
parser.addEdge("node1", "newNode");
parser.outputGraphics("example.png", "PNG");
```

This will parse the graph from the file example.dot, add a node with the label "newNode", add an edge from the node with the label "node1" to the new node, and output the modified graph as a PNG image file named example.png.

Here is a workflow action on github: