1. 문제 분석

문제 정의: 자바 Swing 라이브러리를 이용해 메모장을 구현하는 과제이다. 파일 저장, 파일 불러오기, 문자열 찾기, 문자열 치환의 기능을 구현한다.

문제 해결 방안: Swing에서 제공하는 여러 가지 컴포넌트를 적절히 조합하여 GUI 로직을 구현하고, 적절한 알고리즘을 사용하여 문제를 해결하면 된다.

2. 문제 해결

소스 코드:

[Notepad.java]

```
public class Notepad {
  public static void main(String[] args) {
    App app = new App();
    app.run();
  }
}
```

[App.java]

```
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class App extends JFrame {
```

```
// 사용되는 GUI 컴포넌트들
private JTextArea memoTextArea;

private JMenuBar menuBar;

private JMenu fileMenu;
private JMenuItem saveMenuItem;
private JMenuItem loadMenuItem;
private JMenuItem saveAsMenuItem;

private JMenu stringMenu;
private JMenuItem findMenuItem;
private JMenuItem replaceMenuItem;

private String fileName = "Untitled";
private String filePath = "";

// 앱이 실행될 때 호출되는 메소드이다.
public void run() {
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setTitle(fileName + " - Notepad");

  initMenu();

  setLayout(new GridLayout(1, 1));
  memoTextArea = new JTextArea();
  memoTextArea.setLineWrap(true);
  add(new JScrollPane(memoTextArea));

  setSize(500, 500);
  setVisible(true);
}

// 메뉴를 초기화하고 클릭 이벤트와 연결한다.
private void initMenu() {
  menuBar = new JMenuBar();

  fileMenu = new JMenu("File");
  saveMenuItem = new JMenuItem("Save");
  loadMenuItem = new JMenuItem("Load");
  saveAsMenuItem = new JMenuItem("Save As");
```

```
stringMenu = new JMenu("String");
findMenuItem = new JMenuItem("Find");
replaceMenuItem = new JMenuItem("Replace");

menuBar.add(fileMenu);
fileMenu.add(saveMenuItem);
fileMenu.add(loadMenuItem);
fileMenu.add(saveAsMenuItem);

menuBar.add(stringMenu);
stringMenu.add(findMenuItem);
stringMenu.add(replaceMenuItem);

setJMenuBar(menuBar);

saveMenuItem.addActionListener(new ActionListener() {

  @Override
  public void actionPerformed(ActionEvent e) {
    saveText();
  }
});

loadMenuItem.addActionListener(new ActionListener() {

  @Override
  public void actionPerformed(ActionEvent e) {
    loadText();
  }
});

saveAsMenuItem.addActionListener(new ActionListener() {

  @Override
  public void actionPerformed(ActionEvent e) {
    saveAsText();
  }
});

findMenuItem.addActionListener(new ActionListener() {

  @Override
  public void actionPerformed(ActionEvent e) {
```

```
      findStringInText();
    }
  });

  replaceMenuItem.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
      replaceStringInText();
    }
  });
}

// 파일을 저장한다.
void saveText() {
  // 파일이 새로 만들어진 상태일 경우
  if (filePath.equals("")) {
    JFileChooser fileChooser = new JFileChooser();
    int ret = fileChooser.showSaveDialog(this);

    // 확인을 눌렀을 경우
    if (ret == JFileChooser.APPROVE_OPTION) {
      File file = fileChooser.getSelectedFile();

      try {
        Files.write(file.toPath(), memoTextArea.getText().getBytes());

        fileName = file.getName();
        filePath = file.getPath();
        setTitle(fileName + " - Notepad");
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }
  // filePath 에 파일이 저장되어 있는 경우
  else {
    File file = new File(filePath);

    try {
      // 파일에 내용 쓰기
      Files.write(file.toPath(), memoTextArea.getText().getBytes());
```

```
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}

// 파일을 읽어 그 내용을 텍스트 상자에 넣는다.
void loadText() {
  JFileChooser fileChooser = new JFileChooser();
  int ret = fileChooser.showOpenDialog(this);

  if (ret == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();

    try {
      String content = new String(Files.readAllBytes(file.toPath()));
      memoTextArea.setText(content);

      fileName = file.getName();
      filePath = file.getPath();
      System.out.println(file.getPath());
      setTitle(fileName + " - Notepad");
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}

// 다른 이름으로 저장
void saveAsText() {
  JFileChooser fileChooser = new JFileChooser();
  int ret = fileChooser.showSaveDialog(this);

  if (ret == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();

    try {
      Files.write(file.toPath(), memoTextArea.getText().getBytes());

      fileName = file.getName();
      filePath = file.getPath();
      setTitle(fileName + " - Notepad");
```

```
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }

  // 입력된 문자를 텍스트상자에서 찾아서 드래그해준다.
  void findStringInText() {

    String find = JOptionPane.showInputDialog(this, "To find");

    if (find != null) {
      if (memoTextArea.getText().contains(find)) {
        memoTextArea.setCaretPosition(memoTextArea.getText().indexOf(find));
        memoTextArea.moveCaretPosition(memoTextArea.getText().indexOf(find)         +
find.length());
      }
    }
  }

  // 찾을 문자열과 변경할 문자열을 입력받아 찾아서 바꿔 준다.
  void replaceStringInText() {
    String find = JOptionPane.showInputDialog(this, "To find");
    String replace = JOptionPane.showInputDialog(this, "To replace");

    if (find != null && replace != null) {
      memoTextArea.setText(memoTextArea.getText().replaceAll(find, replace));
    }
  }
}
```

3. 결과

사진 1. 처음 실행시켰을 때 화면

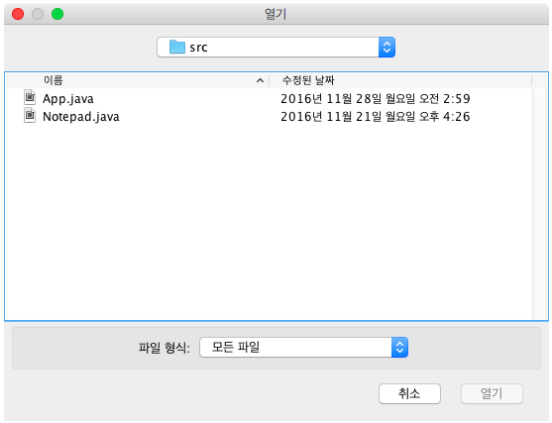

사진 2. File – Open 으로 파일을 선택

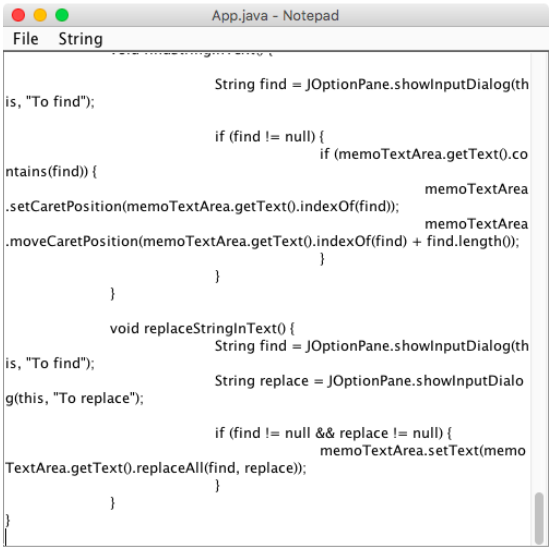

사진 3. 파일을 연 모습

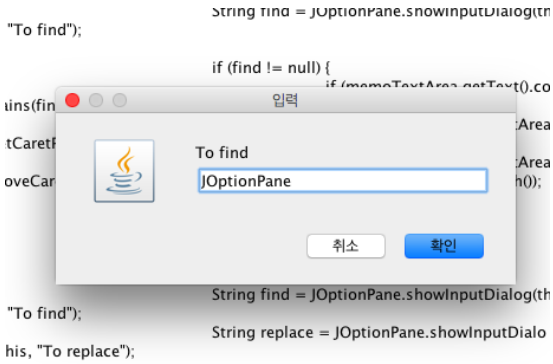

사진 4. String – Find 으로 검색

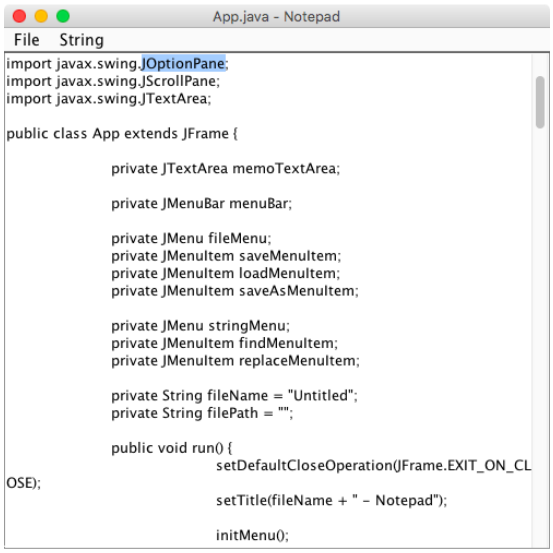

사진 5. 검색된 문자열이 선택됨

사진 6. String – Replace – Find: Tab 입력 – Replace: 공백 두 개 입력 후 실행 결과

사진 7. File – Save 후 저장되었는지 확인

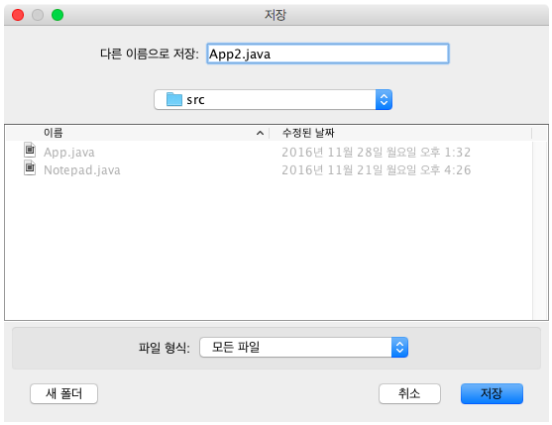

사진 8. File – Save As 사용

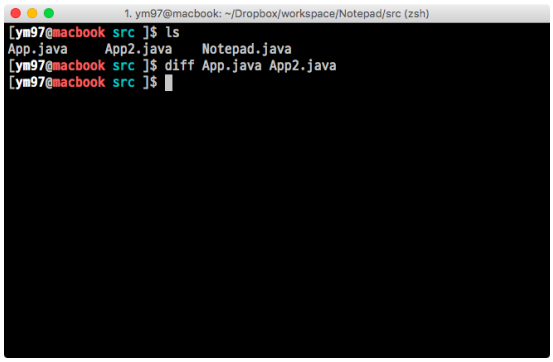

사진 9. 제대로 저장되었는지 확인