User Manual for Two-Pass Assembler GUI Application

Components:

1. **GUI Layout**:
    - There are four text areas for displaying:
        - Input source code
        - SymbTab
        - Intermediate code
        - Object code
    - A button named "Assemble" starts the two-pass assembly process.

1. Introduction

This is a GUI-based two-pass assembler written in Java using Swing. It simulates the assembly process by converting source code into object code through two passes:

- **Pass One**: Creates an intermediate representation and a symbol table.
- **Pass Two**: Generates the final object code from the intermediate representation and symbol table.

---

2. System Requirements

- **Java Development Kit (JDK)**: Version 8 or higher.
- A text editor or IDE that supports Java.
- Basic knowledge of assembly language programming.

---

3. How to Run the Program

1. **Compile the Java Code**:
    - Open a command line terminal.
    - Navigate to the folder where the source code is stored.

Run the following command:

```
javac GUI.java
```

2. **Run the Assembler GUI**:

- After compiling, execute the program with:

```
java Twopassassembler
```

4. GUI Overview

Main Interface Components:

1. **Input Source Code Area**: A text area where the user can input assembly source code.
2. **SymTab (Symbol Table) Area**: Displays the symbol table generated after the first pass.
3. **Intermediate Code Area**: Shows the intermediate code produced after the first pass.
4. **Object Code Area**: Displays the final object code after the second pass.
5. **Assemble Button**: Click this button to begin the two-pass assembly process.

5. How to Use

Step 1: Input Source Code

- Enter the assembly code in the "Input Source Code" text area. Ensure the code follows the expected format, with labels, opcodes, and operands.

Step 2: Assemble

- Once the code is entered, click the **Assemble** button.
- The application will:
    1. **Pass One**: Parse the source code, create a symbol table, and generate intermediate code.
    2. **Pass Two**: Use the intermediate code and symbol table to generate object code.

Step 3: View the Results

- **SymTab Area**: The symbol table generated during Pass One will be displayed here.
- **Intermediate Code Area**: The intermediate code generated during Pass One will be shown.
- **Object Code Area**: The final object code will be displayed after Pass Two.

6. Files Involved

- **input.txt**: Stores the input source code temporarily.
- **optab.txt**: Contains the list of supported opcodes and their corresponding machine code.
- **symtab.txt**: Stores the symbol table generated during Pass One.
- **intermediate.txt**: Stores the intermediate code generated during Pass One.
- **objectcode.txt**: Stores the final object code generated during Pass Two.

7. Pass One and Pass Two Explanation

Pass One:

- The assembler reads the source code line by line.
- It calculates the addresses of labels and stores them in the symbol table (symtab.txt).

- The intermediate code is written to intermediate.txt.

Pass Two:

- The assembler uses the intermediate file and symbol table to generate machine code.
- The final object code is written to object_code.txt.

8. Customizing the Program

Modifying Opcodes:

- The optab.txt file stores the opcode-to-machine-code mapping. You can modify this file to add or remove opcodes.

Editing the GUI:

- You can customize the GUI components, like colors, layout, or button positions, by modifying the relevant sections in the code.

10. Troubleshooting

Common Issues:

- **File Not Found Errors**: Ensure that the program can read/write to the necessary files (input.txt, optab.txt, etc.).
- **Incorrect Object Code**: Check the input assembly code for errors, such as typos in opcodes or incorrect operands.

Conclusion

- This Two-Pass Assembler GUI simplifies the process of assembling basic assembly language programs into object code. It demonstrates how a two-pass assembler works, with a focus on readability and user interaction through a graphical interface.