CodeConverter.java
CodeParser.java
CodePrinter.java
ConvertPrograssEvent.java
CopyRightEvent.java
IConvertPrograssListener.java
KeywordSet.java
Token.java
Convert.java
Copyright.java
FileSelect.java
JExtendTextField.java
Logo.java
Main.java
Secure.java
Setting.java
Style.java
Checker.java
FontOption.java
Option.java
OptionDefault.java
OptionFile.java
PageOption.java
Security.java

CodeConverter.java

```
1  package converter;
2  import java.io.File;
3  import java.io.IOException;
4  import java.util.EventListener;
5  import java.util.Vector;
6
7  import javax.swing.event.EventListenerList;
8
9  import com.itextpdf.text.DocumentException;
10
11  import loader.Checker;
12  import option.*;
13
14  /**
15   * 소스코드를 PDF로 변환하기 위한 클래스 입니다.<br>
16   * 이 클래스는 분석기,출력기와 UI 사이에서 동작하며 중간 작업들을 처리합니다.<p>
17   * 이 클래스와 하위 Parser, Printer는 별도의 스레드로 동작하며 이벤트를 발생시킵니다.
18   * @author 강승민
19   *
20   */
21  public class CodeConverter extends Thread{
22
23      /**
24       * 코드 분석기
25       */
26      private CodeParser parser;
27
28      /**
29       * 코드 출력기
30       */
31      private CodePrinter printer;
32
33      /**
34       * 분석기에서 생성된 토큰 벡터
35       */
36      private Vector<Vector<Vector<Token>>> codeToken = new Vector<Vector<Vector<Token>>>();
37
38      /**
39       * 파일 정보를 가져올 Checker입니다.
40       */
41      private Checker checker;
42
43      /**
44       * 옵션 정보 속성
45       */
46      private Option option;
47
```

```
48      /**
49       * 보안 정보 속성
50       */
51      private Security security;
52
53      /**
54       * 변환 이벤트 처리 리스너 리스트
55       */
56      private EventListenerList listenerList = new EventListenerList();
57
58      /**
59       * 현재 변환 진행률
60       */
61      private int currentPrograss = 0;
62
63      /**
64       * 전체 변환량
65       */
66      private int totalPrograss;
67
68      /**
69       * CodeConverter 생성자
70       * @param chker Checker
71       * @param op Option
72       * @param sec Security
73       */
74      public CodeConverter(Checker chker, Option op, Security sec){
75          checker = chker;
76          option = op;
77          security = sec;
78          parser = new CodeParser(option);
79          printer = new CodePrinter(option, security);
80          totalPrograss = 0;
81          for(int i = 0 ; i < checker.fileList.size() ; i++){
82              totalPrograss += checker.fileList.get(i).size();
83          }
84          if(option.getConvertFile()){
85              for(int i = 0 ; i < checker.fileList.size() ; i++){
86                  totalPrograss += checker.fileList.get(i).size();
87              }
88          }
89          if(option.getConvertPackage() && checker.dirList.size() > 0){
90              totalPrograss += checker.dirList.size();
91          }
92          if(option.getConvertAll() && checker.dirList.size() > 0){
93              totalPrograss++;
94          }
95      }
```

```
96
97          /**
98           * 변환 작업을 시작합니다.
99           */
100          private void convert(){
101              int size = checker.fileList.size();
102              if(checker.dirList.size() == 0){
103                  Vector<File> fVector = checker.fileList.get(0);
104                  for(int j = 0 ; j < fVector.size() ; j++){
105                      try {
106                          Vector<Vector<Token>> packageToken = new Vector<Vector<Token>>();
107                          Vector<Token> parseResult = parser.parse(fVector.get(j));
108                          packageToken.add(parseResult);
109                          codeToken.addElement(packageToken);
110                          currentPrograss++;
111                          ConvertPrograssEvent evt = new ConvertPrograssEvent(this, currentPrograss,
totalPrograss);
112                          fireConvertPrograssEvent(evt);
113                      } catch (IOException e) {
114                          e.printStackTrace();
115                      }
116                  }
117              }else{
118                  for(int i = 0 ; i < size ; i++){
119                      Vector<File> fVector = checker.fileList.get(i);
120                      Vector<Vector<Token>> packageToken = new Vector<Vector<Token>>();
121                      for(int j = 0 ; j < fVector.size() ; j++){
122                          try {
123                              Vector<Token> parseResult = parser.parse(fVector.get(j));
124                              packageToken.add(parseResult);
125                              currentPrograss++;
126                              ConvertPrograssEvent evt = new ConvertPrograssEvent(this, currentPrograss,
totalPrograss);
127                              fireConvertPrograssEvent(evt);
128                          } catch (IOException e) {
129                              e.printStackTrace();
130                          }
131                      }
132                      codeToken.addElement(packageToken);
133                  }
134              }
135
136              if(checker.dirList.size() == 0){
137                  convertFile();
138              }else{
139                  if(option.getConvertAll()){
140                      convertAll();
141                  }
```

```
142             if(option.getConvertPackage()){
143                 convertPackage();
144             }
145             if(option.getConvertFile()){
146                 convertFile();
147             }
148         }
149     }
150 
151     /**
152      * 변환 진행 이벤트 리스너를 추가합니다.
153      * @param listener 추가 할 이벤트 리스너
154      */
155     public void addConvertPrograssListener(IConvertPrograssListener listener){
156         listenerList.add(IConvertPrograssListener.class, listener);
157     }
158 
159     /**
160      * 변환 진행 이벤트 리스너를 제거합니다.
161      * @param listener 제거 할 이벤트 리스너
162      */
163     public void removeConvertPrograssListener(IConvertPrograssListener listener){
164         listenerList.remove(IConvertPrograssListener.class, listener);
165     }
166 
167     /**
168      * 변환 이벤트를 발생시킵니다.
169      * @param event 변환 이벤트
170      */
171     private void fireConvertPrograssEvent(ConvertPrograssEvent event){
172         EventListener[] listeners = listenerList.getListeners(IConvertPrograssListener.class);
173         for(int i = 0 ; i < listeners.length ; i++){
174             ((IConvertPrograssListener)listeners[i]).convertPrograss((ConvertPrograssEvent)event);
175         }
176     }
177 
178 
179     /**
180      * 로드된 모든 파일을 한개의 PDF 문서로 만듭니다.
181      */
182     private void convertAll(){
183         try {
184             Vector<String> subName = new Vector<String>();
185             for(int i = 0 ; i < checker.fileList.size() ; i++){
186                 for(int j = 0 ; j < checker.fileList.get(i).size() ; j++){
187                     subName.add(checker.fileList.get(i).get(j).getName());
188                 }
189             }
```

```
190             if(checker.dirList.size() != 0){
191                 printer.printAll(codeToken, subName);
192             }
193             currentPrograss++;
194             ConvertPrograssEvent evt = new ConvertPrograssEvent(this, currentPrograss, totalPrograss);
195             fireConvertPrograssEvent(evt);
196         } catch (DocumentException | IOException e) {
197             e.printStackTrace();
198         }
199     }
200 
201     /**
202      * 로드된 파일을 패키지 단위로 PDF 문서로 만듭니다.
203      */
204     private void convertPackage(){
205         int size = codeToken.size();
206         for(int i = 0 ; i < size ; i++){
207             Vector<Vector<Token>> packageToken = codeToken.get(i);
208             Vector<File> fVector = checker.fileList.get(i);
209             Vector<String> subName = new Vector<String>();
210             for(int j = 0 ; j < fVector.size() ; j++){
211                 subName.add(fVector.get(j).getName());
212             }
213             try {
214                 printer.printPackage(packageToken, checker.dirList.get(i).getName(), subName);
215                 currentPrograss++;
216                 ConvertPrograssEvent evt = new ConvertPrograssEvent(this, currentPrograss, totalPrograss);
217                 fireConvertPrograssEvent(evt);
218             } catch (DocumentException | IOException e) {
219                 e.printStackTrace();
220             }
221         }
222     }
223 
224     /**
225      * 로드된 파일을 각 파일 단위로 PDF 문서로 만듭니다.
226      */
227     private void convertFile(){
228         int size = codeToken.size();
229         for(int i = 0 ; i < size ; i++){
230             Vector<File> fVector = checker.fileList.get(i);
231             Vector<Vector<Token>> packageToken = codeToken.get(i);
232             for(int j = 0 ; j < packageToken.size() ; j++){
233                 try {
234                     if(packageToken.get(j) != null){
235                         String tempHeader = option.getHeaderTitle();
236                         option.setHeaderTitle(fVector.get(j).getName());
237                         printer.printSingleFile(packageToken.get(j), fVector.get(j).getName());
```

```
238                         option.setHeaderTitle(tempHeader);
239                     }
240                     currentPrograss++;
241                     ConvertPrograssEvent evt = new ConvertPrograssEvent(this, currentPrograss,
totalPrograss);
242                     fireConvertPrograssEvent(evt);
243                 } catch (DocumentException | IOException e) {
244                     e.printStackTrace();
245                 }
246             }
247
248         }
249     }
250
251     /**
252      * 변환 진행 스레드를 시작합니다.
253      */
254     @Override
255     public void run() {
256         convert();
257     }
258 }
259
```

CodeParser.java

```java
package converter;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.Vector;

import option.Option;

 /**
  * 주어진 소스코드 파일을 분석합니다.
  * @author 강승민
  */
 public class CodeParser {
    /**
     * 임시 저장용 토큰
     */
    private Vector<Token> tmpTokens = new Vector<Token>();
    /**
     * 최종 변환된 토큰
     */
    private Vector<Token> tokens = new Vector<Token>();

    /**
     * 옵션 값
     */
    private Option option;

    /**
     * 생성자
     * @param op 옵션 정보입니다.
     */
    public CodeParser(Option op){
        option = op;
    }

    /**
     * 소스코드 파일을 분석합니다.
     * @param code 분석할 소스코드 파일
     * @return 분석이 끝난 소스코드 파일의 토큰
     * @throws IOException 파일 읽기에 실패 했을 때 발생합니다.
     */
    @SuppressWarnings("resource")
    public Vector<Token> parse(File code) throws IOException{
        // 토큰 리스트를 초기화
        tmpTokens = new Vector<Token>();
        tokens = new Vector<Token>();
```

```
48          BufferedReader br = null;
49          try {
50              br = new BufferedReader(new FileReader(code));
51          } catch (IOException e) {
52              Token t = new Token();
53              t.setType(Token.tokenType.NORMAL);
54              t.setData("파일이 손상되었거나 접근 권한이 없어 변환에 실패하였습니다.");
55              tokens.add(t);
56              return tokens;
57          }
58
59          Token result = new Token();
60          String s = "";
61          String tabCharactor = "";
62          for(int i = 0 ; i < option.getTabSize() ; i++){
63              tabCharactor += " ";
64          }
65
66          while((s = br.readLine()) != null){
67              s += "₩n";
68              // PDF에서 ₩t 문자가 지원되지 않으므로 적절한 길이의 공백으로 치환합니다.
69              s = s.replaceAll("₩t", tabCharactor);
70              int index = 0;
71              if(result.getType() == Token.tokenType.MULTIPLE_LINE_STRING || result.getType() ==
Token.tokenType.MULTIPLE_LINE_COMMENT){ //이전 토큰이 문자열인 경우 계속해서 문자열 토큰으로 처리하도
록 지정
72                  result = getIndent(s,index,result.getType()); //토큰 분할
73              }else{
74                  result = getIndent(s,index,Token.tokenType.NORMAL); //토큰 분할
75              }
76              while(result.getData().length() > 0 && result != null){ //불할 된 문자열 길이가 1 이상이고 null이 아
닌 경우 계속 분할 시도
77                  tmpTokens.add(result); //분할된 문자열을 리스트에 저장
78                  index += result.getData().length(); //문자열의 인덱스를 증가시켜 최초 분할 이후의 계속 분할
79                  if(result.getType() == Token.tokenType.MULTIPLE_LINE_STRING){ //이전 토큰이 문자열인 경우
계속해서 문자열 토큰으로 처리하도록 지정
80                      result = new Token(); //새롭게 분할된 토큰을 생성
81                      result = getIndent(s,index,Token.tokenType.MULTIPLE_LINE_STRING); //토큰 분할
82                  }else if(result.getType() == Token.tokenType.MULTIPLE_LINE_COMMENT){ //이전 토큰이 문자열
인 경우 계속해서 문자열 토큰으로 처리하도록 지정
83                      result = new Token(); //새롭게 분할된 토큰을 생성
84                      result = getIndent(s,index,Token.tokenType.MULTIPLE_LINE_COMMENT); //토큰 분할
85                  }else{
86                      result = new Token(); //새롭게 분할된 토큰을 생성
87                      result = getIndent(s,index,Token.tokenType.NORMAL); //토큰 분할
88                  }
89              }
90          }
```

```
91
92          for(int i = 0 ; i < tmpTokens.size() ; i++){ //분할된 토큰 크기만큼 반복
93              Token filter = tmpTokens.get(i); //토큰을 가져옴
94              while(filter.getData().length() > 0){ //토큰의 텍스트 길이가 1 이상인 경우 키워드 분할 시도
95                  int tokenIndex = 0; //키워드 위치를 지정할 인덱스 지정
96                  if(filter.getType() == Token.tokenType.NORMAL || filter.getType() ==
Token.tokenType.KEYWORD){ //분할된 토큰 타입 값이 2,3(문자열)이 아닌경우에만 키워드 분할 시도 (이후 주석 관
련된 부분 추가해야함)
97                      int[] hasKeyword = keywordMatch(filter); //키워드 분할을 시도하여 인덱스를 구함
98                      if(hasKeyword[0] == -1){ //분할되지 않은 경우 (-1) 의 처리
99                          if(filter.getData().length() > 0){ //분할 되지 않은 토큰의 길이가 1 이상인 경우에만 처리
100                             Token tmp1 = new Token(); //새로운 토큰 생성
101                             tmp1.setData(filter.getData().substring(tokenIndex)); //분할되지 않은 토큰을 저장
102                             tokens.add(tmp1); //분할되지 않은 토큰을 저장
103                             break; //다음 토큰 처리를 위해 While 반복문 탈출
104                         }
105                     }else{ //토큰이 분할 된 경우의 처리
106                         Token tmp1 = new Token(); //키워드의 앞 부분을 저장할 토큰
107                         tmp1.setData(filter.getData().substring(tokenIndex, hasKeyword[0])); //토큰의 인덱스부터
키워드의 시작부분 -1까지의 문자열 저장
108                         Token tmp2 = new Token(); //키워드를 저장할 토큰
109                         tmp2.setData(filter.getData().substring(hasKeyword[0], hasKeyword[0]+hasKeyword[1]));
//키워드의 시작부분부터 끝부분 -1까지의 문자열 저장
110                         tmp2.setType(Token.tokenType.KEYWORD); //키워드 타입인 1을 지정
111                         tokenIndex += hasKeyword[0]; //새로운 키워드 탐색 위치 지정
112                         filter.setData(filter.getData().substring(hasKeyword[0]+hasKeyword[1])); //키워드 까지 분
할 되고 남은 토큰을 저장. 이후 다시 키워드 처리 반복
113                         if(tmp1.getData().length() > 0){ //키워드 앞 부분의 잘린 문자열의 길이가 1 이상인 경우
에만
114                             tokens.add(tmp1); //키워드 앞 부분 토큰을 리스트에 저장
115                         }
116                         tokens.add(tmp2); //키워드 토큰을 저장
117                     }
118                 }else{
119                     tokens.add(filter); //토큰 타입 값이 문자열,주석인 경우에는 별도 처리 없이 저장
120                     break; //더 이상 분할할 필요가 없으므로 While 반복문 탈출
121                 }
122             }
123         }
124         return tokens;
125     }
126
127     /**
128      * 주어진 문자열을 공백, 문자열, 주석을 기준으로 토큰화 합니다.
129      * @param line 토큰으로 변환할 문자열입니다.
130      * @param startIndex 분석을 시작할 문자열의 위치입니다.
131      * @param type 분석되는 문자열 성질입니다. 여러줄 문자열, 여러줄 주석에서 사용되어집니다. 그 외엔
NORMAL 타입입니다.
```



```
132         * @return 공백, 문자열, 주석을 기준으로 분석된 토큰입니다.
133         */
134        private Token getIndent(String line, int startIndex, Token.tokenType type){
135            Token t = new Token(); //분할할 토큰을 지정
136            if(type == Token.tokenType.MULTIPLE_LINE_STRING || type ==
Token.tokenType.MULTIPLE_LINE_COMMENT){
137                t.setType(type);
138            }
139            //문자열이 null인경우, 길이가 0인경우, 문자 인덱스가 문자열의 끝인경우 처리하지 않음.
140            if (line == null || line.length() == 0 || startIndex == line.length()) {
141                return t;
142            }
143            int i = startIndex; //시작 인덱스 값을 복사하여 저장
144            while (line.charAt(i) == ' ') { //문자열이 스페이스바인 경우 계속 인덱스를 증가시킴
145                i++;
146            }
147            //인덱스가 문자열을 초과하지 않고 스페이스바가 아닌 경우의 처리
148            while (i < line.length() && line.charAt(i) != ' ') {
149                if(t.getType() == Token.tokenType.STRING || t.getType() ==
Token.tokenType.MULTIPLE_LINE_STRING){
150                    //해당 라인의 문자열 끝까지 조사했으나 닫는 " 가 없는 경우의 처리를 위한 try..catch문
151                    try{
152                        //문자열 인덱스를 계속 증가시키며 ', "가 나올 때 까지 반복
153                        while((line.charAt(i) != '₩"' && line.charAt(i) != '₩') && (i>0 ? line.charAt(i-1) == '₩₩' :
true)){
154                            i++;
155                            System.out.println(line.charAt(i-1));
156                        }
157                        //"..." 구성의 문자열 데이터를 토큰에 저장 후 반환
158                        t.setType(Token.tokenType.STRING);
159                        t.setData(line.substring(startIndex, (i+1)));
160                        return t;
161                    }catch(StringIndexOutOfBoundsException e){
162                        //"..." 구성의 문자열 데이터를 토큰에 저장 후 반환
163                        t.setType(Token.tokenType.MULTIPLE_LINE_STRING); //다중열을 가진 문자열로 지정
164                        t.setData(line.substring(startIndex, i));
165                        return t;
166                    }
167                }else if(t.getType() == Token.tokenType.MULTIPLE_LINE_COMMENT){
168                    //해당 라인의 문자열 끝까지 조사했으나 닫는 */ 가 없는 경우의 처리를 위한 try..catch문
169                    try{
170                        //문자열 인덱스를 계속 증가시키며 */가 나올 때 까지 반복
171                        i++;
172                        while(line.charAt(i-1) != '*' || line.charAt(i) != '/'){
173                            i++;
174                        }
175                        t.setType(Token.tokenType.COMMENT);
176                        t.setData(line.substring(startIndex, (i+1)));
```

```
177                     return t;
178                 }catch(StringIndexOutOfBoundsException e){
179                     t.setType(Token.tokenType.MULTIPLE_LINE_COMMENT);
180                     t.setData(line.substring(startIndex));
181                     return t;
182                 }
183             }
184             //스페이스 바 이후의 문자열이 문자열을 나타내는 ' 혹은 " 인 경우의 처리
185             if((line.charAt(i) == '₩"' || line.charAt(i) == '₩'') && (i>0 ? line.charAt(i-1) != '₩₩' : true)){
186                 //증가 된 인덱스가 최초 시작 인덱스 보다 큰 경우, 즉 최초 분할 시작 된 문자열부터 ', " 직전
까지의 문자열에 대한 처리
187                 if(i > startIndex){
188                     t.setData(line.substring(startIndex, i)); //', " 앞 까지의 문자열을 토큰에 담아 반환
189                     return t;
190                 }
191                 t.setType(Token.tokenType.STRING); //', " 이후의 경우 토큰의 타입을 문자열로 지정
192                 if(line.charAt(i) == '₩"'){ //발견된 문자가 " 인 경우 그 뒤에 나오는 " 까지의 반복
193                     i++;
194                     //해당 라인의 문자열 끝까지 조사했으나 닫는 " 가 없는 경우의 처리를 위한 try..catch문
195                     try{
196                         //문자열 인덱스를 계속 증가시키며 "가 나올 때 까지 반복
197                         while(line.charAt(i) != '₩"' || (i>0 ? (line.charAt(i-1) == '₩₩' && (i > 1 ? line.charAt(i-2)
!= '₩₩' : true)) : true)){
198                             i++;
199                         }
200                     }catch(StringIndexOutOfBoundsException e){
201                         //"..." 구성의 문자열 데이터를 토큰에 저장 후 반환
202                         t.setType(Token.tokenType.MULTIPLE_LINE_STRING); //다중열을 가진 문자열로 지정
203                         t.setData(line.substring(startIndex, i));
204                         return t;
205                     }
206                     //"..." 구성의 문자열 데이터를 토큰에 저장 후 반환
207                     t.setData(line.substring(startIndex, (i+1)));
208                     return t;
209                 }else if(line.charAt(i) == '₩''){ //발견된 문자가 '인 경우 그 뒤에 나오는 '까지의 반복
210                     i++;
211                     //해당 라인의 문자열 끝까지 조사했으나 닫는 ' 가 없는 경우의 처리를 위한 try..catch문
212                     try{
213                         //문자열 인덱스를 계속 증가시키며 '가 나올 때 까지 반복
214                         while(line.charAt(i) != '₩'' || (i>0 ? (line.charAt(i-1) == '₩₩' && (i > 1 ? line.charAt(i-2)
!= '₩₩' : true)) : true)){
215                             i++;
216                         }
217                     }catch(StringIndexOutOfBoundsException e){
218                         //'...' 구성의 문자열 데이터를 토큰에 저장 후 반환
219                         t.setType(Token.tokenType.MULTIPLE_LINE_STRING); //다중열을 가진 문자열로 지정
220                         t.setData(line.substring(startIndex, i));
221                         return t;
```



```
222                 }
223                 //'...' 구성의 문자열 데이터를 토큰에 저장 후 반환
224                 t.setData(line.substring(startIndex, (i+1)));
225                 return t;
226             }
227         }
228         //싱글라인 주석에 관한 처리
229         if(i > 0){
230             if(line.charAt(i-1) == '/' && line.charAt(i) == '/'){ // 싱글라인 주석에 관한 처리
231                 if(i-1 > startIndex){ //i-1이 시작 인덱스보다 크면 (최초 시작된 인덱스가 주석과 관련된 키워
드가 아니면)
232                     t.setData(line.substring(startIndex, i-1)); // // 앞 까지의 문자열을 토큰에 담아 반환
233                     return t;
234                 }
235                 t.setType(Token.tokenType.COMMENT); // 타입을 싱글라인 주석으로 지정
236                 i++;
237                 while(line.charAt(i) != '₩n'){ // 라인피드가 나오기 전 까지 반복하여 i를 증가
238                     i++;
239                 }
240                 t.setData(line.substring(startIndex, i+1));
241                 return t; // 최종적으로 확정된 라인을 반환
242             }
243             if(line.charAt(i-1) == '/' && line.charAt(i) == '*'){ //다중라인 주석에 관한 처리
244                 if(i-1 > startIndex){ //i-1이 시작 인덱스보다 크면 (최초 시작된 인덱스가 주석과 관련된 키워
드가 아니면)
245                     t.setData(line.substring(startIndex, i-1)); // // 앞 까지의 문자열을 토큰에 담아 반환
246                     return t;
247                 }
248                 t.setType(Token.tokenType.MULTIPLE_LINE_COMMENT); // 타입을 다중라인 주석으로 지정
249                 i++;
250                 //해당 라인의 문자열 끝까지 조사했으나 닫는 */ 가 없는 경우의 처리를 위한 try..catch문
251                 try{
252                     //문자열 인덱스를 계속 증가시키며 */가 나올 때 까지 반복
253                     while(line.charAt(i-1) != '*' || line.charAt(i) != '/'){
254                         i++;
255                     }
256                 }catch(StringIndexOutOfBoundsException e){
257                     ///*...*/ 구성의 문자열 데이터를 토큰에 저장 후 반환
258                     t.setType(Token.tokenType.MULTIPLE_LINE_COMMENT); //다중라인 주석을 위한 처리
259                     t.setData(line.substring(startIndex));
260                     return t;
261                 }
262                 ///*...*/ 구성의 문자열 데이터를 토큰에 저장 후 반환
263                 t.setType(Token.tokenType.COMMENT); //다중라인 주석을 위한 처리
264                 t.setData(line.substring(startIndex, (i+1)));
265                 return t;
266             }
267         }
```

```
268             i++; //문자열 인덱스를 계속 증가시킴.
269         }
270         t.setData(line.substring(startIndex, i)); //문자열을 토큰에 담아 반환
271         return t;
272     }
273 
274     /**
275      * 문자열과 주석으로 구분되어진 토큰 이외의 일반 토큰에서 키워드가 있는 토큰을 찾습니다.
276      * @param inp 키워드가 있는지의 여부를 탐색할 토큰입니다.
277      * @return 키워드를 기준으로 새롭게 분할된 토큰입니다.
278      */
279     private int[] keywordMatch(Token inp){
280         //키워드 리스트
281         String[] key;
282         switch (option.getCodeLanguage()){
283         case JAVA:
284             key = KeywordSet.keyword_JAVA;
285             break;
286         case C:
287             key = KeywordSet.keyword_C;
288             break;
289         case CPP:
290             key = KeywordSet.keyword_CPP;
291             break;
292         default:
293             key = KeywordSet.keyword_JAVA;
294             break;
295         }
296         int[] ret = {-1,-1}; //최초 문자 인덱스 설정. -1은 분할되지 않음을 뜻함
297         if(inp == null || inp.getData().length() == 0) { //입력값이 null이 거나 길이가 0인경우 분할되지 않은
상태로 반환
298             return ret;
299         }
300         int startIndex = 0; //시작 인덱스를 0으로 초기화
301         int keyLength = key[0].length(); //키워드의 문자 길이를 저장
302 
303         while(startIndex <= inp.getData().length()){ //인덱스를 문자열 끝까지 증가시키며 반복
304             for(int i = 0 ; i < key.length ; i++){ //각 키에 따른 반복
305                 keyLength = key[i].length(); //키워드의 길이를 확인
306                 try{ //인덱스 초과에 대한 에러 처리가 있음.
307                     //시작 인덱스부터 시작인덱스+키인덱스 까지의 문자열이 키워드 값과 같은 경우의 처리
308                     if(inp.getData().substring(startIndex, startIndex+keyLength).equals(key[i])){
309                         //만약 시작 인덱스가 0이고 전체 문자의 길이가 키워드의 길이와 같은 경우
310                         if(startIndex == 0 && startIndex+keyLength == inp.getData().length()){
311                             ret[0] = startIndex; //시작 인덱스 값을 지정
312                             ret[1] = keyLength; //끝 인덱스 값을 지정
313                             return ret; //토큰 반환
314                         }
```

```
315                     //만약 시작 인덱스가 1 이상이고 인덱스+키 길이가 전체 문자열 길이와 같은 경우
316                     //즉 키워드 앞 쪽에는 문자가 있으나 키워드 뒤에는 문자가 없는 경우
317                     if(startIndex > 0 && startIndex+keyLength == inp.getData().length()){
318                         //키워드 앞쪽의 문자 1개가 알파벳,숫자,한글인 경우 해당 키워드는 실질적인 키워드가 아니므로 처리하지 않음
319                         if(!String.valueOf(inp.getData().charAt(startIndex - 1)).matches("[A-Za-z0-9가-힣ㄱ-ㅎ]")){
320                             //키워드 앞쪽의 문자 1개가 알파벳,숫자,한글이 아닌 경우 해당 키워드의 위치를 저장하여 반환
321                             ret[0] = startIndex;
322                             ret[1] = keyLength;
323                             return ret;
324                         }
325                     }
326                     //만약 시작인덱스 + 키워드 길이가 전체 문자열 길이보다 작은 경우
327                     //즉 키워드 뒤 쪽에 문자가 있는 경우
328                     if(startIndex+keyLength < inp.getData().length()){
329                         //해당 키워드 뒤쪽의 문자가 알파벳,숫자,한글인 경우 해당 키워드는 실질적인 키워드가 아니므로 처리하지 않음
330                         if(!String.valueOf(inp.getData().charAt(startIndex+keyLength)).matches("[A-Za-z0-9가-힣ㄱ-ㅎ]") && !String.valueOf(inp.getData().charAt(startIndex - 1)).matches("[A-Za-z0-9가-힣ㄱ-ㅎ]")){
331                             //키워드 뒤쪽의 문자 1개가 알파벳,숫자,한문이 아닌 경우 해당 키워드의 위치를 저장하여 반환
332                             ret[0] = startIndex;
333                             ret[1] = keyLength;
334                             return ret;
335                         }
336                     }
337                 }
338             }catch(StringIndexOutOfBoundsException e){
339                 continue; //인덱스를 초과한 경우 다음 키워드 검사 시작
340             }
341         }
342         startIndex++; //인덱스에 대한 모든 키 검사가 끝난 경우 다음 인덱스에서부터 새롭게 조사
343     }
344     return ret; //모든 조사가 끝난 토큰을 반환
345   }
346 }
347
```

CodePrinter.java

```
1  package converter;
2  import java.io.*;
3  import java.util.Vector;
4  import com.itextpdf.text.*;
5  import com.itextpdf.text.pdf.*;
6
7  import option.*;
8
9  /**
10   * 토큰화된 소스코드 파일을 PDF로 출력합니다.
11   * @author 강승민
12   *
13   */
14  public class CodePrinter {
15
16      /**
17       * 옵션 정보
18       */
19      private Option option;
20
21      /**
22       * 보안 정보
23       */
24      private Security security;
25
26      /**
27       * 생성자
28       * @param op 옵션 정보
29       * @param sec 보안 정보
30       */
31      public CodePrinter(Option op, Security sec){
32          option = op;
33          security = sec;
34      }
35
36      /**
37       * 파싱된 모든 토큰을 한개의 PDF 문서에 출력합니다.
38       * @param tokens 토큰 벡터 / [패키지 리스트[패키지 내 파일 리스트 [파일 내 토큰 벡터]]]의 구성입니다.
39       * @param subName PDF 문서에 삽입되는 파일 이름 리스트 입니다.
40       * @throws DocumentException
41       * @throws IOException
42       */
43      public void printAll(Vector<Vector<Vector<Token>>> tokens, Vector<String> subName) throws
DocumentException, IOException{
44          Document doc = new Document(option.getPageType(), option.getMarginLeft(),
option.getMarginRight(), option.getMarginTop(), option.getMarginBottom());
45          PdfWriter writer = null;
```

```
46          try {
47              writer = PdfWriter.getInstance(doc, new
FileOutputStream(option.getSaveLocate()+option.getFileName() + ".pdf"));
48          } catch (FileNotFoundException e1) {
49              writer = PdfWriter.getInstance(doc, new FileOutputStream(option.getSaveLocate()+
option.getFileName() + (int)(Math.random() * 10000) + ".pdf"));
50          } catch (DocumentException e2) {
51              e2.printStackTrace();
52          }
53          if(security.getUseSecurityOption()){
54              if(security.getUseCopyMode()){
55                  writer.setEncryption(security.getPassword().getBytes(), setOwnerPassword().getBytes(),
PdfWriter.ALLOW_COPY, PdfWriter.STANDARD_ENCRYPTION_40);
56              }else{
57                  writer.setEncryption(security.getPassword().getBytes(), setOwnerPassword().getBytes(),
PdfWriter.ALLOW_SCREENREADERS, PdfWriter.STANDARD_ENCRYPTION_40);
58              }
59          }
60          CopyRightEvent event = new CopyRightEvent(option);
61          writer.setPageEvent(event);
62
63          doc.addTitle(option.getFileName());
64
65          doc.open();
66
67          for(int i = 0 ; i < subName.size() ; i++){
68              Anchor anc = new Anchor(subName.get(i));
69              anc.setReference("#" + subName.get(i));
70              Paragraph p = new Paragraph();
71              p.add(anc);
72              doc.add(p);
73          }
74          doc.newPage();
75
76          int cnt = 0;
77
78          for(int k = 0 ; k < tokens.size() ; k++){
79              Vector<Vector<Token>> packageToken = tokens.get(k);
80              for(int j = 0 ; j < packageToken.size() ; j++, cnt++){
81                  int codeLine = 1;
82                  Vector<Token> docToken = packageToken.get(j);
83
84                  Anchor anc = new Anchor(subName.get(cnt) + "₩n");
85                  anc.setName(subName.get(cnt));
86                  Paragraph para = new Paragraph();
87                  para.add(anc);
88                  doc.add(para);
89
```

```
90                  Phrase p = new Phrase();
91                  if(option.getShowLineNumber()){
92                      p.add(new Chunk(codeLine + "     ", option.getLineNumberFont().getFont()));
93                      codeLine++;
94                  }
95                  doc.add(p);
96                  for(int i = 0 ; i < docToken.size() ; i++){
97                      Token t = docToken.get(i);
98                      p = new Phrase();
99                      String codeLineString = String.valueOf(codeLine) + "     ";
100                      if(t.getType() == Token.tokenType.NORMAL){
101                          if(t.getData().contains("₩n")){
102                              p.add(new Chunk(t.getData(), option.getBasicFont().getFont()));
103                              if(option.getShowLineNumber()){
104                                  p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
105                                  codeLine++;
106                              }
107                          }else{
108                              p.add(new Chunk(t.getData(), option.getBasicFont().getFont()));
109                          }
110                      }else if(t.getType() == Token.tokenType.KEYWORD){
111                          if(t.getData().contains("₩n")){
112                              p.add(new Chunk(t.getData(), option.getKeywordFont().getFont()));
113                              if(option.getShowLineNumber()){
114                                  p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
115                                  codeLine++;
116                              }
117                          }else{
118                              p.add(new Chunk(t.getData(), option.getKeywordFont().getFont()));
119                          }
120                      }else if(t.getType() == Token.tokenType.STRING || t.getType() ==
Token.tokenType.MULTIPLE_LINE_STRING){
121                          if(t.getData().contains("₩n")){
122                              p.add(new Chunk(t.getData(), option.getStringFont().getFont()));
123                              if(option.getShowLineNumber()){
124                                  p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
125                                  codeLine++;
126                              }
127                          }else{
128                              p.add(new Chunk(t.getData(), option.getStringFont().getFont()));
129                          }
130                      }else if(t.getType() == Token.tokenType.COMMENT || t.getType() ==
Token.tokenType.MULTIPLE_LINE_COMMENT){
131                          if(t.getData().contains("₩n")){
132                              p.add(new Chunk(t.getData(), option.getCommentFont().getFont()));
133                              if(option.getShowLineNumber()){
134                                  p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
135                                  codeLine++;
```

```
136                             }
137                         }else{
138                             p.add(new Chunk(t.getData(), option.getCommentFont().getFont()));
139                         }
140                     }
141                     doc.add(p);
142                 }
143                 doc.newPage();
144             }
145         }
146         doc.close();
147     }
148 
149     /**
150      * 파싱된 모든 토큰을 각 패키지(폴더) 별로 PDF 문서로 만듭니다.
151      * @param tokens 토큰 벡터 / [패키지 내 파일 리스트 [파일 내 토큰 벡터]] 구성입니다.
152      * @param name 패키지 이름입니다.
153      * @param subName 패키지 내에 포함된 파일 이름입니다.
154      * @throws DocumentException
155      * @throws IOException
156      */
157     public void printPackage(Vector<Vector<Token>> tokens, String name, Vector<String> subName)
throws DocumentException, IOException{
158         Document doc = new Document(option.getPageType(), option.getMarginLeft(),
option.getMarginRight(), option.getMarginTop(), option.getMarginBottom());
159         PdfWriter writer = null;
160         try {
161             writer = PdfWriter.getInstance(doc, new FileOutputStream(option.getSaveLocate()+ name +
".pdf"));
162         } catch (FileNotFoundException e1) {
163             writer = PdfWriter.getInstance(doc, new FileOutputStream(option.getSaveLocate()+ name + (int
)(Math.random() * 10000) + ".pdf"));
164         } catch (DocumentException e2) {
165             e2.printStackTrace();
166         }
167         if(security.getUseSecurityOption()){
168             if(security.getUseCopyMode()){
169                 writer.setEncryption(security.getPassword().getBytes(), setOwnerPassword().getBytes(),
PdfWriter.ALLOW_COPY, PdfWriter.STANDARD_ENCRYPTION_40);
170             }else{
171                 writer.setEncryption(security.getPassword().getBytes(), setOwnerPassword().getBytes(),
PdfWriter.ALLOW_SCREENREADERS, PdfWriter.STANDARD_ENCRYPTION_40);
172             }
173         }
174         CopyRightEvent event = new CopyRightEvent(option);
175         writer.setPageEvent(event);
176 
177         doc.addTitle(name);
```



```
178
179         doc.open();
180
181         for(int i = 0 ; i < subName.size() ; i++){
182             Anchor anc = new Anchor(subName.get(i));
183             anc.setReference("#" + subName.get(i));
184             Paragraph p = new Paragraph();
185             p.add(anc);
186             doc.add(p);
187         }
188         doc.newPage();
189
190         for(int k = 0 ; k < tokens.size() ; k++){
191             int codeLine = 1;
192             Vector<Token> docToken = tokens.get(k);
193             Anchor anc = new Anchor(subName.get(k) + "₩n");
194             anc.setName(subName.get(k));
195             Paragraph para = new Paragraph();
196             para.add(anc);
197             doc.add(para);
198
199             Phrase p = new Phrase();
200             if(option.getShowLineNumber()){
201                 p.add(new Chunk(codeLine + "     ", option.getLineNumberFont().getFont()));
202                 codeLine++;
203             }
204             doc.add(p);
205             for(int i = 0 ; i < docToken.size() ; i++){
206                 Token t = docToken.get(i);
207                 p = new Phrase();
208                 String codeLineString = String.valueOf(codeLine) + "     ";
209                 if(t.getType() == Token.tokenType.NORMAL){
210                     if(t.getData().contains("₩n")){
211                         p.add(new Chunk(t.getData(), option.getBasicFont().getFont()));
212                         if(option.getShowLineNumber()){
213                             p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
214                             codeLine++;
215                         }
216                     }else{
217                         p.add(new Chunk(t.getData(), option.getBasicFont().getFont()));
218                     }
219                 }else if(t.getType() == Token.tokenType.KEYWORD){
220                     if(t.getData().contains("₩n")){
221                         p.add(new Chunk(t.getData(), option.getKeywordFont().getFont()));
222                         if(option.getShowLineNumber()){
223                             p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
224                             codeLine++;
225                         }
```

```
226                     }else{
227                         p.add(new Chunk(t.getData(), option.getKeywordFont().getFont()));
228                     }
229                 }else if(t.getType() == Token.tokenType.STRING || t.getType() ==
Token.tokenType.MULTIPLE_LINE_STRING){
230                     if(t.getData().contains("₩n")){
231                         p.add(new Chunk(t.getData(), option.getStringFont().getFont()));
232                         if(option.getShowLineNumber()){
233                             p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
234                             codeLine++;
235                         }
236                     }else{
237                         p.add(new Chunk(t.getData(), option.getStringFont().getFont()));
238                     }
239                 }else if(t.getType() == Token.tokenType.COMMENT || t.getType() ==
Token.tokenType.MULTIPLE_LINE_COMMENT){
240                     if(t.getData().contains("₩n")){
241                         p.add(new Chunk(t.getData(), option.getCommentFont().getFont()));
242                         if(option.getShowLineNumber()){
243                             p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
244                             codeLine++;
245                         }
246                     }else{
247                         p.add(new Chunk(t.getData(), option.getCommentFont().getFont()));
248                     }
249                 }
250                 doc.add(p);
251             }
252             doc.newPage();
253         }
254         doc.close();
255     }
256 
257     /**
258      * 단일 파일에 대한 토큰을 PDF로 출력합니다.
259      * @param tokens 출력할 토큰 리스트입니다.
260      * @param name 출력할 파일 명 입니다.
261      * @throws DocumentException
262      * @throws IOException
263      */
264     public void printSingleFile(Vector<Token> tokens, String name) throws DocumentException,
IOException{
265         int codeLine = 1;
266         Document doc = new Document(option.getPageType(), option.getMarginLeft(),
option.getMarginRight(), option.getMarginTop(), option.getMarginBottom());
267         PdfWriter writer = null;
268         try {
269             writer = PdfWriter.getInstance(doc, new FileOutputStream(option.getSaveLocate()+name + ".pdf"
```



```
));
270             } catch (FileNotFoundException e1) {
271                 writer = PdfWriter.getInstance(doc, new FileOutputStream(option.getSaveLocate()+ name + (int
)(Math.random() * 10000) + ".pdf"));
272             } catch (DocumentException e2) {
273                 e2.printStackTrace();
274             }
275 
276             if(security.getUseSecurityOption()){
277                 if(security.getUseCopyMode()){
278                     writer.setEncryption(security.getPassword().getBytes(), setOwnerPassword().getBytes(),
PdfWriter.ALLOW_COPY, PdfWriter.STANDARD_ENCRYPTION_40);
279                 }else{
280                     writer.setEncryption(security.getPassword().getBytes(), setOwnerPassword().getBytes(),
PdfWriter.ALLOW_SCREENREADERS, PdfWriter.STANDARD_ENCRYPTION_40);
281                 }
282             }
283             CopyRightEvent event = new CopyRightEvent(option);
284             writer.setPageEvent(event);
285 
286             doc.addTitle(name);
287 
288             doc.open();
289             Phrase p = new Phrase();
290             if(option.getShowLineNumber()){
291                 p.add(new Chunk(codeLine + "     ", option.getLineNumberFont().getFont()));
292                 codeLine++;
293             }
294             doc.add(p);
295             for(int i = 0 ; i < tokens.size() ; i++){
296                 Token t = tokens.get(i);
297                 p = new Phrase();
298                 String codeLineString = String.valueOf(codeLine) + "     ";
299                 if(t.getType() == Token.tokenType.NORMAL){
300                     if(t.getData().contains("₩n")){
301                         p.add(new Chunk(t.getData(), option.getBasicFont().getFont()));
302                         if(option.getShowLineNumber()){
303                             p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
304                             codeLine++;
305                         }
306                     }else{
307                         p.add(new Chunk(t.getData(), option.getBasicFont().getFont()));
308                     }
309                 }else if(t.getType() == Token.tokenType.KEYWORD){
310                     if(t.getData().contains("₩n")){
311                         p.add(new Chunk(t.getData(), option.getKeywordFont().getFont()));
312                         if(option.getShowLineNumber()){
313                             p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
```



```
314                         codeLine++;
315                     }
316                 }else{
317                     p.add(new Chunk(t.getData(), option.getKeywordFont().getFont()));
318                 }
319             }else if(t.getType() == Token.tokenType.STRING || t.getType() ==
Token.tokenType.MULTIPLE_LINE_STRING){
320                 if(t.getData().contains("₩n")){
321                     p.add(new Chunk(t.getData(), option.getStringFont().getFont()));
322                     if(option.getShowLineNumber()){
323                         p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
324                         codeLine++;
325                     }
326                 }else{
327                     p.add(new Chunk(t.getData(), option.getStringFont().getFont()));
328                 }
329             }else if(t.getType() == Token.tokenType.COMMENT || t.getType() ==
Token.tokenType.MULTIPLE_LINE_COMMENT){
330                 if(t.getData().contains("₩n")){
331                     p.add(new Chunk(t.getData(), option.getCommentFont().getFont()));
332                     if(option.getShowLineNumber()){
333                         p.add(new Chunk(codeLineString, option.getLineNumberFont().getFont()));
334                         codeLine++;
335                     }
336                 }else{
337                     p.add(new Chunk(t.getData(), option.getCommentFont().getFont()));
338                 }
339             }
340             doc.add(p);
341         }
342         doc.close();
343     }
344
345     /**
346      * 임의의 ownerPassword를 생성합니다.
347      * @return 생성된 ownerPassword 입니다.
348      */
349     private String setOwnerPassword(){
350         StringBuffer randStr = new StringBuffer();
351         for(int i = 0 ; i < 30 ; i++){
352             int lower = (int) Math.round(Math.random());
353             int r = (int) Math.floor(Math.random() * 26);
354             if(lower == 0){
355                 randStr.append((char) (r + 'A'));
356             }else{
357                 randStr.append((char) (r + 'a'));
358             }
359         }
```



```
360             return security.getPassword() + randStr.toString();
361         }
362     }
363
```


ConvertPrograssEvent.java

```java
package converter;

import java.util.EventObject;

/**
 * 변환 진행률에 관련된 이벤트 입니다.
 * @author 강승민
 *
 */
 public class ConvertPrograssEvent extends EventObject {

	private static final long serialVersionUID = 7231650191710081939L;

	/**
	 * 현재 진행률
	 */
	private int currentPrograss;

	/**
	 * 전체 변환량
	 */
	private int totalPrograss;

	/**
	 * 이벤트 생성자
	 * @param source 이벤트 발생 주체입니다.
	 * @param currentPrograss 현재 진행률 입니다.
	 * @param totalPrograss 전체 변환량 입니다.
	 */
	public ConvertPrograssEvent(Object source, int currentPrograss, int totalPrograss) {
		super(source);
		this.currentPrograss = currentPrograss;
		this.totalPrograss = totalPrograss;
	}

	/**
	 * 현재 변환 진행률을 확인합니다.
	 * @return 현재 변환 진행률
	 */
	public int getCurrentPrograss(){
		return this.currentPrograss;
	}

	/**
	 * 전체 변환 량을 확인합니다.
	 * @return 전체 변환량
	 */
```

```
48      public int getTotalPrograss(){
49          return this.totalPrograss;
50      }
51  }
52
```


CopyRightEvent.java

```java
package converter;
import com.itextpdf.text.Chunk;
import com.itextpdf.text.Document;
import com.itextpdf.text.Element;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.pdf.ColumnText;
import com.itextpdf.text.pdf.PdfPageEventHelper;
import com.itextpdf.text.pdf.PdfWriter;

 import option.Option;

 /**
  * 헤더 정보, 저작권 정보, 페이지 번호 표시를 위한 문서 이벤트 처리 클래스 입니다.
  * @author 강승민
  *
  */
 public class CopyRightEvent extends PdfPageEventHelper{
     /**
      * 저작권 정보입니다.
      */
     private Phrase footer;

     /**
      * 헤더 정보입니다.
      */
     private Phrase header;

     /**
      * 옵션 정보
      */
     private Option option;

     /**
      * 생성자 : 위치를 기본값으로 정의하는 객체 생성자 입니다.
      * @param op Option 인스턴스입니다.
      */
     public CopyRightEvent(Option op){
         option = op;
         footer = new Phrase(new Chunk(option.getCopyright()));
         header = new Phrase(new Chunk(option.getHeaderTitle()));
     }

     /**
      * 이벤트 헨들러 메소드입니다.
      * 헤더, 저작권 정보의 처리를 진행합니다.
      */
     public void onEndPage(PdfWriter writer, Document doc){
```

```
48              if(option.getHeaderAlign() == Option.Align.LEFT){
49                  ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_LEFT, header, doc.left(),
doc.top() + 10, 0);
50              }else if(option.getHeaderAlign() == Option.Align.CENTER){
51                  ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_CENTER, header,
(doc.right() - doc.left()) / 2 + doc.leftMargin(), doc.top() + 10, 0);
52              }else if(option.getHeaderAlign() == Option.Align.RIGHT){
53                  ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_RIGHT, header, doc.right(),
doc.top() + 10, 0);
54              }
55
56              if(option.getCopyrightAlign() == Option.Align.LEFT){
57                  ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_LEFT, footer, doc.left(),
doc.bottom() - 20, 0);
58                  if(option.getShowPageNumber()){
59                      ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_CENTER, new Phrase(
new Chunk("Page " + writer.getPageNumber())), (doc.right() - doc.left()) / 2 + doc.leftMargin(), doc.bottom() - 20,
0);
60                  }
61              }else if(option.getCopyrightAlign() == Option.Align.CENTER){
62                  ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_CENTER, footer,
(doc.right() - doc.left()) / 2 + doc.leftMargin(), doc.bottom() - 20, 0);
63                  if(option.getShowPageNumber()){
64                      ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_CENTER, new Phrase(
new Chunk("Page " + writer.getPageNumber())), doc.left(), doc.bottom() - 20, 0);
65                  }
66              }else if(option.getCopyrightAlign() == Option.Align.RIGHT){
67                  ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_RIGHT, footer, doc.right(),
doc.bottom() - 20, 0);
68                  if(option.getShowPageNumber()){
69                      ColumnText.showTextAligned(writer.getDirectContent(), Element.ALIGN_CENTER, new Phrase(
new Chunk("Page " + writer.getPageNumber())), (doc.right() - doc.left()) / 2 + doc.leftMargin(), doc.bottom() - 20,
0);
70                  }
71              }
72          }
73      }
74
```

IConvertPrograssListener.java

```java
package converter;

import java.util.EventListener;

/**
 * 변환 진행 이벤트 리스너입니다.
 * @author 강승민
 *
 */
 public interface IConvertPrograssListener extends EventListener{

    /**
     * 핸들러 메소드 입니다.
     * @param event 변환 진행 이벤트
     */
    public void convertPrograss(ConvertPrograssEvent event);
 }

```

KeywordSet.java

```
1   package converter;
2   /**
3    * 각 언어별 키워드 문자들을 저장합니다.
4    * @author 오다솜
5    *
6    */
7   public class KeywordSet {
8
9       /**
10       * 자바 키워드
11       */
12      public static String[] keyword_JAVA = {"abstract", "boolean", "break", "byte", "case", "catch", "char",
"class", "const", "continue", "do", "double", "enum", "extends", "final", "finally", "float", "for", "default",
"implements", "import", "insterface", "long", "native", "new", "goto", "int", "if", "public", "short", "super", "switch",
"synchronized", "package", "private", "protected", "transient", "return", "void", "static", "while", "this", "throw",
"throws", "try", "volatile", "stricftp", "true", "false", "else", "null"};
13
14      /**
15       * C 키워드
16       */
17      public static String[] keyword_C = {"auto", "break", "case", "char", "const", "continue", "default", "do",
"double", "else", "enum", "extern", "float", "for", "goto", "if", "int", "long", "register", "return", "short", "signed",
"sizeof", "static", "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while", "#include", "#define",
 "#undef", "#if", "#ifdef", "#ifndef", "#else", "#elif", "#endif", "#error", "#line", "#pragma", "#import"};
18
19      /**
20       * C++ 키워드
21       */
22      public static String[] keyword_CPP = {"auto", "break", "case", "char", "const", "continue", "default", "do",
"double", "else", "enum", "extern", "float", "for", "goto", "if", "int", "long", "register", "return", "short", "signed",
"sizeof", "static", "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while", "and", "and_eq",
"asm", "bitand", "bitor", "bool", "catch", "class", "compl", "const_cast", "delete", "dynamic_cast", "explicit", "export"
, "false", "friend", "inline", "mutable", "namespace", "new", "not", "not_eq", "operator", "or", "or_eq", "private",
"protected", "public", "reinterpret_cast", "static_cast", "template", "this", "throw", "true", "try", "typeid",
"typename", "using", "virtual", "wchar_t", "xor", "xor_eq", "#include", "#define", "#undef", "#if", "#ifdef", "#ifndef",
"#else", "#elif", "#endif", "#error", "#line", "#pragma", "#import"};
23  }
24
```

Token.java

```
1  package converter;
2  /**
3   * 처리기의 규칙을 통해 생성되는 문자열의 최소 단위입니다.
4   * @author 강승민
5   *
6   */
7  public class Token {
8      /**
9       * 문자열 값입니다.
10       */
11      private String data = "";
12      /**
13       * 문자열의 타입입니다.
14       */
15      private tokenType type;
16
17      /**
18       * 토큰이 가지는 문자열 타입입니다.<p>
19       * 타입 종류<br>
20       * NORMAL : 특정한 성질이 없는 기본 토큰입니다.<br>
21       * KEYWORD : if, for등과 같은 언어에서 지정된 키워드 성질입니다.<br>
22       * STRING : ' ' 혹은 " " 형태로 감싸진 문자열 성질입니다.<br>
23       * MULTIPLE_LINE_STRING : ' ' 혹은 " " 형태가 여러줄에 걸쳐 나타나는 문자열 성질입니다.<br>
24       * COMMENT : 주석 성질입니다.<br>
25       * MULTIPLE_LINE_COMMENT : 여러줄 주석 성질입니다.<br>
26       * @author 강승민
27       *
28       */
29      public enum tokenType {NORMAL, KEYWORD, STRING, MULTIPLE_LINE_STRING, COMMENT,
MULTIPLE_LINE_COMMENT};
30
31      /**
32       * 토큰을 생성합니다.
33       * 생성된 토큰은 빈 문자열에 NORMAL 성질을 가집니다.
34       */
35      public Token(){
36          setType(Token.tokenType.NORMAL);
37      }
38
39      /**
40       * 토큰의 문자열 데이터를 반환합니다.
41       */
42      @Override
43      public String toString(){
44          return data;
45      }
46
```

```
47      /**
48       * 토큰의 타입을 반환합니다.
49       * @return 토큰의 타입
50       */
51      public tokenType getType() {
52          return type;
53      }
54
55      /**
56       * 토큰의 타입을 지정합니다.
57       * @param type 토큰의 타입
58       */
59      public void setType(tokenType type) {
60          this.type = type;
61      }
62
63      /**
64       * 토큰의 문자열 데이터를 지정합니다.
65       * @param data 토큰 문자열
66       */
67      public void setData(String data) {
68          this.data = data;
69      }
70      /**
71       * 토큰의 문자열 데이터를 반환합니다.
72       * @return 토큰 문자열
73       */
74      public String getData() {
75          return data;
76      }
77  }
78
```

Convert.java

```
1  package display;
2  import java.awt.Dimension;
3  import java.awt.Toolkit;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6
7  import javax.swing.*;
8
9  import converter.CodeConverter;
10  import converter.ConvertPrograssEvent;
11  import converter.IConvertPrograssListener;
12  import loader.Checker;
13  import option.Option;
14  import option.Security;
15
16  /**
17   * 변환 다이얼로그 창
18   * @author 김선일
19   *
20   */
21  public class Convert extends JDialog {
22      /**
23       * serialUID
24       */
25      private static final long serialVersionUID = -6990134427787874178L;
26
27      /**
28       * 변환 버튼
29       */
30      private JButton convert_convert_btn;
31
32      /**
33       * 변환 창 닫기 버튼
34       */
35      private JButton convert_cancel_btn;
36
37      /**
38       * 변환 진행 바
39       */
40      private JProgressBar convert_progressbar;
41
42      /**
43       * 변환할 소스코드 리스트 뷰 패널
44       */
45      private JScrollPane convert_source_scp;
46
47      /**
```

```
48      * 변환할 소스코드 리스트
49      */
50     private JList<String> convert_source_list;
51
52     /**
53      * 변환 소스코드 정보
54      */
55     private Checker checker;
56
57     /**
58      * 옵션 정보
59      */
60     private Option option;
61
62     /**
63      * 보안 정보
64      */
65     private Security security;
66
67     /**
68      * 변환기
69      */
70     private CodeConverter converter;
71
72     /**
73      * 생상지<p>
74      * 변수를 초기화 하고 이벤트를 등록합니다.
75      * @param chker 소스코드 정보
76      * @param op 옵션 정보
77      * @param sec 보안 정보
78      */
79     public Convert(Checker chker, Option op, Security sec){
80         checker = chker;
81         option = op;
82         security = sec;
83         converter = new CodeConverter(checker, option, security);
84         converter.addConvertPrograssListener(new IConvertPrograssListener() {
85             @Override
86             public void convertPrograss(ConvertPrograssEvent event) {
87                 convert_progressbar.setValue(event.getCurrentPrograss());
88                 convert_progressbar.setMaximum(event.getTotalPrograss());
89
90                 //변환완료시 완료다이얼로그 출력
91                 //확인버튼 클릭 후 변환창 종료
92                 if(event.getCurrentPrograss() == event.getTotalPrograss()){
93                     showFinishDialog();
94                     dispose();
95                 }
```

```
96              }
97          });
98      }
99
100     /**
101      * 디스플레이 셋업
102      */
103     public void setConvertDialog(){
104         setTitle("소스코드 변환");
105         //변환프레임 컴포넌트 설정
106         convert_progressbar = new JProgressBar();
107         convert_source_scp = new JScrollPane();
108         convert_convert_btn = new JButton();
109         convert_cancel_btn = new JButton();
110
111
112         convert_source_list = new JList<String>();
113
114         setModal(true);
115         setPreferredSize(new Dimension(500, 300));
116         this.setLocation((int)(Toolkit.getDefaultToolkit().getScreenSize().getWidth() / 2 - this
.getPreferredSize().getWidth() / 2),
117                 (int)(Toolkit.getDefaultToolkit().getScreenSize().getHeight() / 2 - this
.getPreferredSize().getHeight() / 2));
118         setName("변환");
119
120         setLayout(null);
121
122         convert_progressbar.setStringPainted(true);
123         add(convert_progressbar);
124         convert_progressbar.setBounds(60, 54, 290, 20);
125         add(convert_source_scp);
126         convert_source_scp.setBounds(60, 80, 290, 140);
127         convert_source_scp.setViewportView(convert_source_list);
128         convert_source_list.setListData(checker.fileListString);
129
130         convert_convert_btn.setText("변환");
131         convert_convert_btn.addActionListener(new ActionListener() {
132             public void actionPerformed(ActionEvent evt) {
133                 convertBtnEvent(evt);
134             }
135         });
136         add(convert_convert_btn);
137         convert_convert_btn.setBounds(360, 50, 57, 23);
138
139         convert_cancel_btn.setText("닫기");
140         convert_cancel_btn.addActionListener(new ActionListener() {
141             public void actionPerformed(ActionEvent evt) {
```

```
142                     convertCancleBtnEvent(evt);
143                 }
144             });
145             add(convert_cancel_btn);
146             convert_cancel_btn.setBounds(360, 230, 57, 23);
147             setResizable(false);
148             pack();
149         }
150 
151         /**
152          * 변환 버튼 이벤트 헨들러
153          * @param evt 버튼 이벤트
154          */
155         private void convertBtnEvent(ActionEvent evt) {
156             if(!converter.isAlive()){
157                 converter.start();
158                 convert_convert_btn.setEnabled(false);
159             }
160         }
161 
162         /**
163          * 닫기 버튼 이벤트 핸들러
164          * @param evt 버튼 이벤트
165          */
166         private void convertCancleBtnEvent(ActionEvent evt) {
167             dispose();
168         }
169 
170         /**
171          * 변환 완료시 처리 작업
172          */
173         private void showFinishDialog(){
174             convert_convert_btn.setEnabled(true);
175             JOptionPane.showMessageDialog(null, "변환이 완료되었습니다!", "변환완료",
JOptionPane.INFORMATION_MESSAGE);
176         }
177 
178     }
179 
```

Copyright.java

```
1  package display;
2  import java.awt.Toolkit;
3
4  import javax.swing.*;
5
6  import option.Option;
7
8  /**
9   * 저작권 설정에 관한 디스플레이 입니다.
10   * @author 김선일
11   *
12   */
13  public class Copyright extends JDialog {
14
15      /**
16       * 다이얼로그 시리얼 UID
17       */
18      private static final long serialVersionUID = 8905550640457174616L;
19
20      /**
21       * 옵션 지정에 사용될 옵션입니다.
22       */
23      private Option option;
24
25      /**
26       * 저작권 사용 여부 설정 버튼
27       */
28      private JCheckBox copyright_option_enable = new JCheckBox();
29
30      /**
31       * 저작권 정보 표시 위치 콤보박스
32       */
33      private JComboBox<String> copyright_location_cb = new JComboBox<String>();
34
35      /**
36       * 저작권 정보 표시 위치 라벨
37       */
38      private JLabel copyright_location_lb = new JLabel();
39
40      /**
41       * 저작권 정보 라벨
42       */
43      private JLabel copyright_imfo_lb = new JLabel();
44
45      /**
46       * 저작권 정보 입력 필드
47       */
```

```
48      private JTextField copyright_info_tf = new JTextField();
49
50      /**
51       * 저작권 정보 저장 버튼
52       */
53      private JButton copyright_save_btn = new JButton();
54
55      /**
56       * 저작권 정보 닫기 버튼
57       */
58      private JButton copyright_cancel_btn = new JButton();
59
60      /**
61       * 생성자
62       * @param op 옵션 정보
63       */
64      public Copyright(Option op){
65          option = op;
66      }
67
68      /**
69       * 디스플레이 셋업
70       */
71      public void setCopyrightPanel() {
72          //저작권설정 프레임의 컴포넌트 설정
73          setTitle("저작권 설정");
74          setModal(true);
75          setPreferredSize(new java.awt.Dimension(360, 250));
76          this.setLocation((int)(Toolkit.getDefaultToolkit().getScreenSize().getWidth() / 2 - this
.getPreferredSize().getWidth() / 2),
77                  (int)(Toolkit.getDefaultToolkit().getScreenSize().getHeight() / 2 - this
.getPreferredSize().getHeight() / 2));
78          enableCopyrightOption();
79          setLayout(null);
80
81          copyright_option_enable.setText("저작권 옵션 사용");
82          copyright_option_enable.addActionListener(new java.awt.event.ActionListener() {
83              public void actionPerformed(java.awt.event.ActionEvent evt) {
84                  enableCopyrightOption();
85              }
86          });
87          if(!option.getCopyright().equals("")){
88              copyright_option_enable.setSelected(true);
89              enableCopyrightOption();
90          }
91          add(copyright_option_enable);
92          copyright_option_enable.setBounds(30, 30, 200, 23);
93
```

```
94            copyright_location_cb.setModel(new DefaultComboBoxModel<>(new String[] { "왼쪽", "중앙", "오른쪽
" }));
95            copyright_location_cb.setSelectedIndex(convertAlignToItemIndex(option.getCopyrightAlign()));
96            add(copyright_location_cb);
97            copyright_location_cb.setBounds(170, 70, 80, 21);
98
99            copyright_location_lb.setText("저작자 정보 표시 위치");
100           add(copyright_location_lb);
101           copyright_location_lb.setBounds(40, 70, 130, 15);
102
103           add(copyright_info_tf);
104           copyright_info_tf.setBounds(170, 110, 150, 21);
105           copyright_info_tf.setText(option.getCopyright());
106           copyright_info_tf.setDocument(new JExtendTextField("^ㄱ-ㅎㅏ-ㅣ가-힣",40));
107
108           copyright_imfo_lb.setText("저작자 정보");
109           add(copyright_imfo_lb);
110           copyright_imfo_lb.setBounds(40, 110, 64, 15);
111           setResizable(false);
112
113           copyright_save_btn.setText("적용");
114           copyright_save_btn.setBounds(150, 180, 83, 23);
115           add(copyright_save_btn);
116           copyright_save_btn.addActionListener(new java.awt.event.ActionListener() {
117               public void actionPerformed(java.awt.event.ActionEvent evt) {
118                   saveBtnEvent();
119               }
120           });
121           copyright_cancel_btn.setText("닫기");
122           copyright_cancel_btn.setBounds(250, 180, 83, 23);
123           add(copyright_cancel_btn);
124           copyright_cancel_btn.addActionListener(new java.awt.event.ActionListener() {
125               public void actionPerformed(java.awt.event.ActionEvent evt) {
126                   closeBtnEvent();
127               }
128           });
129
130           pack();
131
132       }
133
134       /**
135        * 저작권 옵션 사용 여부에 따른 하위 컴포넌트들의 사용 여부 설정
136        */
137       private void enableCopyrightOption() {
138           //저작권설정 사용여부에 따라 하위 옵션들을 확성화, 비활성화 하는 메소드
139           if (copyright_option_enable.isSelected()) {
140               copyright_location_lb.setEnabled(true);
```

```
141             copyright_imfo_lb.setEnabled(true);
142             copyright_info_tf.setEnabled(true);
143             copyright_location_cb.setEnabled(true);
144         } else {
145             copyright_location_lb.setEnabled(false);
146             copyright_imfo_lb.setEnabled(false);
147             copyright_info_tf.setEnabled(false);
148             copyright_location_cb.setEnabled(false);
149         }
150     }
151 
152     /**
153      * 저장 버튼 이벤트 핸들러
154      */
155     private void saveBtnEvent() {
156         //적용버튼 이벤트: 설정값 저장
157         if (copyright_option_enable.isSelected()) {
158             option.setCopyright(copyright_info_tf.getText());
159             option.setCopyrightAlign(convertItemToAlign(copyright_location_cb.getSelectedItem()));
160             option.saveOption();
161         } else {
162             option.setCopyright("");
163             option.setCopyrightAlign(Option.Align.RIGHT);
164             option.saveOption();
165         }
166         dispose();
167     }
168 
169     /**
170      * 닫기 버튼 이벤트 핸들러
171      */
172     private void closeBtnEvent() {
173         dispose();
174     }
175 
176     /**
177      * 콤보박스 내의 값을 옵션 설정 값으로 변환하는 메소드
178      * @param obj 콤보박스 내 오브젝트
179      * @return 옵션 설정 값
180      */
181     private Option.Align convertItemToAlign(Object obj){
182         String s = obj.toString();
183         if(s.equals("왼쪽")){
184             return Option.Align.LEFT;
185         }else if(s.equals("중앙")){
186             return Option.Align.CENTER;
187         }else if(s.equals("오른쪽")){
188             return Option.Align.RIGHT;
```

```
189         }else{
190             return Option.Align.CENTER;
191         }
192     }
193 
194     /**
195      * 옵션 설정 값을 콤보 박스 내 인덱스로 변환하는 메소드
196      * @param align 옵션 설정 값
197      * @return 콤보 박스 내 인덱스
198      */
199     private int convertAlignToItemIndex(Option.Align align){
200         if(align == Option.Align.LEFT){
201             return 0;
202         }else if(align == Option.Align.CENTER){
203             return 1;
204         }else if(align == Option.Align.RIGHT){
205             return 2;
206         }else{
207             return 0;
208         }
209     }
210 
211 }
212 
```

FileSelect.java

```
1  package display;
2  import java.io.File;
3  import java.io.IOException;
4
5  import javax.swing.JOptionPane;
6
7  import loader.Checker;
8  import option.Option;
9
10  /**
11   * 선택된 파일 혹은 폴더에 대한 처리 클래스
12   * @author 유병호
13   *
14   */
15  public class FileSelect{
16      /**
17       * 소스코드 정보
18       */
19      private Checker checker;
20
21      /**
22       * 옵션 정보
23       */
24      private Option option;
25
26      /**
27       * 생성자
28       * @param chker 소스코드 정보
29       * @param op 옵션 정보
30       */
31      public FileSelect(Checker chker, Option op) {
32          checker = chker;
33          option = op;
34      }
35
36      /**
37       * 소스코드 정보를 반환한다.
38       * @return 소스코드 정보
39       */
40      public Checker getChecker(){
41          return checker;
42      }
43
44      /**
45       * 선택한 파일을 처리한다.
46       * @param f 선택 된 파일
47       */
```

```
48      public void selectFile(File f) {
49          //단일파일 선택 메소드
50          if(option.getCodeLanguage() == Option.Language.JAVA){
51              checker.explorerFile(f.toString(), new String[] {".java"});
52          }else if(option.getCodeLanguage() == Option.Language.C){
53              checker.explorerFile(f.toString(), new String[] {".c", ".h"});
54          }else if(option.getCodeLanguage() == Option.Language.CPP){
55              checker.explorerFile(f.toString(), new String[] {".cpp", ".h"});
56          }
57
58          if(checker.fileList.size() == 0){//설정한 확장자와 선택한 확장자가 다를시 나타나는 경고 다이얼로그
59              JOptionPane.showMessageDialog(null, "옵션에 설정된 언어의 확장자와 일치하지 않습니다!", "오류
발생", JOptionPane.WARNING_MESSAGE);
60          }
61      }
62
63      /**
64       * 선택한 디렉토리를 처리한다.
65       * @param source 선택된 디렉토리
66       */
67      public void selectDirectory(File source){
68          checker.clearAllList();
69          //디렉토리 선택 메소드
70          checker.rootPath = source.getName();
71          if(option.getCodeLanguage() == Option.Language.JAVA){
72              try {
73                  checker.explorerDirectory(source.toString(), new String[] {".java"});
74              } catch (IOException e) {
75                  JOptionPane.showMessageDialog(null, "폴더가 손상되었거나 읽기 권한이 없습니다.", "오류발생
", JOptionPane.ERROR_MESSAGE);
76              }
77          }else if(option.getCodeLanguage() == Option.Language.C){
78              try {
79                  checker.explorerDirectory(source.toString(), new String[] {".c", ".h"});
80              } catch (IOException e) {
81                  JOptionPane.showMessageDialog(null, "폴더가 손상되었거나 읽기 권한이 없습니다.", "오류발생
", JOptionPane.ERROR_MESSAGE);
82              }
83          }else if(option.getCodeLanguage() == Option.Language.CPP){
84              try {
85                  checker.explorerDirectory(source.toString(), new String[] {".cpp", ".h"});
86              } catch (IOException e) {
87                  JOptionPane.showMessageDialog(null, "폴더가 손상되었거나 읽기 권한이 없습니다.", "오류발생
", JOptionPane.ERROR_MESSAGE);
88              }
89          }
90      }
91  }
92
```

JExtendTextField.java

```
1  package display;
2
3  import java.util.ArrayList;
4  import javax.swing.text.*;
5
6  /**
7   * JTextField를 보조하는 클래스 입니다.
8   * @author 강승민
9   *
10   */
11  public class JExtendTextField extends PlainDocument {
12      private static final long serialVersionUID = -2596672324494951659L;
13      /** 차단 또는 허용할 문자를 담고있는 규격화된 문자열 입니다. */
14      private String restrict = null;
15
16      /** 차단 또는 허용할 문자를 담고있는 규격화된 문자열을 처리하는 배열 입니다. */
17      private char[] restrictSet;
18      /** 허용할 문자를 담는 배열입니다. */
19      private ArrayList<Character> permissionChar = new ArrayList<Character>();
20      /** 허용할 문자의 범위를 지정하는 배열의 시작 문자입니다. */
21      private ArrayList<Character> permissionCharStart = new ArrayList<Character>();
22      /** 허용할 문자의 범위를 지정하는 배열의 끝 문자입니다. */
23      private ArrayList<Character> permissionCharEnd = new ArrayList<Character>();
24      /** 차단할 문자를 담는 배열입니다. */
25      private ArrayList<Character> blockChar = new ArrayList<Character>();
26      /** 차단할 문자의 범위를 지정하는 배열의 시작 문자입니다. */
27      private ArrayList<Character> blockCharStart = new ArrayList<Character>();
28      /** 차단할 문자의 범위를 지정하는 배열의 끝 문자입니다. */
29      private ArrayList<Character> blockCharEnd = new ArrayList<Character>();
30
31      /** 입력 가능한 문자의 수를 나타냅니다. */
32      private int limit = 10;
33
34      /**
35       * 입력 가능한 문자열 길이 10, 모든 문자 입력 가능합니다.
36       */
37      public JExtendTextField(){
38          super();
39      }
40      /**
41       * 입력 가능한 문자열 길이 n, 모든 문자 입력 가능합니다.
42       * @param limit 제한 길이
43       */
44      public JExtendTextField(int limit){
45          setLimit(limit);
46      }
47      /**
```

```
48          * 입력 가능한 문자열 길이 10, restrict 규격에 따라 허용/차단 문자열이 정해집니다.
49          * @param restrict 규격화된 문자열
50          */
51         public JExtendTextField(String restrict){
52             setRestrict(restrict);
53         }
54 
55         /**
56          * 입력 가능한 문자열 길이 n, restrict 규격에 따라 허용/차단 문자열이 정해집니다.
57          * @param restrict 규격화된 문자열
58          * @param limit 제한 길이
59          */
60         public JExtendTextField(String restrict, int limit){
61             setRestrict(restrict);
62             setLimit(limit);
63         }
64 
65         /**
66          * 입력 가능한 길이를 반환합니다.
67          * @return 제한 길이
68          */
69         public int getLimit(){
70             return limit;
71         }
72         /**
73          * 입력 가능한 문자열의 길이를 설정합니다.
74          * @param limit 길이를 설정합니다. 1 이상의 값이어야 합니다.
75          */
76         public void setLimit(int limit){
77             this.limit = Math.abs(limit);
78         }
79 
80         /**
81          * 허용/차단된 문자열 식을 반환합니다.
82          * @return 문자열 식
83          */
84         public String getRestrict(){
85             return restrict;
86         }
87         /**
88          * 문자열의 식에 따라 문자를 허용/차단 시킬 수 있습니다.<p>
89          * <div><b><span style="font-size: 12pt;">문자의 입력을 차단,허용을 하는 Restrict에 대한 상세 설명
</span></b></div><div><br></div><div>JExtendTextField 안에 존재하는 restrict 라는 문자열 변수는 특정한
규격에 따라 일부 문자만 입력되도록 하거나 혹은 특정 문자를 차단할 수 있도록 합니다
.</div><div><br></div><div>생성자에서 문자열을 전달하거나 setRestrict 를 통해 설정할 수 있습니다
.</div><div><br></div><div>다음의 예제는 영문 소문자만 입력할 수 있도록 합니다. 하이픈 ('-') 을 입력하면 문
자의 범위를 지정할 수 있습니다.</div><div><br></div><div>setRestrict("a-
z")</div><div><br></div><div><br></div><div>다음의 예제는 영문 대,소문자 및 숫자와 공백을 입력할 수 있
```


도록 합니다.
 setRestrict("A-Za-z0-9
")

 다음의 예제는 한글만 입력할 수 있도록 합니다. 한글의 경우 입력 방식의 차이로 ㄱ-ㅎ 은 필히 추가해야하는 문제가 있습니다.
 setRestrict("ㄱ-ㅎ가-힣")

 다음의 예제는 특수문자 (! @ ^ ( ) < > ₩ - _) 만 입력할 수 있도록 합니다.
 setRestrict("!@₩₩^()<>₩₩₩₩-
_")
 특수문자 - 및 ^ 는 문자열 규칙에 이용되어지므로 특정된 두 문자를 차단하고 싶을 경우에는 백슬래시(₩₩)를 붙여서 입력합니다.

 다음의 예제는 영문 대문자만 입력되도록 하되 문자 S는 차단합니다. 캐럿('^') 이후의 모든 문자들은 차단 문자로 등록 되어집니다.
 setRestrict("A-Z^S")

 다음의 예제는 모든 문자의 입력을 허용하지만 특수문자 ^와 &는 차단합니다
.
 setRestrict("^₩₩^&")

 다음의 예제는 직접 유니코드의 코드를 입력하여 문자열을 허용 차단하는 방식입니다. 소문자 a ~ g 까지 입력 받지만 소문자 e는 차단합니다.
 setRestrict("₩u0061-
₩u0067^₩u0065")

 restrict 가 null이거나 공백인 경우에는 모든 입력값을 허용합니다. 어떠한 문자도 입력받고 싶지 않다면 다음의 코드를 사용합니다
.
 setRestrict("^₩u0000-
₩uFFFF")

 restrict는 일반적으로 허용보다는 차단의 우선순위가 더 높습니다. 특정 문자가 허용과 차단에 모두 들어있을 경우 해당 문자는 차단됩니다. setRestrict("A-Z^S") 위 코드는 대문자 A부터 Z까지의 입력을 허용하므로 대문자 S 역시 허용 문자에 포함되어지나 차단문자에도 들어있으므로 결과적으로는 입력이 차단 됩니다.

```
90          * @param rest 규격화된 문자열 입니다.
91          */
92         public void setRestrict(String rest){
93             restrict = rest;
94 
95             if(restrict != null){
96                 restrictSet = restrict.toCharArray();
97                 int len = restrictSet.length;
98                 boolean blocked = false;
99                 for(int i = 0 ; i < len ; i++){
100                     if(restrictSet[i] == '^'){
101                         if(i > 0){
102                             if(restrictSet[i-1] == '₩₩'){
103                                 if(!blocked){
104                                     permissionChar.add(restrictSet[i]);
105                                 }else{
106                                     blockChar.add(restrictSet[i]);
107                                 }
108                                 continue;
109                             }
110                         }
111                         blocked = true;
112                         continue;
113                     }
114                     if(!blocked){
115                         if(restrictSet[i] == '-' && i != 0 && i != len-1){
116                             if(restrictSet[i-1] == '₩₩'){
117                                 permissionChar.add(restrictSet[i]);
```



```
                        continue;
                    }
                    permissionCharStart.add(restrictSet[i-1]);
                    permissionCharEnd.add(restrictSet[i+1]);
                }else{
                    if(restrictSet[i] == '₩' && i < len-1){
                        if(restrictSet[i+1] != '-' && restrictSet[i+1] != '^'){
                            permissionChar.add(restrictSet[i]);
                        }
                    }else{
                        permissionChar.add(restrictSet[i]);
                    }
                }
            }else{
                if(restrictSet[i] == '-' && i != 0 && i != len-1){
                    if(restrictSet[i-1] == '₩'){
                        blockChar.add(restrictSet[i]);
                        continue;
                    }
                    blockCharStart.add(restrictSet[i-1]);
                    blockCharEnd.add(restrictSet[i+1]);
                }else{
                    if(restrictSet[i] == '₩' && i < len-1){
                        if(restrictSet[i+1] != '-' && restrictSet[i+1] != '^'){
                            blockChar.add(restrictSet[i]);
                        }
                    }else{
                        blockChar.add(restrictSet[i]);
                    }
                }
            }
        }
    }
}

/**
 * 등록된 JTextField에 문자가 입력될 경우 발생하는 메소드 입니다.
 */
public void insertString(int offset, String str, AttributeSet attr) throws BadLocationException{
    if(str == null) return;

    boolean pass = true;
    if(restrict != null){
        pass = false;
        for(int i = 0 ; i < str.length() ; i++){
            char c = str.charAt(i);
            if(permissionChar.size() + permissionCharStart.size() > 0){
                for(int j = 0 ; j < permissionChar.size() ; j++){
```

```
166                     if(c == permissionChar.get(j)){
167                         pass = true;
168                         break;
169                     }
170                 }
171                 if(!pass){
172                     for(int j = 0 ; j < permissionCharStart.size() ; j++){
173                         if(c >= permissionCharStart.get(j) && c <= permissionCharEnd.get(j)){
174                             pass = true;
175                             break;
176                         }
177                     }
178                 }
179             }else{
180                 pass = true;
181             }
182 
183             if(blockChar.size() + blockCharStart.size() > 0){
184                 if(pass){
185                     for(int j = 0 ; j < blockChar.size() ; j++){
186                         if(c == blockChar.get(j)){
187                             pass = false;
188                             break;
189                         }
190                     }
191                     if(pass){
192                         for(int j = 0 ; j < blockCharStart.size() ; j++){
193                             if(c >= blockCharStart.get(j) && c <= blockCharEnd.get(j)){
194                                 pass = false;
195                                 break;
196                             }
197                         }
198                     }
199                 }
200             }
201             if(!pass){
202                 return;
203             }
204         }
205     }
206 
207     if(this.getLength() + str.length() <= limit && pass){
208         super.insertString(offset, str, attr);
209     }
210   }
211 }
212 
```

Logo.java

```
1  package display;
2
3  import java.awt.*;
4  import javax.swing.*;
5
6  import option.OptionFile;
7  import option.Option;
8
9  /**
10   * 최초 실행 시 보이는 로고 클래스입니다.
11   * @author 김선일
12   *
13   */
14  public class Logo implements Runnable {
15
16      /**
17       * 사용자 모니터 해상도를 가져오기 위한 속성
18       */
19      private Toolkit tk = Toolkit.getDefaultToolkit();
20
21      /**
22       * 사용자 모니터의 가로 해상도
23       */
24      private int screensize_width = (int) tk.getScreenSize().getWidth();
25
26      /**
27       * 사용자 모니터의 세로 해상도
28       */
29      private int screensize_height = (int) tk.getScreenSize().getHeight();//화면 해상도의 세로
30
31      /**
32       * 프레임 가로 크기
33       */
34      private int framesize_width;
35
36      /**
37       * 프레임 세로 크기
38       */
39      private int framesize_height;
40
41      /**
42       * 로고 표시 프레임
43       */
44      private JDialog logo_frame;
45
46      /**
47       * 옵션 파일 처리
```

```
48      */
49     private OptionFile of = new OptionFile();
50 
51     /**
52      * 옵션 파일
53      */
54     private Option option;
55 
56     /**
57      * 초기화를 위한 생성자
58      */
59     public Logo(){
60         init();
61     }
62 
63     /**
64      * 초기화 메서드
65      * 옵션 파일 로드와 프레임 위치 지정 작업 수행
66      */
67     public void init(){
68         option = of.readOptionFile();
69 
70         ImageIcon logo_img_icon = new ImageIcon("logo.png");//로고에 쓰일 이미지 변수 선언
71 
72         logo_frame = new JDialog(); // 로고 프레임 생성
73         logo_frame.setPreferredSize(new Dimension(300, 300)); // 로고 프레임의 크기 설정
74         setFrameLocation(logo_frame.getPreferredSize(), logo_frame); // 로고를 화면 정 가운데 위치하게 함
75         JLabel logo_img_label = new JLabel(logo_img_icon);    // 로고이미지를 라벨로 선언
76         logo_frame.add(logo_img_label);    // 로고이미지 라벨을 프레임에 부착
77     }
78 
79     /**
80      * 프레임 위치 지정 메서드
81      * @param d 프레임의 위치
82      * @param c 프레임
83      */
84     public void setFrameLocation(Dimension d, Component c) {
85         framesize_width = (int) d.getWidth();    //프레임의 가로깅이
86         framesize_height = (int) d.getHeight();    //프레임의 세로길이
87 
88         c.setLocation(screensize_width / 2 - framesize_width / 2, screensize_height / 2 - framesize_height /
2);
89     }
90 
91     /**
92      * 로고 표시 작업을 위한 스레드 실행 메소드
93      */
94     public void run() {
```

```
95          //로고 애니메이션 실행 후 메인프레임 실행
96          logo_frame.setUndecorated(true);    // 로고프레임의 테두리를 제거
97          logo_frame.setOpacity(0);           // 로고 프레임의 불투명도를 0으로 설정
98          logo_frame.pack();
99          logo_frame.setVisible(true);
100          try {
101              //로고 프레임이 서서히 나타나도록 표시함.
102              logo_frame.setAlwaysOnTop(true);
103              for (float i = 0; i <= 50; i++) {
104                  logo_frame.setOpacity((float) (0.02 * i));
105                  Thread.sleep(5);
106              }
107              Main P = new Main(option);
108              P.setMainFrame();    // 메인 프레임을 화면에 띄움
109              P.setVisible(true);
110
111              logo_frame.dispose();
112
113          } catch (InterruptedException e) {
114              //스레드 인터럽트 에러
115              System.exit(0);
116          }
117
118      }
119
120
121      /**
122       * main 메소드
123       * @param args 이 값은 사용하지 않습니다.
124       */
125      public static void main(String[] args) {
126          Logo logo = new Logo();
127          new Thread(logo).start();
128      }
129
130  }
131
```

Main.java

```
1  package display;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.io.*;
6
7  import javax.swing.*;
8  import javax.swing.filechooser.FileNameExtensionFilter;
9
10  import loader.Checker;
11  import option.Option;
12  import option.OptionDefault;
13  import option.Security;
14
15  /**
16   * 메인 프레임과 패널
17   * @author 김선일
18   *
19   */
20  public class Main extends JFrame {
21
22
23      /**
24       * 프레임 시리얼 UID
25       */
26      private static final long serialVersionUID = -320493438894721666L;
27
28      /**
29       * 타이틀 정보
30       */
31      private static final String label_Title = "SourceCode to PDF";
32
33      /**
34       * 버전 정보
35       */
36      private static final String label_Version = "Ver.Quater";
37
38      /**
39       * 소스파일 선택시 특정 파일만 로드할 것인지 디렉토리를 로드할 것인지에 대한 값<br>
40       * 0 : 단일 파일<br>
41       * 1 : 디렉토리
42       */
43      private int dialog_Result;
44
45      /**
46       * 선택한 파일 (혹은 디렉토리) 정보를 표시하는 라벨
47       */
```

```
48      private JLabel main_selectedfile;
49
50      /**
51       * 소스코드 선택 시 파일을 선택할지 폴더를 선택할지 묻는 다이얼로그 창
52       */
53      private JOptionPane main_filechoose_dialog;
54
55      /**
56       * 파일 (혹은 디렉토리)을 선택하는 창
57       */
58      private JFileChooser main_filechooser;
59
60      /**
61       * 소스코드 로드 창 버튼<p>
62       * 이 버튼을 누를 시 main_filechoose_dialog 가 표시됩니다.
63       */
64      private JButton main_load_btn;
65
66      /**
67       * 변환 설정 창 버튼
68       */
69      private JButton main_setting_btn;
70
71      /**
72       * 서식 설정 창 버튼
73       */
74      private JButton main_style_btn;
75
76      /**
77       * 저작권 설정 창 버튼
78       */
79      private JButton main_copyright_btn;
80
81      /**
82       * 보안 설정 창 버튼
83       */
84      private JButton main_secure_btn;
85
86      /**
87       * 변환 창 버튼
88       */
89      private JButton main_convert_btn;
90
91      /**
92       * 도움말 (README) 창 표시 버튼
93       */
94      private JButton main_help_btn;
95
```

```
96      /**
97       * 메인 창에 표시할 로드된 소스코드 정보
98       */
99      private JList<String> main_file_list;
100
101      /**
102       * 메인 창에 표시할 로드된 소스코드 정보 (스크롤)
103       */
104      private JScrollPane main_file_scl;
105
106      /**
107       * 옵션 정보
108       */
109      private Option option = OptionDefault.getDefaultOption();
110
111      /**
112       * 보안 정보
113       */
114      private Security security = new Security();
115
116      /**
117       * 소스코드 관련 정보
118       */
119      private Checker checker = new Checker();
120
121      /**
122       * 메인 생성자
123       * @param op 로드된 옵션
124       */
125      public Main(Option op){
126          option = op;
127      }
128
129
130      /**
131       * 메인 프레임 설정
132       */
133      public void setMainFrame() {
134          try {
135              UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
136          } catch (Exception e) {
137              System.out.println("LookAndFeel Setting Fail");
138          }
139          setTitle(label_Title + " " + label_Version);
140          setResizable(false);
141
142          setLayout(null);
143          setPreferredSize(new Dimension(550, 300));
```

```
144             this.setLocation((int)(Toolkit.getDefaultToolkit().getScreenSize().getWidth() / 2 - this
.getPreferredSize().getWidth() / 2),
145                     (int)(Toolkit.getDefaultToolkit().getScreenSize().getHeight() / 2 - this
.getPreferredSize().getHeight() / 2));
146 
147             main_selectedfile = new JLabel();
148             main_load_btn = new JButton();
149             main_style_btn = new JButton();
150             main_setting_btn = new JButton();
151             main_copyright_btn = new JButton();
152             main_secure_btn = new JButton();
153             main_convert_btn = new JButton();
154             main_help_btn = new JButton();
155             main_filechoose_dialog = new JOptionPane();
156             main_filechooser = new JFileChooser("./");
157 
158             //16.06.05 05:18 추가된 Main 컴포넌트
159             main_file_list = new JList<String>();
160             main_file_scl = new JScrollPane(main_file_list);
161             add(main_file_scl);
162             main_file_scl.setBounds(38, 100, 345, 100);
163 
164             //16.06.05 05:10 추가된 Main 컴포넌트
165             main_selectedfile.setText("디렉토리 경로");
166             add(main_selectedfile);
167             main_selectedfile.setBounds(38, 39, 500, 15);
168 
169             main_load_btn.setText("소스파일로드");
170             main_load_btn.addActionListener(new ActionListener() {
171                 public void actionPerformed(ActionEvent evt) {
172                     loadBtnEvent(evt);
173                 }
174             });
175             add(main_load_btn);
176 
177             main_load_btn.setBounds(420, 35, 105, 23);
178 
179             main_style_btn.setText("서식설정");
180             main_style_btn.addActionListener(new ActionListener() {
181                 public void actionPerformed(ActionEvent evt) {
182                     styleBtnEvent(evt);
183                 }
184             });
185             add(main_style_btn);
186             main_style_btn.setBounds(38, 220, 81, 23);
187 
188 
189             main_setting_btn.setText("변환설정");
```

```
190         main_setting_btn.addActionListener(new ActionListener() {
191             public void actionPerformed(ActionEvent evt) {
192                 settingBtnEvent(evt);
193             }
194         });
195         add(main_setting_btn);
196         main_setting_btn.setBounds(420, 220, 100, 23);
197 
198 
199 
200         main_copyright_btn.setText("저작권설정");
201         main_copyright_btn.addActionListener(new ActionListener() {
202             public void actionPerformed(ActionEvent evt) {
203                 copyrightBtnEvent(evt);
204             }
205         });
206         add(main_copyright_btn);
207         main_copyright_btn.setBounds(126, 220, 93, 23);
208 
209         main_secure_btn.setText("보안설정");
210         main_secure_btn.addActionListener(new ActionListener() {
211             public void actionPerformed(ActionEvent evt) {
212                 secureBtnEvent(evt);
213             }
214         });
215         add(main_secure_btn);
216         main_secure_btn.setBounds(226, 220, 81, 23);
217 
218         main_convert_btn.setText("변환하기");
219         main_convert_btn.addActionListener(new ActionListener() {
220             public void actionPerformed(ActionEvent evt) {
221                 mainConvertBtnEvent(evt);
222             }
223         });
224         enableConvertBtn();
225 
226         add(main_convert_btn);
227         main_convert_btn.setBounds(420, 130, 100, 70);
228 
229         main_help_btn.setText("도움말");
230         main_help_btn.addActionListener(new ActionListener() {
231             public void actionPerformed(ActionEvent evt) {
232                 helpBtnEvent(evt);
233             }
234         });
235         add(main_help_btn);
236         main_help_btn.setBounds(314, 220, 69, 23);
237 
```

```
238             setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
239             pack();
240         }
241 
242         /**
243          * 소스코드 로드 버튼 이벤트 핸들러
244          * @param evt 버튼 이벤트
245          */
246         private void loadBtnEvent(ActionEvent evt) {
247             // 로드버튼 이벤트
248             showDialog();
249             FileSelect fileSelect = null;
250             if (dialog_Result == 0) {
251                 if(option.getCodeLanguage() == Option.Language.C){
252                     main_filechooser.setFileFilter(new FileNameExtensionFilter("C Files OR Header Files", "c", "h"));
253                 }else if(option.getCodeLanguage() == Option.Language.CPP){
254                     main_filechooser.setFileFilter(new FileNameExtensionFilter("C++ Files OR Header Files", "cpp",
"h"));
255                 }else if(option.getCodeLanguage() == Option.Language.JAVA){
256                     main_filechooser.setFileFilter(new FileNameExtensionFilter("Java Files", "java"));
257                 }
258                 main_filechooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
259                 main_filechooser.showOpenDialog(this);
260 
261                 fileSelect = new FileSelect(checker, option);
262                 if (main_filechooser.getSelectedFile() != null) {
263                     fileSelect.selectFile(main_filechooser.getSelectedFile());
264                 }
265             }
266 
267             if (dialog_Result == 1) {
268                 main_filechooser.setSelectedFile(null);
269 
270                 main_filechooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
271                 main_filechooser.showOpenDialog(this);
272 
273                 fileSelect = new FileSelect(checker, option);
274 
275                 if (main_filechooser.getSelectedFile() != null) {
276                     fileSelect.selectDirectory(main_filechooser.getSelectedFile());
277                 }
278             }
279             if(main_filechooser.getSelectedFile() != null){
280                 main_selectedfile.setText(main_filechooser.getSelectedFile().getPath());
281             }
282 
283             enableConvertBtn();
284 
```

```
285         main_file_list.setListData(checker.fileListString);
286     }
287 
288     /**
289      * 소스코드 로드 여부에 따라 변환버튼 활성화 지정
290      */
291     public void enableConvertBtn() {
292         //소스파일 로드여부에 따라 변환버튼 활성화,비활성화하는 메소드
293         if (checker.fileList.isEmpty())
294             main_convert_btn.setEnabled(false);
295         else
296             main_convert_btn.setEnabled(true);
297     }
298 
299     /**
300      * 변환 버튼 처리 핸들러
301      * @param evt 버튼 이벤트
302      */
303     private void mainConvertBtnEvent(ActionEvent evt) {
304         // 변환버튼 이벤트: 변환버튼을 누르면 변환프레임이 띄워지게 됨
305         Convert c = new Convert(checker, option, security);
306         c.setConvertDialog();
307         c.setVisible(true);
308     }
309 
310     /**
311      * 서식 설정 버튼 처리 핸들러
312      * @param evt 버튼 이벤트
313      */
314     private void styleBtnEvent(ActionEvent evt) { //
315         // 서식설정버튼 이벤트: 서식설정버튼을 누르면 서식성정프레임이 띄워지게 됨
316         Style s = new Style(option);
317         s.setStylePanel();
318         s.setVisible(true);
319     }
320 
321     /**
322      * 변환 설정 버튼 처리 핸들러
323      * @param evt 버튼 이벤트
324      */
325     private void settingBtnEvent(ActionEvent evt) { //
326         // 서식설정버튼 이벤트: 서식설정버튼을 누르면 서식성정프레임이 띄워지게 됨
327         Setting t = new Setting(option);
328         t.setSettingPanel();
329         t.setVisible(true);
330     }
331 
332     /**
```

```
333         * 저작권 설정 버튼 처리 핸들러
334         * @param evt 버튼 이벤트
335         */
336        private void copyrightBtnEvent(ActionEvent evt) {
337            // 저작권설정 버튼 이벤트: 저작권설정 버튼을 누르면 저작권설정프레임이 띄워지게 됨
338            Copyright c = new Copyright(option);
339            c.setCopyrightPanel();
340            c.setVisible(true);
341        }
342
343        /**
344         * 보안 설정 버튼 처리 핸들러
345         * @param evt 버튼 이벤트
346         */
347        private void secureBtnEvent(ActionEvent evt) {
348            // 보안설정 버튼 이벤트: 보안설정버튼을 누르면 보안설정프레임이 띄워지게 됨
349            Secure s = new Secure(security);
350            s.setSecurePanel();
351            s.setVisible(true);
352
353        }
354
355        /**
356         * 도움말 버튼 처리 핸들러
357         * @param evt 버튼 이벤트
358         */
359        private void helpBtnEvent(ActionEvent evt) {
360            // 도움말 버튼 이벤트:도움말 버튼을 누르면 지정된 경로의 파일이 실행되게 됨
361            try {
362                File f = new File("README.txt");//도움말 파일. 문자열 수정으로 변경 가능
363                Desktop.getDesktop().open(f);
364            } catch (IOException e) {
365                System.out.println("README 파일 로드에 실패했습니다");
366            }
367        }
368
369        /**
370         * 다이얼로그 출력 메소드
371         */
372        @SuppressWarnings("static-access")
373        private void showDialog() {
374            // 로드버튼 누를시 파일선택모드를 묻는 다이얼로그 (파일:0, 다랙토리 :1)
375            String[] options = { "  파일  ", "  폴더  ", "  취소  " };
376            dialog_Result = main_filechoose_dialog.showOptionDialog(this, "  파일을 변환 합니까? 폴더 전체를
변환합니까?  ", "소스코드 로드",
377                    JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null, options, "파
일");
378        }
```

```
379
380 }
381
```


Secure.java

```
1  package display;
2  import java.awt.Dimension;
3  import java.awt.Toolkit;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6
7  import javax.swing.*;
8
9  import option.Security;
10
11  /**
12   * 보안 설정 창입니다.
13   * @author 김선일
14   *
15   */
16  public class Secure extends JDialog {
17
18      /**
19       * SerialUID
20       */
21      private static final long serialVersionUID = -5059371536601380535L;
22
23      /**
24       * 복사 가능 여부 설정 체크 박스
25       */
26      private JCheckBox secure_copy_cb = new JCheckBox();
27
28      /**
29       * 보안 옵션 사용 여부 설정 체크박스
30       */
31      private JCheckBox secure_enable_cb = new JCheckBox();
32
33      /**
34       * 비밀번호 입력 필드
35       */
36      private JPasswordField jPasswordField1 = new JPasswordField();
37
38      /**
39       * 입력 비밀번호 재확인 필드
40       */
41      private JPasswordField jPasswordField2 = new JPasswordField();
42
43      /**
44       * 비밀번호 입력 라벨
45       */
46      private JLabel jLabel1 = new JLabel();
47
```

```
48      /**
49       * 비밀번호 재확인 라벨
50       */
51      private JLabel jLabel2 = new JLabel();
52
53      /**
54       * 보안 정보 저장 버튼
55       */
56      private JButton secure_save_btn = new JButton();
57
58      /**
59       * 보안 창 닫기 버튼
60       */
61      private JButton cancelBtn = new JButton();
62
63      /**
64       * 보안 정보
65       */
66      private Security security;
67
68      /**
69       * 생성자
70       * @param sec 보안 정보
71       */
72      public Secure(Security sec){
73          security = sec;
74      }
75
76      /**
77       * 디스플레이 셋업
78       */
79      public void setSecurePanel() {
80          //보안설정 프레임의 컴포넌트 설정
81          setModal(true);
82          setTitle("보안 설정");
83
84          setPreferredSize(new Dimension(360, 250));
85          this.setLocation((int)(Toolkit.getDefaultToolkit().getScreenSize().getWidth() / 2 - this
.getPreferredSize().getWidth() / 2),
86                  (int)(Toolkit.getDefaultToolkit().getScreenSize().getHeight() / 2 - this
.getPreferredSize().getHeight() / 2));
87
88          setLayout(null);
89
90          secure_enable_cb.setBounds(35, 9, 117, 23);
91
92          secure_enable_cb.setText("보안설정 사용");
93
```

```
94          secure_enable_cb.setSelected(security.getUseSecurityOption());
95          enableSecureOption();
96
97          secure_enable_cb.addActionListener(new ActionListener() {
98              public void actionPerformed(ActionEvent evt) {
99                  enableSecureOption();
100             }
101         });
102         add(secure_enable_cb);
103         secure_copy_cb.setBounds(35, 39, 121, 23);
104         secure_copy_cb.setText("복사 가능 여부");
105         secure_copy_cb.setSelected(security.getUseCopyMode());
106         add(secure_copy_cb);
107
108
109         add(jPasswordField1);
110         jPasswordField1.setBounds(130, 84, 186, 21);
111         jPasswordField1.setDocument(new JExtendTextField(20));
112         add(jPasswordField2);
113         jPasswordField2.setBounds(130, 111, 186, 21);
114         jPasswordField2.setDocument(new JExtendTextField(20));
115
116         jLabel1.setText("비밀번호 입력");
117         add(jLabel1);
118         jLabel1.setBounds(49, 87, 76, 15);
119
120         jLabel2.setText("비밀번호 확인");
121         add(jLabel2);
122         jLabel2.setBounds(49, 111, 76, 15);
123
124         secure_save_btn.setText("적용");
125         secure_save_btn.addActionListener(new ActionListener() {
126             public void actionPerformed(ActionEvent evt) {
127                 saveBtnEvent();
128             }
129         });
130         add(secure_save_btn);
131         secure_save_btn.setBounds(120, 170, 77, 23);
132
133         cancelBtn.setText("닫기");
134         cancelBtn.addActionListener(new ActionListener() {
135             public void actionPerformed(ActionEvent evt) {
136                 closeBtnEvent();
137             }
138         });
139         add(cancelBtn);
140         cancelBtn.setBounds(240, 170, 77, 23);
141         setResizable(false);
```

```
142
143             pack();
144         }
145
146         /**
147          * 보안 정보 사용 여부에 따른 하위 컴포넌트 활성화 여부 지정 메소드
148          */
149         private void enableSecureOption() {
150             //보안설정옵션 사용여부에 따라 하위 옵션들을 활성화, 비활성화 하는 메소드
151             if (secure_enable_cb.isSelected()) {
152                 //secure_enable_cb.setEnabled(true);
153                 jPasswordField1.setEnabled(true);
154                 jPasswordField2.setEnabled(true);
155                 secure_copy_cb.setEnabled(true);
156                 jLabel1.setEnabled(true);
157                 jLabel2.setEnabled(true);
158
159             } else {
160                 //secure_enable_cb.setEnabled(false);
161                 jPasswordField1.setEnabled(false);
162                 jPasswordField2.setEnabled(false);
163                 secure_copy_cb.setEnabled(false);
164                 jLabel1.setEnabled(false);
165                 jLabel2.setEnabled(false);
166             }
167
168         }
169
170         /**
171          * 비밀번호 확인 메소드
172          * @return 비밀번호가 입력되지 않았거나 일치하지 않으면 false, 일치하면 true
173          */
174         private boolean checkPassword(){
175             //사용자가 입력한 암호가 동일한지여부와 비밀번호의 사용 여부
176             if
(String.valueOf(jPasswordField1.getPassword()).equals(String.valueOf(jPasswordField2.getPassword()))){
177                 if(String.valueOf(jPasswordField1.getPassword()).equals("")){
178                     security.setUsePassword(false);
179                 }else{
180                     security.setUsePassword(true);
181                 }
182                 return true;
183             }else{//일치하지 않을 경우 경고 다이얼로그 출력.
184                 JOptionPane.showMessageDialog(null, "비밀번호가 일치하지 않습니다!", "비밀번호 입력 확인",
JOptionPane.WARNING_MESSAGE);
185
186                 return false;
187             }
```



```
188     }
189
190     /**
191      * 저장 버튼 이벤트 핸들러
192      */
193     private void saveBtnEvent() {
194         // 적용버튼 이벤트: 설정값 저장
195         boolean chk = checkPassword();
196         if(!chk){
197             return;
198         }
199         if(secure_enable_cb.isSelected()){
200             security.setUseSecurityOption(secure_enable_cb.isSelected());
201             security.setUseCopyMode(secure_copy_cb.isSelected());
202             security.setPassword(String.valueOf(jPasswordField1.getPassword()));
203             dispose();
204         }else if(!secure_enable_cb.isSelected()){
205             security.setUseSecurityOption(false);
206             security.setUsePassword(false);
207             dispose();
208         }
209     }
210
211     /**
212      * 창 닫기 버튼 이벤트 핸들러
213      */
214     private void closeBtnEvent() {
215         //닫기버튼 이벤트: 현재 창을 닫음
216         dispose();
217     }
218
219 }
220
```

Setting.java

```
1  package display;
2
3  import java.awt.Toolkit;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6
7  import javax.swing.*;
8  import option.Option;
9  import option.OptionDefault;
10
11  /**
12   * 변환 설정 창
13   * @author 유병호
14   *
15   */
16  public class Setting extends JDialog {
17     /**
18      * SerialUID
19      */
20     private static final long serialVersionUID = 8905550640457174616L;
21
22     /**
23      * 옵션 정보
24      */
25     private Option option;
26
27     /**
28      * 저장 위치 지정을 위한 폴더 선택기
29      */
30     private JFileChooser locate_chooser = new JFileChooser();
31
32     /**
33      * 줄번호 표시 여부 체크박스
34      */
35     private JCheckBox linenumber_show_jb = new JCheckBox();
36
37     /**
38      * 변환 방식 표시 라벨
39      */
40     private JLabel conver_form_lb = new JLabel();
41
42     /**
43      * 변환 언어 표시 라벨
44      */
45     private JLabel type_form_lb = new JLabel();
46
47     /**
```

```
48      * 변환 언어 관련 콤보박스
49      */
50     private JComboBox<String> type_form_cb = new JComboBox<String>();
51
52     /**
53      * 탭 길이 지정 라벨
54      */
55     private JLabel tab_size_lb = new JLabel();
56
57     /**
58      * 탭 길이 지정 콤보박스
59      */
60     private JComboBox<Integer> tab_size_cb = new JComboBox<Integer>();
61
62     /**
63      * 헤더 내용 지정 라벨
64      */
65     private JLabel header_content_lb = new JLabel();
66
67     /**
68      * 헤더 내용 입력 필드
69      */
70     private JTextField header_content_tf = new JTextField();
71
72     /**
73      * 헤더 위치 지정 라벨
74      */
75     private JLabel header_location_lb = new JLabel();
76
77     /**
78      * 헤더 위치 지정 콤보박스
79      */
80     private JComboBox<String> header_location_cb = new JComboBox<String>();
81
82     /**
83      * 저장 위치 지정 라벨
84      */
85     private JLabel save_locate_lb = new JLabel();
86
87     /**
88      * 저장 위치 변경 버튼
89      */
90     private JButton save_locate_btn = new JButton();
91
92     /**
93      * 저장 위치 표시 텍스트 필드
94      */
95     private JTextField save_locate_tf = new JTextField();
```



```
96
97      /**
98       * 전체 변환 옵션 시 출력될 PDF 파일의 파일명 지정 라벨
99       */
100      private JLabel PDF_name_lb = new JLabel();
101
102      /**
103       * 전체 변환 옵션 시 출력될 PDF 파일명 지정 텍스트 필드
104       */
105      private JTextField PDF_name_tf = new JTextField();
106
107      /**
108       * 변환 정보 저장 버튼
109       */
110      private JButton setting_reset_btn = new JButton();
111
112      /**
113       * 변환 정보 저장 버튼
114       */
115      private JButton setting_save_btn = new JButton();
116
117      /**
118       * 변환 설정 창 닫기 버튼
119       */
120      private JButton setting_cancel_btn = new JButton();
121
122      /**
123       * 종합 변환 옵션 체크박스
124       */
125      private JCheckBox convert_all_rb = new JCheckBox();
126
127      /**
128       * 패키지 변환 옵션 체크박스
129       */
130      private JCheckBox convert_package_rb = new JCheckBox();
131
132      /**
133       * 파일 변환 옵션 체크박스
134       */
135      private JCheckBox convert_file_rb = new JCheckBox();
136
137      /**
138       * 생성자
139       * @param op 옵션 정보
140       */
141      public Setting(Option op) {
142          option = op;
143      }
```

```
144
145     /**
146      * 디스플레이 셋업
147      */
148     public void setSettingPanel() {
149         setTitle("변환 설정");
150         setModal(true);
151         setPreferredSize(new java.awt.Dimension(500, 350));
152           this.setLocation((int)(Toolkit.getDefaultToolkit().getScreenSize().getWidth() / 2 - this
.getPreferredSize().getWidth() / 2),
153                 (int)(Toolkit.getDefaultToolkit().getScreenSize().getHeight() / 2 - this.getPreferredSize().getHeight()
/ 2));
154         setLayout(null);
155
156         linenumber_show_jb.setText("라인 넘버 표시");
157         add(linenumber_show_jb);
158         linenumber_show_jb.setBounds(40, 20, 200, 23);
159         linenumber_show_jb.addActionListener(new ActionListener() {
160             public void actionPerformed(ActionEvent evt) {
161                 enableLinenumberOptoin();
162             }
163         });
164         if (option.getShowLineNumber()==true) {
165             linenumber_show_jb.setSelected(true);
166             enableLinenumberOptoin();
167         }
168
169         conver_form_lb.setText("변환 방식");
170         add(conver_form_lb);
171         conver_form_lb.setBounds(40, 50, 130, 20);
172
173         convert_all_rb.setText("종합 변환");
174         add(convert_all_rb);
175         convert_all_rb.setBounds(170, 50, 100, 20);
176         convert_all_rb.addActionListener(new ActionListener() {
177             public void actionPerformed(ActionEvent evt) {
178                 converAllOption();
179             }
180         });
181         if (option.getConvertAll()==true) {
182             convert_all_rb.setSelected(true);
183             converAllOption();
184         }
185
186         convert_package_rb.setText("패키지 변환");
187         add(convert_package_rb);
188         convert_package_rb.setBounds(270, 50, 100, 20);
189         convert_package_rb.addActionListener(new ActionListener() {
```

```
190             public void actionPerformed(ActionEvent evt) {
191                 converPackageOption();
192             }
193         });
194         if (option.getConvertPackage()==true) {
195             convert_package_rb.setSelected(true);
196             converPackageOption();
197         }
198 
199         convert_file_rb.setText("개별 변환");
200         add(convert_file_rb);
201         convert_file_rb.setBounds(370, 50, 100, 20);
202         convert_file_rb.addActionListener(new ActionListener() {
203             public void actionPerformed(ActionEvent evt) {
204                 converFileOption();
205             }
206         });
207         if (option.getConvertFile()==true) {
208             convert_file_rb.setSelected(true);
209             converFileOption();
210         }
211 
212         type_form_lb.setText("언어 선택");
213         add(type_form_lb);
214         type_form_lb.setBounds(40, 80, 130, 20);
215 
216         type_form_cb.setModel(new DefaultComboBoxModel<>(new String[] { "자바", "C", "C++" }));
217         type_form_cb.setSelectedIndex(converLanguageToItemIndex(option.getCodeLanguage()));
218         add(type_form_cb);
219         type_form_cb.setBounds(170, 80, 100, 20);
220 
221         tab_size_lb.setText("탭 길이");
222         add(tab_size_lb);
223         tab_size_lb.setBounds(40, 110, 130, 20);
224 
225         tab_size_cb.setModel(new DefaultComboBoxModel<>(new Integer[] {1, 2, 3, 4, 5, 6}));
226         tab_size_cb.setSelectedIndex(converNumToItemIndex(option.getTabSize()));
227         add(tab_size_cb);
228         tab_size_cb.setBounds(170, 110, 100, 20);
229 
230         header_content_lb.setText("헤더 내용");
231         add(header_content_lb);
232         header_content_lb.setBounds(40, 140, 100, 20);
233 
234         add(header_content_tf);
235         header_content_tf.setBounds(170, 140, 100, 20);
236         header_content_tf.setDocument(new JExtendTextField("^ㄱ-ㅎㅏ-ㅣ가-힣", 40));
237         header_content_tf.setText(option.getHeaderTitle());
```

```
238
239         header_location_lb.setText("헤더 위치");
240         add(header_location_lb);
241         header_location_lb.setBounds(40, 170, 100, 20);
242
243         header_location_cb.setModel(new DefaultComboBoxModel<>(new String[] { "왼쪽", "중앙", "오른쪽"
}));
244         header_location_cb.setSelectedIndex(convertAlignToItemIndex(option.getHeaderAlign()));
245         add(header_location_cb);
246         header_location_cb.setBounds(170, 170, 100, 20);
247
248         save_locate_lb.setText("저장 경로");
249         add(save_locate_lb);
250         save_locate_lb.setBounds(40, 200, 130, 20);
251
252         save_locate_btn.setText("변경하기");
253         save_locate_btn.setBounds(170, 200, 100, 20);
254         add(save_locate_btn);
255
256         save_locate_tf.setText(option.getSaveLocate());
257         save_locate_tf.setBounds(290, 200, 170, 20);
258         save_locate_tf.setEditable(false);
259         add(save_locate_tf);
260
261         save_locate_btn.addActionListener(new ActionListener() {
262             public void actionPerformed(ActionEvent evt) {
263                 locateBtnEvent(evt);
264             }
265         });
266
267         PDF_name_lb.setText("파일 이름 설정");
268         add(PDF_name_lb);
269         PDF_name_lb.setBounds(40, 230, 100, 20);
270
271         add(PDF_name_tf);
272         PDF_name_tf.setBounds(170, 230, 100, 20);
273         PDF_name_tf.setDocument(new JExtendTextField("^/₩₩:*?₩"<>|.",200));
274         PDF_name_tf.setText(option.getFileName());
275
276         setResizable(false);
277
278
279         setting_reset_btn.setText("기본값 복원");
280         setting_reset_btn.setBounds(40, 270, 150, 23);
281         add(setting_reset_btn);
282         setting_reset_btn.addActionListener(new ActionListener() {
283             public void actionPerformed(ActionEvent evt) {
284                 OptionDefault.getDefaultOption(option);
```

```
285                 option.saveOption();
286                 dispose();
287             }
288         });
289         setting_save_btn.setText("적용");
290         setting_save_btn.setBounds(250, 270, 83, 23);
291         add(setting_save_btn);
292         setting_save_btn.addActionListener(new ActionListener() {
293             public void actionPerformed(ActionEvent evt) {
294                 saveBtnEvent();
295             }
296         });
297         setting_cancel_btn.setText("닫기");
298         setting_cancel_btn.setBounds(350, 270, 83, 23);
299         add(setting_cancel_btn);
300         setting_cancel_btn.addActionListener(new ActionListener() {
301             public void actionPerformed(ActionEvent evt) {
302                 closeBtnEvent();
303             }
304         });
305         pack();
306     }
307 
308     /**
309      * 종합 변환 체크박스 변화 이벤트 핸들러
310      */
311     private void converAllOption() {
312         if (convert_all_rb.isSelected()) {
313             option.setConvertAll(true);
314         }
315         else {
316             if(!(option.getConvertPackage() || option.getConvertFile())){
317                 JOptionPane.showMessageDialog(null, "변환 옵션중 한 가지는 선택해야 합니다.", "에러",
JOptionPane.ERROR_MESSAGE);
318                 convert_all_rb.setSelected(true);
319             }else{
320                 option.setConvertAll(false);
321             }
322         }
323     }
324 
325     /**
326      * 패키지 변환 체크박스 변화 이벤트 핸들러
327      */
328     private void converPackageOption() {
329         if (convert_package_rb.isSelected()) {
330             option.setConvertPackage(true);
331         }
```



```
332         else {
333             if(!(option.getConvertAll() || option.getConvertFile())){
334                 JOptionPane.showMessageDialog(null, "변환 옵션중 한 가지는 선택해야 합니다.", "에러",
JOptionPane.ERROR_MESSAGE);
335                 convert_package_rb.setSelected(true);
336             }else{
337                 option.setConvertPackage(false);
338             }
339         }
340     }
341 
342     /**
343      * 개별 변환 체크박스 변화 이벤트 핸들러
344      */
345     private void converFileOption() {
346         if (convert_file_rb.isSelected()) {
347             option.setConvertFile(true);
348         }
349         else {
350             if(!(option.getConvertAll() || option.getConvertPackage())){
351                 JOptionPane.showMessageDialog(null, "변환 옵션중 한 가지는 선택해야 합니다.", "에러",
JOptionPane.ERROR_MESSAGE);
352                 convert_file_rb.setSelected(true);
353             }else{
354                 option.setConvertFile(false);
355             }
356         }
357     }
358 
359     /**
360      * 줄번호 표시 체크박스 변화 이벤트 핸들러
361      */
362     private void enableLinenumberOptoin() {
363         if (linenumber_show_jb.isSelected()) {
364             option.setShowLineNumber(true);
365         }
366         else {
367             option.setShowLineNumber(false);
368         }
369     }
370 
371     /**
372      * 저장버튼 이벤트 핸들러
373      */
374     private void saveBtnEvent() {
375         enableLinenumberOptoin();
376         converAllOption();
377         converPackageOption();
```



```
378         converFileOption();
379         option.setCodeLanguage(convertItemToLanguage(type_form_cb.getSelectedItem()));
380         option.setTabSize((int) tab_size_cb.getSelectedItem());
381         option.setHeaderTitle(header_content_tf.getText());
382         option.setHeaderAlign(convertItemToAlign(header_location_cb.getSelectedItem()));
383         option.setFileName(PDF_name_tf.getText());
384         option.setSaveLocate(save_locate_tf.getText());
385         option.saveOption();
386         dispose();
387     }
388 
389     /**
390      * 창 닫기 버튼 이벤트 핸들러
391      */
392     private void closeBtnEvent() {
393         dispose();
394     }
395 
396     /**
397      * 탭 길이 변환 메소드
398      * @param ts 탭 길이
399      * @return 변환된 탭 길이
400      */
401     private int converNumToItemIndex(int ts){
402         if (ts == 1) {
403             return 0;
404         } else if (ts == 2) {
405             return 1;
406         }else if (ts == 3) {
407             return 2;
408         }else if (ts == 4) {
409             return 3;
410         }else if (ts == 5) {
411             return 4;
412         }else if (ts == 6) {
413             return 5;
414         }else {
415             return 3;
416         }
417     }
418 
419     /**
420      * 콤보박스의 언어 내용을 옵션 타입으로 변경하는 메소드
421      * @param obj 콤보박스에 지정된 언어 명칭
422      * @return 변환된 언어 타입
423      */
424     private Option.Language convertItemToLanguage(Object obj) {
425         String s = obj.toString();
```



```
426         if (s.equals("자바")) {
427             return Option.Language.JAVA;
428         } else if (s.equals("C")) {
429             return Option.Language.C;
430         } else if (s.equals("C++")) {
431             return Option.Language.CPP;
432         } else {
433             return Option.Language.JAVA;
434         }
435     }
436 
437     /**
438      * 옵션의 언어 타입을 콤보박스의 인덱스로 변경하는 메소드
439      * @param lang 옵션 정보의 언어 타입
440      * @return 콤보박스의 인덱스
441      */
442     private int converLanguageToItemIndex(Option.Language lang) {
443         if (lang == Option.Language.JAVA) {
444             return 0;
445         } else if (lang == Option.Language.C) {
446             return 1;
447         } else if (lang == Option.Language.CPP) {
448             return 2;
449         } else {
450             return 0;
451         }
452     }
453 
454     /**
455      * 콤보박스의 헤더 위치 내용을 옵션 설정 값으로 바꾸는 메소드
456      * @param obj 콤보박스 값
457      * @return 옵션 설정 값
458      */
459     private Option.Align convertItemToAlign(Object obj){
460         String s = obj.toString();
461         if(s.equals("왼쪽")){
462             return Option.Align.LEFT;
463         }else if(s.equals("중앙")){
464             return Option.Align.CENTER;
465         }else if(s.equals("오른쪽")){
466             return Option.Align.RIGHT;
467         }else{
468             return Option.Align.CENTER;
469         }
470     }
471 
472     /**
473      * 옵션 설정 값을 콤보박스의 인덱스로 바꾸는 메소드
```



```
474      * @param align 옵션 설정 값
475      * @return 콤보박스 인덱스
476      */
477     private int convertAlignToItemIndex(Option.Align align){
478         if(align == Option.Align.LEFT){
479             return 0;
480         }else if(align == Option.Align.CENTER){
481             return 1;
482         }else if(align == Option.Align.RIGHT){
483             return 2;
484         }else{
485             return 0;
486         }
487     }
488 
489     /**
490      * 저장 위치 변경 버튼 이벤트 핸들러
491      * @param evt 버튼 이벤트
492      */
493     private void locateBtnEvent(ActionEvent evt) {
494         locate_chooser.setSelectedFile(null);
495         locate_chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
496         locate_chooser.showOpenDialog(this);
497 
498         if (locate_chooser.getSelectedFile() != null) {
499             save_locate_tf.setText(locate_chooser.getSelectedFile().toString().replaceAll("₩₩₩₩", "/")+"/");
500         }
501     }
502 
503 }
504 
```

Style.java

```
1  package display;
2
3  import java.awt.Toolkit;
4  import java.io.File;
5  import java.io.IOException;
6  import java.util.HashMap;
7  import java.util.Vector;
8
9  import javax.swing.*;
10
11  import com.itextpdf.text.DocumentException;
12  import com.itextpdf.text.Font;
13  import com.itextpdf.text.pdf.BaseFont;
14
15  import option.FontOption;
16  import option.Option;
17  import option.PageOption;
18
19  /**
20   * 서식 설정 창 입니다.
21   * @author 김선일
22   *
23   */
24  public class Style extends JDialog {
25      /**
26       * SerialUID
27       */
28      private static final long serialVersionUID = 529200692386985004L; //
29
30      /**
31       * 옵션 정보
32       */
33      private Option option;
34
35      /**
36       * 각 서식 별 정보를 가지는 탭 패널
37       */
38      private JTabbedPane jTabbedPane1 = new JTabbedPane();
39
40      /**
41       * 용지 설정 탭 패널
42       */
43      private JPanel style_paperset_tab = new JPanel();
44
45      /**
46       * 일반 문자 폰트 설정 탭 패널
47       */
```

```
48      private JPanel style_general_tab = new JPanel();
49
50      /**
51       * 문자열 문자 폰트 설정 탭 패널
52       */
53      private JPanel style_string_tab = new JPanel();
54
55      /**
56       * 키워드 문자 폰트 설정 탭 패널
57       */
58      private JPanel style_keyword_tab = new JPanel();
59
60      /**
61       * 주석 문자 폰트 설정 탭 패널
62       */
63      private JPanel style_comment_tab = new JPanel();
64
65      /**
66       * 줄번호 문자 폰트 설정 탭 패널
67       */
68      private JPanel style_linenumber_tab = new JPanel();
69
70      /**
71       * 서식 저장 버튼
72       */
73      private JButton saveBtn = new JButton();
74
75      /**
76       * 서식 설정 창 닫기 버튼
77       */
78      private JButton cancelBtn = new JButton();
79
80      /**
81       * 지정된 위치의 폰트 [이름,경로] 쌍으로 된 Map 데이터
82       */
83      private HashMap<String,String> fontData = new HashMap<String,String>();
84
85      /**
86       * 폰트 이름 벡터
87       */
88      private Vector<String> fontName = new Vector<String>();
89
90      /**
91       * 일반 문자 폰트의 사용중인 폰트 정보
92       */
93      private String general_FontFamily;
94
95      /**
```

```
96          * 문자열 문자 폰트의 사용중인 폰트 정보
97          */
98         private String string_FontFamily;
99
100         /**
101          * 주석 문자 폰트의 사용중인 폰트 정보
102          */
103         private String comment_FontFamily;
104
105         /**
106          * 키워드 문자 폰트의 사용중인 폰트 정보
107          */
108         private String keyword_FontFamily;
109
110         /**
111          * 줄번호 문자 폰트의 사용중인 폰트 정보
112          */
113         private String linenumber_FontFamily;
114
115         /**
116          * 생성자<p>
117          * 지정된 경로의 폰트 폴더의 폰트들을 로드합니다.
118          * @param op 옵션 정보
119          */
120         public Style(Option op) { //
121             option = op;
122
123             File[] files = new File("fonts/").listFiles();
124             for(int i = 0 ; i < files.length ; i++){
125                 if(files[i].getName().endsWith(".ttf") || files[i].getName().endsWith(".TTF")){
126                     try {
127                         BaseFont bf = BaseFont.createFont(files[i].getPath(), BaseFont.IDENTITY_H,
BaseFont.EMBEDDED);
128                         Font f = new Font(bf);
129                         if(fontData.get(f.getFamilyname()) == null){
130                             fontData.put(f.getFamilyname(), files[i].getPath());
131                             fontName.add(f.getFamilyname());
132                         }
133                         if(option.getBasicFont().getFontName().equals(files[i].getPath())){
134                             general_FontFamily = f.getFamilyname();
135                         }
136                         if(option.getStringFont().getFontName().equals(files[i].getPath())){
137                             string_FontFamily = f.getFamilyname();
138                         }
139                         if(option.getCommentFont().getFontName().equals(files[i].getPath())){
140                             comment_FontFamily = f.getFamilyname();
141                         }
142                         if(option.getKeywordFont().getFontName().equals(files[i].getPath())){
```

```
143                         keyword_FontFamily = f.getFamilyname();
144                     }
145                     if(option.getLineNumberFont().getFontName().equals(files[i].getPath())){
146                         linenumber_FontFamily = f.getFamilyname();
147                     }
148                 } catch (DocumentException | IOException e) {
149                     System.out.println("Font Load Error");
150                 }
151             }
152         }
153     }
154 
155     /**
156      * 서식 설정 창 초기화 메소드
157      */
158     public void setStylePanel() {
159         // 서식설정프레임의 컴ㅁ포넌트 설정
160 
161         setTitle("서식 설정");
162         setModal(true);
163         setPreferredSize(new java.awt.Dimension(500, 370));
164         this.setLocation((int)(Toolkit.getDefaultToolkit().getScreenSize().getWidth() / 2 - this
.getPreferredSize().getWidth() / 2),
165                 (int)(Toolkit.getDefaultToolkit().getScreenSize().getHeight() / 2 - this
.getPreferredSize().getHeight() / 2));
166         setLayout(null);
167 
168         style_paperset_tab.setLayout(null);
169         setPapersetTab(style_paperset_tab);
170         setGeneralTab(style_general_tab);
171         setCommentTab(style_comment_tab);
172         setStringTab(style_string_tab);
173         setKeywordTab(style_keyword_tab);
174         setLinenumberTab(style_linenumber_tab);
175         saveBtn.setText("적용");
176         add(saveBtn);
177         saveBtn.setBounds(300, 300, 83, 23);
178         saveBtn.addActionListener(new java.awt.event.ActionListener() {
179             public void actionPerformed(java.awt.event.ActionEvent evt) {
180                 saveBtnEvent();
181             }
182         });
183         cancelBtn.setText("취소");
184         add(cancelBtn);
185         cancelBtn.addActionListener(new java.awt.event.ActionListener() {
186             public void actionPerformed(java.awt.event.ActionEvent evt) {
187                 closeBtnEvent();
188             }
```

```
189         });
190         cancelBtn.setBounds(400, 300, 83, 23);
191 
192         jTabbedPane1.setBounds(0, 0, 500, 270);
193         jTabbedPane1.addTab("용지설정", style_paperset_tab);
194 
195         jTabbedPane1.addTab("일반", style_general_tab);
196 
197         jTabbedPane1.addTab("문자열", style_string_tab);
198 
199         jTabbedPane1.addTab("키워드", style_keyword_tab);
200 
201         jTabbedPane1.addTab("주석", style_comment_tab);
202 
203         jTabbedPane1.addTab("라인넘버", style_linenumber_tab);
204         add(jTabbedPane1);
205         setResizable(false);
206         pack();
207 
208     }
209 
210     // PaperSetTab 컴포넌트 선언
211     /**
212      * 용지 크기 설정 라벨
213      */
214     private JLabel paper_size_label = new JLabel();
215 
216     /**
217      * 용지 상단 여백 설정 라벨
218      */
219     private JLabel paper_upblank_label = new JLabel();
220 
221     /**
222      * 용지 하단 여백 설정 라벨
223      */
224     private JLabel paper_downblank_label = new JLabel();
225 
226     /**
227      * 용지 좌측 여백 설정 라벨
228      */
229     private JLabel paper_leftblank_label = new JLabel();
230 
231     /**
232      * 용지 우측 여백 설정 라벨
233      */
234     private JLabel paper_rightblank_label = new JLabel();
235 
236     /**
```

```
237         * 용지 상단 여백 설정 입력 필드
238         */
239        private JTextField paper_upblank_tf = new JTextField();
240
241        /**
242         * 용지 하단 여백 설정 입력 필드
243         */
244        private JTextField paper_downblank_tf = new JTextField();
245
246        /**
247         * 용지 좌측 여백 설정 입력 필드
248         */
249        private JTextField paper_leftblank_tf = new JTextField();
250
251        /**
252         * 용지 우측 여백 설정 입력 필드
253         */
254        private JTextField paper_rightblank_tf = new JTextField();
255
256        /**
257         * 용지 크기 콤보박스
258         */
259        private JComboBox<PageOption.PageType> paper_size_cb = new
JComboBox<PageOption.PageType>();
260
261        /**
262         * 용지 설정 탭 초기화 메소드
263         * @param j 삽입 되어질 부모 패널
264         */
265        private void setPapersetTab(JPanel j) {
266            // PapersetTab 컴포넌트 설정
267            j.setLayout(null);
268
269            paper_size_label.setBounds(20, 50, 76, 15);
270            paper_size_label.setText("용지 크기");
271            j.add(paper_size_label);
272
273            paper_size_cb.setBounds(120, 50, 80, 20);
274            paper_size_cb.setModel(new
DefaultComboBoxModel<PageOption.PageType>(PageOption.PageType.values()));
275            paper_size_cb.setSelectedItem(option.getPageOption().getPageType());
276            j.add(paper_size_cb);
277
278            paper_upblank_label.setBounds(20, 80, 76, 15);
279            paper_upblank_label.setText("위쪽 여백");
280            j.add(paper_upblank_label);
281
282            paper_upblank_tf.setBounds(120, 75, 30, 21);
```

```
283             paper_upblank_tf.setDocument(new JExtendTextField("0-9", 2));
284             j.add(paper_upblank_tf);
285             paper_upblank_tf.setText(String.valueOf(option.getMarginTop()));
286 
287             paper_downblank_label.setBounds(20, 110, 76, 15);
288             j.add(paper_downblank_label);
289             paper_downblank_label.setText("아래쪽 여백");
290             paper_downblank_tf.setDocument(new JExtendTextField("0-9", 2));
291             paper_downblank_tf.setText(String.valueOf(option.getMarginBottom()));
292             paper_downblank_tf.setBounds(120, 105, 30, 21);
293             j.add(paper_downblank_tf);
294 
295             paper_leftblank_label.setBounds(20, 140, 76, 15);
296             paper_leftblank_label.setText("왼쪽 여백");
297             j.add(paper_leftblank_label);
298 
299             paper_leftblank_tf.setBounds(120, 135, 30, 21);
300             paper_leftblank_tf.setDocument(new JExtendTextField("0-9", 2));
301             j.add(paper_leftblank_tf);
302             paper_leftblank_tf.setText(String.valueOf(option.getMarginLeft()));
303 
304             paper_rightblank_label.setBounds(20, 170, 76, 15);
305             paper_rightblank_label.setText("오른쪽 여백");
306             j.add(paper_rightblank_label);
307 
308             paper_rightblank_tf.setDocument(new JExtendTextField("0-9", 2));
309             paper_rightblank_tf.setText(String.valueOf(option.getMarginRight()));
310             paper_rightblank_tf.setBounds(120, 165, 30, 21);
311             j.add(paper_rightblank_tf);
312         }
313 
314         // GeneralTab의 컴포넌트 선언
315         /**
316          * 일반 문자에 지정할 폰트 리스트
317          */
318         private JList<String> general_font_list = new JList<String>();
319 
320         /**
321          * 일반 문자에 지정할 폰트 크기 리스트
322          */
323         private JList<Integer> general_size_list = new JList<Integer>();
324 
325         /**
326          * 일반 문자에 지정할 컬러 정보 지정 라벨
327          */
328         private JLabel general_color_label = new JLabel();
329 
330         /**
```

```
331         * 일반 문자에 지정할 컬러 정보 입력 필드
332         */
333        private JTextField general_color_tf = new JTextField();
334
335        /**
336         * 폰트 리스트 스크롤 패널
337         */
338        private JScrollPane general_font_scl = new JScrollPane();
339
340        /**
341         * 폰트 크기 리스트 스크롤 패널
342         */
343        private JScrollPane general_size_scl = new JScrollPane();
344
345        /**
346         * 일반 문자에 지정할 폰트 스타일
347         */
348        private JComboBox<FontOption.FontStyle> general_style_cb = new
JComboBox<FontOption.FontStyle>(FontOption.FontStyle.values());
349
350        /**
351         * 일반 문자 폰트 설정 탭 초기화 메소드
352         * @param j 삽입 되어질 부모 패널
353         */
354        private void setGeneralTab(JPanel j) {
355            // GeneralTab의 컴포넌트 설정
356            j.setLayout(null);
357
358            //위치 설정 부분
359            general_font_scl.setBounds(30, 50, 150, 190);
360            general_size_scl.setBounds(210, 50, 50, 190);
361            general_color_label.setBounds(290, 50, 40, 21);
362            general_color_tf.setBounds(330, 50, 76, 21);
363            general_style_cb.setBounds(290, 100, 100, 21);
364
365            general_style_cb.setSelectedItem(option.getBasicFont().getFontStyle());
366            j.add(general_style_cb);
367
368            //폰트 설정 부분
369            j.add(general_font_scl);
370            general_font_scl.setViewportView(general_font_list);
371
372            general_font_list.setListData(fontName);
373            general_font_list.setSelectedValue(general_FontFamily, true);
374
375            //폰트 크기 설정 부분
376            j.add(general_size_scl);
377
```

```
378         Integer[] sizeArray = new Integer[99];
379         for (int i = 1; i < 100; i++)
380             sizeArray[i - 1] = i;
381         general_size_scl.setViewportView(general_size_list);
382         general_size_list.setListData(sizeArray);
383         general_size_list.setSelectedValue(option.getBasicFont().getFontSize(), true);
384 
385 
386         general_color_label.setText("색상  #");
387         general_color_tf.setDocument(new JExtendTextField("0-9a-fA-F", 6));
388         general_color_tf.setText(fullColorName(option.getBasicFont().getFontColor()));
389         j.add(general_color_tf);
390         j.add(general_color_label);
391 
392         general_font_scl.setViewportView(general_font_list);
393     }
394 
395     // stringTab의 컴포넌트 선언
396     /**
397      * 문자열 문자에 지정할 폰트 리스트
398      */
399     private JList<String> string_font_list = new JList<String>();
400 
401     /**
402      * 문자열 문자에 지정할 폰트 크기 리스트
403      */
404     private JList<Integer> string_size_list = new JList<Integer>();
405 
406     /**
407      * 문자열 문자에 지정할 컬러 정보 지정 라벨
408      */
409     private JLabel string_color_label = new JLabel();
410 
411     /**
412      * 문자열 문자에 지정할 컬러 정보 입력 필드
413      */
414     private JTextField string_color_tf = new JTextField();
415 
416     /**
417      * 폰트 리스트 스크롤 패널
418      */
419     private JScrollPane string_font_scl = new JScrollPane();
420 
421     /**
422      * 폰트 크기 리스트 스크롤 패널
423      */
424     private JScrollPane string_size_scl = new JScrollPane();
425 
```



```
426     /**
427      * 문자열 문자에 지정할 폰트 스타일
428      */
429     private JComboBox<FontOption.FontStyle> string_style_cb = new
JComboBox<FontOption.FontStyle>(FontOption.FontStyle.values());
430
431     /**
432      * 문자열 문자 폰트 설정 탭 초기화 메소드
433      * @param j 삽입 되어질 부모 패널
434      */
435     private void setStringTab(JPanel j) {
436         // stringTab의 컴포넌트 설정
437         j.setLayout(null);
438         string_font_scl.setBounds(30, 50, 150, 190);
439         string_size_scl.setBounds(210, 50, 50, 190);
440         string_color_label.setBounds(290, 50, 40, 21);
441         string_color_tf.setBounds(330, 50, 76, 21);
442         string_style_cb.setBounds(290, 100, 100, 21);
443
444         string_style_cb.setSelectedItem(option.getStringFont().getFontStyle());
445         j.add(string_style_cb);
446
447         //폰트 설정 부분
448         j.add(string_font_scl);
449         string_font_scl.setViewportView(string_font_list);
450
451         string_font_list.setListData(fontName);
452         string_font_list.setSelectedValue(string_FontFamily, true);
453
454         //폰트 크기 설정 부분
455         j.add(string_size_scl);
456
457         Integer[] sizeArray = new Integer[99];
458         for (int i = 1; i < 100; i++)
459             sizeArray[i - 1] = i;
460         string_size_scl.setViewportView(string_size_list);
461         string_size_list.setListData(sizeArray);
462         string_size_list.setSelectedValue(option.getStringFont().getFontSize(), true);
463
464
465         string_color_label.setText("색상  #");
466         string_color_tf.setDocument(new JExtendTextField("0-9a-fA-F", 6));
467         string_color_tf.setText(fullColorName(option.getStringFont().getFontColor()));
468         j.add(string_color_tf);
469         j.add(string_color_label);
470
471         string_font_scl.setViewportView(string_font_list);
472     }
```

```
473
474         // keywordTab의 컴포넌트 선언
475         /**
476          * 키워드 문자에 지정할 폰트 리스트
477          */
478         private JList<String> keyword_font_list = new JList<String>();
479
480         /**
481          * 키워드 문자에 지정할 폰트 크기 리스트
482          */
483         private JList<Integer> keyword_size_list = new JList<Integer>();
484
485         /**
486          * 키워드 문자에 지정할 컬러 정보 지정 라벨
487          */
488         private JLabel keyword_color_label = new JLabel();
489
490         /**
491          * 키워드 문자에 지정할 컬러 정보 입력 필드
492          */
493         private JTextField keyword_color_tf = new JTextField();
494
495         /**
496          * 폰트 리스트 스크롤 패널
497          */
498         private JScrollPane keyword_font_scl = new JScrollPane();
499
500         /**
501          * 폰트 크기 리스트 스크롤 패널
502          */
503         private JScrollPane keyword_size_scl = new JScrollPane();
504
505         /**
506          * 키워드 문자에 지정할 폰트 스타일
507          */
508         private JComboBox<FontOption.FontStyle> keyword_style_cb = new
JComboBox<FontOption.FontStyle>(FontOption.FontStyle.values());
509
510         /**
511          * 키워드 문자 폰트 설정 탭 초기화 메소드
512          * @param j 삽입 되어질 부모 패널
513          */
514         private void setKeywordTab(JPanel j) {
515             // ketwordTab의 컴포넌트 설정
516             j.setLayout(null);
517
518             keyword_font_scl.setBounds(30, 50, 150, 190);
519             keyword_size_scl.setBounds(210, 50, 50, 190);
```



```
520         keyword_color_label.setBounds(290, 50, 40, 21);
521         keyword_color_tf.setBounds(330, 50, 76, 21);
522         keyword_style_cb.setBounds(290, 100, 100, 21);
523 
524         keyword_style_cb.setSelectedItem(option.getKeywordFont().getFontStyle());
525         j.add(keyword_style_cb);
526 
527         // 폰트 설정 부분
528         j.add(keyword_font_scl);
529         keyword_font_scl.setViewportView(keyword_font_list);
530 
531         keyword_font_list.setListData(fontName);
532         keyword_font_list.setSelectedValue(keyword_FontFamily, true);
533 
534         // 폰트 크기 설정 부분
535         j.add(keyword_size_scl);
536 
537         Integer[] sizeArray = new Integer[99];
538         for (int i = 1; i < 100; i++)
539             sizeArray[i - 1] = i;
540         keyword_size_scl.setViewportView(keyword_size_list);
541         keyword_size_list.setListData(sizeArray);
542         keyword_size_list.setSelectedValue(option.getKeywordFont().getFontSize(), true);
543 
544         keyword_color_label.setText("색상  #");
545         keyword_color_tf.setDocument(new JExtendTextField("0-9a-fA-F", 6));
546         keyword_color_tf.setText(fullColorName(option.getKeywordFont().getFontColor()));
547         j.add(keyword_color_tf);
548         j.add(keyword_color_label);
549 
550         keyword_font_scl.setViewportView(keyword_font_list);
551     }
552 
553     // commentTab의 컴포넌트 선언
554     /**
555      * 주석 문자에 지정할 폰트 리스트
556      */
557     private JList<String> comment_font_list = new JList<String>();
558 
559     /**
560      * 주석 문자에 지정할 폰트 크기 리스트
561      */
562     private JList<Integer> comment_size_list = new JList<Integer>();
563 
564     /**
565      * 주석 문자에 지정할 컬러 정보 지정 라벨
566      */
567     private JLabel comment_color_label = new JLabel();
```



```
568
569         /**
570          * 주석 문자에 지정할 컬러 정보 입력 필드
571          */
572         private JTextField comment_color_tf = new JTextField();
573
574         /**
575          * 폰트 리스트 스크롤 패널
576          */
577         private JScrollPane comment_font_scl = new JScrollPane();
578
579         /**
580          * 폰트 크기 리스트 스크롤 패널
581          */
582         private JScrollPane comment_size_scl = new JScrollPane();
583
584         /**
585          * 주석 문자에 지정할 폰트 스타일
586          */
587         private JComboBox<FontOption.FontStyle> comment_style_cb = new
JComboBox<FontOption.FontStyle>(FontOption.FontStyle.values());
588
589         /**
590          * 주석 문자 폰트 설정 탭 초기화 메소드
591          * @param j 삽입 되어질 부모 패널
592          */
593         private void setCommentTab(JPanel j) {
594
595             // commentTab의 컴포넌트 설정
596             j.setLayout(null);
597             comment_font_scl.setBounds(30, 50, 150, 190);
598             comment_size_scl.setBounds(210, 50, 50, 190);
599             comment_color_label.setBounds(290, 50, 40, 21);
600             comment_color_tf.setBounds(330, 50, 76, 21);
601             comment_style_cb.setBounds(290, 100, 100, 21);
602
603             comment_style_cb.setSelectedItem(option.getCommentFont().getFontStyle());
604             j.add(comment_style_cb);
605
606             //폰트 설정 부분
607             j.add(comment_font_scl);
608             comment_font_scl.setViewportView(comment_font_list);
609
610             comment_font_list.setListData(fontName);
611             comment_font_list.setSelectedValue(comment_FontFamily, true);
612
613             //폰트 크기 설정 부분
614             j.add(comment_size_scl);
```



```
615
616         Integer[] sizeArray = new Integer[99];
617         for (int i = 1; i < 100; i++)
618             sizeArray[i - 1] = i;
619         comment_size_scl.setViewportView(comment_size_list);
620         comment_size_list.setListData(sizeArray);
621         comment_size_list.setSelectedValue(option.getCommentFont().getFontSize(), true);
622
623
624         comment_color_label.setText("색상  #");
625         comment_color_tf.setDocument(new JExtendTextField("0-9a-fA-F", 6));
626         comment_color_tf.setText(fullColorName(option.getCommentFont().getFontColor()));
627         j.add(comment_color_tf);
628         j.add(comment_color_label);
629
630         comment_font_scl.setViewportView(comment_font_list);
631     }
632
633     // linenumberTab의 컴포넌트 선언
634     /**
635      * 줄번호 문자에 지정할 폰트 리스트
636      */
637     private JList<String> linenumber_font_list = new JList<String>();
638
639     /**
640      * 줄번호 문자에 지정할 폰트 크기 리스트
641      */
642     private JList<Integer> linenumber_size_list = new JList<Integer>();
643
644     /**
645      * 줄번호 문자에 지정할 컬러 정보 지정 라벨
646      */
647     private JLabel linenumber_color_label = new JLabel();
648
649     /**
650      * 줄번호 문자에 지정할 컬러 정보 입력 필드
651      */
652     private JTextField linenumber_color_tf = new JTextField();
653
654     /**
655      * 폰트 리스트 스크롤 패널
656      */
657     private JScrollPane linenumber_font_scl = new JScrollPane();
658
659     /**
660      * 폰트 크기 리스트 스크롤 패널
661      */
662     private JScrollPane linenumber_size_scl = new JScrollPane();
```



```
663
664         /**
665          * 줄번호 문자에 지정할 폰트 스타일
666          */
667         private JComboBox<FontOption.FontStyle> linenumber_style_cb = new
JComboBox<FontOption.FontStyle>(FontOption.FontStyle.values());
668
669         /**
670          * 줄번호 문자 폰트 설정 탭 초기화 메소드
671          * @param j 삽입 되어질 부모 패널
672          */
673         private void setLinenumberTab(JPanel j) {
674             // linenumverTab의 컴포넌트 설정
675             j.setLayout(null);
676             linenumber_font_scl.setBounds(30, 50, 150, 190);
677             linenumber_size_scl.setBounds(210, 50, 50, 190);
678             linenumber_color_label.setBounds(290, 50, 40, 21);
679             linenumber_color_tf.setBounds(330, 50, 76, 21);
680             linenumber_style_cb.setBounds(290, 100, 100, 21);
681
682             linenumber_style_cb.setSelectedItem(option.getLineNumberFont().getFontStyle());
683             j.add(linenumber_style_cb);
684
685             //폰트 설정 부분
686             j.add(linenumber_font_scl);
687             linenumber_font_scl.setViewportView(linenumber_font_list);
688             linenumber_font_list.setListData(fontName);
689             linenumber_font_list.setSelectedValue(linenumber_FontFamily, true);
690
691             //폰트 크기 설정 부분
692             j.add(linenumber_size_scl);
693
694             Integer[] sizeArray = new Integer[99];
695             for (int i = 1; i < 100; i++)
696                 sizeArray[i - 1] = i;
697             linenumber_size_scl.setViewportView(linenumber_size_list);
698             linenumber_size_list.setListData(sizeArray);
699             linenumber_size_list.setSelectedValue(option.getLineNumberFont().getFontSize(), true);
700
701
702             linenumber_color_label.setText("색상  #");
703             linenumber_color_tf.setDocument(new JExtendTextField("0-9a-fA-F", 6));
704             linenumber_color_tf.setText(fullColorName(option.getLineNumberFont().getFontColor()));
705             j.add(linenumber_color_tf);
706             j.add(linenumber_color_label);
707
708             linenumber_font_scl.setViewportView(linenumber_font_list);
709         }
```



```
710
711         /**
712          * 닫기 버튼 이벤트 핸들러
713          */
714         private void closeBtnEvent() {
715             // 닫기버튼 이벤트: 현재 창을 닫음
716             dispose();
717         }
718
719         /**
720          * 저장 버튼 이벤트 핸들러
721          */
722         private void saveBtnEvent() {
723             if(paper_upblank_tf.getText().equals("")){
724                 paper_upblank_tf.setText("0");
725             }
726             if(paper_downblank_tf.getText().equals("")){
727                 paper_downblank_tf.setText("0");
728             }
729             if(paper_leftblank_tf.getText().equals("")){
730                 paper_leftblank_tf.setText("0");
731             }
732             if(paper_rightblank_tf.getText().equals("")){
733                 paper_rightblank_tf.setText("0");
734             }
735             if(general_color_tf.getText().equals("")){
736                 general_color_tf.setText("0");
737             }
738             if(string_color_tf.getText().equals("")){
739                 string_color_tf.setText("0");
740             }
741             if(keyword_color_tf.getText().equals("")){
742                 keyword_color_tf.setText("0");
743             }
744             if(comment_color_tf.getText().equals("")){
745                 comment_color_tf.setText("0");
746             }
747             if(linenumber_color_tf.getText().equals("")){
748                 linenumber_color_tf.setText("0");
749             }
750
751             // 적용버튼 이벤트:현재 설정값들을 저장함
752             option.setMargin(Integer.valueOf(paper_upblank_tf.getText()),
Integer.valueOf(paper_downblank_tf.getText()), Integer.valueOf(paper_leftblank_tf.getText()),
Integer.valueOf(paper_rightblank_tf.getText()));
753             option.setPageType((PageOption.PageType) paper_size_cb.getSelectedItem());
754             option.setBasicFont(fontData.get(general_font_list.getSelectedValue()),
general_size_list.getSelectedValue(), Integer.valueOf(general_color_tf.getText(), 16), (FontOption.FontStyle)
```



```
general_style_cb.getSelectedItem());
755             option.setStringFont(fontData.get(string_font_list.getSelectedValue()),
string_size_list.getSelectedValue(), Integer.valueOf(string_color_tf.getText(), 16), (FontOption.FontStyle)
string_style_cb.getSelectedItem());
756             option.setKeywordFont(fontData.get(keyword_font_list.getSelectedValue()),
keyword_size_list.getSelectedValue(), Integer.valueOf(keyword_color_tf.getText(), 16), (FontOption.FontStyle)
keyword_style_cb.getSelectedItem());
757             option.setCommentFont(fontData.get(comment_font_list.getSelectedValue()),
comment_size_list.getSelectedValue(), Integer.valueOf(comment_color_tf.getText(), 16), (FontOption.FontStyle)
comment_style_cb.getSelectedItem());
758             option.setLineNumberFont(fontData.get(linenumber_font_list.getSelectedValue()),
linenumber_size_list.getSelectedValue(), Integer.valueOf(linenumber_color_tf.getText(), 16), (FontOption.FontStyle)
linenumber_style_cb.getSelectedItem());
759
760             option.saveOption();
761             dispose();
762
763         }
764
765         /**
766          * 정수 컬러값을 문자열로 바꾸는 메소드
767          * @param color 정수로 된 컬러값
768          * @return 16진수로 형태의 문자열로 변환된 컬러 값
769          */
770         private String fullColorName(int color){
771             if(color < 16){
772                 return "00000" + Integer.toHexString(color).toUpperCase();
773             }else if(color < 256){
774                 return "0000" + Integer.toHexString(color).toUpperCase();
775             }else if(color < 4096){
776                 return "000" + Integer.toHexString(color).toUpperCase();
777             }else if(color < 65536){
778                 return "00" + Integer.toHexString(color).toUpperCase();
779             }else if(color < 1048576){
780                 return "0" + Integer.toHexString(color).toUpperCase();
781             }else{
782                 return Integer.toHexString(color).toUpperCase();
783             }
784         }
785     }
786
```

Checker.java

```
1  package loader;
2  import java.io.File;
3  import java.io.IOException;
4  import java.util.Vector;
5
6  /**
7   * 소스코드 검사 클래스
8   * @author 유병호
9   *
10   */
11  public class Checker {
12
13      /**
14       * 변환할 디렉토리(패키지) 경로 리스트
15       */
16      public Vector<String> dirListString = new Vector<String>(); // 변환할 패키지 목록 저장 (스트링 벡터)
17
18      /**
19       * 변환할 파일 경로 리스트
20       */
21      public Vector<String> fileListString = new Vector<String>(); // 변환할 파일 목록 저장 (스트링 벡터)
22
23      /**
24       * 변환되지 않는 필터링 된 파일 경로 리스트
25       */
26      public Vector<String> failListString = new Vector<String>(); // 필터링 될 파일 목록 (스트링 벡터)
27
28      /**
29       * 변환할 디렉토리(패키지) 리스트
30       */
31      public Vector<File> dirList = new Vector<File>(); // 변환할 패키지 목록 저장 (파일 벡터)
32
33      /**
34       * 변환할 파일 리스트 [각 패키지 별 [파일 리스트]] 형태
35       */
36      public Vector<Vector<File>> fileList = new Vector<Vector<File>>(); // 변환할 파일 목록 저장 (파일 벡
터)
37
38      /**
39       * 변환되지 않는 필터링 된 파일 리스트
40       */
41      public Vector<File> failList = new Vector<File>(); // 필터링 될 파일 목록 (파일 벡터)
42
43      /**
44       * 사용자가 선택한 경로
45       */
46      public String rootPath;
```

```
47
48      /**
49       * 필터링 용 임시 경로
50       */
51      private String source = "";
52
53      /**
54       * 파일 필터링 메소드
55       * @param source 필터링 할 파일 경로
56       * @param ext 필터링 확장자 리스트
57       */
58      public void explorerFile(String source, String[] ext){
59          clearAllList();
60          boolean find = false;
61          for(int i = 0 ; i < ext.length ; i++){
62              find = source.endsWith(ext[i]);
63              if(find) break;
64          }
65
66          if(find){
67              Vector<File> newVector = new Vector<File>();
68              File f = null;
69              try{
70                  f = new File(source);
71              }catch(NullPointerException e){
72                  return;
73              }
74              newVector.addElement(f);
75              this.fileList.addElement(newVector);
76              this.fileListString.addElement(stringReplace(f));
77          }else{
78              File f = new File(source);
79              this.failList.addElement(f);
80              this.failListString.addElement(stringReplace(f));
81          }
82      }
83
84      /**
85       * 디렉토리 필터링 메소드
86       * @param source 필터링 할 디렉토리 경로
87       * @param ext 필터링 확장자 리스트
88       * @throws IOException 파일 읽기에 문제가 발생한 경우 발생합니다.
89       */
90      public void explorerDirectory(String source, String[] ext) throws IOException { // 지정된 경로부터 탐색
하는 메소드
91          File start = null;
92          try{
93              start = new File(source);
```



```
94          }catch(NullPointerException e){
95              throw new IOException();
96          }
97
98          File[] file_list = null;
99          try{
100             file_list = start.listFiles();
101          }catch(SecurityException e){
102             throw new IOException();
103          }
104          Vector<File> fileVector = new Vector<File>();
105          try {
106             if(file_list == null){
107                 throw new IOException();
108             }
109             for (int i = 0; i < file_list.length; i++) { // 시작 폴더부터 모든 하위 폴더 까지 검사
110                 File file_list_tmp = null;
111                 try{
112                     file_list_tmp = file_list[i]; // file_list_tmp에 하나씩 파일을 받음
113                 }catch(NullPointerException e){
114                     continue;
115                 }
116
117                 String file_name = file_list[i].toString().toLowerCase(); // file_name에 파일 이름을 string으로 저
장
118                 boolean check_type = false;
119                 for(int typeChk = 0 ; typeChk < ext.length ; typeChk++){
120                     check_type = file_name.endsWith(ext[typeChk]);
121                     if(check_type) break;
122                 }
123                 if (file_list_tmp.isFile() && !check_type) { // 파일이지만 타입이 일치하지 않을 경우
124                     this.failListString.addElement(stringReplace(file_list_tmp)); // 해당하는 파일을 필터링 파일
목록에 저장
125                     this.failList.addElement(file_list_tmp); // 해당하는 파일을 필터링 파일 목록에 저장 (파일 벡
터)
126                 }
127                 if (file_list_tmp.isFile() && check_type) { // 파일이면서 타입이 일치하는 경우
128                     this.fileListString.addElement(stringReplace(file_list_tmp)); // 해당하는 파일을 변환할 파일
목록에 저장
129                     fileVector.addElement(file_list_tmp); // 해당하는 파일을 변환할 파일 목록에 저장 (파일 벡터
)
130                     String dir = source.replaceAll("₩₩₩₩", "/"); // 문자열 처리
131                     if (!dirListString.contains(dir)) { // 같은 이름의 패키지 명이 없을 경우
132                         dirListString.addElement(dir); // 패키지 그룹에 경로 추가
133                         dirList.addElement(start); // 패키지 그룹에 경로 추가 (파일 벡터)
134                     }
135                 } else if (file_list_tmp.isDirectory()) { // 폴더일 경우
136                     explorerDirectory(file_list_tmp.getCanonicalPath().toString(), ext); // 하위 경로로 이동
```

```
137                 }
138             }
139             if(fileVector.size() != 0){
140                 this.fileList.addElement(fileVector);
141             }
142         } catch (IOException e) {
143             return;
144         }
145     }
146 
147     /**
148      * 모든 리스트를 초기화 합니다.
149      */
150     public void clearAllList(){
151         dirList.removeAllElements();
152         fileList.removeAllElements();
153         failList.removeAllElements();
154         dirListString.removeAllElements();
155         fileListString.removeAllElements();
156         failListString.removeAllElements();
157     }
158 
159     /**
160      * ₩ 형태의 파일 경로 구분을 / 형태로 치환
161      * @param file 경로를 치환할 파일
162      * @return 치환된 경로
163      */
164     private String stringReplace(File file) { // 문자열 처리
165         String rep = this.source.replaceAll("₩₩₩₩", "/");
166         String path = this.source + file.getPath().replaceAll("₩₩₩₩", "/").replaceAll(rep, "");
167         return path;
168     }
169 
170 }
171 
```

FontOption.java

```
1  package option;
2  import java.io.IOException;
3
4  import com.itextpdf.text.DocumentException;
5  import com.itextpdf.text.Font;
6  import com.itextpdf.text.pdf.BaseFont;
7
8  /**
9   * 폰트에 관련된 옵션 클래스 입니다.
10   * @author 유병호
11   *
12   */
13  public class FontOption {
14
15      /**
16       * 폰트 스타일 데이터입니다.<p>
17       * NORMAL : 아무런 스타일도 정의되어 있지 않은 상태입니다.<br>
18       * BOLD : 굵게 표시됩니다.<br>
19       * ITALIC : 기울임꼴로 표시됩니다.<br>
20       * BOLDITALIC : 굵은 스타일에 기울임꼴이 적용됩니다.<br>
21       * UNDERLINE : 밑줄이 그어집니다. <br>
22       * STRIKETHRE : 취소선이 그어집니다.
23       * @author 유병호
24       *
25       */
26      public enum FontStyle {NORMAL, BOLD, ITALIC, BOLDITALIC, UNDERLINE, STRIKETHRU};
27
28      /**
29       * 폰트의 모양, 표시 방법이 정의되는 개체입니다.
30       */
31      private BaseFont baseFont;
32
33      /**
34       * 폰트의 색, 스타일, 크기등이 정의되는 개체입니다.
35       */
36      private Font font;
37
38      /**
39       * 폰트파일의 경로입니다.
40       */
41      private String fontPath;
42
43      /**
44       * 빈 생성자
45       */
46      public FontOption(){
47
```

```
48      }
49
50      /**
51       * 폰트 옵션을 생성합니다.
52       * @param fontName 폰트 파일이 있는 경로입니다.
53       * @param fontSize 폰트의 크기입니다.
54       * @param fontColor 폰트의 RGB 색상값입니다.
55       * @param style 폰트의 스타일입니다.
56       */
57      public FontOption(String fontName, int fontSize, int fontColor, FontStyle style){
58          setFont(fontName, fontSize, fontColor, style);
59      }
60
61      /**
62       * 폰트 옵션을 지정합니다.
63       * @param fontName 폰트 이름 입니다.
64       * @param fontSize 폰트의 크기입니다.
65       * @param fontColor 폰트의 RRGGBB 형태의 색상값입니다.
66       * @param style 폰트의 스타일입니다.
67       */
68      public void setFont(String fontName, int fontSize, int fontColor, FontStyle style){
69          //baseFont = FontFactory.getFont(fontName, BaseFont.IDENTITY_H,
BaseFont.EMBEDDED).getBaseFont();
70          try {
71              baseFont = BaseFont.createFont(fontName, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
72          } catch (DocumentException | IOException e) {
73              baseFont = OptionDefault.getDefaultOption().getBasicFont().getBaseFont();
74          }
75          fontPath = fontName;
76          if(fontSize < 1 || fontSize >= 100){
77              fontSize = 11;
78          }
79          font = new Font(baseFont, (float)fontSize);
80          // RGB 색상값은 저장이나 관리에 유용하도록 합쳐서 저장하지만 iText 라이브러리의 파라메터는 각각
R, G, B 값이 따로 지정되므로
81          // 비트 연산을 통해 각각의 RGB 값을 분리합니다.
82          if(fontColor < 0 || fontColor > 0xFFFFFF){
83              fontColor = 0;
84          }
85          int r = fontColor >> 16;
86          int g = (fontColor >> 8) & (0x00FF);
87          int b = fontColor & 0x0000FF;
88          font.setColor(r,g,b);
89
90          //폰트 스타일을 라이브러리 값으로 치환합니다.
91          switch (style){
92          case NORMAL:
93              font.setStyle(Font.NORMAL);
```



```
94              break;
95          case BOLD:
96              font.setStyle(Font.BOLD);
97              break;
98          case BOLDITALIC:
99              font.setStyle(Font.BOLDITALIC);
100             break;
101         case ITALIC:
102             font.setStyle(Font.ITALIC);
103             break;
104         case STRIKETHRU:
105             font.setStyle(Font.STRIKETHRU);
106             break;
107         case UNDERLINE:
108             font.setStyle(Font.UNDERLINE);
109             break;
110         default:
111             font.setStyle(Font.NORMAL);
112         }
113     }
114
115     /**
116      * 폰트 개체를 반환합니다.
117      * @return 폰트 개체
118      */
119     public Font getFont(){
120         return font;
121     }
122
123     /**
124      * 폰트 기본 정보가 있는 개체를 반환합니다.
125      * @return 폰트 기본 정보 개체
126      */
127     public BaseFont getBaseFont(){
128         return baseFont;
129     }
130
131     /**
132      * 폰트 파일의 경로를 반환합니다.
133      * @return 폰트 파일 경로
134      */
135     public String getFontName(){
136         return fontPath;
137     }
138
139     /**
140      * 폰트의 크기를 반환합니다.
141      * @return 폰트의 크기
```



```
142         */
143         public int getFontSize(){
144             return (int) font.getSize();
145         }
146 
147         /**
148          * 폰트의 색상값을 RRGGBB 형태로 반환합니다.
149          * @return 폰트 색상
150          */
151         public int getFontColor(){
152             return (~font.getColor().getRGB()) ^ 0xFFFFFF;
153         }
154 
155         /**
156          * 폰트 스타일을 반환합니다.
157          * @return 폰트 스타일
158          */
159         public FontStyle getFontStyle(){
160             // 라이브러리에서 지정되었던 폰트 스타일이 내부적으로 구현된 폰트 스타일로 치환됩니다.
161             int s = font.getStyle();
162             if(s == 0){
163                 return FontStyle.NORMAL;
164             }else if(s == 1){
165                 return FontStyle.BOLD;
166             }else if(s == 2){
167                 return FontStyle.ITALIC;
168             }else if(s == 3){
169                 return FontStyle.BOLDITALIC;
170             }else if(s == 4){
171                 return FontStyle.UNDERLINE;
172             }else if(s == 8){
173                 return FontStyle.STRIKETHRU;
174             }else{
175                 return FontStyle.NORMAL;
176             }
177         }
178     }
179 
```

Option.java

```
1  package option;
2
3  import java.io.File;
4  import java.util.regex.Pattern;
5
6  import com.itextpdf.text.Rectangle;
7
8  /**
9   * 옵션을 저장하는 클래스입니다.
10   * @author 오다솜
11   *
12   */
13  public class Option {
14
15      /**
16       * 위치 정보입니다.<p>
17       * LEFT / CENTER / RIGHT
18       * @author 오다솜
19       *
20       */
21      public enum Align {LEFT, CENTER, RIGHT};
22
23      /**
24       * 언어 종류입니다.
25       * @author 오다솜
26       *
27       */
28      public enum Language {JAVA, C, CPP};
29
30      /**
31       * 지정되는 옵션의 갯수입니다.<br>
32       * 이 값은 옵션 파일의 RW 작업에 사용되며 새로운 옵션의 추가나 기존 옵션의 삭제가 있을 시 값의 변동이 필요합니다.
33       */
34      public static final int optionCount = 38;
35
36      /**
37       * 옵션 파일 인스턴스
38       */
39      private OptionFile of = new OptionFile();
40
41      /**
42       * 일반 문자를 표시하는 폰트의 옵션입니다.<p>
43       * 기본값<br>
44       * 폰트 : 맑은 고딕<br>
45       * 크기 : 11<br>
46       * 색상 : 0x000000<br>
```

```
47      * 스타일 : 없음
48      */
49     private FontOption basicFont = new FontOption();
50
51     /**
52      * 키워드 문자를 표시하는 폰트 옵션입니다.<p>
53      * 기본값<br>
54      * 폰트 : 맑은 고딕<br>
55      * 크기 : 11<br>
56      * 색상 : 0xC13A55<br>
57      * 스타일 : 없음
58      */
59     private FontOption keywordFont = new FontOption();
60
61     /**
62      * ' ' 혹은 " " 형태로 감싸진 문자열 내부를 표시하는 폰트 옵션입니다.<p>
63      * 기본값<br>
64      * 폰트 : 맑은 고딕<br>
65      * 크기 : 11<br>
66      * 색상 : 0x0000FF<br>
67      * 스타일 : 없음
68      */
69     private FontOption stringFont = new FontOption();
70
71     /**
72      * 주석 형태의 문자열을 표시하는 폰트 옵션입니다.<p>
73      * 기본값<br>
74      * 폰트 : 맑은 고딕<br>
75      * 크기 : 11<br>
76      * 색상 : 0x3F7F5F<br>
77      * 스타일 : 없음
78      */
79     private FontOption commentFont = new FontOption();
80
81     /**
82      * 라인 넘버를 표시하는 폰트 옵션입니다.<p>
83      * 기본값<br>
84      * 폰트 : 맑은 고딕<br>
85      * 크기 : 11<br>
86      * 색상 : 0xCCCCCC<br>
87      * 스타일 : 없음
88      */
89     private FontOption lineNumberFont = new FontOption();
90
91     /**
92      * 탭의 길이를 나타내는 옵션입니다.<p>
93      * 기본값 : 4
94      */
```

```
95      private int tabSize;
96
97      /**
98       * 생성되는 PDF 파일의 헤더에 들어갈 문구를 저장하는 옵션입니다.<p>
99       * 기본값 : 빈 문자열
100       */
101      private String headerInfo;
102
103      /**
104       * 저작자 정보를 저장하는 옵션입니다.<p>
105       * 기본값 : 빈 문자열
106       */
107      private String copyrightInfo;
108
109      /**
110       * PDF 문서의 크기, 여백등을 지정하는 옵션입니다.<p>
111       * 기본값<br>
112       * 문서 용지 : A4<br>
113       * 여백(위, 아래, 왼쪽, 오른쪽) : 30, 30, 40, 40
114       */
115      private PageOption pageOption = new PageOption();
116
117      /**
118       * 라인넘버를 표시할지 여부를 지정하는 옵션입니다.<p>
119       * 기본값 : true
120       */
121      private boolean showLineNumber;
122
123      /**
124       * 페이지 번호를 표시할지 여부를 지정하는 옵션입니다.<p>
125       * 기본값 : true
126       */
127      private boolean showPageNumber;
128
129      /**
130       * 소스코드의 언어 종류입니다.
131       */
132      private Language codeLanguage;
133
134      /**
135       * 헤더 문구 표시 위치입니댜.
136       */
137      private Align headerAlign;
138
139      /**
140       * 저작권 문구 표시 위치입니다.
141       */
142      private Align copyrightAlign;
```



```
143
144     /**
145      * 소스코드 변환 시 모든 소스파일을 한개의 문서로 출력할것인지에 대한 옵션입니다.
146      */
147     private boolean convertAll;
148
149     /**
150      * 소스코드 변환 시 각 패키지 당 한개의 문서로 출력할것인지에 대한 옵션입니다.
151      */
152     private boolean convertPackage;
153
154     /**
155      * 소스코드 변환 시 각 파일당 한개의 문서로 출력할 것인지에 대한 옵션입니다.
156      */
157     private boolean convertFile;
158
159     /**
160      * 종합 변환 시 지정되는 파일 명 입니다.
161      */
162     private String file_name;
163
164     /**
165      * 변환되는 PDF 문서가 저장되는 위치입니다.
166      */
167     private String save_locate;
168
169     /**
170      * 옵션을 파일로 저장합니다.
171      */
172     public void saveOption(){
173         of.writeOptionFile(this);
174     }
175
176     /**
177      * 일반 문자열을 표시하는 폰트 옵션을 지정합니다.
178      * @param fontName 폰트 파일의 경로
179      * @param fontSize 폰트의 크기
180      * @param fontColor 폰트의 색상
181      * @param style 폰트의 스타일
182      */
183     public void setBasicFont(String fontName, int fontSize, int fontColor, FontOption.FontStyle style){
184         basicFont.setFont(fontName, fontSize, fontColor, style);
185     }
186
187     /**
188      * 일반 문자열을 표시하는 폰트 옵션을 반환합니다.
189      * @return 일반 문자열 폰트
190      */
```



```
191     public FontOption getBasicFont(){
192         return basicFont;
193     }
194 
195     /**
196      * 키워드 문자를 표시하는 폰트 옵션을 지정합니다.
197      * @param fontName 폰트 파일의 경로
198      * @param fontSize 폰트의 크기
199      * @param fontColor 폰트의 색상
200      * @param style 폰트의 스타일
201      */
202     public void setKeywordFont(String fontName, int fontSize, int fontColor, FontOption.FontStyle style) {
203         keywordFont.setFont(fontName, fontSize, fontColor, style);
204     }
205     /**
206      * 키워드 문자를 표시하는 폰트 옵션을 반환합니다.
207      * @return 키워드 문자열 폰트
208      */
209     public FontOption getKeywordFont(){
210         return keywordFont;
211     }
212 
213     /**
214      * ' ' 혹은 " " 형태로 감싸진 문자열 내부를 표시하는 폰트 옵션을 지정합니다.
215      * @param fontName 폰트 파일의 경로
216      * @param fontSize 폰트의 크기
217      * @param fontColor 폰트의 색상
218      * @param style 폰트의 스타일
219      */
220     public void setStringFont(String fontName, int fontSize, int fontColor, FontOption.FontStyle style) {
221         stringFont.setFont(fontName, fontSize, fontColor, style);
222     }
223     /**
224      * ' ' 혹은 " " 형태로 감싸진 문자열 내부를 표시하는 폰트 옵션을 반환합니다.
225      * @return 문자열 관련 문자열 폰트
226      */
227     public FontOption getStringFont(){
228         return stringFont;
229     }
230 
231     /**
232      * 주석을 표시하는 폰트 옵션을 지정합니다.
233      * @param fontName 폰트 파일의 경로
234      * @param fontSize 폰트의 크기
235      * @param fontColor 폰트의 색상
236      * @param style 폰트의 스타일
237      */
238     public void setCommentFont(String fontName, int fontSize, int fontColor, FontOption.FontStyle style) {
```



```
239             commentFont.setFont(fontName, fontSize, fontColor, style);
240         }
241 
242         /**
243          * 주석을 표시하는 폰트 옵션을 반환합니다.
244          * @return 주석 문자열 폰트
245          */
246         public FontOption getCommentFont(){
247             return commentFont;
248         }
249 
250         /**
251          * 라인 넘버를 표시하는 폰트 옵션을 지정합니다.
252          * @param fontName 폰트 파일의 경로
253          * @param fontSize 폰트의 크기
254          * @param fontColor 폰트의 색상
255          * @param style 폰트의 스타일
256          */
257         public void setLineNumberFont(String fontName, int fontSize, int fontColor, FontOption.FontStyle style)
{
258             lineNumberFont.setFont(fontName, fontSize, fontColor, style);
259         }
260 
261         /**
262          * 라인 넘버를 표시하는 폰트 옵션을 반환합니다.
263          * @return 줄번호 문자열 폰트
264          */
265         public FontOption getLineNumberFont(){
266             return lineNumberFont;
267         }
268 
269         /**
270          * 탭 크기를 지정합니다.
271          * @param tabSize 탭 크기
272          */
273         public void setTabSize(int tabSize){
274             if(tabSize < 1 || tabSize > 6){
275                 tabSize = 4;
276             }
277             this.tabSize = tabSize;
278         }
279         /**
280          * 지정된 탭 크기를 반환합니다.
281          * @return 탭 크기
282          */
283         public int getTabSize(){
284             return tabSize;
285         }
```



```
286
287     /**
288      * 용지 크기를 설정합니다.
289      * @param type 용지의 종류
290      */
291     public void setPageType(PageOption.PageType type){
292         pageOption.setPageType(type);
293     }
294
295     /**
296      * 페이지 옵션을 반환합니다.
297      * @return 페이지 옵션
298      */
299     public PageOption getPageOption(){
300         return pageOption;
301     }
302     /**
303      * 용지 크기를 반환합니다.
304      * @return 용지 크기
305      */
306     public Rectangle getPageType(){
307         if(pageOption.getPageType() == PageOption.PageType.A1){
308             return com.itextpdf.text.PageSize.A1;
309         }else if(pageOption.getPageType() == PageOption.PageType.A2){
310             return com.itextpdf.text.PageSize.A2;
311         }else if(pageOption.getPageType() == PageOption.PageType.A3){
312             return com.itextpdf.text.PageSize.A3;
313         }else if(pageOption.getPageType() == PageOption.PageType.A4){
314             return com.itextpdf.text.PageSize.A4;
315         }else if(pageOption.getPageType() == PageOption.PageType.A5){
316             return com.itextpdf.text.PageSize.A5;
317         }else if(pageOption.getPageType() == PageOption.PageType.A6){
318             return com.itextpdf.text.PageSize.A6;
319         }else if(pageOption.getPageType() == PageOption.PageType.B1){
320             return com.itextpdf.text.PageSize.B1;
321         }else if(pageOption.getPageType() == PageOption.PageType.B2){
322             return com.itextpdf.text.PageSize.B2;
323         }else if(pageOption.getPageType() == PageOption.PageType.B3){
324             return com.itextpdf.text.PageSize.B3;
325         }else if(pageOption.getPageType() == PageOption.PageType.B4){
326             return com.itextpdf.text.PageSize.B4;
327         }else if(pageOption.getPageType() == PageOption.PageType.B5){
328             return com.itextpdf.text.PageSize.B5;
329         }else if(pageOption.getPageType() == PageOption.PageType.B6){
330             return com.itextpdf.text.PageSize.B6;
331         }else{
332             return com.itextpdf.text.PageSize.A4;
333         }
```



```
334     }
335
336     /**
337      * 용지 여백을 지정합니다.
338      * @param top 상단 여백
339      * @param bottom 하단 여백
340      * @param left 좌측 여백
341      * @param right 우측 여백
342      */
343     public void setMargin(int top, int bottom, int left, int right){
344         pageOption.setMarginTop(top);
345         pageOption.setMarginBottom(bottom);
346         pageOption.setMarginLeft(left);
347         pageOption.setMarginRight(right);
348     }
349
350     /**
351      * 상단 여백을 지정합니다.
352      * @param margin 상단 여백
353      */
354     public void setMarginTop(int margin){
355         pageOption.setMarginTop(margin);
356     }
357     /**
358      * 상단 여백을 반환합니다.
359      * @return 상단 여백
360      */
361     public int getMarginTop(){
362         return pageOption.getMarginTop();
363     }
364
365     /**
366      * 하단 여백을 지정합니다.
367      * @param margin 하단 여백
368      */
369     public void setMarginBottom(int margin) {
370         pageOption.setMarginBottom(margin);
371     }
372     /**
373      * 하단 여백을 반환합니다.
374      * @return 하단 여백
375      */
376     public int getMarginBottom(){
377         return pageOption.getMarginBottom();
378     }
379
380     /**
381      * 좌측 여백을 지정합니다.
```



```
382         * @param margin 좌측 여백
383         */
384        public void setMarginLeft(int margin) {
385            pageOption.setMarginLeft(margin);
386        }
387        /**
388         * 좌측 여백을 반환합니다.
389         * @return 좌측 여백
390         */
391        public int getMarginLeft(){
392            return pageOption.getMarginLeft();
393        }
394
395        /**
396         * 우측 여백을 지정합니다.
397         * @param margin 우측 여백
398         */
399        public void setMarginRight(int margin) {
400            pageOption.setMarginRight(margin);
401        }
402        /**
403         * 우측 여백을 반환합니다.
404         * @return 우측 여백
405         */
406        public int getMarginRight(){
407            return pageOption.getMarginRight();
408        }
409
410        /**
411         * 문서 헤더에 삽입될 문구를 지정합니다.
412         * @param header 헤더 문구
413         */
414        public void setHeaderTitle(String header){
415            headerInfo = header;
416        }
417        /**
418         * 헤더 문구를 반환합니다.
419         * @return 헤더 문구
420         */
421        public String getHeaderTitle(){
422            return headerInfo;
423        }
424
425        /**
426         * 저작자 정보를 지정합니다.
427         * @param copyright 저작자 정보
428         */
429        public void setCopyright(String copyright){
```



```
430         copyrightInfo = copyright;
431     }
432     /**
433      * 저작자 정보를 반환합니다.
434      * @return 저작자 정보
435      */
436     public String getCopyright(){
437         return copyrightInfo;
438     }
439 
440     /**
441      * 라인 넘버 표시 여부를 지정합니다.
442      * @param arg 라인 넘버 표시 여부 / true = 표시 / false = 표시하지 않음
443      */
444     public void setShowLineNumber(boolean arg){
445         showLineNumber = arg;
446     }
447     /**
448      * 라인넘버 표시 여부를 반환합니다.
449      * @return 줄번호 표시 여부
450      */
451     public boolean getShowLineNumber(){
452         return showLineNumber;
453     }
454 
455     /**
456      * 페이지 넘버 표시 여부를 지정합니다.
457      * @param arg 페이지 넘버 표시 여부 / true = 표시 / false = 표시하지 않음
458      */
459     public void setShowPageNumber(boolean arg){
460         showPageNumber = arg;
461     }
462     /**
463      * 페이지 넘버 표시 여부를 반환합니다.
464      * @return 쪽번호 표시 여부
465      */
466     public boolean getShowPageNumber(){
467         return showPageNumber;
468     }
469 
470     /**
471      * 소스 언어 종류를 지정합니다<br>
472      * java, c, cpp
473      * @param lang 소스코드 언어 종류
474      */
475     public void setCodeLanguage(Language lang){
476         codeLanguage = lang;
477     }
```



```
478
479     /**
480      * 소스 언어 종류를 반환합니다.
481      * @return 언어 종류
482      */
483     public Language getCodeLanguage(){
484         return codeLanguage;
485     }
486
487     /**
488      * 헤더 문구의 위치를 지정합니다.
489      * @param align 헤더 위치 정보
490      */
491     public void setHeaderAlign(Align align){
492         headerAlign = align;
493     }
494
495     /**
496      * 헤더 문구의 위치를 반환합니다.
497      * @return 헤더 문구의 위치
498      */
499     public Align getHeaderAlign(){
500         return headerAlign;
501     }
502
503     /**
504      * 저작권 문구의 위치를 지정합니다.
505      * @param align 저작권 위치 정보
506      */
507     public void setCopyrightAlign(Align align){
508         copyrightAlign = align;
509     }
510
511     /**
512      * 저작권 문구의 위치를 반환합니다.
513      * @return 저작권 문주의 위치
514      */
515     public Align getCopyrightAlign(){
516         return copyrightAlign;
517     }
518
519     /**
520      * PDF 출력 시 모든 소스파일을 한개의 문서로 만들지 지정하는 옵션입니다.<br>
521      * 값을 false로 변경할 때 convertPackage나 convertFile 값이 false일 경우 값이 변경되지 않습니다.
522      * @param arg 옵션 설정 값
523      * @return 변경 성공시 true가, 실패시 false가 리턴됩니다.
524      */
525     public boolean setConvertAll(boolean arg){
```

```
526         if(!arg && !(getConvertPackage() || getConvertFile())){
527             return false;
528         }else{
529             convertAll = arg;
530             return true;
531         }
532     }
533 
534     /**
535      * PDF 출력 시 각 패키지 내의 소스파일을 한개의 문서로 만들지 지정하는 옵션입니다.<br>
536      * 값을 false로 변경할 때 convertAll이나 convertFile 값이 false일 경우 값이 변경되지 않습니다.
537      * @param arg 옵션 설정 값
538      * @return 변경 성공시 true가, 실패시 false가 리턴됩니다.
539      */
540     public boolean setConvertPackage(boolean arg){
541         if(!arg && !(getConvertAll() || getConvertFile())){
542             return false;
543         }else{
544             convertPackage = arg;
545             return true;
546         }
547     }
548 
549     /**
550      * PDF 출력 시 각 소스파일별로 문서로 만들지 지정하는 옵션입니다.<br>
551      * 값을 false로 변경할 때 convertAll이나 convertPackage 값이 false일 경우 값이 변경되지 않습니다.
552      * @param arg 옵션 설정 값
553      * @return 변경 성공시 true가, 실패시 false가 리턴됩니다.
554      */
555     public boolean setConvertFile(boolean arg){
556         if(!arg && !(getConvertPackage() || getConvertAll())){
557             return false;
558         }else{
559             convertFile = arg;
560             return true;
561         }
562     }
563 
564     /**
565      * 전체변환 출력 옵션 반환
566      * @return 출력 옵션 값
567      */
568     public boolean getConvertAll(){
569         return convertAll;
570     }
571 
572     /**
573      * 패키지변환 출력 옵션 반환
```



```
574         * @return 출력 옵션 값
575         */
576        public boolean getConvertPackage(){
577            return convertPackage;
578        }
579
580        /**
581         * 개별 변환 출력 옵션 반환
582         * @return 출력 옵션 값
583         */
584        public boolean getConvertFile(){
585            return convertFile;
586        }
587
588        /**
589         * 전체 변환 시 지정할 파일 명 설정
590         * @param filename 파일 명
591         */
592        public void setFileName(String filename) {
593            filename = filename.trim(); //파일 이름의 앞 뒤 공백 제거
594            if(filename.length() > 200){ //길이가 200자 초과인 경우의 예외 처리
595                filename = filename.substring(0, 200);
596            }
597            if(Pattern.compile("[:₩₩₩₩/%*?:|₩"<>]").matcher(filename).find()){ //유효하지 않은 키워드 발견시 기
본값으로 변환
598                filename = "__ConvertAllFile__";
599            }
600            file_name = filename;
601        }
602
603        /**
604         * 전체 변환 시 사용되는 파일 명 값 반환
605         * @return 파일 명
606         */
607        public String getFileName() {
608            return file_name;
609        }
610
611        /**
612         * PDF 저장 위치 설정
613         * @param locate 저장 위치
614         */
615        public void setSaveLocate (String locate) {
616            try{
617                if(!new File(locate).isDirectory()){ //유효하지 않은 폴더일 시 기본값으로 변환
618                    locate = System.getProperty("user.dir") + "₩₩";
619                }
620            }catch(Exception e){
```

```
621             locate = System.getProperty("user.dir") + "₩₩";
622         }
623         save_locate = locate;
624     }
625 
626     /**
627      * PDF 저장 위치 확인
628      * @return 저장 위치
629      */
630     public String getSaveLocate () {
631         return save_locate;
632     }
633 }
634 
```


OptionDefault.java

```java
package option;
/**
 * 옵션의 기본값과 관련된 클래스 입니다.
 * @author 오다솜
 *
 */
public class OptionDefault {

  /**
   * 새롭게 옵션을 생성하고 기본값으로 지정한 뒤 반환합니다.
   * @return 기본값으로 지정된 옵션입니다.
   */
  public static Option getDefaultOption(){
    Option op = new Option();

    //폰트 파일 경로, 크기, 색상, style순
    op.setBasicFont("fonts₩₩malgun.ttf", 11, 0x000000, FontOption.FontStyle.NORMAL); //일반 문자열 표시하는 폰트 옵션 기본값 지정
    op.setKeywordFont("fonts₩₩malgun.ttf", 11, 0xC13A55, FontOption.FontStyle.NORMAL);//키워드 문자 표시하는 폰트 옵션 기본값 지정
    op.setStringFont("fonts₩₩malgun.ttf", 11, 0x0000FF, FontOption.FontStyle.NORMAL); //' ' 혹은 " " 형태로 감싸진 문자열 내부를 표시하는 폰트 옵션 기본값 지정
    op.setCommentFont("fonts₩₩malgun.ttf", 11, 0x3F7F5F, FontOption.FontStyle.NORMAL); //주석 폰트 옵션 기본값 지정
    op.setLineNumberFont("fonts₩₩malgun.ttf", 11, 0xCCCCCC, FontOption.FontStyle.NORMAL);//라인 넘버 를 표시하는 폰트 옵션 기본값 지정

    op.setTabSize(4); //탭 길이 지정

    op.setHeaderTitle("");
    op.setCopyright("");

    op.setPageType(PageOption.PageType.A4);
    op.setMargin(30, 30, 40, 40);

    op.setShowLineNumber(true);
    op.setShowPageNumber(true);

    op.setCodeLanguage(Option.Language.JAVA);

    op.setHeaderAlign(Option.Align.CENTER);
    op.setCopyrightAlign(Option.Align.RIGHT);

    op.setConvertAll(true);
    op.setConvertPackage(true);
    op.setConvertFile(true);

```



```
43          op.setFileName("__ConvertAllFile__");
44          op.setSaveLocate(System.getProperty("user.dir") + "₩₩");
45
46          return op;
47      }
48
49      public static void getDefaultOption(Option op){
50          //폰트 파일 경로, 크기, 색상, style순
51          op.setBasicFont("fonts₩₩malgun.ttf", 11, 0x000000, FontOption.FontStyle.NORMAL); //일반 문자열 표시하는 폰트 옵션 기본값 지정
52          op.setKeywordFont("fonts₩₩malgun.ttf", 11, 0xC13A55, FontOption.FontStyle.NORMAL);//키워드 문자 표시하는 폰트 옵션 기본값 지정
53          op.setStringFont("fonts₩₩malgun.ttf", 11, 0x0000FF, FontOption.FontStyle.NORMAL); //' ' 혹은 " " 형태로 감싸진 문자열 내부를 표시하는 폰트 옵션 기본값 지정
54          op.setCommentFont("fonts₩₩malgun.ttf", 11, 0x3F7F5F, FontOption.FontStyle.NORMAL); //주석 폰트 옵션 기본값 지정
55          op.setLineNumberFont("fonts₩₩malgun.ttf", 11, 0xCCCCCC, FontOption.FontStyle.NORMAL);//라인 넘버 를 표시하는 폰트 옵션 기본값 지정
56
57          op.setTabSize(4); //탭 길이 지정
58
59          op.setHeaderTitle("");
60          op.setCopyright("");
61
62          op.setPageType(PageOption.PageType.A4);
63          op.setMargin(30, 30, 40, 40);
64
65          op.setShowLineNumber(true);
66          op.setShowPageNumber(true);
67
68          op.setCodeLanguage(Option.Language.JAVA);
69
70          op.setHeaderAlign(Option.Align.CENTER);
71          op.setCopyrightAlign(Option.Align.RIGHT);
72
73          op.setConvertAll(true);
74          op.setConvertPackage(true);
75          op.setConvertFile(true);
76
77          op.setFileName("__ConvertAllFile__");
78          op.setSaveLocate(System.getProperty("user.dir") + "₩₩");
79      }
80  }
81
```


OptionFile.java

```
1  package option;
2  import java.io.BufferedReader;
3  import java.io.BufferedWriter;
4  import java.io.FileReader;
5  import java.io.FileWriter;
6  import java.io.IOException;
7
8  /**
9   * 지정된 옵션을 파일로 저장하거나 파일로 저장된 옵션 정보를 읽어들입니다.
10   * @author 유병호
11   *
12   */
13  public class OptionFile {
14
15      /**
16       * 파일에서 읽어들이는 옵션 정보를 저장합니다.
17       */
18      private Option option;
19
20      /**
21       * 지정된 옵션 파일을 읽어들입니다.
22       * @return 읽어들인 옵션 파일입니다.
23       */
24      @SuppressWarnings("resource")
25      public Option readOptionFile(){
26          option = new Option();
27          BufferedReader br = null;
28
29          try{
30              br = new BufferedReader(new FileReader("option.ini")); //option.ini 파일을 읽어들입니다.
31              String[] optionBuffer = new String[Option.optionCount];
32
33              // 정확하게 optionCount 만큼의 반복을 함으로써 옵션 정보가 빠져있는 등의 비정상 적인 상황의
경우
34              // 예외를 발생시키므로 대처가 가능합니다.
35              for(int i = 0 ; i < Option.optionCount ; i++){
36                  optionBuffer[i] = br.readLine();
37              }
38
39              //optionCount 보다 저장되어있는 옵션 정보가 부족한 경우 기본값으로 초기화
40              if(optionBuffer[Option.optionCount - 1] == null){
41                  option = OptionDefault.getDefaultOption();
42                  writeOptionFile(option);
43                  return option;
44              }
45
46              //읽어들인 옵션을 지정합니다.
```

```
47              try{
48                  option.setBasicFont(optionBuffer[0], Integer.valueOf(optionBuffer[1]),
Integer.valueOf(optionBuffer[2]), convertFontStyle(optionBuffer[3]));
49                  option.setKeywordFont(optionBuffer[4], Integer.valueOf(optionBuffer[5]),
Integer.valueOf(optionBuffer[6]), convertFontStyle(optionBuffer[7]));
50                  option.setStringFont(optionBuffer[8], Integer.valueOf(optionBuffer[9]),
Integer.valueOf(optionBuffer[10]), convertFontStyle(optionBuffer[11]));
51                  option.setCommentFont(optionBuffer[12], Integer.valueOf(optionBuffer[13]),
Integer.valueOf(optionBuffer[14]), convertFontStyle(optionBuffer[15]));
52                  option.setLineNumberFont(optionBuffer[16], Integer.valueOf(optionBuffer[17]),
Integer.valueOf(optionBuffer[18]), convertFontStyle(optionBuffer[19]));
53
54                  option.setTabSize(Integer.valueOf(optionBuffer[20]));
55
56                  option.setHeaderTitle(optionBuffer[21]);
57                  option.setCopyright(optionBuffer[22]);
58
59                  option.setPageType(convertPageType(optionBuffer[23]));
60                  option.setMargin(Integer.valueOf(optionBuffer[24]), Integer.valueOf(optionBuffer[25]),
Integer.valueOf(optionBuffer[26]), Integer.valueOf(optionBuffer[27]));
61
62                  option.setShowLineNumber(Boolean.valueOf(optionBuffer[28]));
63                  option.setShowPageNumber(Boolean.valueOf(optionBuffer[29]));
64
65                  option.setCodeLanguage(convertLanguage(optionBuffer[30]));
66
67                  option.setHeaderAlign(convertAlign(optionBuffer[31]));
68                  option.setCopyrightAlign(convertAlign(optionBuffer[32]));
69
70                  option.setConvertAll(Boolean.valueOf(optionBuffer[33]));
71                  option.setConvertPackage(Boolean.valueOf(optionBuffer[34]));
72                  option.setConvertFile(Boolean.valueOf(optionBuffer[35]));
73                  option.setFileName(optionBuffer[36]);
74                  option.setSaveLocate(optionBuffer[37]);
75              }catch(NumberFormatException e){
76                  option = OptionDefault.getDefaultOption();
77                  writeOptionFile(option);
78                  return option;
79              }
80
81              writeOptionFile(option);
82              br.close();
83          } catch(IOException e){
84              // 만약 파일이 없거나 읽어들이는 경과값이 정상적이지 않을경우 기본값으로 지정된 옵션을 생성하
고 그 정보를 파일에 기록합니다.
85              option = OptionDefault.getDefaultOption();
86              writeOptionFile(option);
87          }
```



```
88          return option;
89      }
90
91      /**
92       * 옵션을 파일로 저장합니다.
93       * @param op 파일로 저장할 옵션입니다.
94       */
95      public void writeOptionFile(Option op){
96          BufferedWriter bw = null;
97
98          try{
99              bw = new BufferedWriter(new FileWriter("option.ini"));
100
101             // 각각의 옵션 값은 줄단위로 구별됩니다.
102             bw.write(op.getBasicFont().getFontName()+ "");
103             bw.newLine();
104             bw.write(op.getBasicFont().getFontSize()+ "");
105             bw.newLine();
106             bw.write(op.getBasicFont().getFontColor()+ "");
107             bw.newLine();
108             bw.write(op.getBasicFont().getFontStyle()+ "");
109             bw.newLine();
110
111             bw.write(op.getKeywordFont().getFontName()+ "");
112             bw.newLine();
113             bw.write(op.getKeywordFont().getFontSize()+ "");
114             bw.newLine();
115             bw.write(op.getKeywordFont().getFontColor()+ "");
116             bw.newLine();
117             bw.write(op.getKeywordFont().getFontStyle()+ "");
118             bw.newLine();
119
120             bw.write(op.getStringFont().getFontName()+ "");
121             bw.newLine();
122             bw.write(op.getStringFont().getFontSize()+ "");
123             bw.newLine();
124             bw.write(op.getStringFont().getFontColor()+ "");
125             bw.newLine();
126             bw.write(op.getStringFont().getFontStyle()+ "");
127             bw.newLine();
128
129             bw.write(op.getCommentFont().getFontName()+ "");
130             bw.newLine();
131             bw.write(op.getCommentFont().getFontSize()+ "");
132             bw.newLine();
133             bw.write(op.getCommentFont().getFontColor()+ "");
134             bw.newLine();
135             bw.write(op.getCommentFont().getFontStyle()+ "");
```

```
136             bw.newLine();
137
138             bw.write(op.getLineNumberFont().getFontName()+ "");
139             bw.newLine();
140             bw.write(op.getLineNumberFont().getFontSize()+ "");
141             bw.newLine();
142             bw.write(op.getLineNumberFont().getFontColor()+ "");
143             bw.newLine();
144             bw.write(op.getLineNumberFont().getFontStyle()+ "");
145             bw.newLine();
146
147             bw.write(op.getTabSize()+ "");
148             bw.newLine();
149
150             bw.write(op.getHeaderTitle()+ "");
151             bw.newLine();
152             bw.write(op.getCopyright()+ "");
153             bw.newLine();
154
155             bw.write(op.getPageOption().getPageType() + "");
156             bw.newLine();
157             bw.write(op.getMarginTop()+ "");
158             bw.newLine();
159             bw.write(op.getMarginBottom()+ "");
160             bw.newLine();
161             bw.write(op.getMarginLeft()+ "");
162             bw.newLine();
163             bw.write(op.getMarginRight()+ "");
164             bw.newLine();
165
166             bw.write(op.getShowLineNumber()+ "");
167             bw.newLine();
168             bw.write(op.getShowPageNumber()+ "");
169             bw.newLine();
170             bw.write(op.getCodeLanguage()+ "");
171             bw.newLine();
172
173             bw.write(op.getHeaderAlign()+ "");
174             bw.newLine();
175             bw.write(op.getCopyrightAlign()+ "");
176             bw.newLine();
177
178             bw.write(op.getConvertAll()+ "");
179             bw.newLine();
180             bw.write(op.getConvertPackage()+ "");
181             bw.newLine();
182             bw.write(op.getConvertFile()+ "");
183             bw.newLine();
```



```
184             bw.write(op.getFileName()+ "");
185             bw.newLine();
186             bw.write(op.getSaveLocate()+ "");
187 
188             bw.close();
189         } catch(IOException e){
190             e.printStackTrace();
191         }
192     }
193 
194     /**
195      * 파일에서 읽어들인 문자열을 폰트 스타일로 변환합니다.
196      * @param arg 파일에서 읽은 폰트 스타일과 관련된 문자열입니다.
197      * @return 치환된 폰트 스타일입니다.
198      */
199     private FontOption.FontStyle convertFontStyle(String arg){
200         if(arg.equals("NORMAL")){
201             return FontOption.FontStyle.NORMAL;
202         }else if(arg.equals("BOLD")){
203             return FontOption.FontStyle.BOLD;
204         }else if(arg.equals("ITALIC")){
205             return FontOption.FontStyle.ITALIC;
206         }else if(arg.equals("BOLDITALIC")){
207             return FontOption.FontStyle.BOLDITALIC;
208         }else if(arg.equals("UNDERLINE")){
209             return FontOption.FontStyle.UNDERLINE;
210         }else if(arg.equals("STRIKETHRU")){
211             return FontOption.FontStyle.STRIKETHRU;
212         }else{
213             return FontOption.FontStyle.NORMAL;
214         }
215     }
216 
217     /**
218      * 파일에서 읽어들인 문자열을 용지 크기로 변환합니다.
219      * @param arg 파일에서 읽은 용지 크기와 관련된 문자열입니다.
220      * @return 치환된 용지 크기입니다.
221      */
222     private PageOption.PageType convertPageType(String arg){
223         if(arg.equals("A1")){
224             return PageOption.PageType.A1;
225         }else if(arg.equals("A2")){
226             return PageOption.PageType.A2;
227         }else if(arg.equals("A3")){
228             return PageOption.PageType.A3;
229         }else if(arg.equals("A4")){
230             return PageOption.PageType.A4;
231         }else if(arg.equals("A5")){
```



```
232             return PageOption.PageType.A5;
233         }else if(arg.equals("A6")){
234             return PageOption.PageType.A6;
235         }else if(arg.equals("B1")){
236             return PageOption.PageType.B1;
237         }else if(arg.equals("B2")){
238             return PageOption.PageType.B2;
239         }else if(arg.equals("B3")){
240             return PageOption.PageType.B3;
241         }else if(arg.equals("B4")){
242             return PageOption.PageType.B4;
243         }else if(arg.equals("B5")){
244             return PageOption.PageType.B5;
245         }else if(arg.equals("B6")){
246             return PageOption.PageType.B6;
247         }else{
248             return PageOption.PageType.A4;
249         }
250     }
251 
252     /**
253      * 파일에서 읽어들인 문자열을 언어 종류 타입으로 변환합니다.
254      * @param arg 읽어들인 언어 종류와 관련된 문자열입니다.
255      * @return 치환된 언어 종류 타입입니다.
256      */
257     private Option.Language convertLanguage(String arg){
258         if(arg.equals("JAVA")){
259             return Option.Language.JAVA;
260         }else if(arg.equals("C")){
261             return Option.Language.C;
262         }else if(arg.equals("CPP")){
263             return Option.Language.CPP;
264         }else{
265             return Option.Language.JAVA;
266         }
267     }
268 
269     /**
270      * 파일에서 읽어들인 문자열을 정렬 위치 타입으로 변환합니다.
271      * @param arg 읽어들인 위치와 관련된 문자열입니다.
272      * @return 치환된 정렬 위치 정보 타입입니다.
273      */
274     private Option.Align convertAlign(String arg){
275         if(arg.equals("LEFT")){
276             return Option.Align.LEFT;
277         }else if(arg.equals("RIGHT")){
278             return Option.Align.RIGHT;
279         }else if(arg.equals("CENTER")){
```



```
280                 return Option.Align.CENTER;
281             }else{
282                 return Option.Align.CENTER;
283             }
284         }
285     }
286
```


PageOption.java

```java
package option;
/**
 * 페이지와 관련된 옵션 클래스 입니다.
 * @author 오다솜
 *
 */
public class PageOption {
    /**
     * 용지의 크기(타입) 데이터 입니다.
      * @author 오다솜
      *
      */
     public enum PageType {A1, A2, A3, A4, A5, A6, B1, B2, B3, B4, B5, B6};

     /**
      * 용지의 위쪽 여백입니다.
      */
     private int marginTop;

     /**
      * 용지의 아래쪽 여백입니다.
      */
     private int marginBottom;

     /**
      * 용지의 왼쪽 여백입니다.
      */
     private int marginLeft;

     /**
      * 용지의 오른쪽 여백입니다.
      */
     private int marginRight;

     /**
      * 용지의 크기 입니다.
      */
     private PageType type;

     /**
      * 용지의 크기를 설정합니다.
      * @param type 용지의 크기
      */
     public void setPageType(PageType type){
         this.type = type;
     }

```



```
48      /**
49       * 용지의 크기를 반환합니다.
50       * @return 용지의 크기
51       */
52      public PageType getPageType(){
53          return this.type;
54      }
55
56      /**
57       * 용지의 위쪽 여백을 지정합니다.
58       * @param margin 위쪽 여백
59       */
60      public void setMarginTop(int margin){
61          if(margin < 0 || margin >= 100){
62              margin = 30;
63          }
64          marginTop = margin;
65      }
66      /**
67       * 용지의 위쪽 여백값을 반환합니다.
68       * @return 위쪽 여백
69       */
70      public int getMarginTop(){
71          return marginTop;
72      }
73
74      /**
75       * 용지의 아랫쪽 여백을 지정합니다.
76       * @param margin 아래쪽 여백
77       */
78      public void setMarginBottom(int margin) {
79          if(margin < 0 || margin >= 100){
80              margin = 30;
81          }
82          marginBottom = margin;
83      }
84      /**
85       * 용지의 아래쪽 여백을 반환합니다.
86       * @return 아래쪽 여백
87       */
88      public int getMarginBottom(){
89          return marginBottom;
90      }
91
92      /**
93       * 용지의 왼쪽 여백을 지정합니다.
94       * @param margin 왼쪽 여백
95       */
```



```
96      public void setMarginLeft(int margin) {
97          if(margin < 0 || margin >= 100){
98              margin = 40;
99          }
100         marginLeft = margin;
101     }
102     /**
103      * 용지의 왼쪽 여백을 반환합니다.
104      * @return 왼쪽 여백
105      */
106     public int getMarginLeft(){
107         return marginLeft;
108     }
109
110     /**
111      * 용지의 오른쪽 여백을 지정합니다.
112      * @param margin 오른쪽 여백
113      */
114     public void setMarginRight(int margin) {
115         if(margin < 0 || margin >= 100){
116             margin = 40;
117         }
118         marginRight = margin;
119     }
120     /**
121      * 용지의 오른쪽 여백을 반환합니다.
122      * @return 오른쪽 여백
123      */
124     public int getMarginRight(){
125         return marginRight;
126     }
127 }
128
```

Security.java

```
1  package option;
2
3  /**
4   * 보안 옵션 클래스 입니다.
5   * @author 유병호
6   *
7   */
8  public class Security {
9      /**
10      * 보안 옵션의 사용 여부입니다.
11      */
12     private boolean useSecurityOption = false;
13
14     /**
15      * 복사 가능 여부입니다.
16      */
17     private boolean useCopyMode = true;
18
19     /**
20      * 비밀번호 사용 여부입니다.
21      */
22     private boolean usePassword = false;
23
24     /**
25      * 비밀번호 입니다.
26      */
27     private String password = "";
28
29
30     /**
31      * 보안 옵션 사용 여부를 설정합니다.
32      * @param arg 사용여부<br>
33      * true : 사용<br>
34      * false : 사용하지 않음
35      */
36     public void setUseSecurityOption(boolean arg){
37         useSecurityOption = arg;
38     }
39
40     /**
41      * PDF에 기록된 소스코드의 복사 여부를 지정합니다.
42      * @param arg 복사 여부<br>
43      * true : 복사 가능 <br>
44      * false: 복사 불가능
45      */
46     public void setUseCopyMode(boolean arg){
47         useCopyMode = arg;
```

```
48      }
49
50      /**
51       * 문서의 패스워드 사용 여부를 지정합니다.
52       * @param arg 패스워드 사용 여부<br>
53       * true : 패스워드 사용<br>
54       * false: 패스워드를 사용하지 않음
55       */
56      public void setUsePassword(boolean arg){
57          usePassword = arg;
58      }
59
60      /**
61       * 문서의 패스워드를 지정합니다.
62       * @param pwd 패스워드<br>
63       * 패스워드를 사용하지 않음으로 설정되어 있을 경우 지정된 값은 사용되지 않습니다.
64       */
65      public void setPassword(String pwd){
66          password = pwd;
67      }
68
69      /**
70       * 보안 기능 사용 여부를 반환합니다.
71       * @return 보안 기능 사용 여부
72       */
73      public boolean getUseSecurityOption(){
74          return useSecurityOption;
75      }
76
77      /**
78       * 복사 가능 여부를 반환합니다.
79       * @return 복사 가능 여부
80       */
81      public boolean getUseCopyMode(){
82          return useCopyMode;
83      }
84
85      /**
86       * 비밀번호 사용 여부를 반환합니다.
87       * @return 비밀번호 사용 여부
88       */
89      public boolean getUsePassword(){
90          return usePassword;
91      }
92
93      /**
94       * 비밀번호를 반환합니다.
95       * @return 비밀번호
```



```
96          */
97         public String getPassword(){
98             return password;
99         }
100    }
101
```

