프로그램 개요

자바 Swing을 이용한 마인드맵

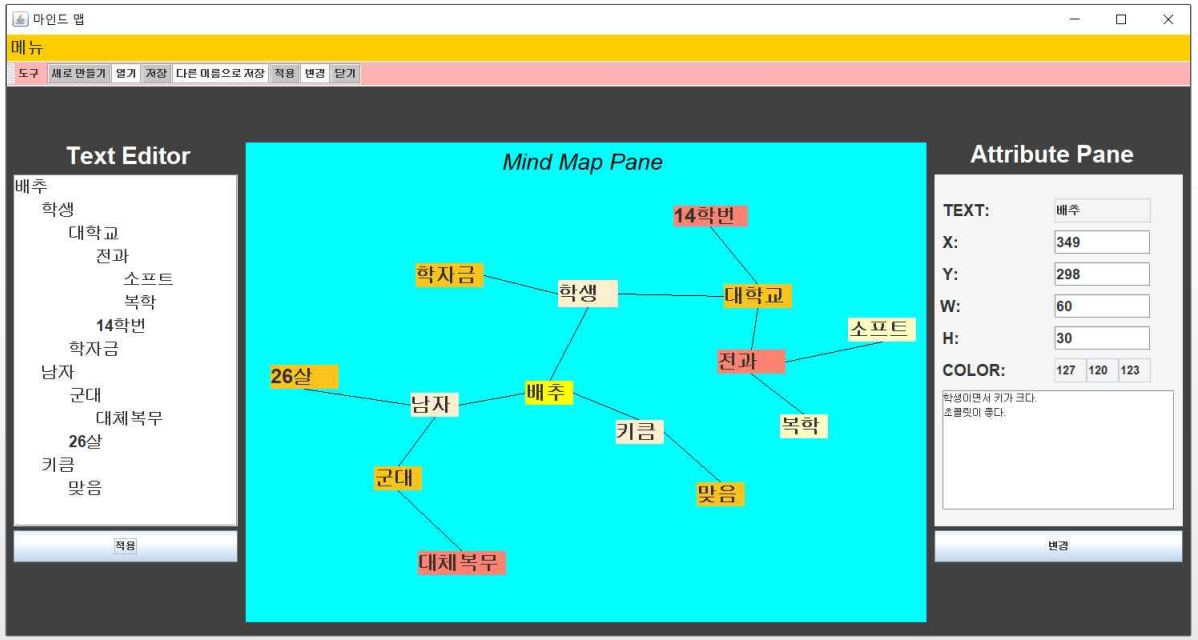
- 자신의 생각을 정리하는 마인드맵을 그림으로 나타낼 수 있는 GUI기반 프로그램.

- 텍스트 입력으로 관계 구조를 완성하고, 마우스와 다른 입력으로 노드의 위치와 정보를 변경할 수 있게 조작할 수 있습니다.

- json 파일로 저장해서 불러올 수 있습니다.

IDE : 이클립스

개발환경 : JDK 1.8.0_151 , json-simple-1.1.1



실행과정과 소스코드 소개

자바 Swing을 이용한 마인드맵

MindMap.java와 **Controler**.java로 나누어 MindMap.java에는 **보이는 비주얼**을 담당하고, Contoler.java는 **트리와 노드의 값을 관리**하는 역활을 합니다.

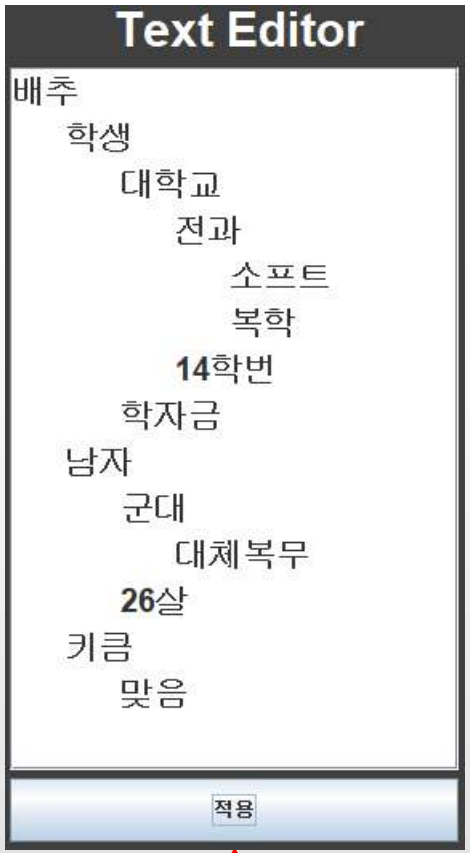
```
public class MindMap extends JFrame {
    //멤버변수
    private Controler mTree = new Controler();
    private MyTextArea tEditor; // JTextArea 상속
    private MyAtrpanel atrPane; // JPanel 상속
    private MindPanel mMapPane; // JPanel 상속
    private MyMenuBar menuBar;  // JMenuBar 상속
    private MyToolBar toolBar;  // JToolBar 상속

    private Container c;                // MindMap 본인을 지칭 하도록 만든 컨테이너.
    private String EditorText = null;   // TextArea의 완성된 문자열 상태를 가지고 있는 문자열.
    private String ourPath = null;      // JSON 저장을 하지 않은 상태라면 그 경로는 null이고, 있다면 그 경로 문자열이다.
    //-----------------------------------------------
```

MindMap 클래스에서 Controler 객체를 만들어 트리구조를 만들거나 검색, 변경 처리를 진행합니다.

MindMap이라는 JFrame 위에 여러 Swing클래스 컨테이너들을 올려두는 구조를 갖게 됩니다.

실행과정과 소스코드 소개

Text Editor



사용자는 'Tab키'를 이용해서 텍스트에 원하는 관계구조를 표현해주고 '적용' 버튼을 누르며 됩니다.

MindMap.java

GUI로부터 사용자가 입력한 텍스트를 간단한 구문분석을 하여 Controler 객체에게 넘깁니다.

```
mMapPane.DeletePrevLabel();
origin = tEditor.getText();

st = new StringTokenizer(origin, "\n");
int num = st.countTokens();
```

```
for(int i=0; i<num; i++) {
    int search=0;
    int count=0;

    buf = new StringBuffer(st.nextToken());
    while(buf.charAt(search)=='\t') {
        count++;
        search++;
    }

    buf.delete(0, search);
    mTree.count.add(count);
    mTree.textLength.add(buf.length());
    mTree.textadd.add(buf.toString());
}
```

Controler.java

받은 내용을 바탕으로 트리구조를 만들어줍니다.

```
if(count.get(i) == 0) {
    System.err.println("마인드맵의 가장 꼭대기 root는 하나만 일때만 가능합니다.\n 탭을
    root = null;
    return false;
}
else if(count.get(i)==count.get(i-1)+1) {
    cur.son = new MindNode(textadd.get(i),count.get(i));
    (cur.son).parent = cur;
    cur.son.width = (textLength.get(i)*25)+10;
    cur.search_next= cur.son;
    cur = cur.son;
}
else if(count.get(i)==count.get(i-1)) {
    cur.brother = new MindNode(textadd.get(i),count.get(i));
    (cur.brother).parent = cur.parent;
    cur.brother.width = (textLength.get(i)*25)+10;
    cur.search_next = cur.brother;
    cur = cur.brother;
}
else if(count.get(i)<count.get(i-1)) {
    MindNode temp2;
    temp2 = toMyParent(cur, count.get(i-1)-count.get(i));
    temp2.brother = new MindNode(textadd.get(i), count.get(i));
    (temp2.brother).parent = temp2.parent;
    temp2.brother.width = (textLength.get(i)*25)+10;
    cur.search_next = temp2.brother;
    cur = temp2.brother;
}
else {
    System.out.println("텍스트 에디터 에서 표준과 잘못된 입력이 있는지 확인해주세요");
    root = null;
    return false;
}
```

실행과정과 소스코드 소개

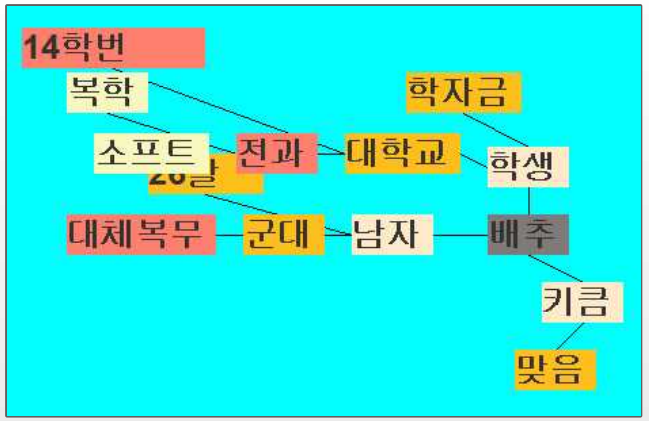
Controler.java

```
/* 만들어둔 트리를 이용해서 Mind Pane에 쓰일 JLabel을 만들어둔다. */
public void MakeJLabelNode() {
    MindNode temp = root;
    mindLabel = new LinkedList<JLabel>();
    for(int i=0; i<count.size(); i++) {
        mindLabel.add(new JLabel());
        mindLabel.get(i).setText(textadd.get(i));
        MindNode dest = FindNode(i);

        if(dest.width==0) {
            mindLabel.get(i).setSize((textLength.get(i)*25)+10,30);
            temp.width = (textLength.get(i)*25)+10;
        }
        else mindLabel.get(i).setSize(dest.width, dest.height);
        mindLabel.get(i).setLocation(dest.x, dest.y);
        temp = temp.search_next;
    }
}
```

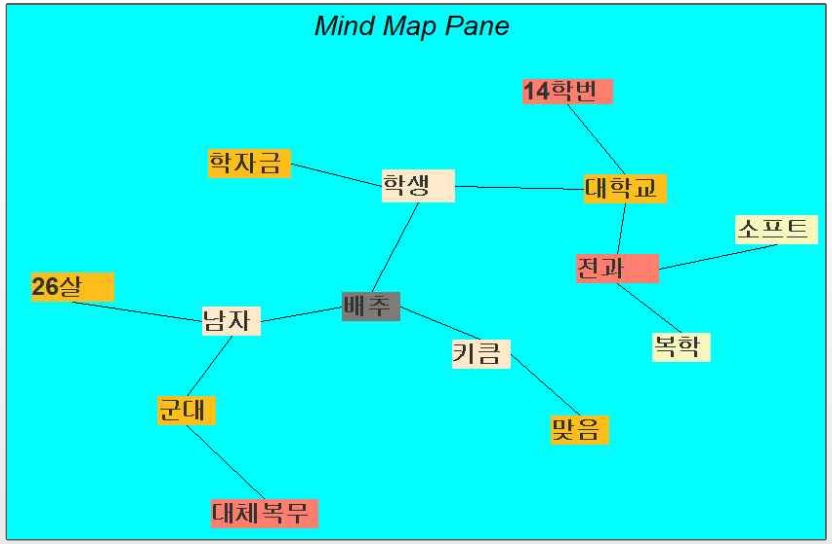
완성된 트리구조를 바탕으로 각각의 라벨노드를 형성시켜줍니다.

이후, MakeXY()메소드를 이용해 마인드맵에 위치를 단순하게 지정해줍니다.

MindMap.java

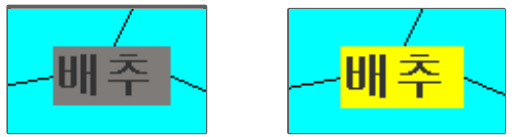


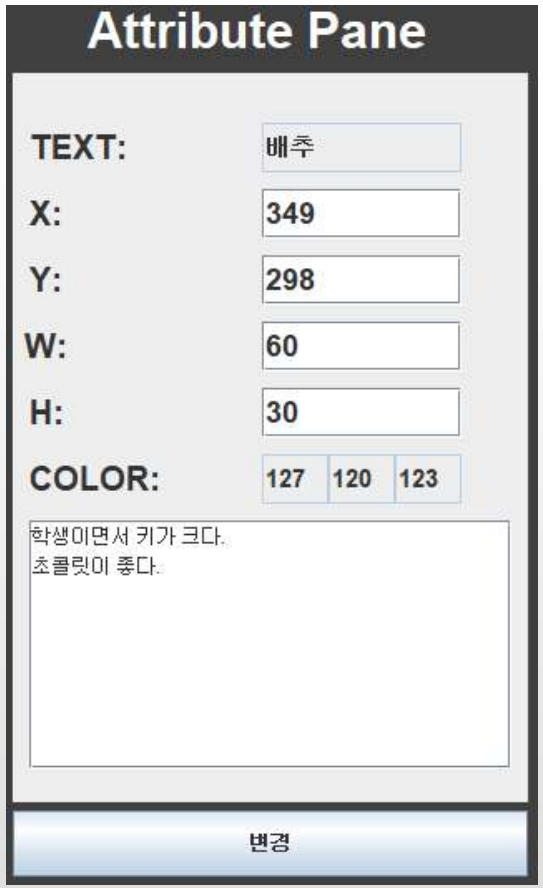
완성된 라벨을 Mind Map Pane위에 그려줍니다. 그리고 관계에 따른 선을 그려줍니다.



마우스이벤트를 활용해서 각 라벨노드의 위치 바꾸어 더 명료하게 볼수 있습니다.

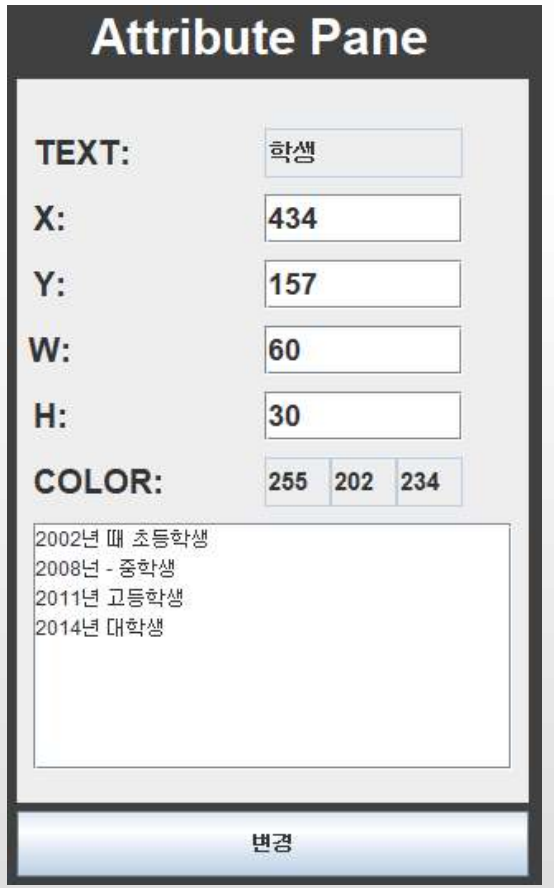
실행과정과 소스코드 소개



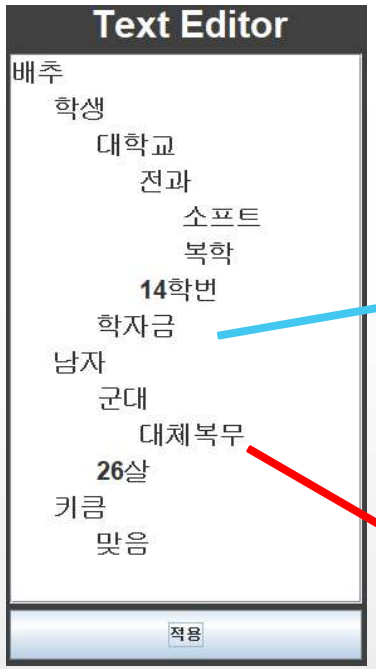


마인드 맵에 라벨을 클릭하면 노란색으로 색이 바뀌어서 해당 라벨의 정보는 Attribute Pane위에 나타나게 됩니다.

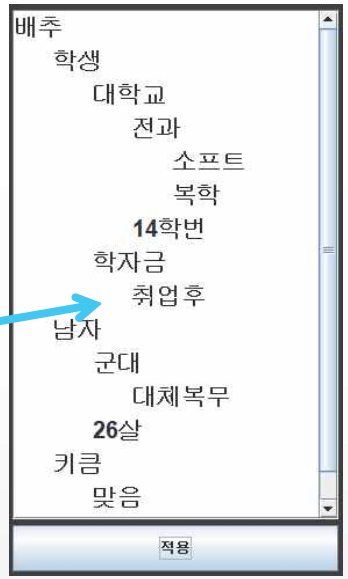
설정이나 라벨의 설명을 변경해서 저장하면 해당 라벨의 정보가 바뀌게 됩니다.

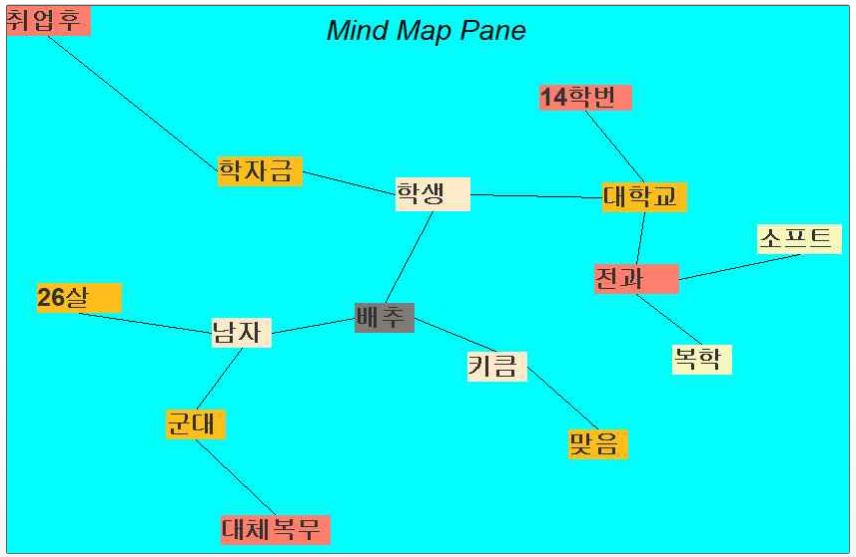


실행과정과 소스코드 소개

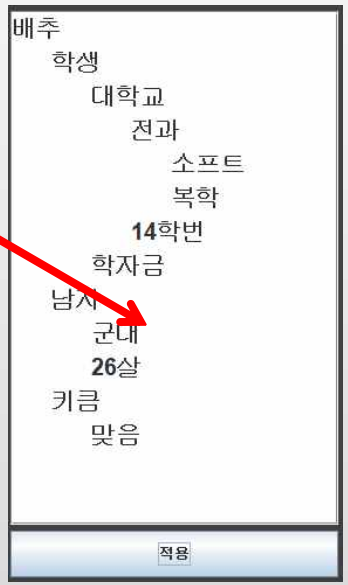


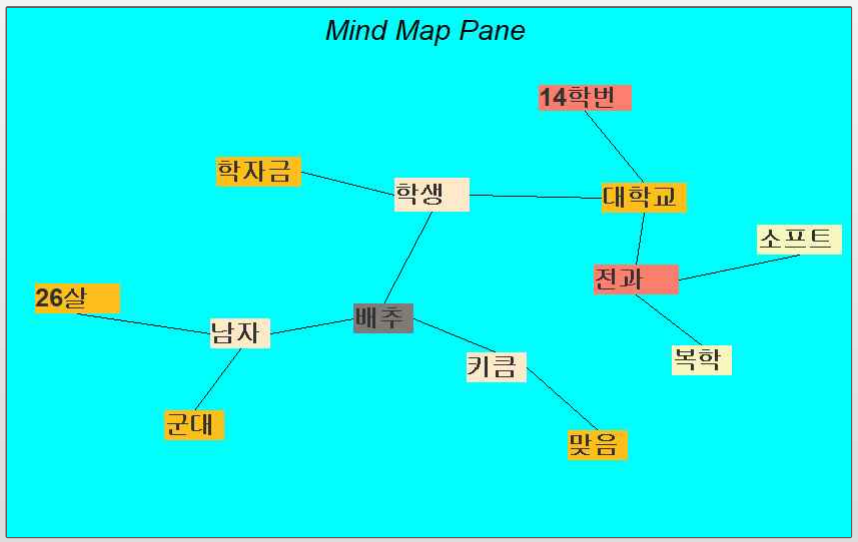
추가





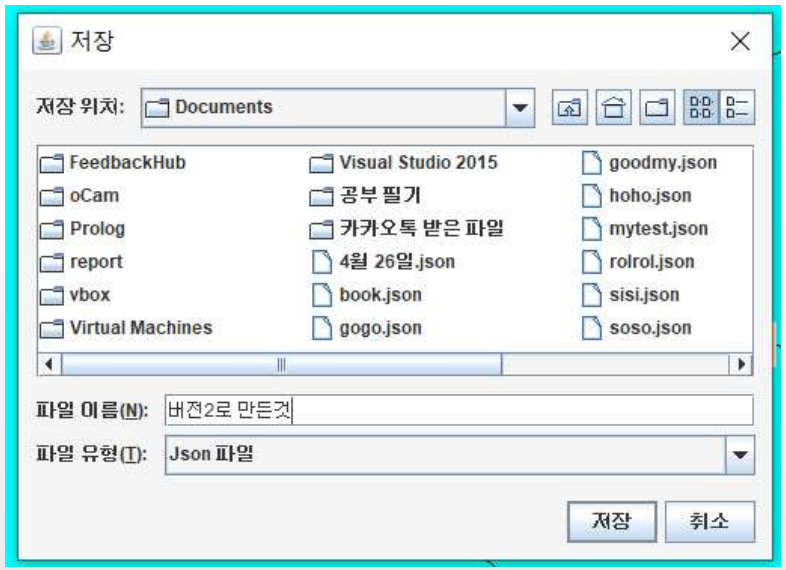
사라짐





기존에 있던 텍스트 관계에서 그 관계를 추가하거나 빼게 되면 원래의 라벨들은 그대로 있고 해당 정보만 마인드맵에서 추가되거나 삭제가 됩니다.

실행과정과 소스코드 소개



해당 라벨과 텍스트 정보들은 json형식의 파일로 저장할 수 있습니다.

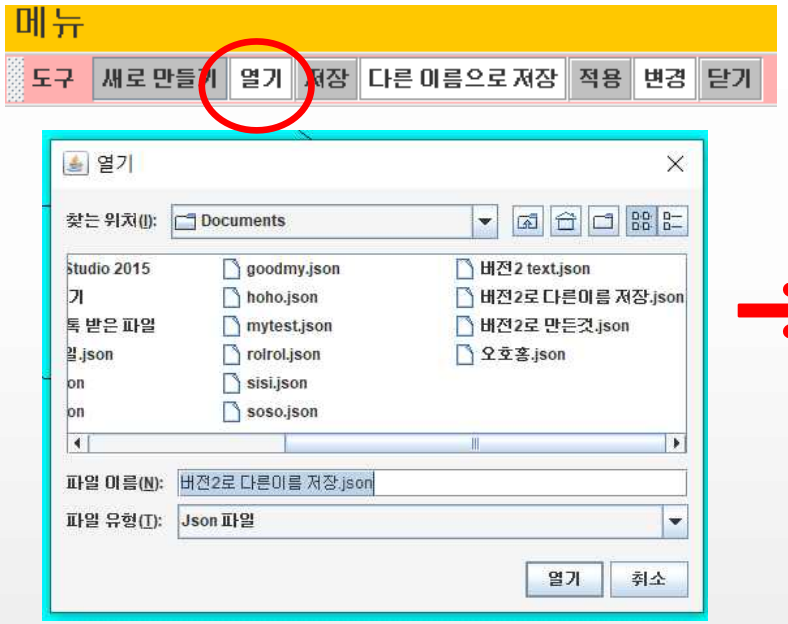
```
FileNameExtensionFilter filter = new FileNameExtensionFilter(
        "Json 파일","json");

chooser.setFileFilter(filter);
int ret = chooser.showSaveDialog(null);
if(ret != JFileChooser.APPROVE_OPTION) {
    JOptionPane.showMessageDialog(null, "파일을 저장하지 않으셨어요...",
            "경고",JOptionPane.WARNING_MESSAGE);
    return;}

filePath = chooser.getSelectedFile().getPath();
```

```
jobj.put("X", xArray);
jobj.put("Y", yArray);
jobj.put("W", wArray);
jobj.put("H", hArray);
jobj.put("Text", textArray);
jobj.put("Count", countArray);
jobj.put("tLength", tlengthArray);
jobj.put("Totalnum",totalnum);
jobj.put("EditText", Etext);
jobj.put("explainText", explaintext);
```

```
try {
    FileWriter file;
    if( Path==null && ourPath!=filePath)
        file = new FileWriter(filePath+".json");
    else
        file = new FileWriter(filePath);
    file.write(jobj.toJSONString());
    file.flush();
    file.close();
    if(Path==null) ourPath = filePath+".json";
    System.out.println("save 경로:"+ourPath);
}
```

json파일로 저장하도록 해당 파일을 만들어 두고 그 파일에 우리가 특정한 태그에 맞는 정보로 들어가게 해두었습니다.

실행과정과 소스코드 소개



해당 라벨과 텍스트 정보들은 json형식의 파일로 불러올 수 있습니다.

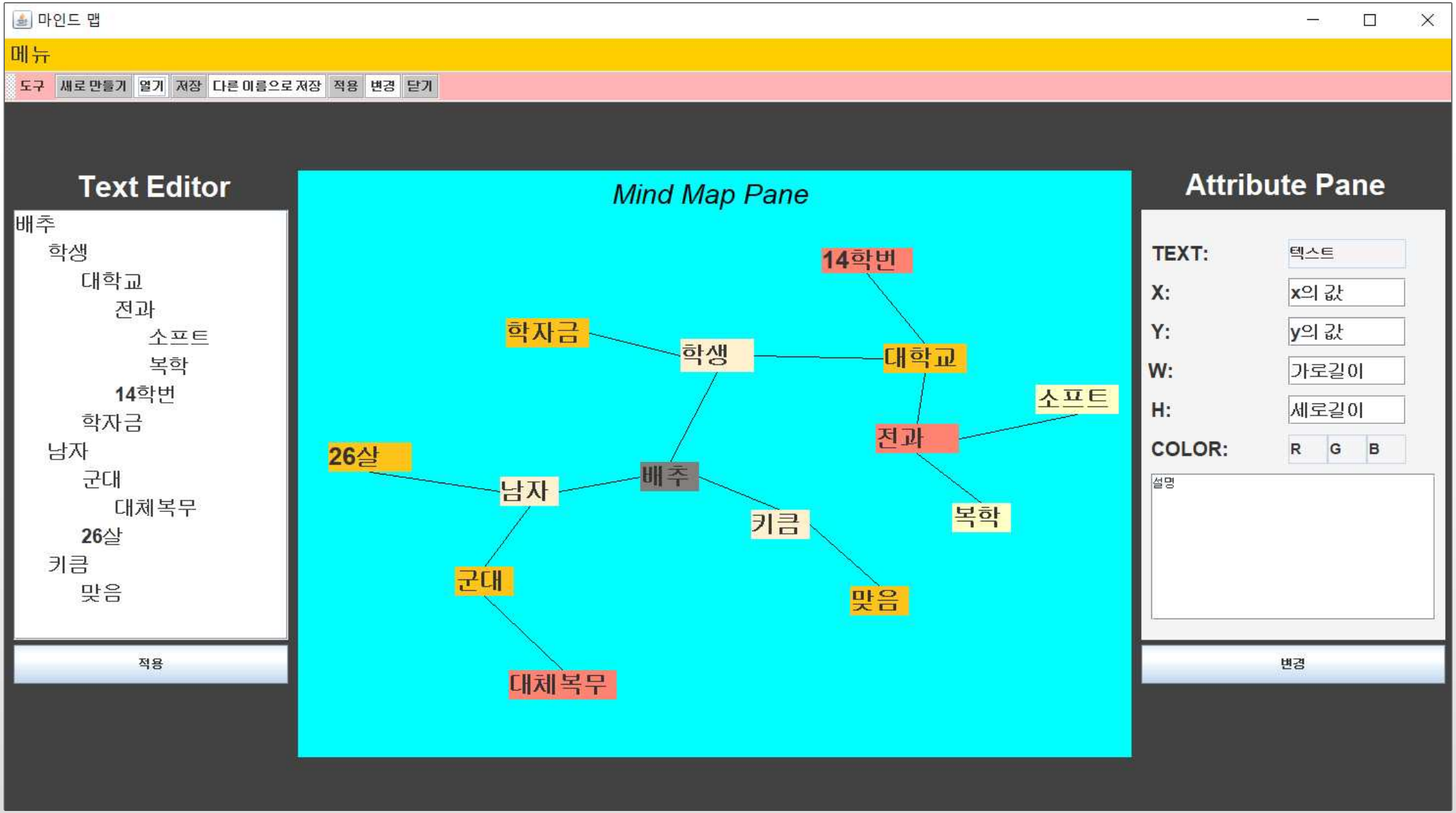
```
// path 경로의 파일을 읽어와 Object로 파싱
Object obj = parser.parse(new FileReader(path));
System.out.println("load의 경로"+path);
JSONObject jsonObject =(JSONObject) obj;

// list가져오기
JSONArray xList =(JSONArray) jsonObject.get("X");
JSONArray yList =(JSONArray) jsonObject.get("Y");
JSONArray wList =(JSONArray) jsonObject.get("W");
JSONArray hList =(JSONArray) jsonObject.get("H");
JSONArray tList =(JSONArray) jsonObject.get("Text");
JSONArray cList =(JSONArray) jsonObject.get("Count");
JSONArray lengthList =(JSONArray) jsonObject.get("tLength");
JSONArray jsonlength =(JSONArray) jsonObject.get("Totalnum");
JSONArray EditText =(JSONArray) jsonObject.get("EditText");
JSONArray explainText =(JSONArray) jsonObject.get("explainText");
```

json파일에서 Load를 할때 해당 태그 정보를 위와 같은 정보로 받아와서 필요한 객체에게 정보를 넘겨줍니다.

```
if(mTree.setNode()) {
    EditorText = nowTextArea;
    for(int i=0; i<num; i++)
        mTree.setXYWH(i, x[i], y[i], w[i], h[i], explain[i]);
    mTree.MakeJLabelNode();
    mMapPane.setMyJLabel();
    mMapPane.repaint();

    tEditor.setText(nowTextArea);
    atrPane.tfield.setText("텍스트");
    atrPane.xfield.setText("x의 값");
    atrPane.yfield.setText("y의 값");
    atrPane.wfield.setText("가로길이");
    atrPane.hfield.setText("세로길이");
    atrPane.rfield.setText("R");
    atrPane.gfield.setText("G");
    atrPane.bfield.setText("B");
    atrPane.atrText.setText("설명");
    ourPath = path;
}
```

넘겨받은 정보를 토대로 해당 마인드맵을 그려주고 해당 정보로 텍스트도 그려줍니다.

실행화면 전체


마인드 맵
메뉴
도구
새로 만들기
열기
저장
다른 이름으로 저장
적용
변경
닫기
Text Editor
배추
학생
대학교
전과
소프트
복학
14학번
학자금
남자
군대
대체복무
26살
키큼
맞음
적용
Mind Map Pane
14학번
학자금
학생
대학교
소프트
전과
26살
배추
남자
키큼
복학
군대
맞음
대체복무
Attribute Pane
TEXT:
텍스트
X:
x의 값
Y:
y의 값
W:
가로길이
H:
세로길이
COLOR:
R
G
B
설명
변경