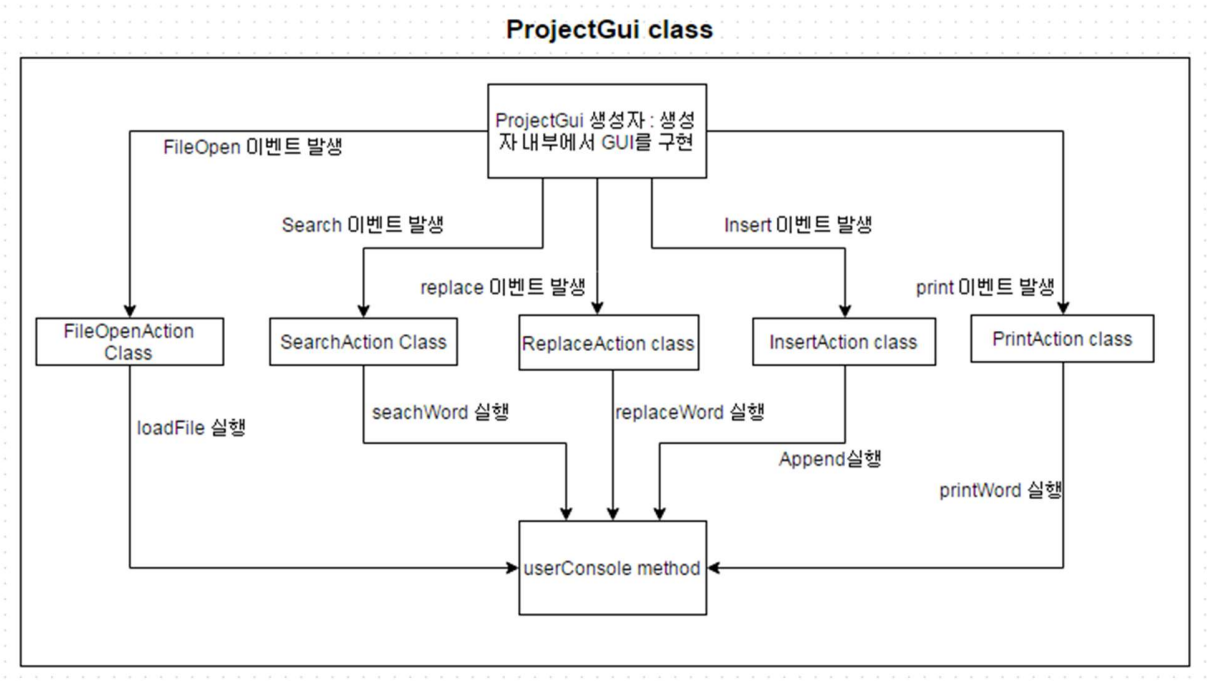
문제 분석 :

Editor 클래스의 기능(단어검색, 단어변환, 검색결과 출력, 단어삽입)과 Swing을 이용한 GUI(메뉴, 검색진행 상태 레이블, 입/출력 파일경로 입력 텍스트필드, 검색/변환 단어 입력 텍스트필드, 검색결과 출력 리스트)을 구현하시오.

: Editor 클래스의 기능은 기존 실습과제 12에서 구현한 기능을 사용한다. 기존 Editor 클래스는 콘솔화면으로 프로그램을 컨트롤했지만 프로젝트에서는 Swing class를 사용하여 GUI로 프로그램을 구성한다. GUI에 컴포넌트에서 발생하는 이벤트를 처리하기 위해 이벤트 처리기를 만들어야 된다. 이벤트 처리기를 만들기 위해서 ActionListener 인터페이스를 상속받아 인터페이스 내부에 actionPerformed 메소드를 오버라이딩 해야 한다. 또한 컴포넌트의 이벤트 처리기를 기존 Editor 클래스의 메소드와 연결시켜서 이벤트가 발생했을 때 그와 관련된 Editor 클래스의 메소드가 실행되도록 해야 한다.

Flow Chart :

프로그램 설계 그림



프로그램 설계/알고리즘 :

Editor 클래스의 단어검색, 단어변환, 검색결과 출력, 단어삽입 기능은 기존 과제에서 구현한 내용을 그대로 사용한다. 그러므로 Editor 설계/알고리즘에 대한 내용은 기존 과제에 작성한 내용과 동일하기 때문에 프로젝트 과제에는 포함하지 않겠다.

기존 Editor 클래스의 실행은 콘솔로 실행되었다. 프로젝트에서는 콘솔로 구현된 Editor 클래스를 Swing 을 사용해서 GUI 로 구현해야 한다. GUI 를 사용하기 위해서 awt 와 swing 를 import 하고 컴포넌트에 대한 이벤트를 처리하기 위해서 awt.event 도 import 한다. Main 메소드가 실행되면 ProjectGui gui=new ProjectGui(" ");가

실행된다. Main 메소드에서 ProjectGui 를 생성하면 ProjectGui 클래스에 생성자가 실행이 된다. ProjectGui 클래스에는 GUI 를 생성하는 생성자와 파일열기 이벤트를 처리하는 FileOpenAction 클래스, 단어검색 이벤트를 처리하는 SearchAction 클래스, 단어변환 이벤트를 처리하는 ReplaceAction 클래스, 단어삽입 이벤트를 처리하는 InsertAction 클래스, 단어출력 이벤트를 처리하는 PrintAction 클래스와 userConsole 메소드가 존재한다.

GUI 의 구현은 1 개의 메뉴바와 5 개의 패널로 구성했다. 메뉴바에는 JMenu 객체를 생성해서 파일, 편집을 만들고 JMenuItem 객체를 사용해서 파일에 열기, 출력 그리고 편집에 검색, 변환을 포함한다. 1 번째 패널에는 검색할 단어, 변환할 단어를 입력할 길이가 15 인 텍스트필드와 검색, 변환 버튼을 포함한다. 2 번째 패널에는 삽입할 단어위치를 지정하기 위한 행, 열 텍스트필드와 삽입 버튼을 포함한다. 3 번째 패널에는 입력파일명을 입력할 길이가 25 인 텍스트필드와 열기버튼, 검색개수 결과를 출력하는 레이블을 포함한다. 4 번째 패널에는 검색결과를 출력하기 위한 textArea 를 포함한다. 5 번째 패널에는 검색결과를 출력하는 레이블과 출력파일명을 입력할 길이가 25 인 텍스트필드와 결과출력 버튼을 포함한다. 5 개의 패널은 FlowLayout 을 사용해서 컴포넌트들을 왼쪽에서 오른쪽 방향으로 배치했다. 패널들은 JFrame 클래스의 객체 mainFrame 에 FlowLayout 속성으로 삽입된다. 검색, 변환, 열기, 출력, 삽입 이벤트를 처리하는 이벤트 처리기의 actionPerformed 메소드 내부에서는 userConsole 메소드를 호출한다. ProjectGui 클래스에 userConsole 메소드를 삽입해서 컴포넌트의 이벤트 내용을 userConsole 메소드가 참조할 수 있도록 하기 위해서 userConsole 메소드를 ProjectGui 클래스에 포함했다. 입력파일명을 입력하고 열기버튼 또는 메뉴-파일-열기를 누르면 FileOpenAction 이벤트 처리기가 생성이 되면서 actionPerformed 메소드 내부에서 userConsole("o");을 실행한다. 해당 메소드가 실행이 되면 userConsole 메소드에서는 editor.loadFile(inputPath.getText());을 실행해서 입력스트림을 생성한다. 메소드의 인자로는 입력 파일명 텍스트필드의 문자열을 getText()메소드로 가져와서 loadFile 메소드의 인자로 전달한다. 단어검색은 검색할 단어를 입력하고 검색버튼 또는 메뉴-편집-검색을 누르면 SearchAction 이벤트처리기가 생성된다. 해당 이벤트처리기의 actionPerformed 메소드에서는 userConsole("s");을 실행해서 userConsole 메소드에서 editor.searchWord(br, searchField.getText());을 실행한다. searchWord 의 첫 번째 인자로 loadFile 에서 생성한 파일스트림과 searchField 에 입력한 검색할 단어를 가져와서 인자로 전달한다. searchWord 메소드가 종료되면 검색결과를 저장하는 ArrayList 가 반환되는데 반환 값을 멤버변수 mArrayList 에 저장한다. 그리고 검색결과 개수를 클래스 멤버변수 resultCount 에 저장한다. 그런 다음 actionPerformed 함수로 PC 가 조정되면서 userConsole("s"); 다음에 아래 코드를 실행한다.

```
for(ISearchItem item : mArrayList)//검색 결과 콘솔화면 출력
{
        textArea.append("행 : "+item.getLineNo()+" 열 : "+item.getIndexNo()+"\n");
}
resultCountLabel.setText("검색결과: "+resultCount);
resultLabel.setText("검색완료");
```

개선된 for 문을 사용해서 ArrayList 에 저장된 내용을 리스트에 출력하고 검색결과 개수를 resultCountLabel 에 출력한다. 그리고 검색결과를 resultLabel 에 출력한다. 출력파일명과 검색할 단어, 변환할 단어를 입력하고 변환버튼 또는 메뉴-편집-변환을 누르면 ReplaceAction 이벤트 처리기가 생성이 된다. 해당 클래스의 actionPerformed 메소드에서는 userConsole("r");을 실행한다. userConsole 이 실행이 되면 editor.replaceWord(br, searchField.getText(), replaceField.getText(), outputPath.getText());을 실행한다. replaceWord 메소드의 첫 번째

인자로 파일스트림을 전달하고 두 번째와 세 번째 인자로는 검색할 단어와 변환할 단어를 전달한다. 네 번째 인자로는 출력파일명을 텍스트필드로부터 가져와서 인자로 전달한다.

이와 같이 삽입버튼을 누르면 InsertAction 이 실행이 되고 결과출력 또는 메뉴-파일-출력을 누르면 PrintAction 이 실행된다. JFrame 을 상속받는 ProjectGui 클래스의 생성자에서 GUI 를 생성하고 GUI 컴포넌트에 입력되는 이벤트 정보를 공유하기 위해 userConsole 메소드를 ProjectGui 에 포함해서 프로그램을 설계했다.

소스 코드/주석 :

```
import java.io.*;//파일 입출력을 위한 class 가져오기
import java.util.Scanner;//표준입력 받기 위해서 Scanner class 가져오기
import java.util.StringTokenizer;//문자열 파싱을 위한 token class 가져오기
import edu.dongguk.cse.pl.reportwritter.ISearchItem;//사용자 정의 인터페이스 가져오기
import java.util.ArrayList;//ArrayList 가져오기
import java.awt.event.*;//GUI 이벤트
import java.awt.*;//GUI
import javax.swing.*;//GUI

/**
 * author      : Jun Su. Lim
 * Student Num : 2010111661
 * Created     : 2015-10-03
 * Modified    : 2015-12-11 (GUI 구현)
 * Class       : CSearchItem -ISearchItem 인터페이스 구현
 *             : Editor - searchWord, printWord, replaceWord method include
 *             : EditorTest - main method include
 *             : MyBufferedReader - BufferedReader 사용자 정의
 *             : ImprovedEditor - Append, SearchWord method include
 *             : ProjectGui - GUI 정의
 * Description : test.txt 파일을 로드한 후 사용자가 입력한 문자열 검색 지원, 문자열 검색 결과 사용자가 지정한
                 출력위치에 text파일로 저장
 *             : test.txt에 저장된 단어를 사용자가 원하는 단어로 변경하는 기능 포함
 *             : GUI로 실행
 */

/**
 * ISearchItem 인터페이스 상속, 인터페이스 구현
 * ISearchItem superclass, CSearchItem subclass
 */
class CSearchItem implements ISearchItem{
        private int lineNo;  //line을 관리하는 변수(라인 수)
        private int indexNo; //index을 관리하는 변수(번째)

        //인터페이스에 대한 메소드 오버라이딩
        @Override
        public int getIndexNo(){
                return indexNo;
        }
        @Override
        public int getLineNo(){
                return lineNo;
        }
        @Override
        public void setIndexNo(int indexNo){
```

```
                this.indexNo=indexNo;
        }
        @Override
        public void setLineNo(int lineNo){
                this.lineNo=lineNo;
        }
}


/**
 * SuperClass : BufferedReader
 * SubClass   : MyBufferedReader
 * readLine() method overriding, exception handling
 * constructor overloading.
 */
class MyBufferedReader extends BufferedReader{

        File file=null;
        InputStreamReader istreamR=null;

        //constructor overloading
        public MyBufferedReader(File file) throws FileNotFoundException {
                super(new FileReader(file));//SuperClass constructor call
                this.file=file;
        }

        public MyBufferedReader(InputStreamReader iStreamR) throws FileNotFoundException {
                super(iStreamR);
                this.istreamR=iStreamR;
        }

        //SuperClass readline() method overriding.
        //readLine() exception handling, EOF 만나면 문자열 출력
        public String readLine() throws IOException
        {
                String temp=null;
                try {
                        temp = super.readLine();
                        if(temp == null)
                                throw new NullPointerException();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch(NullPointerException e){
                        //System.out.println("라인을 모두 읽었습니다");
                }
                return temp;
        }

}

/**
 * loadFile method : 읽어올 파일 경로 지정
 * searchWord method : 문자열 검색지원
 * printWord method : 검색된 문자열 파일 저장 지원
 * replaceWord method : test.txt에 저장된 단어 변환
 * searchMethod method : search method의 subroutine
 */
class Editor{

        static int resultCount;//검색횟수 저장
        static ArrayList<ISearchItem> mArrayList;//검색 결과 저장
```

```
	/**
	 * param	: String filePath, 파일 경로 저장
	 * throws	: IOException, 파일 입출력 예외처리
	 * return	: FileInputStream, 파일 입력스트림 반환
	 * Description : 사용자로부터 검색할 파일 경로를 입력 받으면 경로에 존재하는 파일과 입력스트림 형성 후 입력스트림 반환
	 */
	public static FileInputStream loadFile(String filePath) throws IOException
	{
		FileInputStream fIn=new FileInputStream(filePath);//텍스트파일과 입력스트림 형성

		return fIn;//입력스트림 반환
	}

	/**
	 * param	: BufferedReader br, 버퍼 기능이 추가된 입력스트림을 전달 받는다.
	 * param	: String findWord, 찾을 단어 저장
	 * throws	: IOException, 파일 입출력 예외처리
	 * return	: ArrayList<ISearchItem>, 검색 결과 반환
	 * Description : 문자열 검색
	 */
	public static ArrayList<ISearchItem> searchWord(MyBufferedReader br, String findWord) throws IOException
	{
		mArrayList=searchMethod(br, findWord);//subroutine 호출, 단어검색
		return mArrayList;
	}

	/**
	 * param	: BufferedReader br, 버퍼 기능이 추가된 입력스트림을 전달 받는다.
	 * param	: String findWord, 찾을 단어 저장
	 * param	: String outputPath, 출력 파일 경로 저장
	 * throws	: IOException, 파일 입출력 예외처리
	 * return	: ArrayList<ISearchItem>, 검색 결과 반환
	 * Description : 문자열 검색 결과 파일 저장 method
	 */
	public static ArrayList<ISearchItem> printWord(MyBufferedReader br, String findWord, String outputPath) throws IOException
	{

		mArrayList=searchMethod(br, findWord);//단어 검색
		File file=new File(outputPath);//파일스트림 생성

		//두 번째 매개변수로 true가 전달되면 파일 기존 내용 뒤에 append
		//두 번째 매개변수로 false가 전달되면 파일 기존 내용을 삭제하고 write
		FileWriter fw=new FileWriter(file, true);

		//파일에 결과 출력
		fw.write("<"+outputPath+">"+" 파일에서 "+"<"+findWord+">"+"의 검색결과는 다음과 같습니다.\r\n");
		for(ISearchItem item : mArrayList)
		{
			fw.write(item.getLineNo()+"번째 줄"+item.getIndexNo()+"번째\r\n");
		}
```

```
                fw.write("\r\n");
                fw.write("이상 검색결과로 총"+resultCount+"개의 검색이 완료되었습니다.\r\n\r\n");
                fw.close();//파일스트림 종료

                return mArrayList;//검색 결과 반환
        }

        /**
         * param   : BufferedReader br, 버퍼 입력 스트림
         * param   : String findWord, 사용자가 찾고자 하는 문자열을 저장한다.
         * throws  : IOException, 파일 입출력 시에 예외처리
         * return  : ArrayList<ISearchItem>, 검색결과를 저장하는 ArrayList 반환
         * Description : 사용자로부터 표준입력 받은 문자열을 가지고 텍스트파일에 입력 받은 문자열과 같은 문자열이
있는지 탐색
         *            : 해당 문자열이 텍스트 파일에 존재하면 문자열의 행과 열을 ArrayList에 저장
         */
        public static ArrayList<ISearchItem> searchMethod(MyBufferedReader br, String findWord) throws
IOException
        {
                int lineNum=0, indexNum=0;//행과 열을 표시하기 위한 변수
                String currentLine, nextLine, concatLine1, concatLine2;//문자열 저장 변수
                String secondFindWord=null;
                String thirdLine="\0";//문자열을 저장하기 위한 변수
                StringTokenizer st, fw;//파싱한 토큰을 저장하기 위한 객체
                int count=0;//검색 횟수
                boolean spaceCheck=false;//찾을 단어 공백 확인 플래그
                ArrayList<ISearchItem> mArrayList = new ArrayList<ISearchItem>();//ArrayList 생성

                for(int i=0; i<findWord.length(); i++)//찾을 단어가 공백이 포함된 단어인지 확인
                {
                        if(findWord.charAt(i)==' ')//공백이 포함돼있으면
                        {
                                spaceCheck=true;//공백 플래그 true로 설정
                                break;//loop 종료
                        }
                }

                if(spaceCheck==true)//공백이 포함된 단어만 실행
                {
                        //예를 들어 hello world는 hello, world로 나눠진다.
                        fw=new StringTokenizer(findWord," ");
                        findWord=fw.nextToken();//파싱
                        secondFindWord=fw.nextToken();
                }

                loop://반복문의 label
                while(true)//무한 루프
                {
                        CSearchItem item=new CSearchItem();//CSearchItem 객체 생성

                        //thirdLine객체에 문자열이 없으면 currentLine에 문자열을 읽어와서 저장
                        if(thirdLine=="\0")
                        {
```

```
                currentLine=br.readLine();
            }
            else//thirdLine에 문자열이 존재하면 currentLine에 복사
            {
                currentLine=thirdLine;
                thirdLine="\0";//thirdLine 널 문자로 초기화
            }

            if(currentLine==null)//currentLine에 문자열이 없으면 무한 루프 탈출
                break loop;

            st=new StringTokenizer(currentLine," ");//공백 단위로 문자열 파싱

            lineNum++;//행 1 증가
            while(st.hasMoreTokens())//토큰이 없을 때까지 반복
            {
                if(findWord.equals(st.nextToken()))//토큰을 가져와서 사용자가 입력한
문자열과 비교
                {
                    //공백이 포함된 단어검색 조건
                    if(spaceCheck==true && st.hasMoreTokens() &&
secondFindWord.equals(st.nextToken()))
                    {
                        indexNum=currentLine.indexOf(findWord)+1;//열 번호
                        if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                        {
                            item.setIndexNo(indexNum);//열 번호 저장
                            item.setLineNo(lineNum);//행 번호 저장
                            count++;//검색 횟수 1증가
                            mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장
                        }
                        indexNum=0;
                        continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                    }
                    else if(spaceCheck==false)//공백이 포함되지 않은 단어
                    {
                        indexNum=currentLine.indexOf(findWord)+1;//열 번호
                        if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                        {
                            item.setIndexNo(indexNum);//열 번호 저장
                            item.setLineNo(lineNum);//행 번호 저장
                            count++;//검색 횟수 1증가
                            mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장
                        }
                        indexNum=0;
                        continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                    }
                }
            }
```

```
                        nextLine=br.readLine();//현재 읽어온 문자열 다음에 문자열을 읽어온다.
                        if(nextLine==null)//nextLine이 널이면 무한 루프 탈출
                                break loop;

                        st=new StringTokenizer(nextLine," ");//공백단위로 파싱


                        while(st.hasMoreTokens())
                        {
                                if(findWord.equals(st.nextToken()))
                                {
                                        lineNum++;
                                        //공백이 포함된 단어검색 조건
                                        if(spaceCheck==true && st.hasMoreTokens() &&
secondFindWord.equals(st.nextToken()))
                                        {
                                                indexNum=nextLine.indexOf(findWord)+1;//열 번호
                                                if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                                                {
                                                        item.setIndexNo(indexNum);//열 번호 저장
                                                        item.setLineNo(lineNum);//행 번호 저장
                                                        count++;//검색 횟수 1증가
                                                        mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장

                                                }
                                                indexNum=0;
                                                continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                                        }
                                        else if(spaceCheck==false)
                                        {
                                                indexNum=nextLine.indexOf(findWord)+1;//열 번호
                                                if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                                                {
                                                        item.setIndexNo(indexNum);//열 번호 저장
                                                        item.setLineNo(lineNum);//행 번호 저장
                                                        count++;//검색 횟수 1증가
                                                        mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장

                                                }
                                                indexNum=0;
                                                continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                                        }
                                }
                        }

                        //첫 번째 줄과 두 번째 줄 사이에 개행문자가 포함된 문자열을 찾기 위해서 두 개의
문자열을 연결한다.
                        concatLine1=currentLine+nextLine;
                        st=new StringTokenizer(concatLine1," ");//공백 단위로 파싱

                        while(st.hasMoreTokens())
                        {
                                if(findWord.equals(st.nextToken()))
                                {
```

```
                                        if(spaceCheck==true && st.hasMoreTokens() &&
secondFindWord.equals(st.nextToken()))
                                        {
                                                indexNum=concatLine1.indexOf(findWord)+1;//열 번호
                                                if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                                                {
                                                        item.setIndexNo(indexNum);//열 번호 저장
                                                        item.setLineNo(lineNum);//행 번호 저장
                                                        count++;//검색 횟수 1증가
                                                        mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장
                                                }
                                                indexNum=0;
                                                continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                                        }
                                        else if(spaceCheck==false)
                                        {
                                                indexNum=concatLine1.indexOf(findWord)+1;//열 번호
                                                if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                                                {
                                                        item.setIndexNo(indexNum);//열 번호 저장
                                                        item.setLineNo(lineNum);//행 번호 저장
                                                        count++;//검색 횟수 1증가
                                                        mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장
                                                }
                                                indexNum=0;
                                                continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                                        }
                                }
                        }

                        thirdLine=br.readLine();//두 번째 줄 다음에 세 번째 줄을 읽어온다.
                        if(thirdLine==null)
                                break loop;

                        //두 번째 줄과 세 번째 줄 사이에 개행문자가 포함된 문자열을 찾기 위해서 두 개의
문자열을 연결한다.
                        concatLine2=nextLine+thirdLine;

                        st=new StringTokenizer(concatLine2," ");

                        lineNum++;
                        while(st.hasMoreTokens())
                        {
                                if(findWord.equals(st.nextToken()))
                                {
                                        if(spaceCheck==true && st.hasMoreTokens() &&
secondFindWord.equals(st.nextToken()))
                                        {
                                                indexNum=concatLine2.indexOf(findWord)+1;//열 번호
                                                if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                                                {
                                                        item.setIndexNo(indexNum);//열 번호 저장
```

```
                                                item.setLineNo(lineNum);//행 번호 저장
                                                count++;//검색 횟수 1증가
                                                mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장
                                        }
                                        indexNum=0;
                                        continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                                }
                                else if(spaceCheck==false)
                                {
                                        indexNum=concatLine2.indexOf(findWord)+1;//열 번호
                                        if(indexNum < 129)//한 행의 열은 최대 128까지 설정
                                        {
                                                item.setIndexNo(indexNum);//열 번호 저장
                                                item.setLineNo(lineNum);//행 번호 저장
                                                count++;//검색 횟수 1증가
                                                mArrayList.add(item);//CSearchItem 객체
ArrayList에 저장
                                        }
                                        indexNum=0;
                                        continue loop;//문자열을 탐색하면 무한 루프의 조건으로
jump
                                }
                        }
                }

        }
        resultCount=count;//최종 검색 횟수 멤버변수에 저장
        return mArrayList;//검색결과 반환
    }
  /**
   * param  : MyBufferedReader br - 파일 스트림을 저장
   * param  : String searchWord, 찾을 단어 저장
   * param  : String replaceWord, 변환할 단어 저장
   * param  : String outputPath, 출력 파일경로 저장
   * throws : IOException - 파일 입출력 예외
   * Description : test.txt에 특정 단어를 사용자가 원하는 단어로 변환.
   *            : 개행이 포함된 단어도 수정 가능.
   */
      public static void replaceWord(MyBufferedReader br, String searchWord, String replaceWord, String
outputPath) throws IOException
      {
        String context=null;
        String str=null;
        String result="";
        int ch;
        int i;
        StringBuilder searchTmp=null;
        StringBuilder changeTmp=null;

        BufferedWriter bw=new BufferedWriter (new FileWriter(outputPath));//쓰기 스트림 생성
        String shWord=searchWord;//찾을 단어 입력
        String word=replaceWord;//변환할 단어 입력
```

```
//텍스트파일 첫 번째 단어이면 단어 뒤쪽에만 공백추가
//그 외 단어 앞 뒤에 공백 추가
if(shWord.equals("kLsHxEatRjp")==true)
{
	searchTmp=new StringBuilder(shWord);
	searchTmp.insert(shWord.length(), ' ');
	shWord=searchTmp.toString();

	changeTmp=new StringBuilder(word);
	changeTmp.insert(word.length(), ' ');
	word=changeTmp.toString();
}
else{
	searchTmp=new StringBuilder(shWord);
	searchTmp.insert(0, ' ');
	searchTmp.insert(shWord.length()+1, ' ');
	shWord=searchTmp.toString();

	changeTmp=new StringBuilder(word);
	changeTmp.insert(0, ' ');
	changeTmp.insert(word.length()+1, ' ');
	word=changeTmp.toString();
}


//test.txt 파일에서 개행 단위로 문장을 읽어온다.
//파일의 끝까지 반복
while((context=br.readLine()) != null){

	i=0;
	char[] tmpArr=new char[20];//개행이 포함된 단어를 읽기 위한 배열

	//개행 다음에 단어를 읽어온다.
	//EOF 만나면 실행 안됨
	while((ch=br.read()) != ' ')
	{
		if(ch== -1)
			break;

		tmpArr[i++]=(char)ch;
	}
	//배열에 저장된 단어를 string으로 변환
	//개행 다음에 단어를 문장하고 연결
	str=new String(tmpArr,0, i);
	context += str;
	context = new StringBuilder(context).insert(context.length(), ' ').toString();
	i++;

	//replaceAll함수를 사용해서 단어 변환
	//변환된 문장은 result 변수에 저장
	//result변수가 비어있는지 확인한 후 안 비어있으면 조건 분기 실행
	if((result=context.replaceAll(shWord,word)).equals("")!=true)
	{
		//string result를 StringBuilder로 boxing
		//StringBuilder insert method로 개행 문자 삽입. 첫 번째 param은 문자를
삽입할 위치, 두 번째 param은 삽입할 문자
		//개행 문자를 읽어오기 위해 System.getProperty("line.separator") 사용,
자바에서 지원하는 운영체제별 개행 문자를 가져옴.
		//StringBuilder instance를 String으로 변환한 다음 result 변수에 저장
```

```
                                        result = new StringBuilder(result).insert(result.length()-i,
System.getProperty("line.separator")).toString();
                                        result = result.trim();
                                        bw.write(result);//파일에 변환된 문장 저장
                                        bw.write(" ");//띄어쓰기 저장
                                }

                        }
                }

}
/**
 * SuperClass : Editor
 * SubClass   : ImprovedEditor
 * Append method : 사용자가 원하는 라인에 문자열 추가
 * SearchWord method : 사용자가 원하는 라인에서만 검색
 */
class ImprovedEditor extends Editor{

        /**
         * param : MyBufferedReader br, 파일 스트림 저장
         * param : String appenStr, 삽입할 단어 저장
         * param : String _row, 행
         * param : String _col, 열
         * param : String outputPath, 출력 파일경로 저장
         * throws IOException
         * description : 사용자가 지정한 행과 열에 단어를 삽입한다.
         */
        public static void Append(MyBufferedReader br, String appenStr, String _row, String _col, String
outputPath) throws IOException
        {
                int row, col, i=0;
                String temp;
                StringBuilder bTemp;

                BufferedWriter bw=new BufferedWriter(new FileWriter(outputPath));//쓰기 스트림 생성

                //문자열 앞 뒤에 공백 추가
                bTemp=new StringBuilder(appenStr).insert(0, ' ');
                appenStr=bTemp.insert(appenStr.length()+1, ' ').toString();

                row=Integer.parseInt(_row);//String을 정수로 변환
                col=Integer.parseInt(_col)-1;

                //EOF까지 반복
                while((temp=br.readLine()) != null)
                {
                        i++;
                        if(i==row)//사용자가 지정한 라인에 문자열 추가 후 텍스트 복사
                        {
                                temp=new StringBuilder(temp).insert(col, appenStr).toString();
                        }
                        bw.write(temp);//텍스트 복사
                        bw.newLine();//텍스트파일에 개행 삽입
                }
        }
        /*
        public static void  SearchWord(MyBufferedReader br) throws IOException
        {
```

```
                int line, col;
                String word;
                String temp;
                Scanner sc=new Scanner(System.in);
                System.out.print("찾을 단어를 입력하세요 : ");
                word=sc.nextLine();
                System.out.print("검색할 라인을 입력하세요 : ");
                line=sc.nextInt();

                //사용자가 지정한 라인에서만 검색.
                for(int i=1; i<=line; i++)
                {
                        temp=br.readLine();//file pointer 소비
                        if(i==line)//지정한 위치에 도달하면 검색 실행
                        {
                                if((col=temp.indexOf(word)) != -1)
                                {
                                        col+=1;//텍스프 파일의 열은 1부터 시작하므로 1을 더해준다.
                                        System.out.println("행 : "+line+" 열 : "+col);
                                }
                                else
                                        System.out.println("해당 라인에는 일치하는 단어가 없습니다.");

                        }

                }
        }*/
}
/**
 * superClass : JFrame
 * subClass   : ProjectGui
 * description : JFrame을 상속받아 GUI를 구현
 */
class ProjectGui extends JFrame{
        private static FileInputStream fIn;
        private static ArrayList<ISearchItem> mArrayList;
        private static int resultCount;

        //----------------GUI 변수----------------------
        private JFrame mainFrame;
        private static JTextField inputPath;//입력 경로
        private static JTextField outputPath;//반환 경로
        private static JTextField searchField;
        private static JTextField replaceField;
        private static JTextArea textArea;
        private static JLabel resultCountLabel;
        private static JLabel resultLabel;
        private static JTextField rowField;
        private static JTextField colField;
        //-----------------------------------------

        //ProjectGui 생성자, 생성자에서 GUI를 구현
        public ProjectGui(String title){
                super(title);

                mainFrame=new JFrame("임준수_2010111661");//프레임 생성
                mainFrame.setSize(700, 450);//프레임 사이즈
                mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//종료 설정
                mainFrame.setLayout(new FlowLayout(FlowLayout.LEFT,10,10));
                mainFrame.addWindowListener(new WindowAdapter() {
```

```
        public void windowClosing(WindowEvent windowEvent){
                System.exit(0);
        }
});

//메뉴바 구현
JMenuBar menuBar = new JMenuBar();
JMenu fileMenu=new JMenu("파일");
JMenu toolMenu=new JMenu("편집");
JMenuItem openItem=new JMenuItem("열기");
fileMenu.add(openItem);
openItem.addActionListener(new FileOpenAction());
JMenuItem printItem=new JMenuItem("출력");
fileMenu.add(printItem);
printItem.addActionListener(new PrintAction());
JMenuItem searchItem=new JMenuItem("검색");
toolMenu.add(searchItem);
searchItem.addActionListener(new SearchAction());
JMenuItem replaceItem=new JMenuItem("변환");
toolMenu.add(replaceItem);
replaceItem.addActionListener(new ReplaceAction());
menuBar.add(fileMenu);
menuBar.add(toolMenu);
mainFrame.setJMenuBar(menuBar);//메뉴바 프레임에 삽입

//1번째 패널 구현
JPanel panel=new JPanel();
panel.setLayout(new FlowLayout(FlowLayout.LEFT,10,0));
panel.add(new JLabel("검색할 단어:"));
searchField=new JTextField(15);//텍스트 필드
panel.add(searchField);
panel.add(new JLabel("변환할 단어:"));
replaceField=new JTextField(15);
panel.add(replaceField);
JButton searchButton=new JButton("검색");
panel.add(searchButton);
searchButton.addActionListener(new SearchAction());
JButton replaceButton=new JButton("변환");
panel.add(replaceButton);
replaceButton.addActionListener(new ReplaceAction());
mainFrame.add(panel);//패널 프레임에 삽입

//2번째 패널 구현
JPanel panel2=new JPanel();
panel2.setLayout(new FlowLayout(FlowLayout.LEFT,10,0));
panel2.add(new JLabel("삽입할 단어 위치:"));
panel2.add(new JLabel("행"));
rowField=new JTextField(5);
panel2.add(rowField);
panel2.add(new JLabel("열"));
colField=new JTextField(5);
panel2.add(colField);
JButton insertButton=new JButton("삽입");
panel2.add(insertButton);
insertButton.addActionListener(new InsertAction());
mainFrame.add(panel2);
```

```
		//3번째 패널 구현
		JPanel panel3=new JPanel();
		panel3.setLayout(new FlowLayout(FlowLayout.LEFT,10,0));
		panel3.add(new JLabel("입력 파일명:"));
		inputPath=new JTextField(25);
		panel3.add(inputPath);
		JButton openButton=new JButton("열기");
		panel3.add(openButton);
	openButton.addActionListener(new FileOpenAction());
		resultCountLabel=new JLabel("검색결과:");
		panel3.add(resultCountLabel);
		mainFrame.add(panel3);

		//4번째 패널 구현
		JPanel panel4=new JPanel();
		panel4.setLayout(new FlowLayout());
		textArea=new JTextArea(10,55);//텍스트 area 삽입
		JScrollPane scrollPane=new JScrollPane(textArea);//텍스트 area에 스크롤 삽입
		panel4.add(scrollPane);
		mainFrame.add(panel4);

		//5번째 패널 구현
		JPanel panel5=new JPanel();
		panel5.setLayout(new FlowLayout(FlowLayout.LEFT,10,0));
		resultLabel= new JLabel("검색준비");
		panel5.add(resultLabel);
		panel5.add(new JLabel("출력 파일명:"));
		outputPath=new JTextField(25);
		panel5.add(outputPath);
		JButton printButton=new JButton("결과출력");
		panel5.add(printButton);
		printButton.addActionListener(new PrintAction());
		mainFrame.add(panel5);

		mainFrame.setVisible(true);//GUI 출력
	}

	/**
	 * ActionListener 인터페이스 구현
	 * Description : 열기 버튼 또는 메뉴-파일-열기를 누르면 실행
	 */
	class FileOpenAction implements ActionListener
	{
		@Override
		public void actionPerformed(ActionEvent e) {
			try {
				userConsole("o");//loadFile 실행
				textArea.setText("파일이 로드되었습니다. 원하시는 기능을 선택하세요\n");
				textArea.append("* 단어검색은 검색할 단어를 입력한 후 검색버튼 클릭 또는
메뉴-편집-검색 선택\n");
				textArea.append("* 단어변환은 검색할 단어, 변환할 단어, 출력 파일명을
입력한 후 변환버튼 클릭 또는 메뉴-편집-변환 선택\n");
				textArea.append("* 단어삽입은 변환할 단어, 행, 열, 출력 파일명을 입력한 후
삽입버튼 클릭\n");
				textArea.append("* 결과출력은 검색할 단어, 출력 파일명을 입력한 후 결과출력
버튼 클릭 또는 메뉴-파일-출력 선택\n");
```

```
                        } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                        resultLabel.setText("검색준비");
                        resultCountLabel.setText("검색결과:");
                }
        }

        /**
         * ActionListener 인터페이스 구현
         * Description : 검색버튼 또는 메뉴-편집-검색을 누르면 실행
         */
        class SearchAction implements ActionListener
        {
                @Override
                public void actionPerformed(ActionEvent e) {
                        try {
                                textArea.append("단어검색\n");
                                userConsole("s");//searchWord 실행
                                for(ISearchItem item : mArrayList)//검색 결과 콘솔화면 출력
                                {
                                        textArea.append("행 : "+item.getLineNo()+" 열 :
"+item.getIndexNo()+"\n");
                                }
                                resultCountLabel.setText("검색결과: "+resultCount);
                                resultLabel.setText("검색완료");

                        } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                        textArea.append("단어검색 완료\n");
                }
        }

        /**
         * ActionListener 인터페이스 구현
         * Description : 변환버튼 또는 메뉴-편집-변환을 누르면 실행
         */
        class ReplaceAction implements ActionListener
        {
                @Override
                public void actionPerformed(ActionEvent e){
                        textArea.append("단어변환\n");
                        try {
                                userConsole("r");//replaceWord 실행
                        } catch (IOException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                        textArea.append("단어변환 완료\n");
                        resultLabel.setText("검색완료");
                }
        }

        /**
         * ActionListener 인터페이스 구현
```

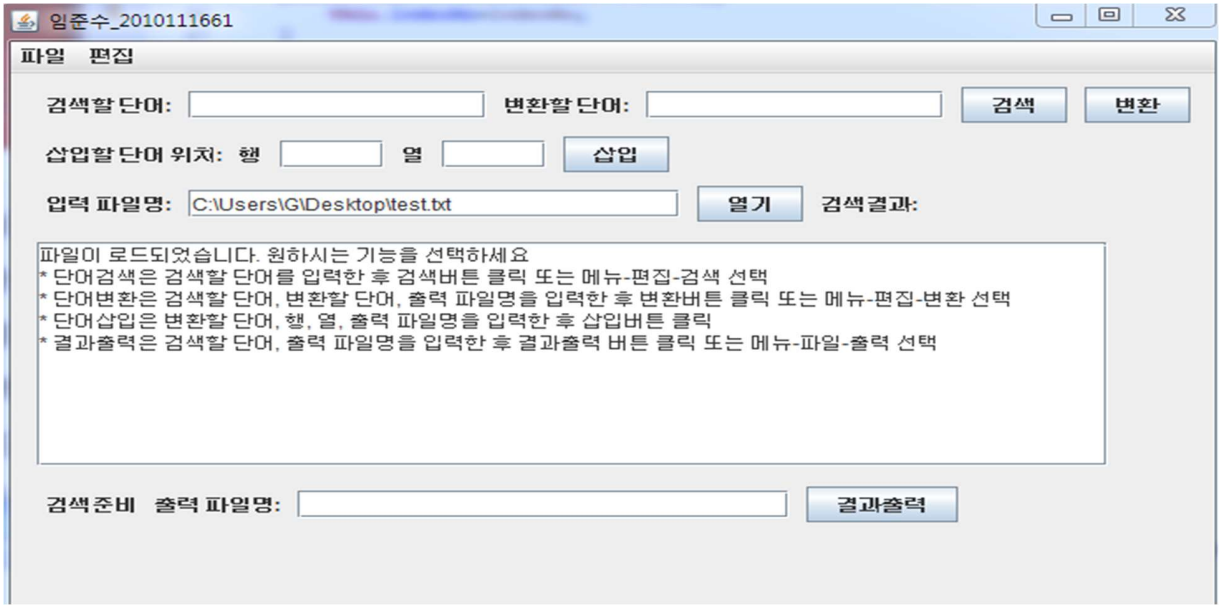
```
	 * Description : 삽입버튼을 누르면 실행
	 */
	class InsertAction implements ActionListener
	{
		@Override
		public void actionPerformed(ActionEvent e){
			textArea.append("단어삽입\n");
			try {
				userConsole("a");//Append 실행
			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}
			textArea.append("단어삽입 완료\n");
			resultLabel.setText("검색완료");
		}
	}

	/**
	 * ActionListener 인터페이스 구현
	 * Description : 결과출력 버튼을 누르거나 메뉴-파일-출력을 누르면 실행
	 */
	class PrintAction implements ActionListener
	{
		@Override
		public void actionPerformed(ActionEvent e){
			textArea.append("검색결과 출력\n");
			try {
				userConsole("p");//printWord 실행
				for(ISearchItem item : mArrayList)//검색 결과 콘솔화면 출력
				{
					textArea.append("행 : "+item.getLineNo()+" 열 :
"+item.getIndexNo()+"\n");
				}
				resultCountLabel.setText("검색결과: "+resultCount);
			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}
			textArea.append("검색결과 출력 완료\n");
			resultLabel.setText("검색완료");
		}
	}

	/**
	 * param	: String selectMenu, 메뉴 선택 단어 저장
	 * throws	: IOException, 파일 입출력 예외처리
	 * return	: void
	 * Description : 컴포넌트에 event listener가 실행되면서 userConsole기능 선택
	 */
	public static void userConsole(String selectMenu) throws IOException
	{
		MyBufferedReader br = null;
		ImprovedEditor imEditor=null;
		Editor editor = new Editor();//Editor instance 생성

			switch(selectMenu){
			case "O": case "o"://파일 로드
```

```
                                fIn=editor.loadFile(inputPath.getText());//editor 인스턴스에 메소드 호출
                                break;
                        case "S": case "s"://문자열 검색
                                fIn.getChannel().position(0);//파일 포인터를 시작 위치로 조정
                                br=new MyBufferedReader(new InputStreamReader(fIn));//버퍼를 이용해서
텍스트를 읽어오는 br 참조형 변수 선언
                                mArrayList=editor.searchWord(br, searchField.getText());
                                resultCount=editor.resultCount;
                                break;
                        case "P": case "p"://검색결과 저장
                                fIn.getChannel().position(0);//파일 포인터를 시작 위치로 조정
                                br=new MyBufferedReader(new InputStreamReader(fIn));//버퍼를 이용해서
텍스트를 읽어오는 br 참조형 변수 선언
                                mArrayList=editor.printWord(br, searchField.getText(),
outputPath.getText());
                                resultCount=editor.resultCount;
                                break;
                        case "R": case "r"://text파일에서 특정 단어 검색 후 사용자가 원하는 단어로 변환
                                fIn.getChannel().position(0);
                                br=new MyBufferedReader(new InputStreamReader(fIn));//버퍼를 이용해서
텍스트를 읽어오는 br 참조형 변수 선언
                                editor.replaceWord(br, searchField.getText(), replaceField.getText(),
outputPath.getText());
                                break;
                        case "A": case "a"://문자열 삽입 기능
                                fIn.getChannel().position(0);
                                br=new MyBufferedReader(new InputStreamReader(fIn));//버퍼를 이용해서
텍스트를 읽어오는 br 참조형 변수 선언
                                imEditor=new ImprovedEditor();//ImprovedEditor instance 생성
                                imEditor.Append(br, replaceField.getText(), rowField.getText(),
colField.getText(), outputPath.getText());
                                break;
                        case "U": case "u"://특정 위치에서만 검색
                                //fIn.getChannel().position(0);
                                //br=new MyBufferedReader(new InputStreamReader(fIn));//버퍼를 이용해서
텍스트를 읽어오는 br 참조형 변수 선언
                                //imEditor=new ImprovedEditor();
                                //imEditor.SearchWord(br);
                                //break;
                        case "Q": case "q"://프로그램 종료
                                System.out.println("프로그램을 종료합니다.");
                                br.close();//스트림 종료
                                return;
                        default://예외처리
                                System.out.println("알수 없는 명령입니다.다시 입력해주세요");
                        }

        }
}
/**
 * main method : 프로그램 실행
 * Description : main을 가지는 class
 */
public class EditorTest {
```

```
	/**
	 * param       : String[] args
	 * throws      : IOException, 파일 입출력 예외처리
	 * return      : void
	 * Description : ProjectGui 인스턴스 생성
	 */
	public static void main(String[] args) throws IOException {
		ProjectGui gui=new ProjectGui(" ");
	}

}
```

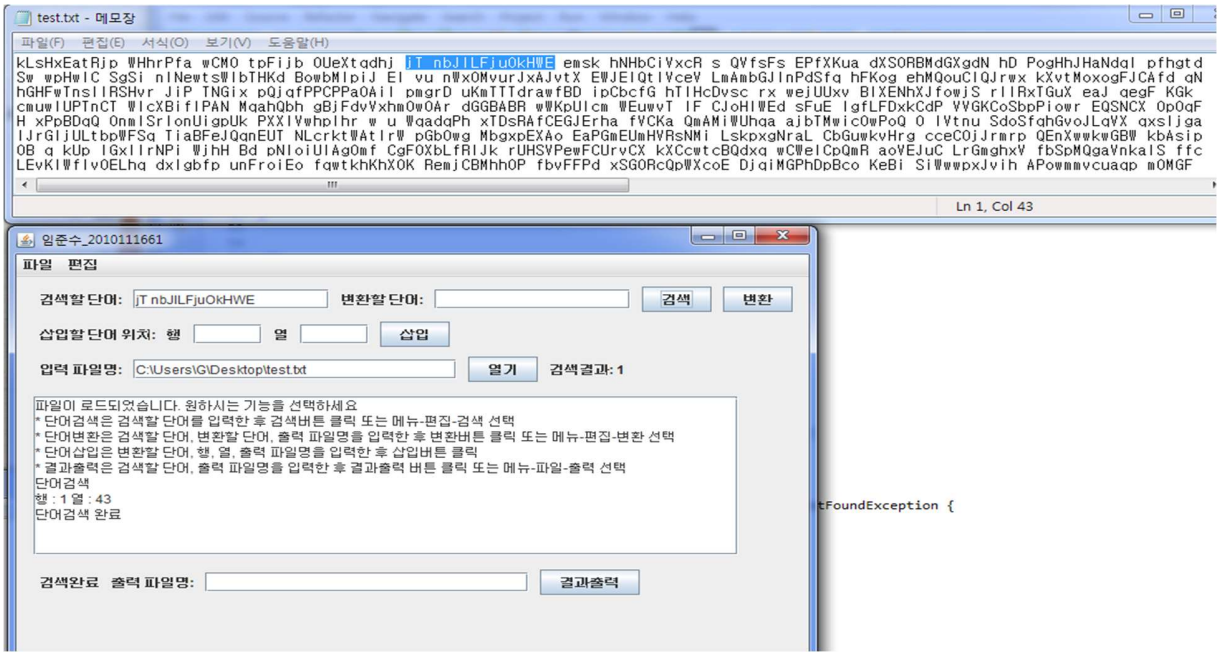
결과 / 결과 분석 :

***프로그램을 실행하고 입력파일명을 입력한 다음 열기 또는 메뉴-파일-열기를 눌렀을 때의 결과화면**



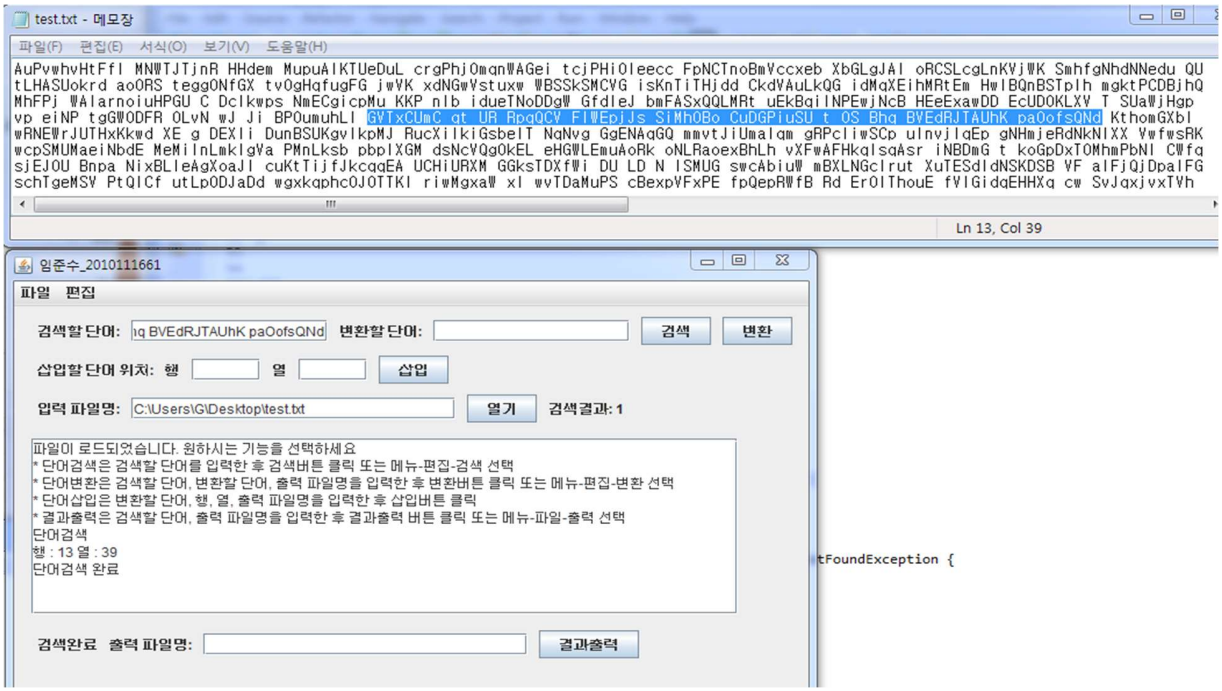
main 메소드가 실행되면 ProjectGui 객체 gui 를 생성한다. Gui 객체가 생성될 때 ProjectGui 의 생성자가 호출되는데 생성자 내부에서 GUI 를 만든다. 그런 다음 입력파일명에 파일경로를 입력한 다음 열기 또는 메뉴-파일-열기를 누르면 FileOpenAction 이벤트처리기가 생성이 되고 해당 처리기안에 actionPerformed 메소드에서 userConsole("o")를 실행한다. 그러면 userConsole 메소드에서는 editor.loadFile()을 호출하고 loadFile 은 파일 스트림을 반환한다. 위와 같은 과정을 모두 거친 다음에 출력되는 결과화면이다.

***공백이 포함된 문자열 검색**

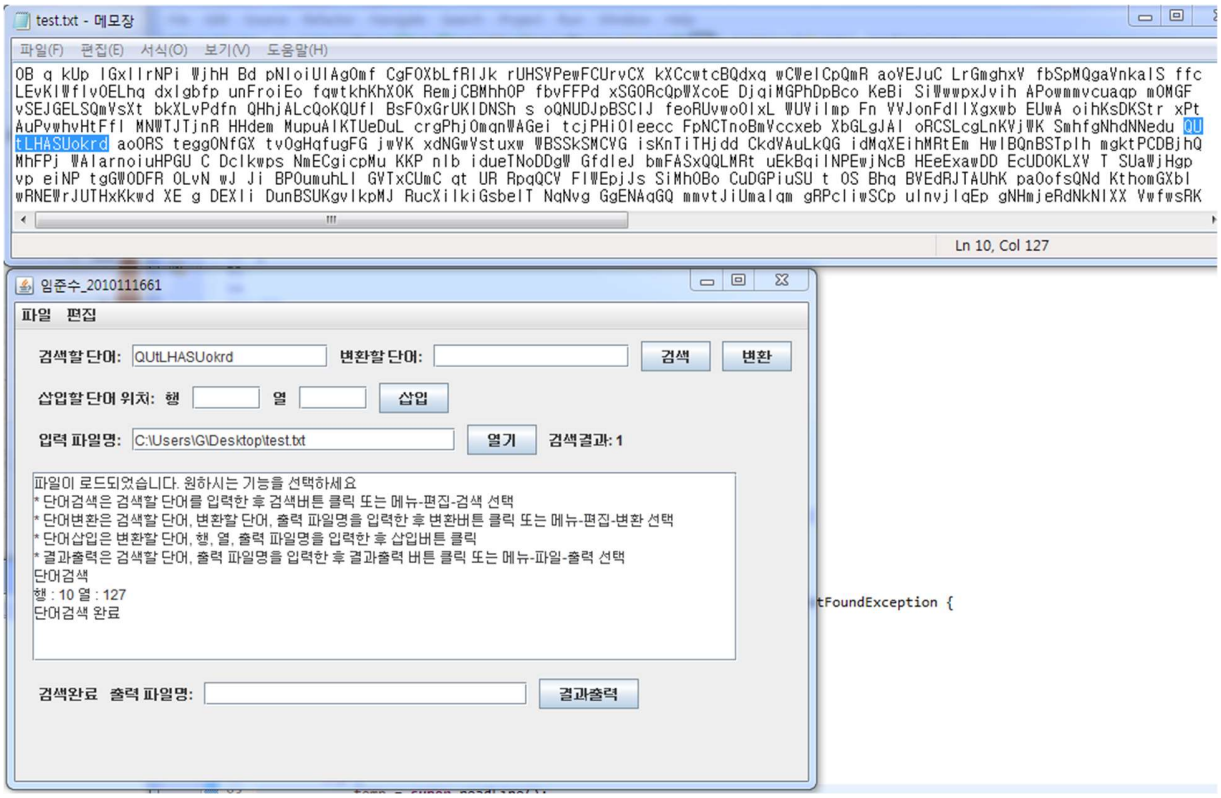


입력파일명을 입력한 다음 열기를 누른 후에 검색할 단어를 입력한 후 검색 또는 메뉴-편집-검색을 누르면 SearchAction class 가 생성된다. 해당 클래스안에 actionPerformed 에서 userConsole("s")를 실행한다. 그러면 userConsole 메소드에서는 switch 분기에 따라 "s" case 가 실행이 되고 해당 분기에서 editor.searchWord 를 호출한다. searchWord 가 종료되면 ArrayList 가 반환되는데 반환된 ArrayList 를 클래스 멤버변수 mArrayList 에 저장하고 검색결과 개수인 editor.resultCount;도 클래스의 멤버변수 resultCount 변수에 저장한다. 그런 다음 actionPerformed 로 PC 가 이동해서 개선된 for 문으로 mArrayList 에 저장된 값을 리스트에 출력한다. 그리고 나서 resultCount 의 값을 resultCountLabel 에 출력하기 때문에 위에 결과화면 열기버튼 옆에 검색결과: 1 이 출력된다. 또한 파일을 열었을 때 출력파일명 옆에 검색결과 레이블은 검색준비였는데 검색을 끝난 후에는 검색완료로 변경된다.

***긴 문자열 검색결과**

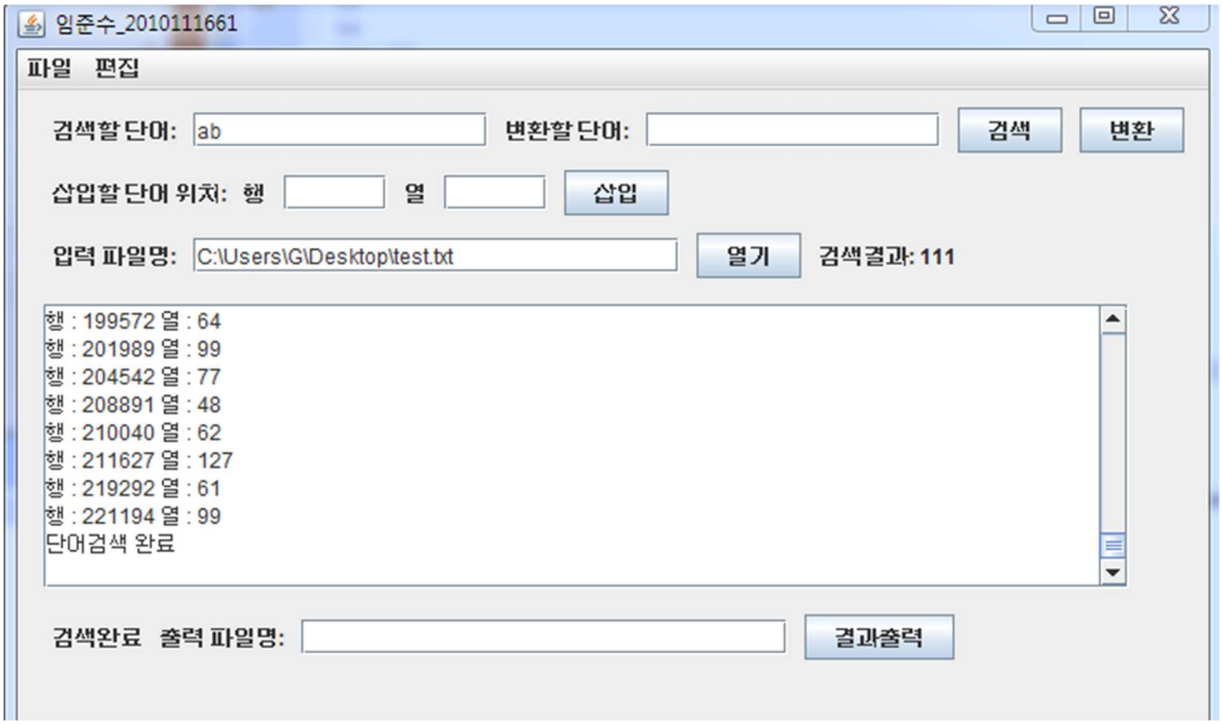


***개행문자가 포함된 문자열 검색 결과**



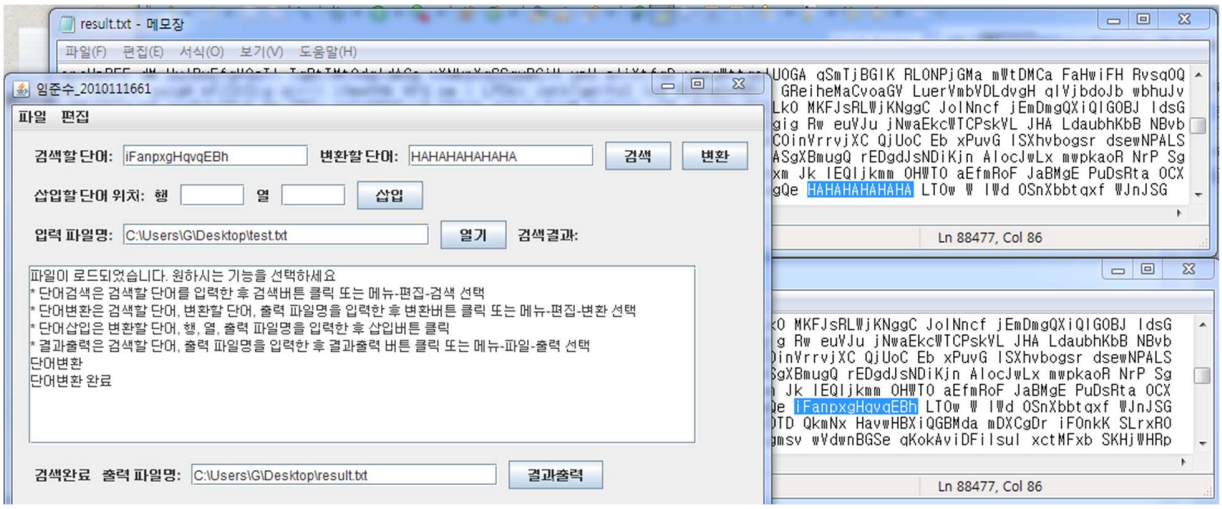
위에 검색결과 화면의 동작은 공백이 포함된 문자열 검색과 동일하다.

***검색결과가 10 개이상이면 리스트에 스크롤 바가 생성됨을 보여주는 결과화면**



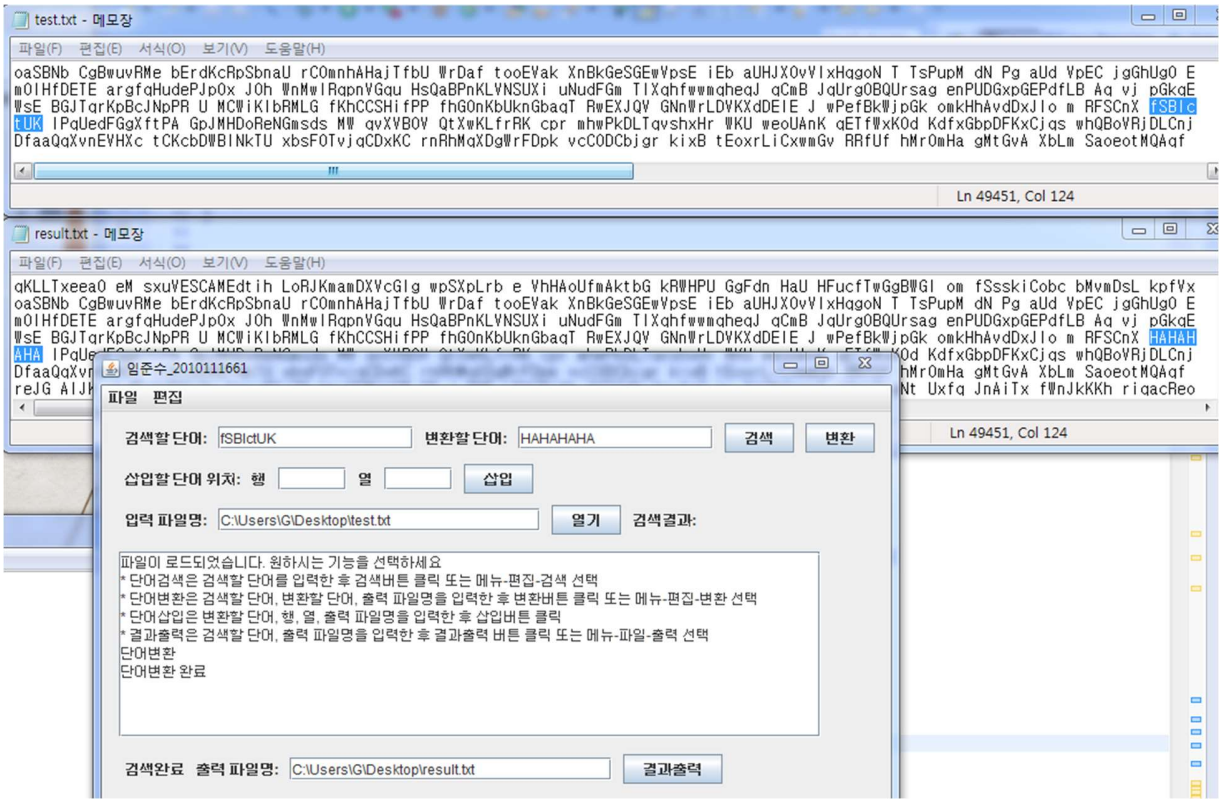
프레임에 `textArea` 를 삽입할 때 JScrollPane scrollPane=**new** JScrollPane(*textArea*); `panel4.add(scrollPane);` `JScrollPane` 으로 박싱한 다음 패널에 삽입하기 때문에 기본 `textArea` 의 크기 `10,55` 을 넘어가는 검색결과에는 스크롤 바가 생성된다.

***단어변환 결과화면**



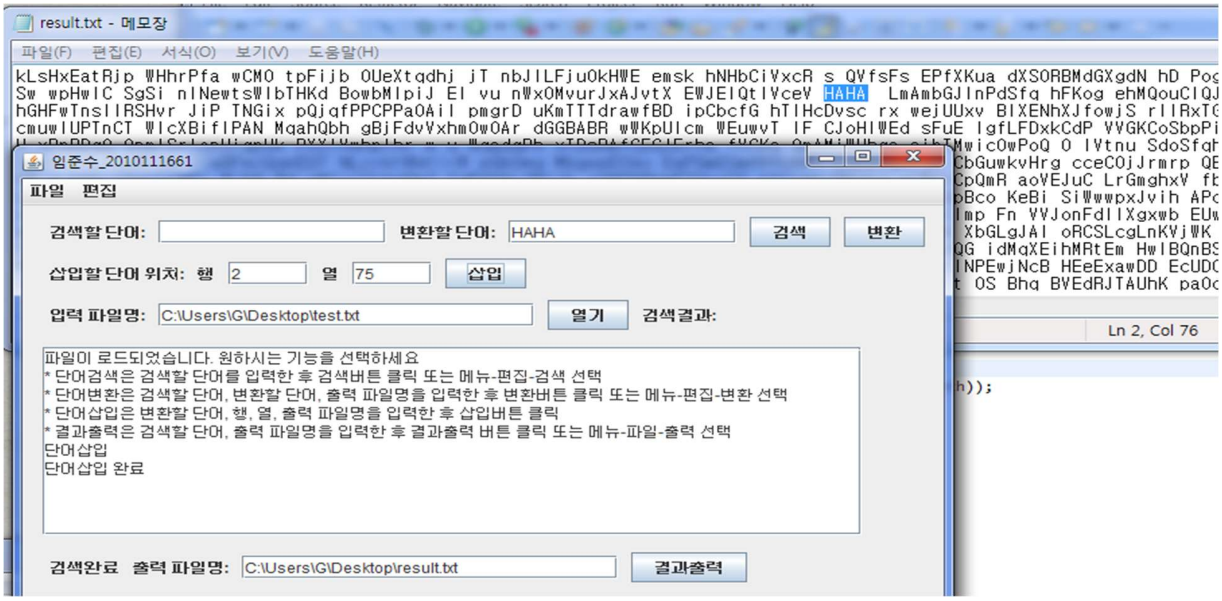
파일을 load 한 후에 검색할 단어와 변환할 단어, 출력파일명을 입력한 다음 변환버튼 또는 메뉴-편집-변환을 누르면 ReplaceAction 클래스가 실행된다. actionPerformed 에서는 userConsole("r");을 실행하고 userConsole 메소드에서는 분기에 의해 editor.replaceWord();를 호출한다. replaceWord 메소드가 실행되고 나면 출력파일명의 경로에 검색할 단어가 변환할 단어로 변환돼서 텍스트파일이 생성된다.

***개행이 포함된 단어 변환 결과화면**



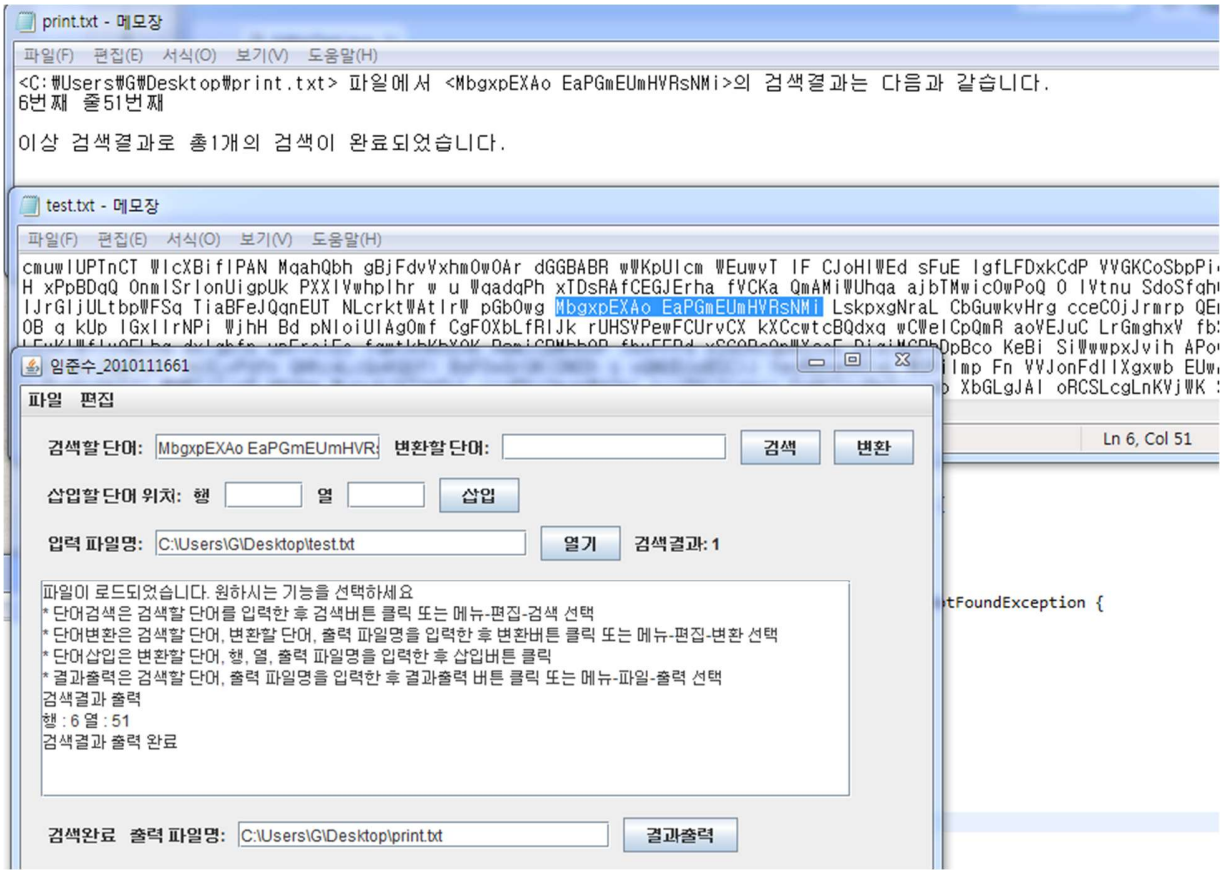
단어변환 과정과 동일하게 검색할 단어, 변환할 단어, 출력파일명을 입력한 다음 변환 버튼이나 메뉴-편집-변환을 누르면 단어변환이 된다. 개행문자가 포함되어 있어도 replaceWord 에서 개행문자를 처리해서 단어를 변환해준다.

***단어삽입 결과화면**



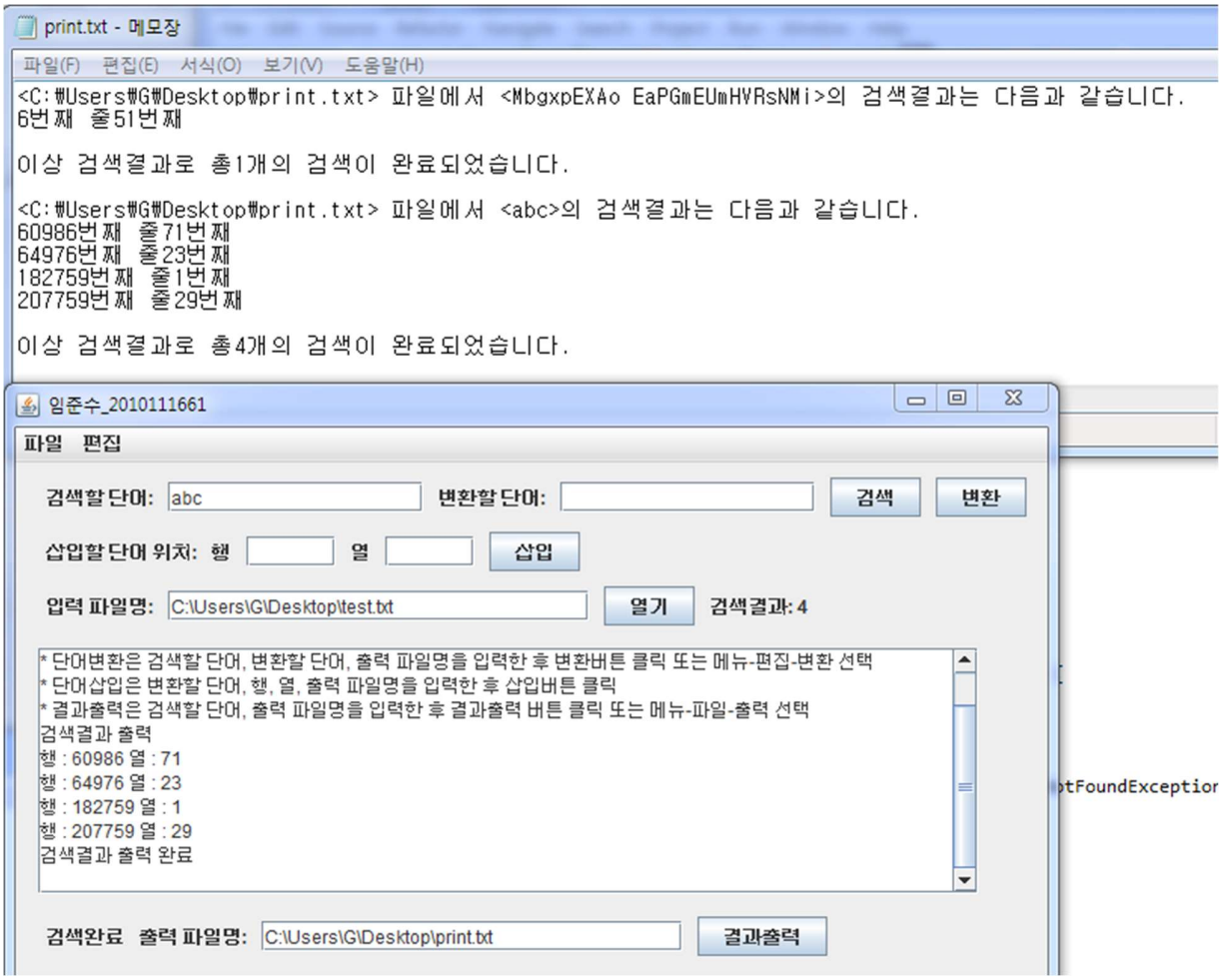
파일을 open 한 후에 변환할 단어 텍스트필드에 단어를 입력하고 행, 열, 출력파일명을 입력한 다음 삽입버튼을 누르면 InsertAction 이벤트처리기가 실행된다. InsertAction 이벤트 처리기에 actionPerformed 메소드에서는 userConsole("a");를 호출한다. 그러면 userConsole 메소드에서는 분기에 의해 case "a"로 분기되고 해당 분기안에서 ImprovedEditor 객체 imEditor 를 생성한 다음 imEditor.Append();를 호출한다. Append 함수 내부에서는 인자로 전달된 변환할 단어와 행, 열을 이용해서 텍스트파일에 단어를 삽입한다. 위에 결과화면을 보면 GUI 에서는 2 행 75 열에 단어를 삽입했는데 텍스트파일에서는 2 행 76 열에 삽입되었다. 왜냐하면 단어를 삽입할 때 단어임을 구분하기 위해서 단어 양옆에 공백을 추가하기 때문에 텍스트파일에 단어를 삽입하면 앞에 공백이 삽입돼서 75 열에서 76 열로 바뀌게 된다.

***검색결과 출력기능 결과화면**



파일을 open 한 후에 검색할 단어와 출력파일명을 입력한 후에 결과출력 버튼 또는 메뉴-파일-출력을 누르면 PrintAction 이벤트처리기가 실행된다. 해당 클래스 내부에서 actionPerformed 메소드는 userConsole("p"); 를 호출하고 userConsole 메소드에서는 분기에 의해 case "p" 분기를 실행한다. 분기 안에서는 editor.printWord()메소드를 호출하고 printWord 메소드에서는 검색할 단어를 searchMethod 를 호출해서 단어를 검색하고 검색결과를 mArrayList 에 저장한다. 그런 다음 FileWriter 를 사용해서 mArrayList 에 저장되어있는 검색결과를 출력파일명의 파일경로에 텍스트파일을 생성한 다음 결과를 저장한다. GUI 에서는 검색결과가 리스트에 출력되고 resultCountLabel 에 검색결과: 1 이 표시된다. resultLabel 도 검색준비에서 검색완료로 변경되었다.

***검색결과는 append 모드로 저장됨을 보여주는 결과화면**



검색결과는 텍스트파일에 저장할 때 출력파일경로에 이미 동일한 이름의 텍스트파일이 있으면 텍스트파일의 내용을 지운 다음에 검색결과를 저장하는 것이 아니라 기존 내용을 보존하고 그 다음부터 검색결과를 추가 저장한다. 만약 출력파일경로에 동일한 이름의 텍스트파일이 없으면 텍스트파일을 새로 생성해서 검색결과를 저장한다.