[객체지향 프로그래밍]

[나만의 영어 단어장 만들기]

과제 체크리스트	확인
1. 이 과제물은 내가 직접 연구하고 작성한 것이다.	예
2. 인용한 모든 자료(책, 논문, 보고서, 인터넷 자료 등)의 출처를 밝혔다.	예
3. 정확한 출처 제시나 사용 허락 없이 다른 사람의 아이디어를 가져오지 않았다.	예
4. 이 과제물을 다른 사람으로부터 받거나 구매하여 제출하지 않았다.	예
5. 이 과제의 결과물에 나의 학번 이름이 출력되어 있다.	예

■ 담당 교수 : 지정희 교수님

■ 학 번 : 202010392

■ 이 름 : 신희곤

■ 제출일 : 2021년 12월 6일

1 문제정의 및 분석

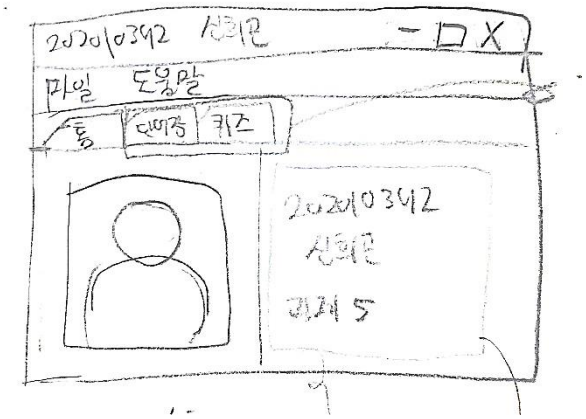
1) 개요

이번 과제의 문제는 과제 4에서 제출한 영어 단어장 프로그램을 콘솔이 아닌 스윙을 기반으로 구현하는 것이다. 먼저, GUI를 어떻게 만들지 구상을 해야할 것이다. 그 후 GUI를 실제로 구현하고 과제 4에서 만든 기능들을 GUI에서도 돌아가도록 적절히 수정작업을 거치면 될 것이다.

1. 객체는 오직 하나의 책임을 가져야 한다.
2. 객체는 확장에 대해서는 개방적이고 수정에 대해서는 폐쇄적이어야 한다.
3. 자식 클래스는 언제나 자신의 부모 클래스를 대체할 수 있다.
4. 범용 인터페이스 하나보다는 특정 클라이언트를 위한 여러 개의 인터페이스 분리가 더 좋다.
5. 프로그래머는 구체화가 아니라 추상화에 의존해야 한다고 한다.

분석 하기 앞서, 이번 과제를 수행하면서 '최대한 객체지향적으로 설계해보자'는 자세로 임했음을 밝히고 싶다. 위의 객체지향 5원칙을 적어두고, 최대한 원칙을 지켜가며 설계하도록 노력했다.

2) GUI 구상



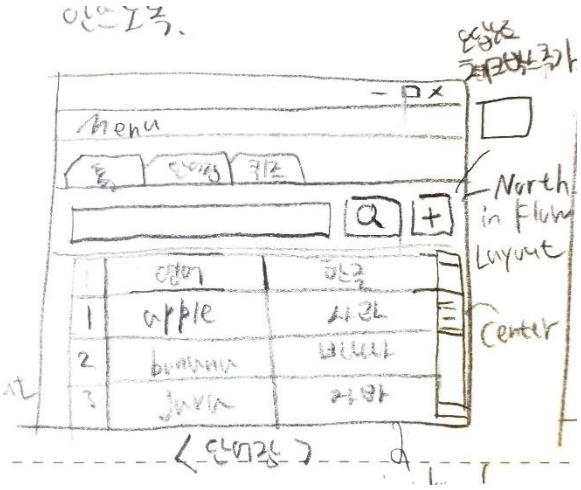
먼저 **메인 프레임**은 크게 **메뉴**와 **Tab Pane** 부분으로 구성했다. 과제 4에서 콘솔로 출력되는 메뉴는 '1) 단어검색 2) 객관식 퀴즈 3) 종료' 다. 따라서 내 사진과 학번이름을 띄울 홈 패널과, 단어리스트를 보여주고 단어검색을 수행할 단어장 패널, 객관식 퀴즈를 진행할 퀴즈 패널 세개가 필요하다. 그러므로 이 세개의 패널을 JTabbedPane으로 묶어서 관리하기로 구상했다.

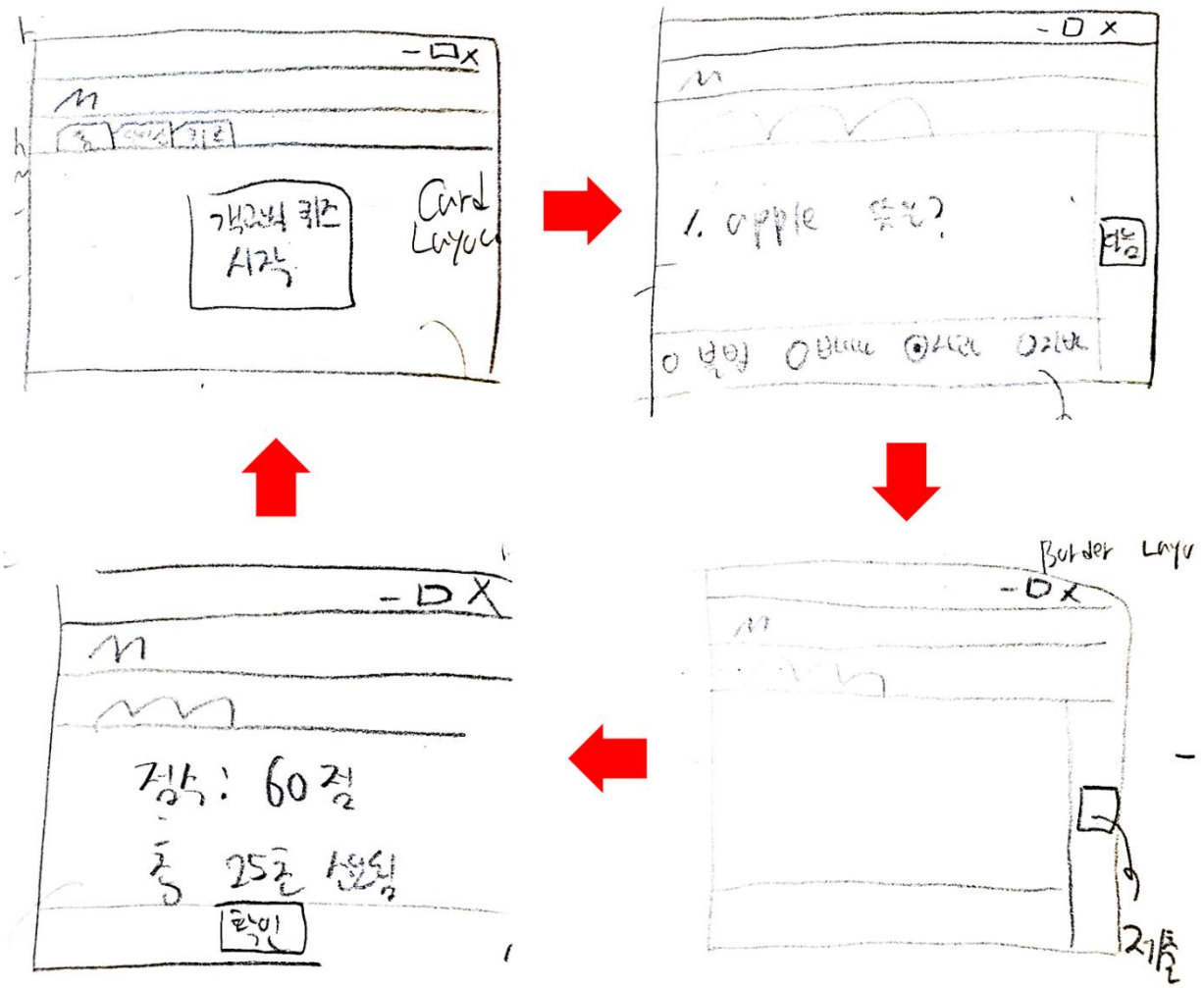
단어장이 생성되지 않았다면 단어장과 퀴즈패널은 이용할 수 없으므로, 이경우 두 tab은 비활성화 처리를 해야할 것이다.

또한 오른쪽 그림의 **홈 패널**은 GridLayout 1x2를 선택해 왼쪽에는 내 사진, 오른쪽은 학번이름을 출력하도록 구성했다.

단어장 패널은 다음과 같이 구성했다. 전체적인 패널의 레이아웃은 BorderLayout으로, 북쪽엔 다양한 기능들을 수행하는 컴포넌트를 배치한 패널이 들어간다. 센터에는 단어 리스트를 보여주는 Jtable을 넣을 계획이다.

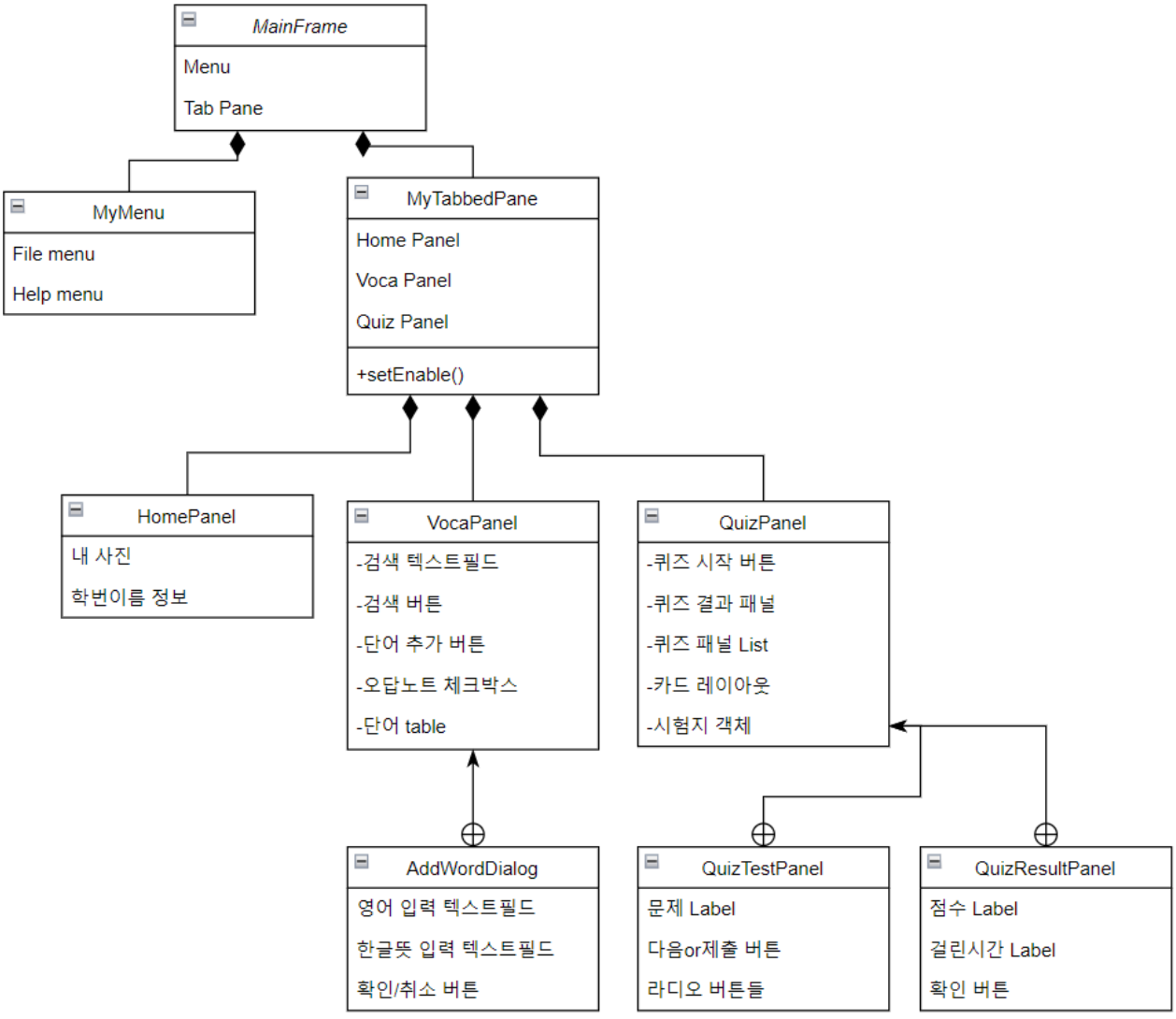
북쪽의 패널에는 단어를 검색하는 TextField와 검색버튼, 단어 추가 버튼, 아래 Table을 틀린순으로 정렬하는 체크박스가 필요하다고 생각하여 이와 같이 구성했다. 특히 단어 추가 버튼을 누르면, 영어와 뜻을 입력하는 다이얼로그를 띄워서 추가할 단어를 입력받을 생각이다.



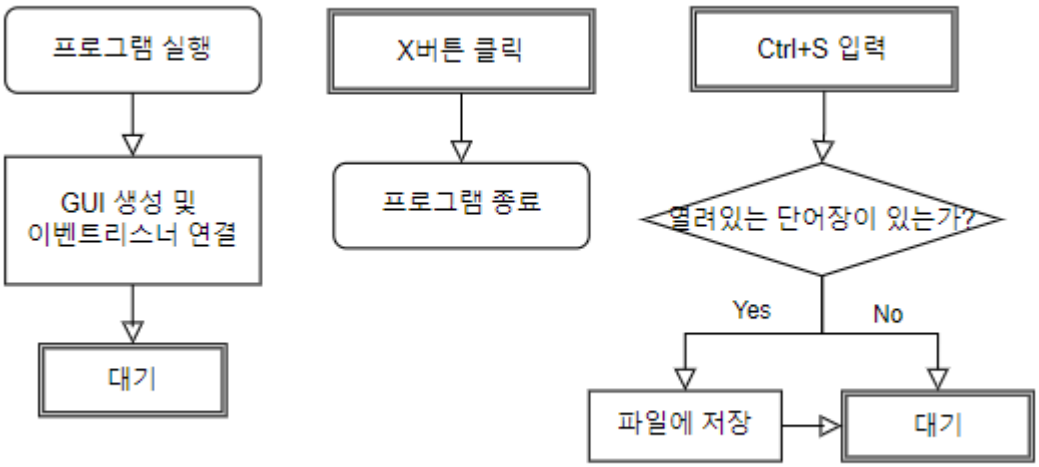


퀴즈 패널은 간단히 퀴즈 시작버튼 하나만 존재하며, 시작 버튼을 누르면 패널이 바뀌며 퀴즈가 시작된다. 여러 패널을 관리하기 위해 CardLayout을 채택했다. 퀴즈를 수행할 때마다 퀴즈 내용이 달라지므로, 퀴즈가 시작되고 종료될 때 패널을 동적으로 생성하여 추가하고 삭제하도록 제작할 계획이다. 마지막 퀴즈 패널은 '다음' 버튼 대신 '제출' 버튼이며, 제출 버튼을 누를 시 결과 화면으로 넘어가 결과를 보여준다. 결과 화면에서 확인 버튼을 누르면 전에 수행했던 패널들을 지워 초기화하고 처음 패널로 돌아온다.

즉, GUI 컨테이너들의 관계를 클래스 다이어그램로 정리하면 다음과 같다.

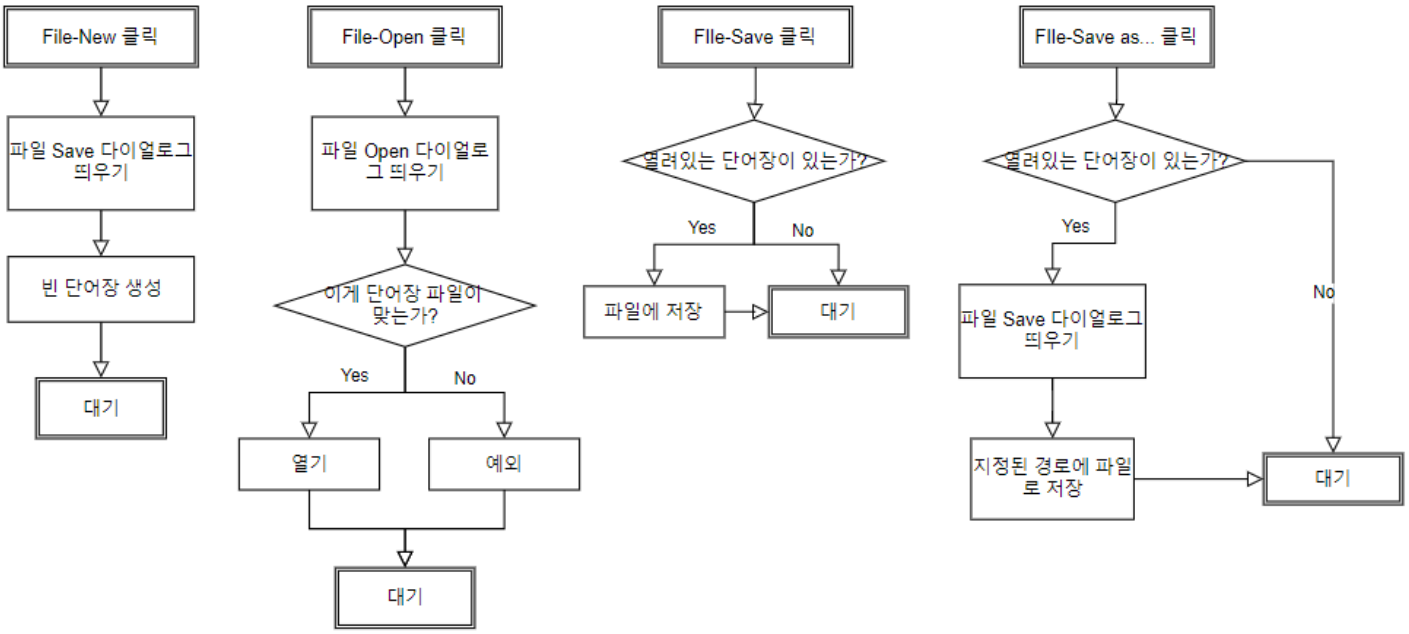


3) 프로그램 흐름



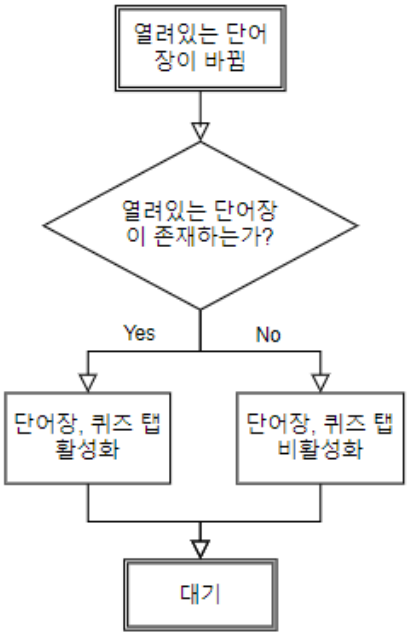
기본적으로 실행하면 GUI를 생성하고 대기한다. 도중 버튼 클릭 등의 이벤트가 발생하면 관련 동작을 실행하는 이벤트 프로그래밍을 기반으로 설계했다. 또한 Ctrl+S를 누르면 단어장이 저장이 되도록 설계했다. 단어를 추가하거나 퀴즈 진행 후 단어장을 저장하면 프로그램을 종료하고 다시 실행해도 단어 틀린 횟수나 추가된 단어가 존재한다.

(1) 메뉴에 연결된 Event Listener



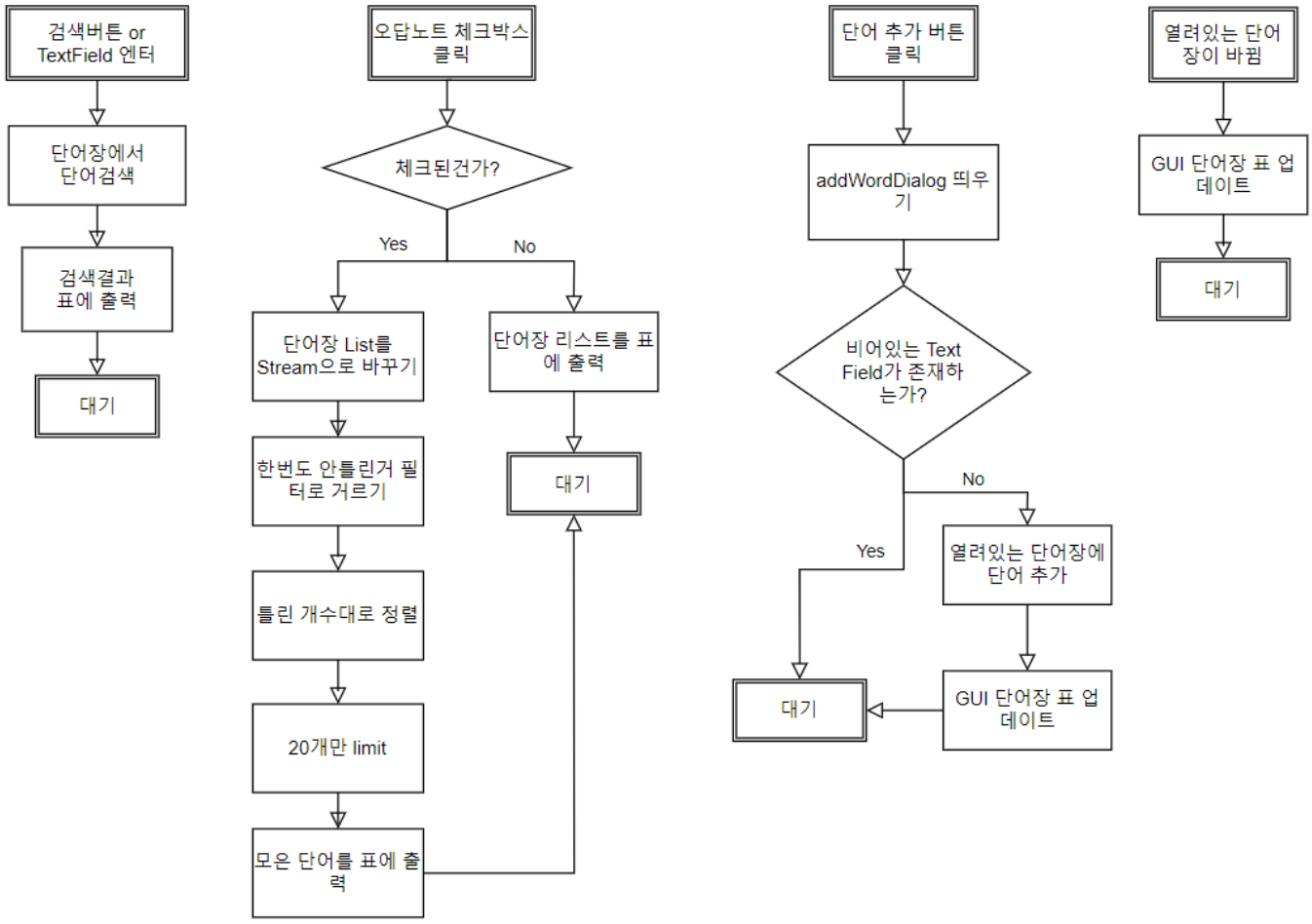
메뉴 부분에선 단어장을 생성하거나 기존의 단어장을 열고, 저장하는 기능을 지원한다.

(2) Tab Pane에 연결된 Event Listener



열려있는 단어장이 없다면 단어장과 퀴즈 탭에 접근을 못하도록 비활성화 해야하고, 단어장을 열었다면 활성화해야 한다. 이를 처리한 방법으로 임의로 열려있는 단어장이 바뀌었을 때 이벤트를 만들었다. 그후 Tab을 관리하는 TabPane 객체에 이벤트 리스너를 달아주는 형식으로 구현하였다.

(3) 단어장 탭에 연결된 Event Listener

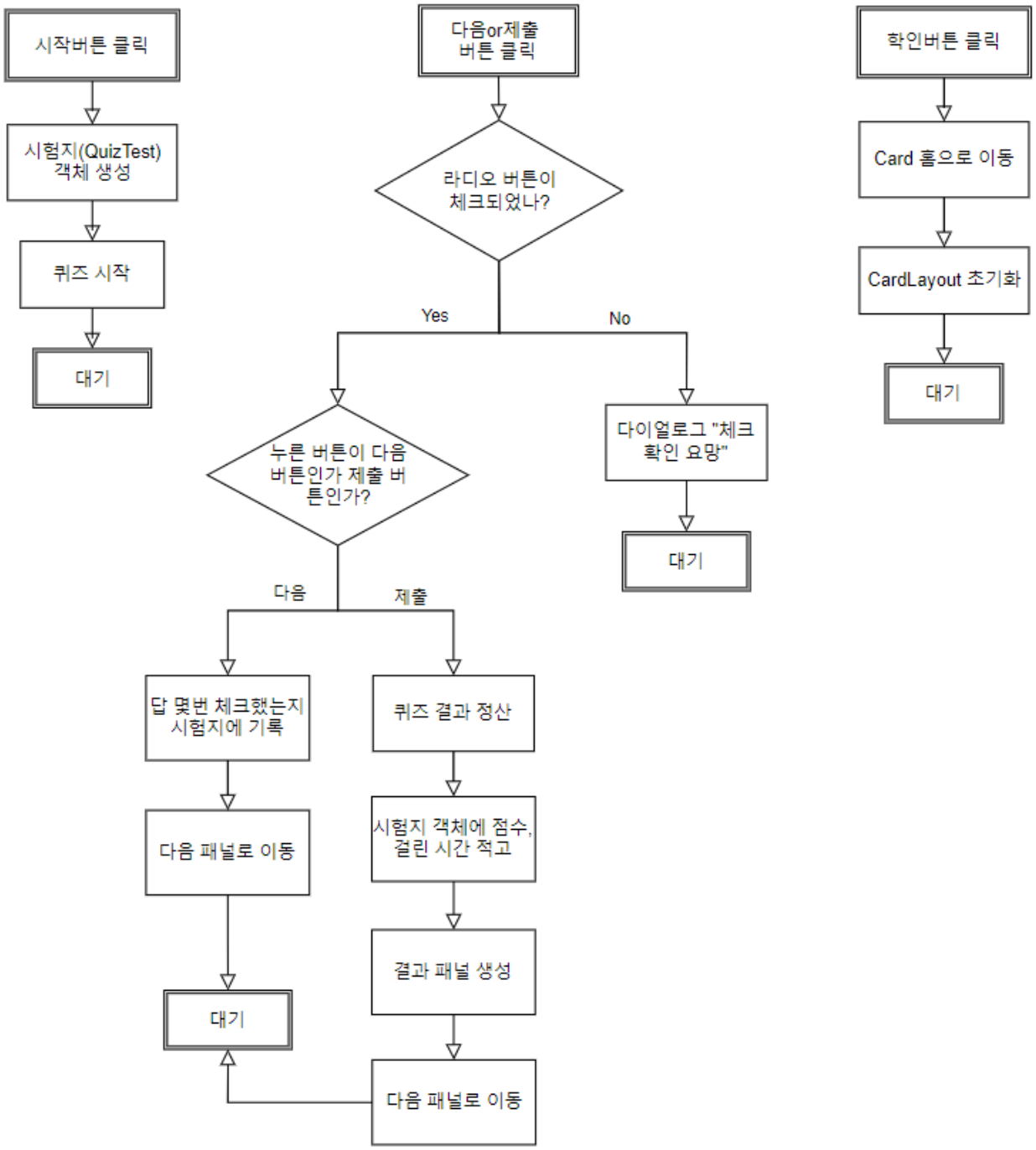


단어장 검색 기능은, 현재 열려있는 단어장에서 해당 단어를 검색하고, 그 결과를 JTable GUI에 출력한다. 오답노트는 단어장 List를 Stream으로 변환해서 정렬과 limit 동작을 수행했다. 체크박스를 해제하면 다시 모든 단어리스트를 JTabel에 출력한다.

단어 추가 버튼을 누를시 단어를 추가하는 다이얼로그가 뜨며, 내용을 전부 채웠을경우 단어를 추가하고 JTable을 업데이트한다.

메뉴에서 New, Open 등의 동작을 수행해서 열려있는 단어장이 바뀌었을 경우 JTable의 값 또한 새로운 단어장의 내용으로 바뀌어야 한다. 이벤트를 이용해 자동으로 표가 업데이트되도록 구현하였다.

(4) 퀴즈 탭에 연결된 Event Listener



퀴즈를 시작하면 QuizTest라는 이름의 빈 '시험지' 객체를 생성한다. 그 후 퀴즈를 실행한다. 퀴즈를 실행하면 문제 10개를 단어장에서 랜덤추출해 시험지에 등록하고, CardLayout에 패널을 추가한 뒤 타이머를 시작해야 한다.

답을 체크하고 '다음' 버튼을 누르면 시험지에 몇번을 체크했는지 기록해야 한다. '제출' 버튼을

누르면 답을 채점하고, 결과를 시험지에 적는 과정이 있어야 한다. 시험이 종료되면 동적으로 생성된 CardLayout의 패널을 제거해서 전에 시험봤던 정보를 초기화 해주어야 한다.

2 주요 소스코드 설명

```
public static void main(String[] args) {
	new MainFrame();
}
```

실행시 MainFrame 객체를 생성한다.

```
public class MainFrame extends JFrame {
	MyMenu menu = new MyMenu();
	MyTabbedPane tapPane = new MyTabbedPane();

	public MainFrame() {
		this("202010392 신희곤");
	}

	public MainFrame(String title) {
		super(title);
		this.setSize(540,500);
		this.setLocationRelativeTo(null); // 창을 중앙에 띄우기
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);
		init();
		this.setVisible(true);
	}
```

MainFrame 생성자 내용은 위와 같다. 객체가 생성됨과 동시에 필드에서 바로 MyMenu와 MyTabbedPane 객체를 생성한다.

```
public void init() {
	Container frame = this.getContentPane();
	frame.add(tapPane);
	this.setJMenuBar(menu);
```

init() 함수에선 생성된 메뉴와 tabPane을 달아준다.

```
	this.getRootPane().registerKeyboardAction(new ActionListener() {

		@Override
		public void actionPerformed(ActionEvent e) {
			try {
				save();
			} catch (NullPointerException ex) {
				JOptionPane.showMessageDialog(null, "생성된 단어장이
```

```
없습니다.", "Message", JOptionPane.WARNING_MESSAGE);
                        }
                }

    }, KeyStroke.getKeyStroke(KeyEvent.VK_S,KeyEvent.CTRL_DOWN_MASK),
JComponent.WHEN_IN_FOCUSED_WINDOW);

        this.addWindowListener(new WindowAdapter() {

                @Override
        public void windowClosing(WindowEvent e) {
                        try { save(); } catch (NullPointerException ex) {}
        }

        });
```

프레임 부분에서 Ctrl+S 키를 누르거나 프로그램을 종료하면 단어장이 저장되도록 이벤트리스너를 추가했다.

Save() 함수의 내용은 다음과 같다.

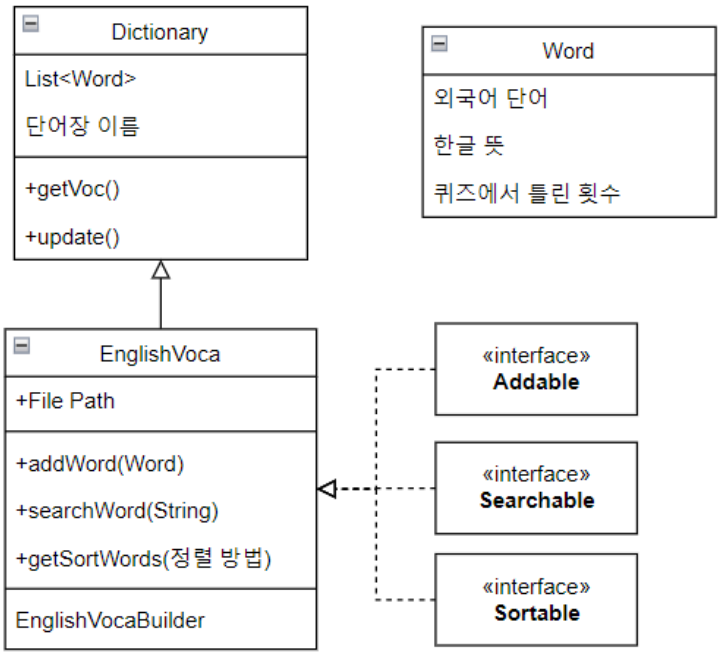
```
public static void save() throws NullPointerException {
        EnglishVoca dictionary = (EnglishVoca)
OpenedDictionary.getInstance();
        DictionaryToFileConverter converter = new
DictionaryToFileConverter(dictionary, dictionary.filePath);
        converter.convert();
}
```

현재 열려있는 단어장을 가져오고,
단어장 객체를 파일 객체로 변환해주는 컨버터 객체를 생성한 뒤
파일로 변환해서 저장하는 코드이다.

이때 사용되는 객체는 EnglishVoca, OpenedDictionary, DictionaryToFileConverter 세가지로, 각 객체가 갖는 역할을 설명하고자 한다.

- EnglishVoca 객체

영어 단어장 객체이며, 단어장 이라는 추상클래스를 상속하고 있다.



단어장은 기본적으로 반드시 단어 목록과 단어장 이름을 가져야 의미가 있다. 그리고 단어리스트를 가져오는 메소드와, 이 단어장 객체를 현재 열려있는 단어장 객체로 업데이트 하는 기능은 모든 단어장에 필요하므로 Dictionary 추상클래스에 정의해두었다.

```
public abstract class Dictionary {
	protected List<Word> voc = new ArrayList<>();
	protected String dictionaryName;

	public Dictionary(List<Word> voc, String dictionaryName) {
		super();
		this.voc = voc;
		this.dictionaryName = dictionaryName;
	}

	public List<Word> getVoc() {
		return voc;
	}

	public void update() {
		OpenedDictionary.setInstance(this);
	}

	static interface Addable {
		public void addWord(Word word);
	}

	static interface Searchable {
		public List<Word> searchWord(String str);
	}

	static interface Sortable {
		public List<Word> getSortWords(Comparator<? super Word> comparator);
	}
}
```

단어장에 필요한 기능은 단어를 추가하는 기능, 단어를 검색하는 기능, 단어를 정렬하는 기능이 필요할 것 같아 단어장 클래스의 내부 인터페이스로 정의해두었다.

EnglishVoca는 단어장 중 영어단어장을 의미하는 객체이며, 단어를 add, search, sort하는 기능을 구현하고 있다.

```
public class EnglishVoca extends Dictionary implements Addable, Searchable, Sortable {
	public String filePath;

	private EnglishVoca(EnglishVocaBuilder builder) {
		super(builder.voc, builder.dictionaryName);
		this.filePath = builder.filePath;
	}
```

단어장 생성부는 빌더 패턴을 적용시켜 보았다. EnglishVocaBuilder 객체를 통해야만 EnglishVoca 객체를 생성할 수 있다. EnglishVocaBuilder는 EnglishVoca 클래스 내부에 정적 필드로 정의되어있다.

```
	public static class EnglishVocaBuilder {
		private List<Word> voc = new ArrayList<>();
		private String dictionaryName;
		private String filePath;

		public EnglishVocaBuilder() {}
		public EnglishVocaBuilder setVoc(List<Word> voc) {
			this.voc = voc;
			return this;
		}

		public EnglishVocaBuilder setDictionaryName(String dictionaryName) {
			this.dictionaryName = dictionaryName;
			return this;
		}

		public EnglishVocaBuilder setFilePath(String filePath) {
			this.filePath = filePath;
			return this;
		}

		public EnglishVoca build() {
			return new EnglishVoca(this);
		}

	}
```

이하 EnglishVoca에서 구현한 메소드들을 설명한다.

```
	@Override
	public void addWord(Word word) {
```

```
            voc.add(word);
        }
```

단어를 추가하는 메소드이다. 단어리스트에 단어를 추가한다.

```
        @Override
        public List<Word> searchWord(String str) {
            List<Word> list = new ArrayList<>();
            str = str.trim();
            for (Word word : voc) {
                if (word != null) {
                    if (word.eng.contains(str)) {
                        list.add(word);
                    }
                }
            }

            return list;
        }
```

단어를 검색하는 메소드이다. 단어 리스트를 전부 탐색해 str를 포함하는 영단어가 있다면 리스트에 추가한다. 반복이 끝나면 그 리스트를 반환한다.

```
        @Override
        public List<Word> getSortWords(Comparator<? super Word> comparator) {
            return voc.stream().sorted(comparator).collect(Collectors.toList());
        }
```

단어들을 정렬하는 메소드이다. 함수형 인터페이스를 입력받아 스트림으로 정렬 후 다시 리스트로 변환해 반환한다.

- OpenedDictionary 객체

```
public class OpenedDictionary {
        private static Dictionary dictionary = null;

        private OpenedDictionary() {}

        public static Dictionary getInstance() {
            return dictionary;
        }

        public static void setInstance(Dictionary dic) {
            dictionary = dic;
            if (dic != null)
                OpenedChangeListener.change();
        }
}
```

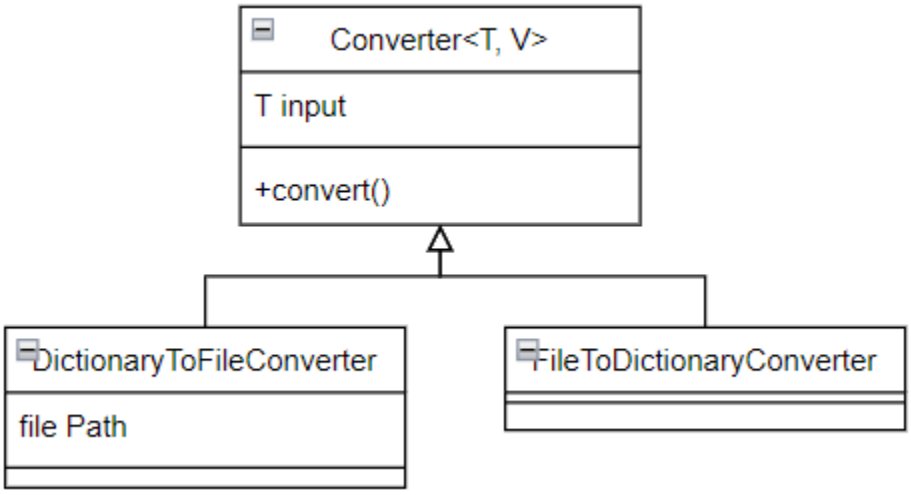
현재 열려있는 단어장을 저장하는 싱글톤 객체다. 어디서든 현재 펼쳐놓은 단어장이 무엇인지 볼 수 있어야 하므로 정적 필드, 정적 메소드로 구현한다.

setInstacne는 다른 단어장을 펼치겠다는 메소드로, 이 메소드가 실행시 열려있는 단어장이 변경되었다는 이벤트가 발생한다. 아래는 OpenedChangeListener의 내용이다.

```
public class OpenedChangeListener {
	private static List<OnOpenedChangeListener> linstener = new ArrayList<>();

	private OpenedChangeListener() {}

	public static void change() {
		for (OnOpenedChangeListener lis : OpenedChangeListener.linstener)
			lis.onOpenedChange();
	}

	public static void addOnOpenedChangeListener(OnOpenedChangeListener listener) {
		OpenedChangeListener.linstener.add(listener);
	}

	public static interface OnOpenedChangeListener {
		void onOpenedChange();
	}
}
```

- DictionaryToFileConverter 객체

단어장 객체를 파일로 변환하는 컨버터 객체이다. 파일을 단어장으로 불러오는 FileToDictionaryConverter 객체도 존재하며, 둘다 Converter 추상클래스를 상속받는다.



- Converter 코드

```
public abstract class Converter<T, V> {
	protected T input;

	public Converter(T input) {
		super();
		this.input = input;
	}
```

```
        public abstract V convert();
}
```

Converter 객체의 명명 규칙은 'TToVConverter'로 정의한다.
Convert() 메소드는 필드에 T타입으로 저장된 객체를 변환해서 V 객체로 출력하는 동작을 담당한다.

단어장을 파일로 변환할 때, 파일객체를 만들기 위해선 파일을 저장할 경로가 필요하므로 특별히 DictionaryToFileConverter 객체는 파일 경로를 저장하는 String 변수를 필드로 가진다.

```
public class DictionaryToFileConverter extends Converter<Dictionary, File> {
        private final String pathName;

        public DictionaryToFileConverter(Dictionary dictionary, String pathName) {
                super(dictionary);
                this.input = dictionary;
                if (!pathName.contains(".txt"))
                        pathName = pathName + ".txt";
                this.pathName = pathName;
        }
```

생성자에서 파일 뒤에 .txt 확장자가 없을 경우 자동으로 .txt가 추가되도록 설계했다.

```
        public File convert() {
                File file = new File(pathName);
                List<Word> voc = input.getVoc();

                try {
                        file.createNewFile();
                    FileWriter fw = new FileWriter(file);
                    BufferedWriter bw = new BufferedWriter(fw);
                    PrintWriter pw = new PrintWriter(bw);
                    for (Word word : voc) {
                        pw.println(word);
                    }

                    pw.flush();
                    pw.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }

                return file;
        }
}
```

객체를 변환하는 내용을 구현한 메소드이다. Writer 스트림을 생성 후 보조스트림 두개를 붙였다. 그 후 input에 저장된 단어장의 단어들을 파일에 출력한다. 그후 스트림을 닫고, 파일 객체를 리턴한다.

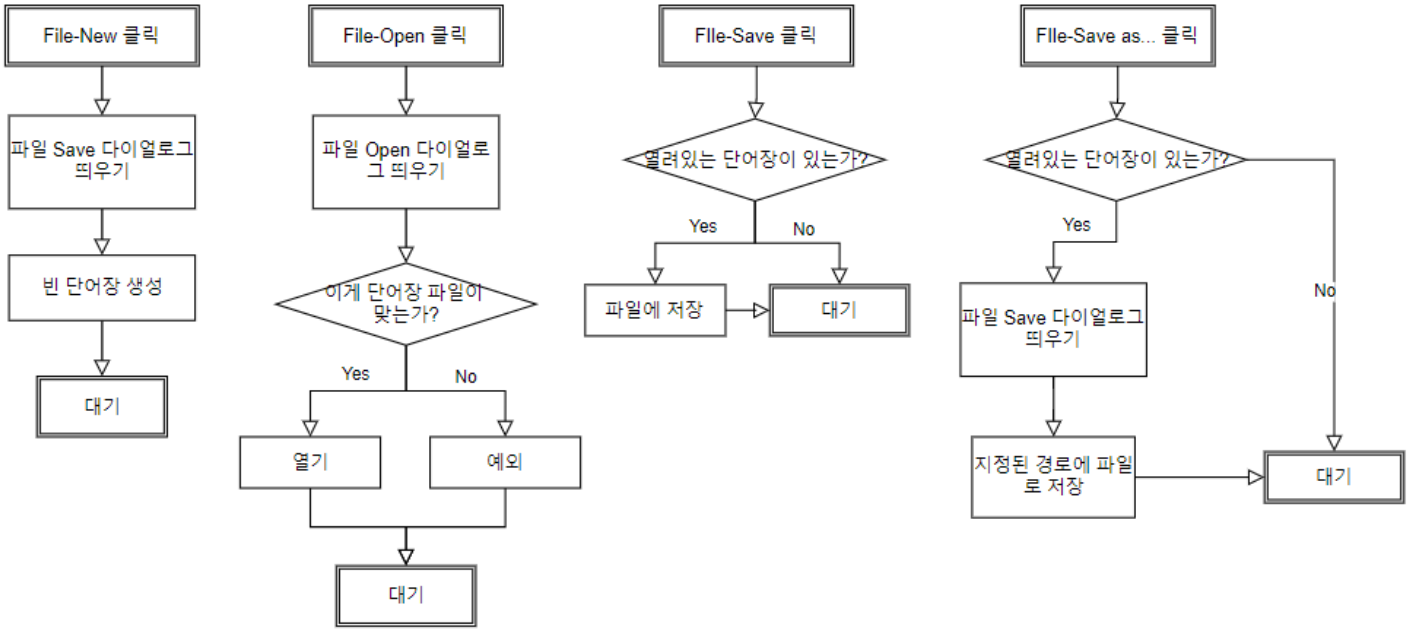
Word 객체에서 toString()를 오버라이딩했기 때문에 println 구문에서 word를 바로 인자로 넘겼다.

```
	@Override
	public String toString() {
		return eng+"\t"+kor+"\t"+wrongCount;
	}
```

다시 본론인 GUI 생성으로 돌아와서, 프레임에서 생성된 MyMenu 객체와 MyTabbedPane 객체를 살펴보겠다.

```
public class MyMenu extends JMenuBar {
	JMenu fileMenu, helpMenu;

	public MyMenu() {
		fileMenu = new JMenu("파일");

		JMenuItem create = new JMenuItem("새로 만들기");
		JMenuItem open = new JMenuItem("열기");
		JMenuItem save = new JMenuItem("저장");
		JMenuItem saveas = new JMenuItem("다른 이름으로 저장...");

		create.addActionListener(new MenuActionListener());
		open.addActionListener(new MenuActionListener());
		save.addActionListener(new MenuActionListener());
		saveas.addActionListener(new MenuActionListener());

		fileMenu.add(create);
		fileMenu.add(open);
		fileMenu.addSeparator();
		fileMenu.add(save);
		fileMenu.add(saveas);

		helpMenu = new JMenu("도움말");
		helpMenu.add(new JMenuItem("미구현"));

		this.add(fileMenu);
		this.add(helpMenu);
	}
```

먼저 메뉴 객체 부분이다. 메뉴 아이템을 생성해 액션리스트를 연결한다.




```
class MenuActionListener implements ActionListener {

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

actionPerformed 함수의 내용은 다음과 같다.

```
	String cmd = e.getActionCommand();

	if (cmd.equals("새로 만들기")) {
		JFileChooser chooser = new JFileChooser();
		FileNameExtensionFilter filter = new FileNameExtensionFilter(
				"TXT Files",
				"txt");
		chooser.setFileFilter(filter);
		int ret = chooser.showSaveDialog(null);

		if (ret == JFileChooser.APPROVE_OPTION) {
			String pathName = chooser.getSelectedFile().getPath();
			String fileName = chooser.getSelectedFile().getName();

			EnglishVoca voca = new EnglishVoca.EnglishVocaBuilder()
					.setVoc(new ArrayList<Word>())
					.setDictionaryName(fileName)
					.setFilePath(pathName)
					.build();
```

```
                        DictionaryToFileConverter converter = new
DictionaryToFileConverter(voca, pathName);
                        converter.convert();
                        OpenedDictionary.setInstance(voca);
                }
        }
```

새로 만들기 버튼을 클릭시, 파일을 저장할 다이얼로그를 띄운다.
정상적인 경로를 받아왔다면, 영어 단어장 객체를 생성한다.
그리고 단어장을 파일로 변환하는 컨버터를 생성 후 컨버트한다.
그 후 열려있는 단어장을 새로 만든 단어장으로 변경한다.

```
        else if (cmd.equals("열기")) {
                try {
                        JFileChooser chooser = new JFileChooser();
                        FileNameExtensionFilter filter = new FileNameExtensionFilter(
                                        "TXT Files",
                                        "txt");
                        chooser.setFileFilter(filter);
                        int ret = chooser.showOpenDialog(null);

                        if (ret == JFileChooser.APPROVE_OPTION) {
                                String pathName = chooser.getSelectedFile().getPath();

                                FileToDictionaryConverter converter = new
FileToDictionaryConverter(new File(pathName));
                                OpenedDictionary.setInstance(converter.convert());
                        }
                } catch (OpenFileException ex) {
                        JOptionPane.showMessageDialog(null, "올바른 파일의 형식이 아닙니다.",
"Message", JOptionPane.WARNING_MESSAGE);
                        return;
                }
        }
```

열기 버튼을 누를 경우 파일 Open 다이어로그를 띄운다.
정상적인 경로를 받아왔다면, 파일을 단어장으로 변환하는 컨버터를 생성한다.
그 후 단어장으로 변환하고, 변환된 단어장을 펼친다. (= 열려있는 단어장을 변환한 단어장으로 변경한다.)
만약 변환과정 도중 OpenFileException이 발생하면, 즉 오픈한 txt 파일이 단어장 형식이 아니라면 예외처리를 해주었다.

FileToDictionaryConverter의 내용은 다음과 같다.

```
public class FileToDictionaryConverter extends Converter<File, Dictionary> {
        public FileToDictionaryConverter(File file) {
                super(file);
        }
```

컨버터를 상속받으며 필드에는 따로 정의한 변수는 없다.

```
	public Dictionary convert() {
		List<Word> voc = new ArrayList<>();

		try {
			FileInputStream fi = new FileInputStream(input);
			EncodeFinder encode = new EncodeFinder(input);
			Scanner scan = new Scanner(fi, encode.find());
```

파일인풋스트림 객체를 생성하고, EncodeFinder 객체를 생성한다. 이 객체는 파일의 인코딩 형식(ASNI, UTF-8)이 무엇인지 찾아내는 객체다.
스캐너 객체를 만들어서 이 스캐너 객체로 파일을 읽어낸다.

```
			while (scan.hasNextLine()) {
				String str = scan.nextLine();
				String[] temp = str.split("\t");
				if (temp.length == 2)
					voc.add(new Word(temp[0].trim(),temp[1].trim()));
				else
					voc.add(new Word(temp[0].trim(),temp[1].trim(),
Integer.parseInt(temp[2].trim())));
			}
```

라인 단위로 불러와 \t을 기준으로 끊어낸다. 단어장 txt 파일이 갖는 한 라인의 규칙은

```
	considerable	상당한	0
```

이지만,

```
	considerable	상당한
```

도 단어장 파일로 인식할 수 있도록 따로 if문을 만들어 처리해주었다.

```
			EnglishVoca voca = new EnglishVoca.EnglishVocaBuilder()
					.setVoc(voc)
					.setDictionaryName(input.getName())
					.setFilePath(input.getPath())
					.build();

			scan.close();
			return voca;
```

그 후 단어장 객체를 생성하여 인자로 단어리스트, 단어장이름, 파일 경로를 설정한다.
마지막으로 스캐너 객체를 닫는다.

```
		} catch(ArrayIndexOutOfBoundsException | NumberFormatException e) {
			throw new OpenFileException("fail open dictionary");
		} catch (IOException e) {
			JOptionPane.showMessageDialog(null, e.getMessage(), "Message",
JOptionPane.WARNING_MESSAGE);
			return null;
		}
	}
```

만약 \t 기준으로 끊어도 temp 배열의 길이가 충분하지 않거나, temp[2]에 숫자가 아닌 문자가

온다면, 그것은 단어장 형식의 파일이 아니라는 뜻이다. 따라서 예외를 받아 OpenFileException 를 던진다.

다시 메뉴의 이벤트 리스너로 돌아와 다음 내용을 살펴보겠다.

```
	else if (cmd.equals("저장")) {
		try {
			MainFrame.save();
		} catch (NullPointerException ex) {
			JOptionPane.showMessageDialog(null, "생성된 단어장이 없습니다.",
"Message", JOptionPane.WARNING_MESSAGE);
		}
	}
```

저장 버튼을 누르면 MainFrame 클래스의 save 함수를 실행한다.
만약 변환과정 도중 NullPointerException가 발생하면 열려있는 단어장이 없다는 의미이므로 다이얼로그를 띄워 예외처리를 해준다.

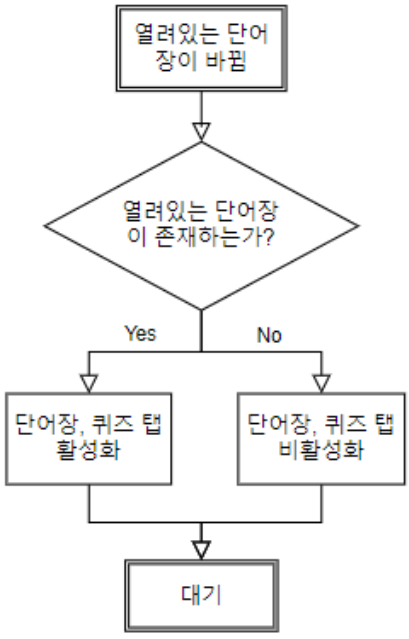
```
	else if (cmd.equals("다른 이름으로 저장...")) {
		try {
			EnglishVoca dictionary = (EnglishVoca)
OpenedDictionary.getInstance();
			JFileChooser chooser = new JFileChooser();
			FileNameExtensionFilter filter = new FileNameExtensionFilter(
					"TXT Files",
					"txt");
			chooser.setFileFilter(filter);
			int ret = chooser.showSaveDialog(null);

			if (ret == JFileChooser.APPROVE_OPTION) {
				String pathName = chooser.getSelectedFile().getPath();

				DictionaryToFileConverter converter = new
DictionaryToFileConverter(dictionary, pathName);
				converter.convert();
			}
		} catch (NullPointerException ex) {
			JOptionPane.showMessageDialog(null, "생성된 단어장이 없습니다.",
"Message", JOptionPane.WARNING_MESSAGE);
		}
	}
```

마지막으로 다른 이름으로 저장 버튼을 누르면 , 어디에 저장할 것인지 경로를 지정하라는 File Save 다이얼로그를 띄운다. 정상적으로 경로를 받아왔다면,
단어장To파일 컨버터를 생성하여 파일로 저장한다.

메뉴 생성부와 추가한 이벤트 리스너를 살펴봤으니, 이번엔 MyTabbedPane 객체를 만드는 부분을 살펴보자.

```
public class MyTabbedPane extends JTabbedPane {
	HomePanel HomePanel = new HomePanel();
	VocaPanel VocaPanel = new VocaPanel();
	QuizPanel QuizPanel = new QuizPanel();

	public MyTabbedPane() {
		super();

		this.add("홈", HomePanel);
		this.add("단어장", VocaPanel);
		this.add("퀴즈", QuizPanel);
		setEnable();

		OpenedChangeListener.addOnOpenedChangeListener(new
OpenedChangeListener.OnOpenedChangeListener() {

			@Override
			public void onOpenedChange() {
				setEnable();
			}

		});
	}
```

필드로 세 개의 패널을 가지며, 필드에서 바로 생성한다.
그 후 TabPane에 세개의 패널을 부착하고, 단어장과 퀴즈 탭을 비활성화한다.
열려있는 단어장이 바뀔때마다 setEnable() 함수를 실행하도록 리스너를 달아주었다.




```
public void setEnable() {
	if (OpenedDictionary.getInstance() == null) {
		this.setEnabledAt(1, false);
		this.setEnabledAt(2, false);
	} else {
		this.setEnabledAt(1, true);
		this.setEnabledAt(2, true);
	}
}
```

다음은 MyTabbedPane에서 생성된 세 개의 패널, 즉 HomePanel, VocaPanel, QuizPanel 객체를

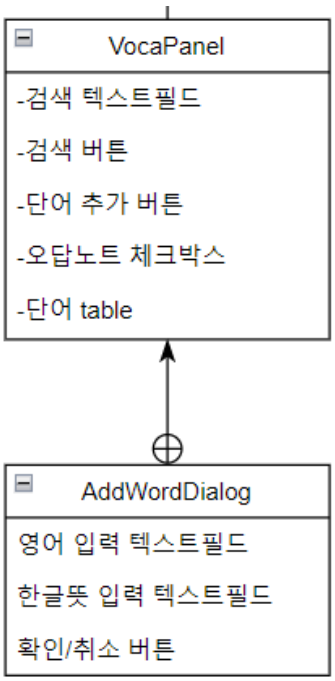
살펴보겠다.

- HomePanel 객체

```
public class HomePanel extends JPanel {
	@java.io.Serial
	private static final long serialVersionUID = -4432412913715438027L;

	private final String imgPath = "img/my_picture.png";
	private final ImageIcon img;

	private JLabel myPicture;
	private JLabel myInfo;

	public HomePanel() {
		this.setLayout(new GridLayout(1, 2, 10, 10));
		img = new ImageIcon(imgPath);
		myPicture = new JLabel(img);
		myPicture.setSize(img.getIconWidth(), img.getIconHeight());

		myInfo = new JLabel("202010392 신희곤");
		myInfo.setHorizontalAlignment(SwingConstants.CENTER);
		myInfo.setFont(new Font("바탕체", Font.BOLD, 24));

		this.add(myPicture);
		this.add(myInfo);
	}
}
```

JPanel을 상속받는 컨테이너 객체다.
홈 패널은 크게 GridLayout 1x2로 구성하였으며, 왼쪽엔 이미지와 오른쪽엔 학번이름을 배치했다.

- VocaPanel 객체




```
public class VocaPanel extends JPanel {
	private JPanel north;
	private JTextField searchField;
	private JButton searchButton;
	private JButton addWordButton;
	private JCheckBox wrongAnswerNoteCB;
	private JTable wordTable;
	private List<TableColumn> columns;
```

VocaPanel의 필드 부분이다.

```
	public VocaPanel() {
		this.setLayout(new BorderLayout());

		north = new JPanel();
		searchField = new JTextField(20);

		searchButton = new JButton("검색");
		addWordButton = new JButton("단어추가");
		wrongAnswerNoteCB = new JCheckBox("오답노트 활성화");
```

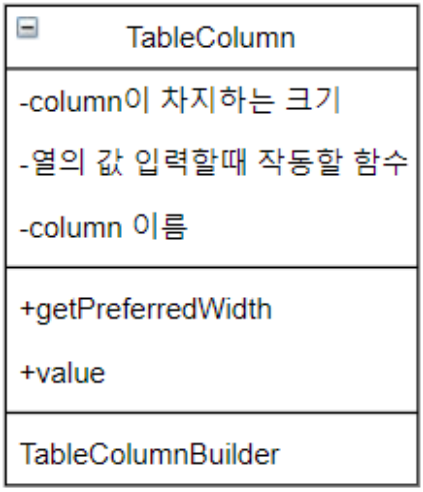
버튼, 텍스트필드를 부착할 north 패널과 각종 컴포넌트를 생성한다.

```
		columns = new ArrayList<>();
		columns.add(new TableColumn.TableColumnBuilder()
				.setName("index")
				.setPreferredWidth(15)
				.setValue((w,m)-
>Integer.toString(m.getRowCount()+1)).build());
		columns.add(new TableColumn.TableColumnBuilder()
				.setName("영어 단어")
				.setPreferredWidth(170)
				.setValue((w,m)->w.eng).build());
		columns.add(new TableColumn.TableColumnBuilder()
				.setName("한글 뜻")
				.setPreferredWidth(170)
				.setValue((w,m)->w.kor).build());
		columns.add(new TableColumn.TableColumnBuilder()
				.setName("틀린 횟수")
				.setPreferredWidth(20)
				.setValue((w,m)-
>Integer.toString(w.getWrongCount())).build());
```

그 후 필드에 있는 List<TableColumn> 리스트에 TableColumn을 생성하여 추가한다.
TableColumn 객체는 awt의 객체가 아닌 임의로 제작한 객체이다.

✓ TableColumn 객체

index	영어 단어	한글 뜻	틀린 횟수
1	at first	처음에는	0
2	castle	성	0
3	consider	고려하다/ 간주하다	0
4	considerable	상당한	2
5	enlighten	계몽시키다 개화시키다	0
6	explain	설명하다	0
7	explanation	설명	0
8	female	여성의	0
9	frankly speaking	솔직히 말하여	0
10	illegal	불법적인	1
11	in the first place	처음으로	0
12	in the place of	~대신에	0
13	intelligent	지적인	0
14	invent	발명하다/ 만들다	0



JTable에서 한 열을 나타내는 객체이다.

열이 차지하는 크기 비율, value, column 이름 정보를 갖는다.

Ex)

width = 180

value = (w, m)->w.eng

name = " 영어 단어"

```
public class TableColumn {
	private final int preferredWidth;
	private ColumnValue value;
	public String name;

	private TableColumn(TableColumnBuilder builder) {
		super();
		this.name = builder.name;
		this.preferredWidth = builder.preferredWidth;
		this.value = builder.value;
	}

	public int getPreferredWidth() {
		return preferredWidth;
	}

	public String value(Word word, DefaultTableModel model) {
		return value.value(word, model);
	}
```

필드의 ColumnValue는 임의로 만든 함수형 인터페이스다.

```
@FunctionalInterface
public interface ColumnValue {
    public String value(Word word, DefaultTableModel model);
}
```

TableColumn은 빌더를 이용해 객체를 생성해야 한다.

```
	public static class TableColumnBuilder {
		private int preferredWidth;
		private ColumnValue value;
		private String name;

		public TableColumnBuilder() {}

		public TableColumnBuilder setPreferredWidth(int preferredWidth) {
			this.preferredWidth = preferredWidth;
			return this;
		}

		public TableColumnBuilder setValue(ColumnValue value) {
			this.value = value;
			return this;
		}

		public TableColumnBuilder setName(String name) {
			this.name = name;
			return this;
		}

		public TableColumn build() {
			return new TableColumn(this);
		}
	}
```

TableColumn 객체를 만들어 관리하는 이유는, VocaPanel 생성부에서 TableColumn 객체를 만들어 리스트에 등록하기만 하면 JTable에 열이 추가되도록 만들기 위함이다. 이를 통해 JTable의 열을 쉽게 추가 or 삭제할 수 있어 유지보수가 간편해진다.

다시 돌아와 VocaPanel 생성자 부분을 이어서 살펴보자.

```
		DefaultTableModel model = new DefaultTableModel();
		for (TableColumn column : columns)
			model.addColumn(column.name);
		wordTable = new JTable(model);
		TableColumnModel columnModel = wordTable.getColumnModel();
		for (int i=0; i<columns.size(); i++)
```

```
        columnModel.getColumn(i).setPreferredWidth(columns.get(i).getPreferredWidth());
        JScrollPane scrollPane = new JScrollPane(wordTable);
```

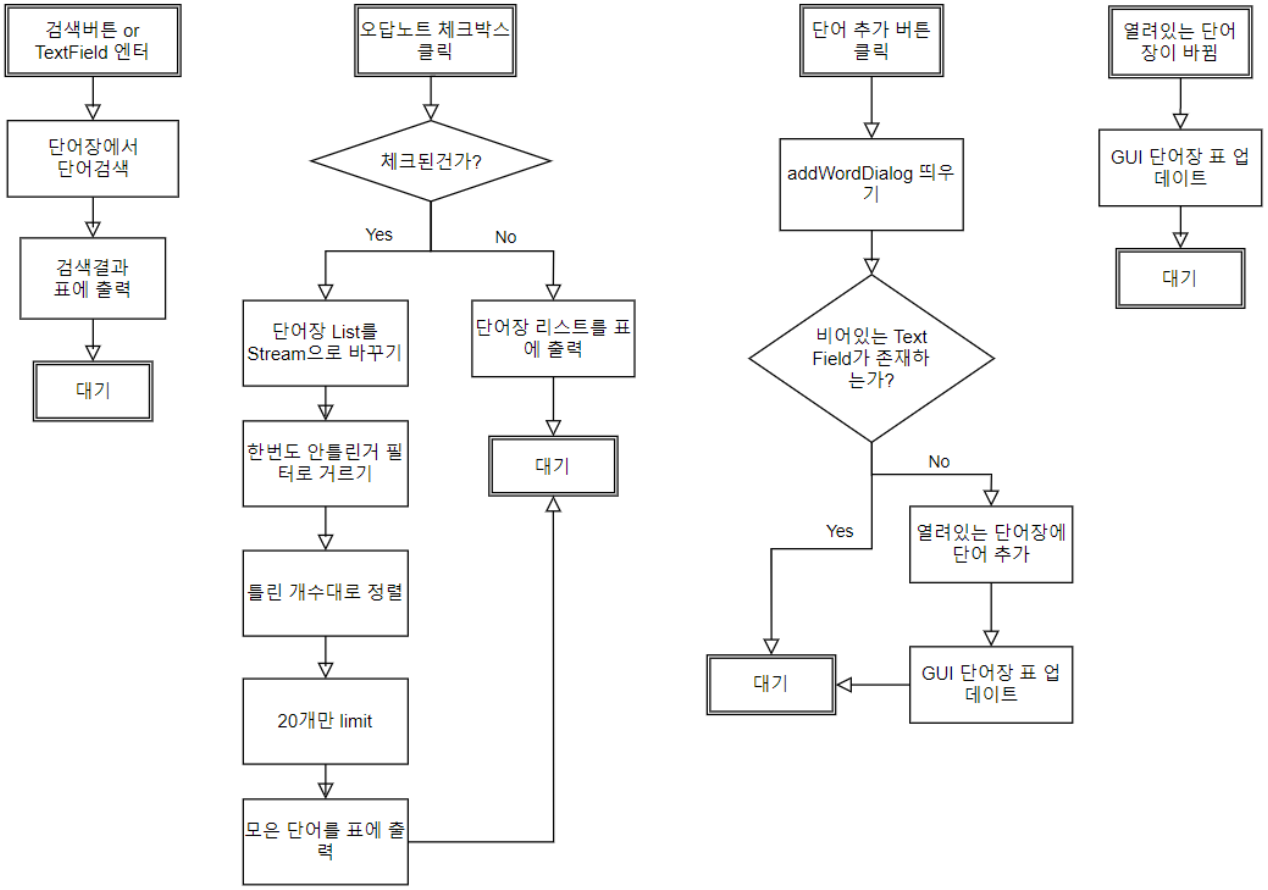
JTable 객체를 생성하는 부분이다. Columns 리스트를 이용해 column의 이름과 폭 비율을 설정한다.
그 후 스크롤패인을 JTable에 부착한다.

```
        north.add(searchField);
        north.add(searchButton);
        north.add(addWordButton);
        north.add(wrongAnswerNoteCB);

        this.add(north, BorderLayout.NORTH);
        this.add(scrollPane, BorderLayout.CENTER);
```

컴포넌트들과 패널을 부착시킨다.

```
        OpenedChangeListener.addOnOpenedChangeListener(new
OpenedChangeListener.OnOpenedChangeListener() {

            @Override
            public void onOpenedChange() {
                PrintToGUI ptg = new PrintToGUI();
                ptg.print(OpenedDictionary.getInstance().getVoc(),
wordTable, columns);
            }

        });

        searchField.addActionListener(new VocaPanelActionListener());
        searchButton.addActionListener(new VocaPanelActionListener());
        addWordButton.addActionListener(new AddWordActionListener());
        wrongAnswerNoteCB.addActionListener(new WorngNoteActionListener());
    }
```

생성자 마지막 부분은 이벤트 리스너들을 등록해주었다.

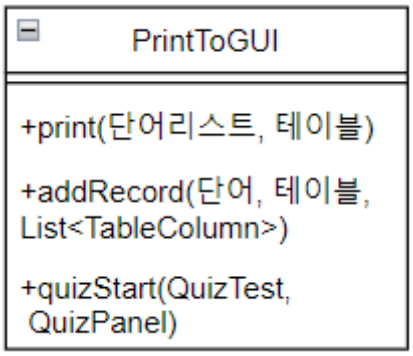


1.

```
class VocaPanelActionListener implements ActionListener {

	@Override
	public void actionPerformed(ActionEvent e) {
		String searchText = searchField.getText();
		if (searchText != null) {
			EnglishVoca dictionary = (EnglishVoca)
OpenedDictionary.getInstance();
			PrintToGUI ptg = new PrintToGUI();
			ptg.print(dictionary.searchWord(searchText), wordTable,
columns);
		}
	}

}
```

검색 버튼을 누르거나 TextField에서 엔터를 입력시 실행되는 리스너다.
searchField에 있는 텍스트를 가져온다.
그리고 펼쳐져 있는 단어장을 가져온다.
PrintToGUI 객체를 생성하고,
단어 검색한 결과를 JTable에 표시한다.

✓ PrintToGUI 객체



관련 객체들을 넣으면, GUI로 출력해주는 기능을 담당하는 객체다.

Print : Jtable에 인자로 받아온 단어리스트 출력

addRecord : 인자로 받아온 단어를 Jtable에 추가

```
public class PrintToGUI {
	public PrintToGUI() {}

	public void print(List<Word> voc, JTable table, List<TableColumn> columns) {
		DefaultTableModel model = (DefaultTableModel) table.getModel();
		model.setRowCount(0);
		for (Word word : voc) {
			addRecord(word, model, columns);
		}
	}
```

JTable에 단어들을 보여지게 하기 위해서 단어리스트와 Table의 정보를 인자로 받는다. TableModel 객체를 생성하여, 현재 table의 행을 모조리 없애고 새로 추가한다.

```
	private void addRecord(Word word, DefaultTableModel model, List<TableColumn>
columns) {
		String[] record = new String[columns.size()];
		for (int i=0; i<columns.size(); i++)
			record[i] = columns.get(i).value(word, model);
		model.addRow(record);
	}
```

Table에 한행 한행 단어를 추가하는 메소드이다. 한 칸의 내용은 TableColumn에 지정된 함수를 이용해 값을 받아와 넣는다.

2.

```
class AddWordActionListener implements ActionListener {

	@Override
	public void actionPerformed(ActionEvent e) {
		JFrame frame = (JFrame) SwingUtilities.getRoot(VocaPanel.this);
		AddWordDialog dialog = new AddWordDialog(frame, "단어 추가");
		dialog.setVisible(true);
		// 모달 다이얼로그 이므로 setVisible() 메소드는
		// 다이얼로그가 닫힐 때까지 리턴하지 않는다.
		String[] texts = dialog.getInput();
		if (texts == null) return;
```

```
                    EnglishVoca dictionary = (EnglishVoca)
OpenedDictionary.getInstance();
                    dictionary.addWord(new Word(texts[0], texts[1]));
                    dictionary.update();
                    PrintToGUI ptg = new PrintToGUI();
                    ptg.print(dictionary.getVoc(), wordTable, columns);
                }

        }
```

단어 추가 버튼을 눌렀을 때 동작하는 이벤트 리스너 부분이다.
먼저 추가할 영단어와 뜻을 가져오는 모달 다이얼로그를 연다.
그 후 입력한 영단어와 뜻을 크기가 2인 String 배열로 받아온다.
이때 내용이 비어있다면 리턴한다.

현재 열려있는 단어장을 가져와서
단어를 추가하고, 열려있는 단어장에 업데이트한다.
그리고 단어가 추가되었으므로 JTable의 내용도 업데이트 되어야 하므로
PrintToGUI 객체를 생성해서 표를 다시 작성한다.

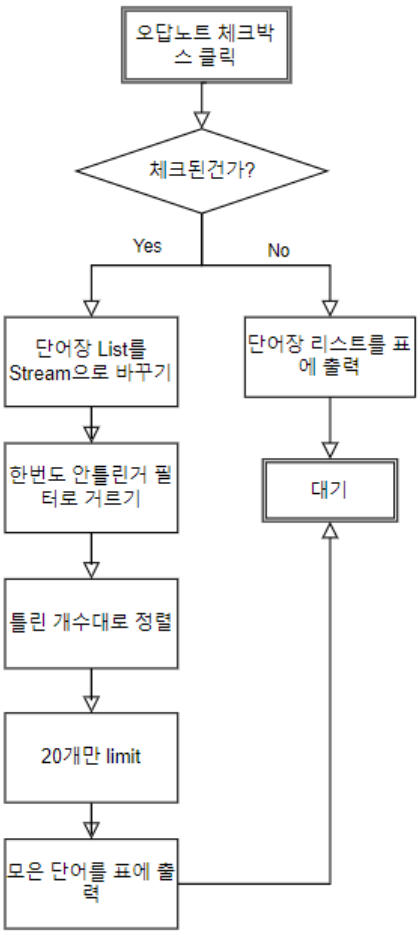
아래는 모달 다이얼로그 객체의 내용이다.

```
        class AddWordDialog extends JDialog {
                JTextField engField = new JTextField(10);
                JTextField korField = new JTextField(10);
                JButton addButton = new JButton("추가");
                JButton cancelButton = new JButton("취소");

                public AddWordDialog(JFrame frame, String title) {
                        super(frame, title, true);
                        this.setLayout(new GridLayout(3,2,10,10));
                        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                        this.setSize(300, 150);

                        this.add(new JLabel("영어 단어"));
                        this.add(engField);
                        this.add(new JLabel("한글 뜻"));
                        this.add(korField);
                        this.add(addButton);
                        this.add(cancelButton);

                        addButton.addActionListener(e->dispose());
                        cancelButton.addActionListener(e->dispose());
                        engField.addActionListener(e->dispose());
                        korField.addActionListener(e->dispose());
                }

                String[] getInput() {
                        if (engField.getText().length() == 0) return null;
                        else if (korField.getText().length() == 0) return null;
```

```
                else {
                        String[] strs = new String[2];
                        strs[0] = engField.getText();
                        strs[1] = korField.getText();
                        return strs;
                }
        }

}
```

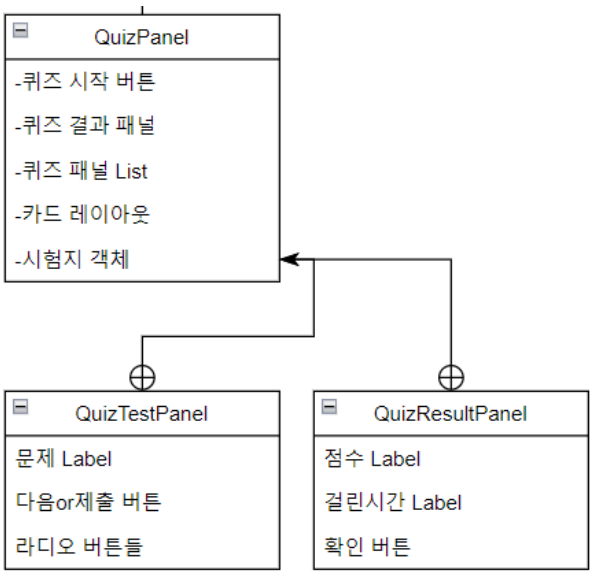
3.

```
class WorngNoteActionListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) {
                EnglishVoca dictionary = (EnglishVoca) OpenedDictionary.getInstance();

                if (wrongAnswerNoteCB.isSelected()) {
                        List<Word> worngWords = dictionary.getSortWords((o1, o2)-
>(o1.getWrongCount()-o2.getWrongCount())*(-1));
                        worngWords = worngWords.stream()
                .filter(word->(word.getWrongCount() > 0))
                .limit(20)
                .collect(Collectors.toList());
                        PrintToGUI ptg = new PrintToGUI();
                        ptg.print(worngWords, wordTable,
columns);
                } else {
                        PrintToGUI ptg = new PrintToGUI();
                        ptg.print(dictionary.getVoc(),
wordTable, columns)
                }
        }

}
```



오답노트 체크박스를 클릭시 동작하는 이벤트 리스너다.
우선 열려있는 단어장을 불러온다.
체크박스가 체크된거라면, 오른쪽의 YES 내용을 수행한다.
아니라면, 활성화된 오답노트에서 원래 단어 리스트를 표에 출력한다.

- QuizPanel 객체



마지막으로 퀴즈 패널 객체를 살펴보겠다.

```
public class QuizPanel extends JPanel {
	private JButton startButton;
	private JPanel home = new JPanel();
	private QuizResultPanel result = new
QuizResultPanel();
	private List<JPanel> panels;
	private CardLayout card = new
CardLayout();
	private QuizTest test;
```

퀴즈 패널의 필드 부분이다.

```
	public QuizPanel() {
		this.setLayout(card);

		startButton = new JButton("객관식 퀴즈 시작");
		home.add(startButton);
		resetPanels();
```

퀴즈 패널의 생성자 부분이다.
시작 버튼을 만들어, 홈 패널에 달아준다. 그리고 CardLayout을 한번 초기화한다.

```
	private void resetPanels() {
		try {
			EnglishVoca dictionary = (EnglishVoca)
OpenedDictionary.getInstance();
			dictionary.update();
		} catch(NullPointerException e) {}
		panels = new ArrayList<>();
		panels.add(home);
		setPanels();
		test = null;
		result = null;
	}


	private void setPanels() {
		this.removeAll();
		int num = 1;
		for (JPanel panel : panels) {
			this.add(panel, Integer.toString(num));
			num++;
		}
	}
```

resetPanels() 함수는 퀴즈가 진행되면서 레이아웃에 추가된 패널들을 모조리 지우고 홈 패널만 남기는 메소드이다.

```
startButton.addActionListener(new ActionListener() {

	@Override
	public void actionPerformed(ActionEvent e) {
		~~
	}

});
```

버튼을 클릭시 퀴즈가 시작되는 이벤트 리스너를 달아주었다. actionPerformed 함수의 내용은 다음과 같다.

```
try {
	MakeQuiz makeQuiz = new MakeQuiz(OpenedDictionary.getInstance());
	List<Quiz> quizList = new ArrayList<>();
```

먼저 MakeQuiz 객체와 만들어진 퀴즈를 담을 빈 리스트를 생성한다.
MakeQuiz 객체는 인자로 받아온 단어장에서 랜덤으로 퀴즈를 추출하는 객체이다.

```
for (int i=0; i < 10; i++)
	quizList.add(makeQuiz.make(4));
```

선지가 4개인 객관식 퀴즈를 10개 생성하여 리스트에 추가한다.

```
	QuizTest test = new QuizTest(quizList);
	PrintToGUI ptg = new PrintToGUI();
	QuizPanel.this.test = test;
	ptg.quizStart(test, QuizPanel.this);
	card.next(QuizPanel.this);
} catch (MakeQuizException ex) {
	JOptionPane.showMessageDialog(null, "단어가 부족해 퀴즈를 만들 수 없습니다.",
"Message", JOptionPane.WARNING_MESSAGE);
	return;
}
```

그후 시험지 객체를 생성하여 퀴즈 리스트를 인자로 넘긴다.
필드에 있는 test 변수에 생성한 QuizTest 객체를 저장한다.
PrintToGUI 객체를 만들어 퀴즈를 시작한다.

만약 MakeQuizException이 발생하면, 단어가 부족해 퀴즈를 만들 수 없는 상태이므로 이를 예외처리 해주었다.

```
	public void quizStart(QuizTest test, QuizPanel panel) {
		panel.settingStartQuizPanel();
		JTabbedPane tp = (JTabbedPane) panel.getParent();
		tp.setEnabledAt(0, false);
		tp.setEnabledAt(1, false);
		Thread timer = new Thread(test);
		test.timer = timer;
		timer.start();
	}
```

```
	public void settingStartQuizPanel() {
		for (int i=0; i<test.quizList.size(); i++) {
			if (i == test.quizList.size()-1)
				this.getPanels().add(this.createQuizTestPanel(i+1,
test.quizList.get(i), true));
			else
				this.getPanels().add(this.createQuizTestPanel(i+1,
test.quizList.get(i), false));

		}

		this.addResultPanel();
		this.setPanels();
	}


	private JPanel createQuizTestPanel(int number, Quiz quiz, boolean lastQuiz) {
		return new QuizTestPanel(number, quiz, lastQuiz);
	}
```

PrintToGUI의 quizStart 메소드이다.
패널에 있는 settingStartQuizPanel() 메소드를 실행한다. 이 메소드는 만들어진 퀴즈를 바탕으로 패널을 실제로 생성하여 카드레이아웃에 추가하는 구현부 이다.

그리고 QuizPanel의 상위에 컨테이너, 즉 TabbedPane을 불러온다.
그 후 홈 탭과 단어장 탭을 비활성화시켜 컨닝을 방지한다.
시험지 객체를 통해 타이머 쓰레드를 생성 후 타이머를 시작한다.

다음은 위에서 사용된 MakeQuiz, Quiz, QuizTest, QuizTestPanel, QuizResultPanel 객체를 설명하고자 한다.

- ✓ MakeQuiz 객체

단어장에서 객관식 퀴즈를 랜덤으로 생성하는 기능을 담당하는 객체다.

```
public class MakeQuiz {
	private Dictionary dictionary;
```

```
	public MakeQuiz(Dictionary dictionary) {
		super();
		this.dictionary = dictionary;
	}
```

생성자로부터 단어장을 받아온다.

```
	public Quiz make(int chioceNum) {
		List<Word> voc = dictionary.getVoc();
		Set<Word> choiceSet = new HashSet<>();
		Random random = new Random();

		if (voc.size() < chioceNum)
			throw new MakeQuizException("fail make quiz (lack word)");
```

인자로 몇 개의 선지를 만들지 정보를 받는다.
단어 집합과 랜덤 객체를 생성한다.
만약 단어장 사이즈가 만들어야할 선지 숫자보다 작으면 예외를 던진다.

```
		try {
			Loop: while (choiceSet.size() < chioceNum) {
				Word randomWord = voc.get(random.nextInt(voc.size()));
				for (Word word : choiceSet) {
					if (word.kor.equals(randomWord.kor))
						continue Loop;
				}
				choiceSet.add(randomWord);
			}
		} catch (IllegalArgumentException e) {
			throw new MakeQuizException("fail make quiz (lack word)");
		}
```

집합의 크기가 선지 개수만큼 충족될때까지 반복한다. 이는 집합이 중복 추가를 허용하지 않는 성질을 이용한 것이다.
단어장에서 랜덤으로 단어를 추출한다.
만약 집합에 이미 한글 뜻이 같은 단어가 있다면 이번 단어는 거르고 다시 추출한다.
아니라면 집합에 단어를 추가한다.

```
		return new
Quiz(choiceSet.stream().skip(random.nextInt(choiceSet.size())).findFirst().orElse(null),
choiceSet);
		/*
		 * Set 자료구조에서 랜덤 값 추출
		 * 출처 : https://stackoverflow.com/questions/124671/picking-a-random-
element-from-a-set
		 */
	}
```

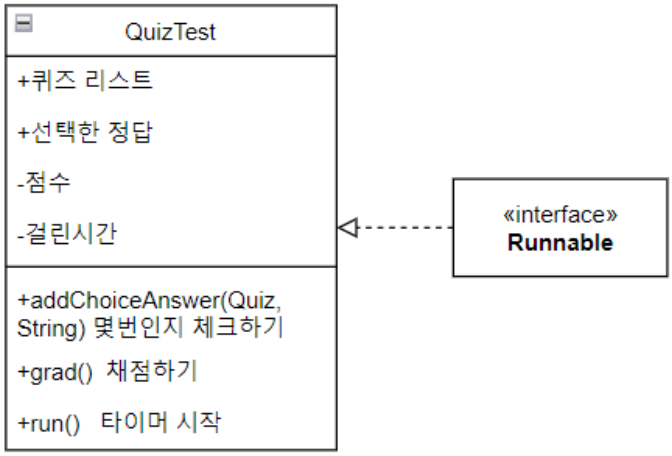
그 후 퀴즈 객체를 생성하여 반환한다.
퀴즈 객체를 생성할 때 Quiz(정답 단어, 선지 집합)을 인자로 받는다.

✓ Quiz 객체

```
public class Quiz {
	public Set<Word> choiceSet;
	private Word answer;

	public Quiz(Word answer, Set<Word> choiceSet) {
		super();
		this.answer = answer;
		this.choiceSet = choiceSet;
	}

	public Word getAnswer() {
		return answer;
	}

	public void setAnswer(Word answer) {
		this.answer = answer;
	}
}
```

하나의 객관식 문제는 n개의 답안과, 그중 정답인 답안의 정보를 가지고 있어야 한다.

✓ QuizTest 객체



시험지는 시험 문제들과, 문제에서 몇번을 찍었는지에 대한 정보와, 몇점인지 정보는 반드시 있어야 한다. 추가로 쓰레드를 추가해 타이머의 기능을 달아주었다.

메소드는 퀴즈에서 몇번을 정답으로 선택했는지 체크하는 메소드와, 채점하는 메소드를 구현해야 한다.

```
public class QuizTest implements Runnable {
	public List<Quiz> quizList = new ArrayList<>();
	public HashMap<Quiz, String> choiceAnswerMap = new HashMap<>();
	public Thread timer;
	private int score = 0;
	private int second = 0;
```

```
public QuizTest(List<Quiz> quizList) {
	super();
	this.quizList = quizList;
```

시험지의 필드 값이다.

```
public void addChoiceAnswer(Quiz quiz, String choiceAnswer) {
	choiceAnswerMap.put(quiz, choiceAnswer);
}
```

퀴즈에서 몇번을 정답으로 선택했는지 체크하는 메소드이다. HashMap에 퀴즈와, 그에 대응하는 체크한 답안을 put 한다.

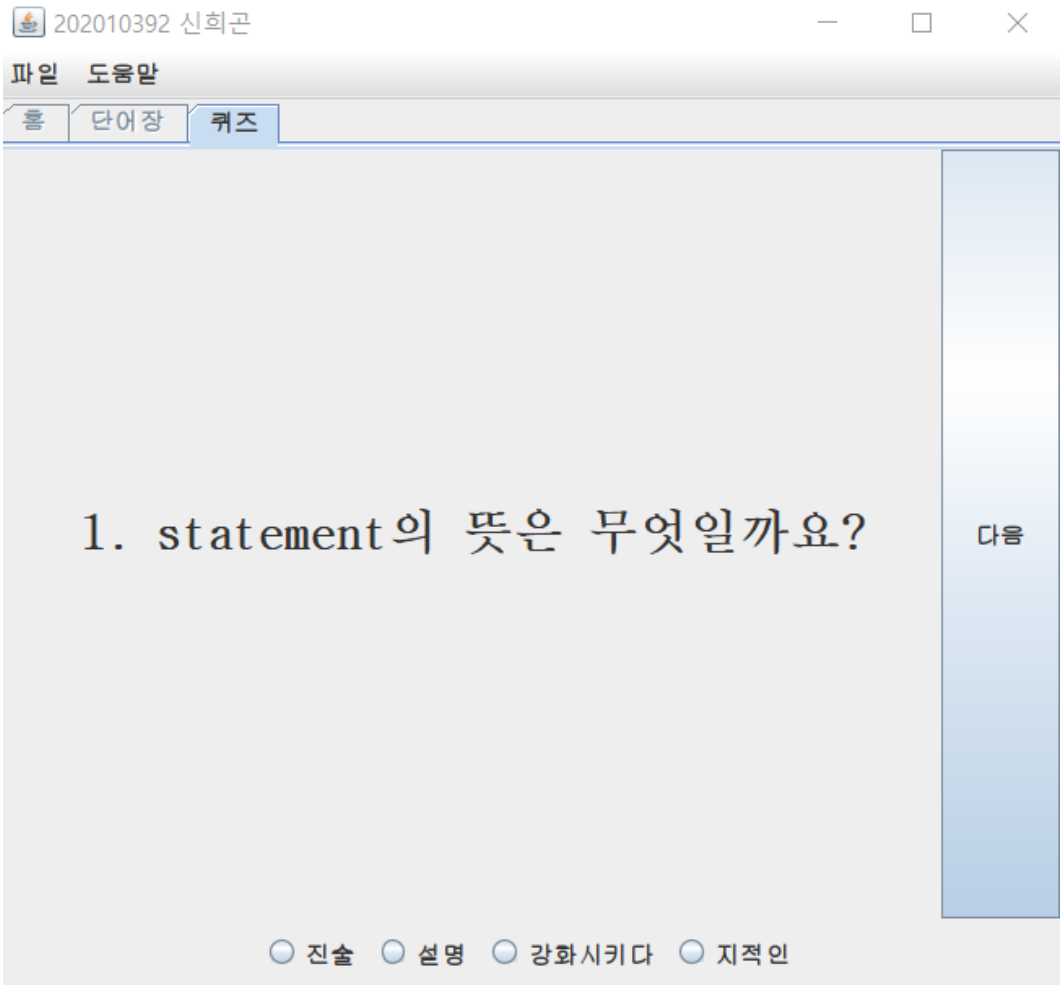
```
public void grade() {
	for (Quiz quiz : quizList) {
		if (quiz.getAnswer().kor.equals(choiceAnswerMap.get(quiz))) {
			score += 1;
		} else {
			quiz.getAnswer().addWrongCount();
		}
	}
}
```

채점하는 기능을 가진 메소드이다. 퀴즈의 답안과, 그 퀴즈에 대응하는 체크했던 답안이 동일하면 1점을 추가한다.
아니면 퀴즈 답안 Word에 틀린 횟수를 추가한다.

```
public int getScore() {
	return score;
}

public int getSecond() {
	return second;
}

@Override
public void run() {
	try {
		while(true) {
			Thread.sleep(1000);
			second++;
		}
	} catch (InterruptedException e) {
		return;
	}
}
```

그후 나머지 getter와 타이머 쓰레드를 구현했다.

- ✓ QuizTestPanel 객체



퀴즈가 시작되면 만들어지는 10개의 패널이다. 직접적으로 퀴즈를 풀게 되는 패널이다.

```
private class QuizTestPanel extends JPanel {
	private JPanel south = new JPanel();
	private ButtonGroup group = new ButtonGroup();
	private JRadioButton[] radioButions;
	private JButton nextButtion;
	private JLabel question;
```

문제 정보와 라디오 버튼, 다음 버튼이 존재한다.

```
	public QuizTestPanel(int number, Quiz quiz, boolean lastQuiz) {
		QuizPanel quizPanel = QuizPanel.this;
```

```
this.radioButions = new JRadioButton[quiz.choiceSet.size()];
this.setLayout(new BorderLayout());

question = new JLabel(number + ". " + quiz.getAnswer().eng + "의
뜻은 무엇일까요?");
question.setHorizontalAlignment(SwingConstants.CENTER);
question.setFont(new Font("바탕체", Font.BOLD, 24));
```

라디오 버든들의 빈 배열을 생성하고,
현재 레이아웃을 보더 레이아웃으로 바꾼다.
문제 Label을 만든다.

```
if (!lastQuiz) {
	nextButtion = new JButton("다음");
} else {
	nextButtion = new JButton("제출");
}
```

만약 마지막 10번 퀴즈가 아니라면 버튼 이름을 '다음'으로 생성하고,
마지막 퀴즈라면 '제출' 버튼으로 생성한다.

```
List<Word> choiceList = new ArrayList<>(quiz.choiceSet);
Collections.shuffle(choiceList);
for (int i=0; i<choiceList.size(); i++) {
	radioButions[i] = new JRadioButton(choiceList.get(i).kor);
	radioButions[i].setActionCommand(choiceList.get(i).kor);
	group.add(radioButions[i]);
	south.add(radioButions[i]);
}
```

라디오 버튼을 하나하나 생성하는 부분이다.
South 패널에도 라디오버튼을 달아준다.

```
this.add(nextButtion, BorderLayout.EAST);
this.add(south, BorderLayout.SOUTH);
this.add(question, BorderLayout.CENTER);
```

최종적으로 패널에 컨테이너와 컴포넌트를 부착한다.

```
nextButtion.addActionListener(new ActionListener() {

	@Override
	public void actionPerformed(ActionEvent e) {
		~~
	}

});
```

다음 or 제출 버튼을 눌렀을 때 동작하는 이벤트 리스너이다.
actionPerformed 함수의 내용은 다음과 같다.

```
	try {
		if (nextButtion.getText().equals("다음")) {
			quizPanel.test.addChoiceAnswer(quiz,
group.getSelection().getActionCommand());
			quizPanel.card.next(QuizTestPanel.this.getParent());
		}
```

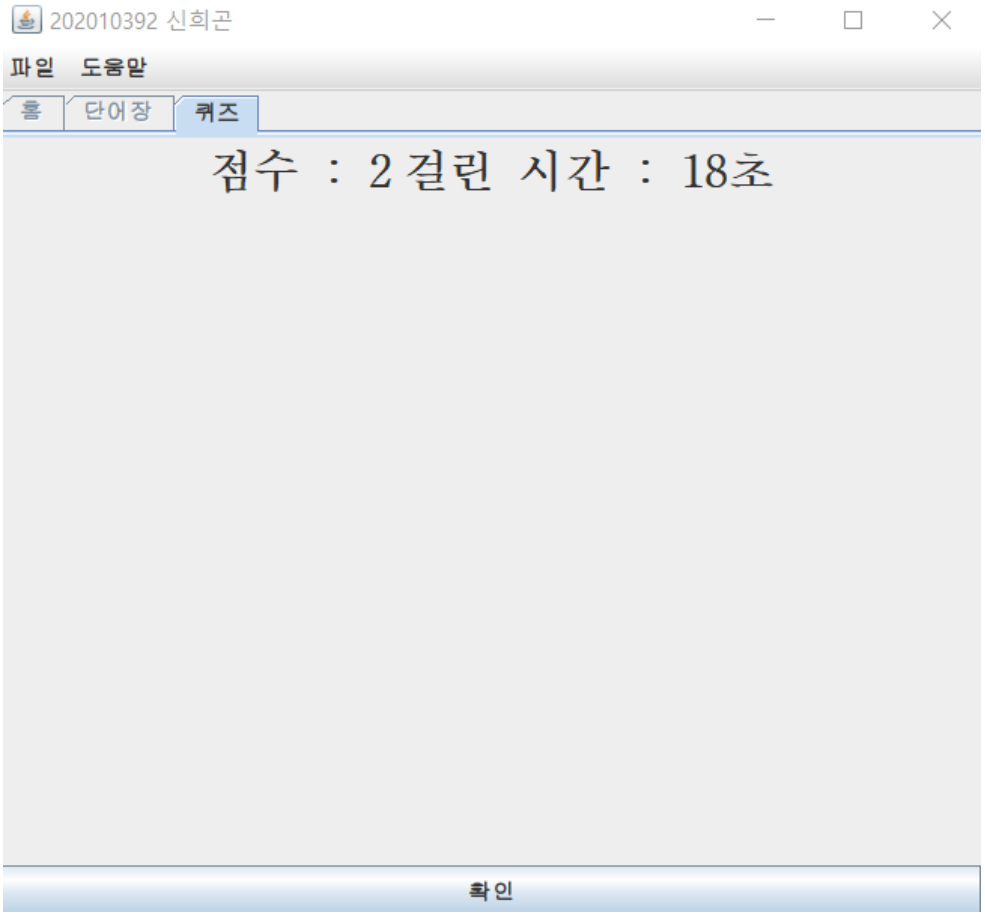
버튼이 다음 버튼이면,
현재 시험지에 몇번을 찍었는지 체크하고
다음 문제로 넘어간다.

```
		else {
			quizPanel.test.addChoiceAnswer(quiz,
group.getSelection().getActionCommand());
			quizPanel.test.timer.interrupt();
			quizPanel.test.grade();
			quizPanel.result.init();
			quizPanel.card.next(QuizTestPanel.this.getParent());
		}
	} catch (NullPointerException ex) {
		JOptionPane.showMessageDialog(null, "정답을 선택해주세요!", "Message",
JOptionPane.WARNING_MESSAGE);
		return;
	}
```

다음 버튼이 아니라 제출 버튼이라면,
일단 마지막 문제를 몇번 찍었는지 체크한다.
그후 타이머를 종료하고,
시험지를 채점한다.
그리고 퀴즈 결과 패널을 생성한 뒤
결과 화면으로 넘어간다.

정답을 선택하지 않을 시 NullPointerException이 발생하고, 이를 예외처리 해주었다.

✓ QuizResultPanel 객체



퀴즈 결과를 보여주는 패널이다.

```
	private class QuizResultPanel extends JPanel {
		JPanel center = new JPanel();
		JLabel scoreLabel, secondLabel;
		JButton okButton;

		public QuizResultPanel() {}
```

점수, 몇초 걸렸는지를 보여주는 레이블과 확인 버튼이 존재한다.

```
		public void init() {
			this.setLayout(new BorderLayout());

			scoreLabel = new JLabel("점수 : " + test.getScore(),
SwingConstants.CENTER);
			scoreLabel.setFont(new Font("바탕체", Font.BOLD, 24));
			secondLabel = new JLabel("걸린 시간 : " + test.getSecond() + "초",
SwingConstants.CENTER);
			secondLabel.setFont(new Font("바탕체", Font.BOLD, 24));
```

```
                        center.add(scoreLabel);
                        center.add(secondLabel);
```

결과 패널을 생성하는 메소드이다.
점수와 걸린 시간 Label을 생성하고 center 패널에 붙인다.

```
                        okButton = new JButton("확인");

                        this.add(center, BorderLayout.CENTER);
                        this.add(okButton, BorderLayout.SOUTH);
```

확인 버튼도 생성하고, 각각 Center와 South 위치에 붙인다.

```
                        okButton.addActionListener(new ActionListener() {

                                @Override
                                public void actionPerformed(ActionEvent e) {
                                        JTabbedPane tp = (JTabbedPane)
QuizPanel.this.getParent();

                                        tp.setEnabledAt(0, true);
                                        tp.setEnabledAt(1, true);

                                        resetPanels();
                                }

                        });
                }
        }
}
```

확인 버튼을 누르면 동작하는 이벤트 리스너이다.
확인 버튼을 누르면 퀴즈가 종료되는 것이므로,
비활성화 시켜뒀던 홈 탭과 단어장 탭을 다시 활성화한다
그리고 패널을 초기화한다.

3 실행결과

Mp4 파일로 제출

4 느낀점 및 토의사항

객체지향 지식이 전무할 때는 코드를 항상 절차지향적으로 밖에 구현할 수 없었다. 그러다 보니 매번 스파게티 코드가 되기 일쑤였다. 이런 코드들은 나중에 무언가 추가하기도 번거롭고, 수정하기도 어려우며 버그가 예상치도 못한 곳에서 튀어나왔다. 그러나 이번 학기에 JAVA프로그래밍을 배우고 나서 한번 제대로 객체지향적으로 코드를 설계해보고 싶은 생각이 들었다. 기존에는 구현할 기능들의 체크리스트만 적어두고 코드를 작성했지만, 이번 프로젝트에선 플로우 다이어그램과 클래스 다이어그램도 그려보고 어떤 객체가 필요할지 생각하고 코드를 작성하기 시작했다. 최대한 하나의 객체는 오직 하나의 책임을 가지도록 설계하고, 다형성과 캡슐화가 잘 이루어지도록 노력했다. 클래스가 점점 많아지면서 '오히려 더 복잡해지는게 아닌가?' 라는 생각이 들기도 했지만, 일단 객체지향의 원칙을 지켜가며 코드를 작성했다.

그렇게 프로그램을 완성했다. 그 결과 코드 작성 도중, 전에 비해 버그가 거의 발생하지 않았다. 또한 각 객체가 처리하는 내용도 명확하여 처리하는 내용을 이해하기 훨씬 수월했다. 기능을 수정할 때에도 그 기능을 맡고있는 객체의 부분만 수정하면 되므로 유지보수도 간단해졌다. 미흡한 부분이 많지만, 처음으로 객체지향적으로 설계해 놀라운 효과를 얻으니 굉장히 뿌듯해지는 기분이 들었다. 이번 과제를 끝으로 JAVA 프로그래밍 수업이 마무리되어 아쉽지만 많은 지식을 쌓아갈 수 있었고, 많은 깨달음을 얻을 수 있는 유익한 시간이었다고 생각한다.