

# 2019 소프트웨어프로젝트

## Project 6

친구목록 관리 프로그램 최종 설계 및 구현



중앙대학교

창의ICT공과대학 소프트웨어학부

20185659 김혜성

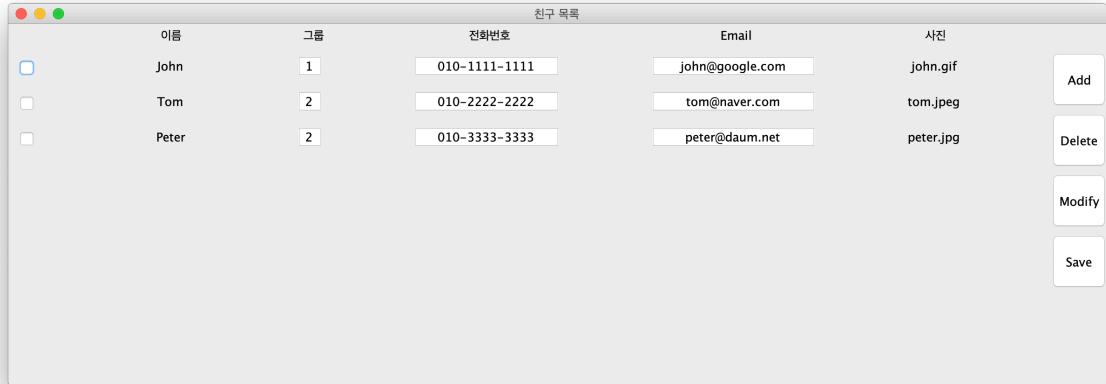
# 목차

## *Table of contents*

1. Abstract
2. Introduction
3. Proposed Program
  - 3.1. Class diagram
  - 3.2. Analyzation of classes
    - 3.2.1. Friend class
      - 3.2.1.1. Analyzation of the source code
      - 3.2.1.2. Point of view
    - 3.2.2. FriendList class
      - 3.2.2.1. Analyzation of the source code
      - 3.2.2.2. Point of view
    - 3.2.3. FriendListFile class
      - 3.2.3.1. Analyzation of the source code
      - 3.2.3.2. Point of view
    - 3.2.4. FriendListFrame class
      - 3.2.4.1. Analyzation of the source code
      - 3.2.4.2. Point of view
    - 3.2.5. FriendListInformationPanel class
      - 3.2.5.1. Analyzation of the source code
      - 3.2.5.2. Point of view
    - 3.2.6. AddFriendListInformationFrame class
      - 3.2.6.1. Analyzation of the source code
      - 3.2.6.2. Point of view
  4. Program Results
  5. Conclusion
  6. References
  7. Evaluation

## 1. Abstract

이 프로젝트에서는 프로젝트 5의 GUI에 이벤트 핸들링을 추가해서 친구목록 관리 프로그램을 완성했다. 이 프로그램의 GUI 구성은 다음과 같다.



▲ Fig. 1. 친구 목록 정보 화면

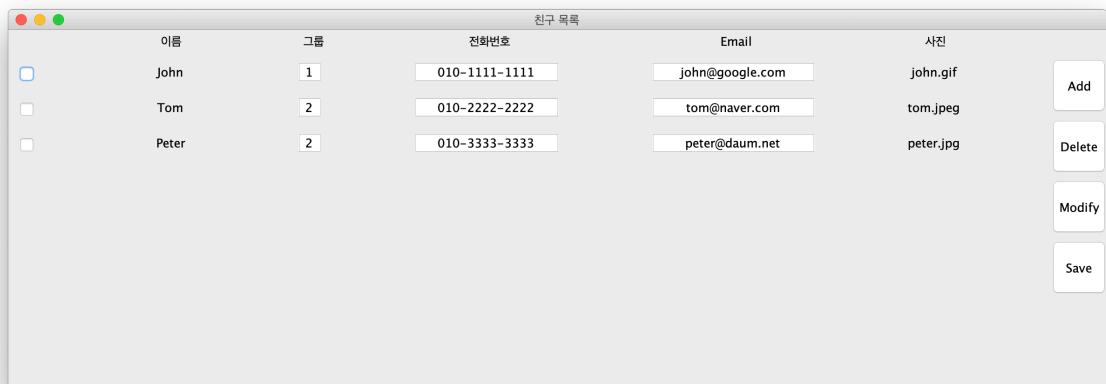


▲ Fig. 2. 추가할 친구 정보 입력 화면

프로그램 실행 과정은 다음과 같다.

1. 정해진 파일 (**friendlist-norm.data**)에 있는 친구 정보들을 읽어와 메인화면의 리스트에 표시 한다.

(파일 parsing 과정은 project3의 FriendListFile class를 사용했다)

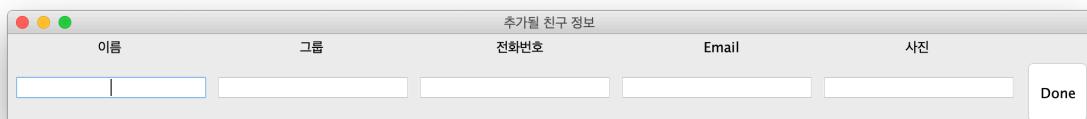


▲ Fig. 3. friendlist-norm.data 파일 parsing 후 메인 화면

**2. “Add” 버튼을 누르면, 새로운 친구정보 입력창(Frame)이 생긴다.**



▲ Fig. 4. “Add” 버튼을 눌렀을 때 메인화면

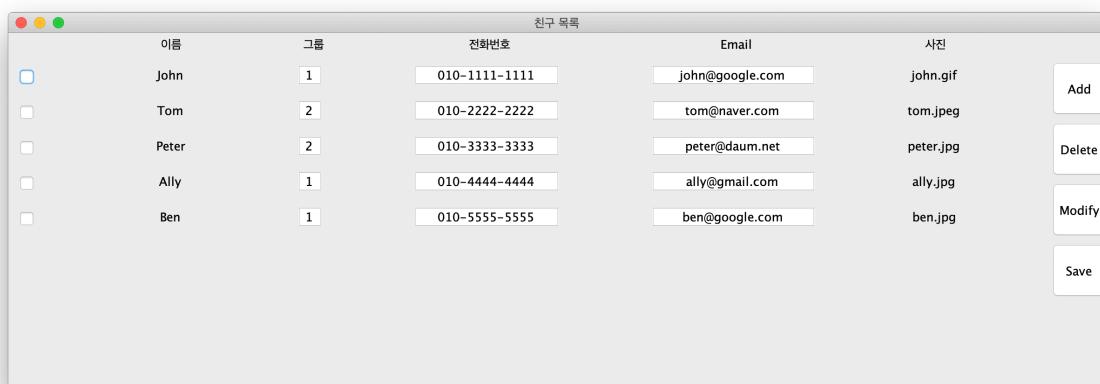


▲ Fig. 5. 메인화면에서 “Add” 버튼을 눌렀을 때 생성된 친구정보 입력창(Frame)

**3. 입력을 마치고 “Done”을 누르면 새로운 친구 Record(Friend 객체)가 FriendList에 추가되고 메인화면의 리스트에도 해당 친구가 추가된다.**

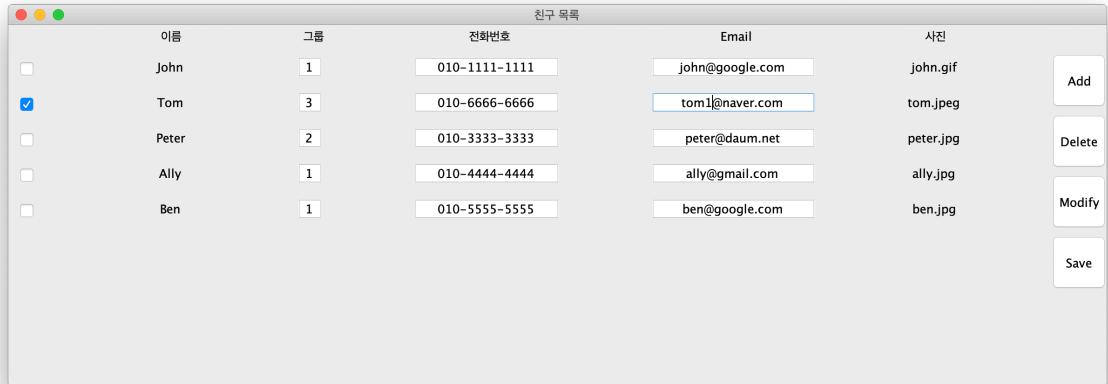


▲ Fig. 6. 새로운 친구정보를 입력한 친구정보 입력창

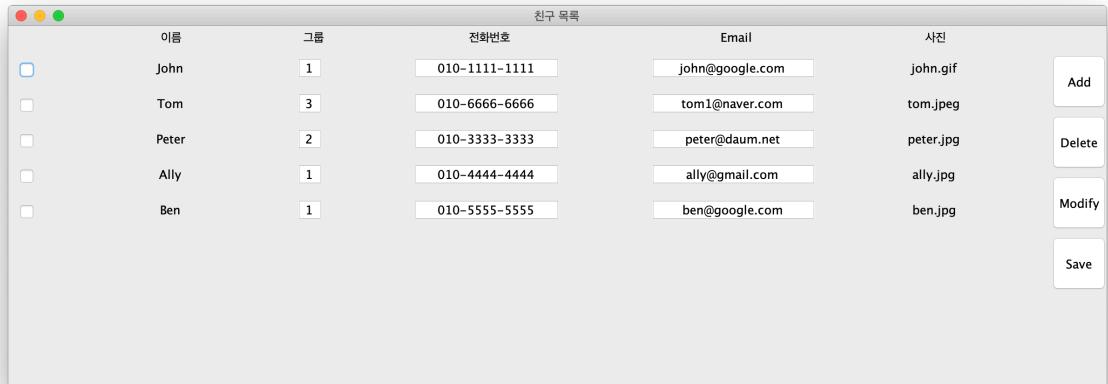


▲ Fig. 7. “Done” 버튼을 눌렀을 때 친구정보가 추가된 메인화면

- 4. “Modify” 버튼을 누르면, 리스트 중에 선택된 친구 정보가 화면 내용으로 수정된다. 즉, 리스트 상에서 친구를 선택하고 내용을 수정한 후, “Modify” 버튼을 누른다. (이 때, 복수 선택 불가)**

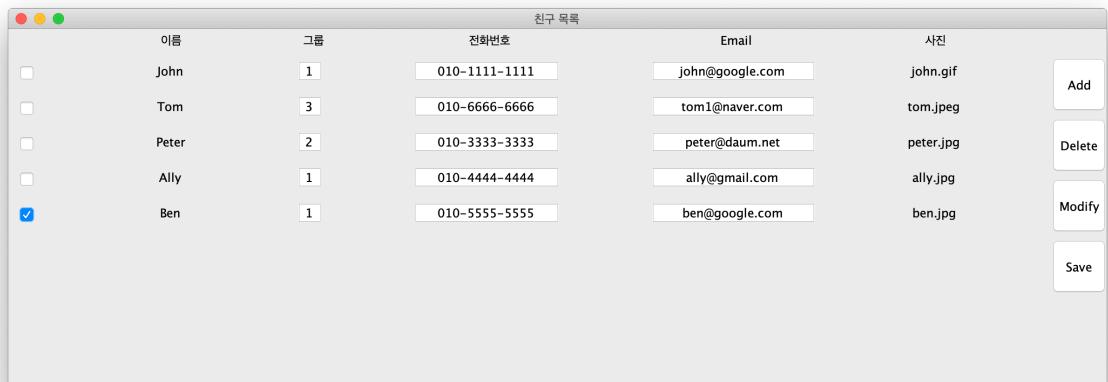


▲ Fig. 8. checkBox 선택 후 textField를 수정한 상태의 메인화면



▲ Fig. 9. “Modify” 버튼을 눌렀을 때 메인화면

- 5. “Delete” 버튼을 누르면, 리스트 중에 선택된 친구 정보가 제거되고, 메인화면의 리스트에서도 제거된다.**



친구 목록					
	이름	그룹	전화번호	Email	사진
<input type="checkbox"/>	John	1	010-1111-1111	john@google.com	john.gif
<input type="checkbox"/>	Tom	3	010-6666-6666	tom1@naver.com	tom.jpeg
<input type="checkbox"/>	Peter	2	010-3333-3333	peter@daum.net	peter.jpg
<input type="checkbox"/>	Ally	1	010-4444-4444	ally@gmail.com	ally.jpg

▲ Fig. 11. “Delete” 버튼을 눌렀을 때 메인화면

6. “Save” 버튼을 누르면 현재 메모리에 있는 Record들(화면 상의 친구 리스트에 있는 친구 정보들)을 파일 양식에 맞게 저장한다.

```
friendlist-norm.data
John : 1 : 010-1111-1111 : john@google.com : john.gif
Tom : 2 : 010-2222-2222 : tom@naver.com : tom.jpeg
Peter : 2 : 010-3333-3333 : peter@daum.net : peter.jpg
```

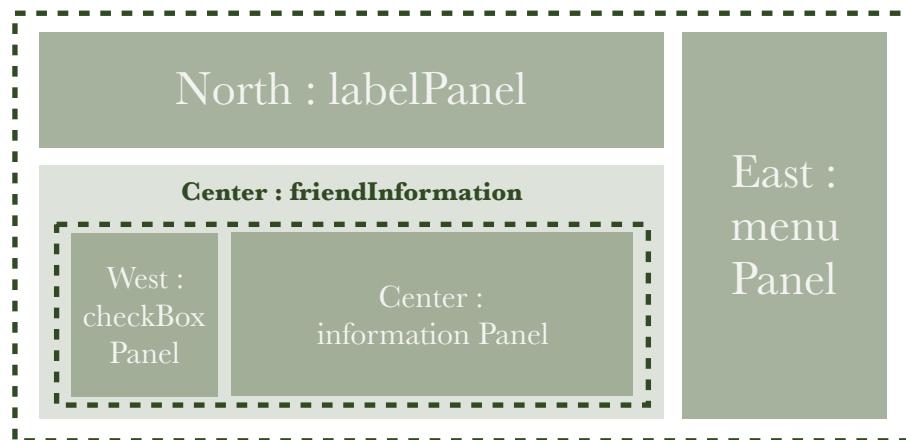
▲ Fig. 12. “Save” 버튼을 누르기 전, 초기 friendlist-norm.data 파일

```
friendlist-norm.data
John:1:010-1111-1111:john@google.com:john.gif
Tom:3:010-6666-6666:tom1@naver.com:tom.jpeg
Peter:2:010-3333-3333:peter@daum.net:peter.jpg
Ally:1:010-4444-4444:ally@gmail.com:ally.jpg
```

▲ Fig. 11. “Save” 버튼을 누른 후, 저장된 friendlist-norm.data 파일

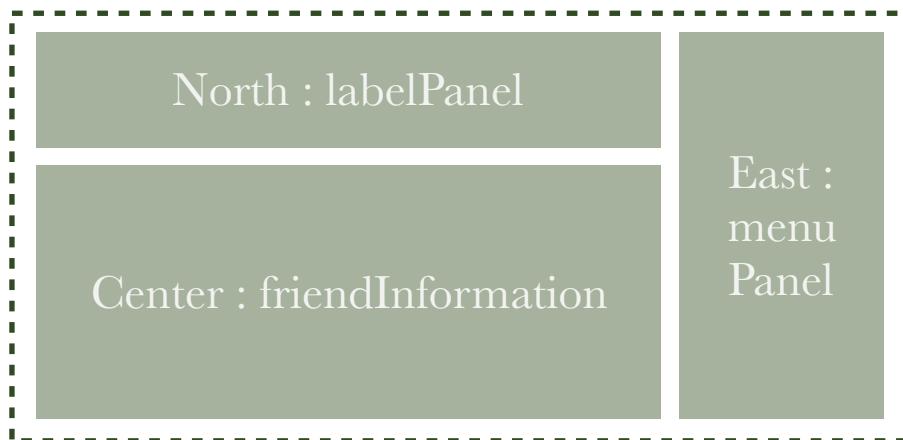
프로그램 화면의 레이아웃 설정은 다음과 같다.

### 1. 친구 목록 정보 화면



▲ Fig. 12. Layout of FriendListFrame

### 2. 추가할 친구 정보 입력화면



▲ Fig. 13. Layout of FriendListFrame with FriendListInformationPanel

## 2. Introduction

이 프로젝트의 목적은 다음과 같다.

### 1. 객체지향 프로그램 설계 연습

- 1.1. 필요 **Class** 정의, 클래스 별로 기능 나누기
- 1.2. 프로그램 크기 증대에 따른 관리 문제 및 **Java**의 해결법 체득

### 2. 한 학기 동안 배운 모든 주요 **Java** 기능 체득

- 2.1. 기본 **GUI, File I/O**, 스트링 처리 (간단한 **DB** 프로그래밍)
- 2.2. **Event - driven programming, object reference** 처리
- 2.3. **Incremental programming**

### 3. 개발 도구 활용법, 디버깅

### 4. 프로그램 개발 과정 체득

이 프로젝트의 추진 방법은 다음과 같다.

### 1. 서로 독립적인 **Class**들 사이에서 자료를 공유하는 방법 연구

- 1.1. **JFrame**을 기반으로 하는 **Class** 복수개 생성
- 1.2. 그들 간의 정보 교환  
(ex. “**Add**”의 경우, 입력창 **frame**에 입력한 내용이 메인 **frame**으로 전달되어야 한다)

- 1.3. 프로그램 구조의 틀을 유지하면서 정보 교환

### 2. 화면 내용 수정의 경우, 화면의 재정비 필요

- (ex. “**Delete**” 작업)

### 3. 프로그램 내부에 있는 친구 **DB(friend list)**와 화면에 보여지는 내용은 항상 일치

### 4. 위의 두가지 문제를 동시에 해결

- 4.1. **JTable** 사용 불가
- 4.2. ex. 매 작업마다 두 내용을 동기화, 화면 갱신

이 프로젝트를 진행하면서 panel 생성과 레이아웃 설정과 관련한 시행착오를 통해 독립적인 Class들 사이에서 자료를 공유하는 방법을 이해할 수 있었다.

따라서 친구목록 관리 프로그램에 있어서 고찰점은 다음과 같다.

### 1. 객체 생성에 유의해야 한다.

### 3. Proposed Program

#### 3.1. Class diagram



▲ Fig. 14. Class diagram

### 3.2. Analyzation of classes

#### 3.2.1. Friend class

```
1 public class Friend {  
2  
3     // private variables of informations of a contact  
4     private String nameOfContact;  
5     private int groupOfContact;  
6     private String phoneNumberOfContact;  
7     private String emailAddressOfContact;  
8     private String pictureOfContact;  
9  
10    // make setters, getters of private variables of informations of a contact  
11    public void setName(String nameOfContact) {  
12        this.nameOfContact = nameOfContact;  
13    }  
14    public String getName() {  
15        return nameOfContact;  
16    }  
17    public void setGroup(int groupOfContact) {  
18        this.groupOfContact = groupOfContact;  
19    }  
20    public int getGroup() {  
21        return groupOfContact;  
22    }  
23    public void setPhoneNumber(String phoneNumberOfContact) {  
24        this.phoneNumberOfContact = phoneNumberOfContact;  
25    }  
26    public String getPhoneNumber() {  
27        return phoneNumberOfContact;  
28    }  
29    public void setEmailAddress(String emailAddressOfContact) {  
30        this.emailAddressOfContact = emailAddressOfContact;  
31    }  
32    public String getEmailAddress() {  
33        return emailAddressOfContact;  
34    }  
35    public void setPicture(String pictureOfContact) {  
36        this.pictureOfContact = pictureOfContact;  
37    }  
38    public String getPicture() {  
39        return pictureOfContact;  
40    }  
41  
42    // make a form of information of a contact  
43    public String formOfInformation() {  
44        return (nameOfContact + ":" + groupOfContact + ":" + phoneNumberOfContact  
45                + ":" + emailAddressOfContact + ":" + pictureOfContact);  
46    }  
47    public void print() {  
48        System.out.println(this.formOfInformation());  
49    }  
50 }
```

cs

▲ Fig. 15. Source code of Friend class

### 3.2.3.1. Analyzation of the source code

1-50 : Friend class

- 4-8 : private 항목 변수 선언
- 11-40 : private 변수의 setter, getter 생성
- 43-46 : public formOfInformation method 정의
  - 44-45 : 구분자(“:”)를 포함하여 형식을 맞춤
- 47-49 : public print method 정의
  - 48 : formOfInformation method로 정리된 형식을 콘솔에 출력

### 3.2.3.2. Point of view

1. 구분자(“:"), 5항목으로 contact 정보 private으로 구성

- 1.1. 이름
- 1.2. Group
- 1.3. 전화번호
- 1.4. Email
- 1.5. 사진

### 3.2.2. FriendList class

```
1 import java.util.ArrayList;
2
3 public class FriendList {
4
5     private int numberofContactLists = 0;
6
7     private ArrayList<Friend> contactInformation = new ArrayList<Friend>();
8
9     public int numFriends() {
10         return numberofContactLists;
11     }
12
13     public Friend getFriend(int i) {
14         if (isRightIndexNumber(i)) {
15             return contactInformation.get(i);
16         }
17         else {
18             return null;
19         }
20     }
21
22     public void removeFriend(int index) {
23         try {
24             contactInformation.remove(index);
25             numberofContactLists--;
26         } catch (ArrayIndexOutOfBoundsException e) {
27         }
28     }
29
30     public void modifyFriendInformation(int index, int modifiedGroup, String modifiedPhone
31     number, String modifiedEmailAddress) {
32         contactInformation.get(index).setGroup(modifiedGroup);
33         contactInformation.get(index).setPhoneNumber(modifiedPhoneNumber);
34         contactInformation.get(index).setEmailAddress(modifiedEmailAddress);
35     }
36
37     public void printContactInformation() {
38         for (int i = 0; i < numberofContactLists; i++) {
39             contactInformation.get(i).print();
40         }
41     }
42
43     public void addFriendInformation(Friend parsedFriendInformation) {
```

```

45     contactInformation.add(parsedFriendInformation);
46     numberOfWorkLists++;
47 }
48
49
50 private boolean isRightIndexNumber(int index) {
51     if (index >= 0 && index <= 100 && index <= numberOfWorkLists) {
52         return true;
53     }
54     else {
55         return false;
56     }
57 }
58
59 public boolean isConflictedName(String nameOfContact) {
60     for (int i = 0; i < numberOfWorkLists; i++) {
61         if (nameOfContact.equals(contactInformation.get(i).getName())) {
62             System.out.println("The name of the contact is conflicted.");
63             return true;
64         }
65     }
66     return false;
67 }
68 }

```

▲ Fig. 16. Source code of FriendList class

### 3.2.2.1. Analyzation of the source code

#### 1-48 : FriendList class

- 5 : numberOfWorkLists 변수 정의 (친구목록의 갯수를 구하기 위한 변수)
- 7 : Friend class type의 contactInformation ArrayList 정의 (모든 Friend를 저장하기 위한 ArrayList)
- 9 - 12 : public numFriends method 정의 (친구목록의 갯수를 반환하는 method)
- 12-20 : public getFriend method 정의
  - (contactInformation 안의 특정 Friend 정보를 반환하는 method)
  - 14 : index의 형식이 옳은지 판단
- 22 - 28 : public removeFriend method 정의
  - (contactInformation 안의 특정 Friend 정보를 제거하는 method)
- 30 - 35 : public modifyFriendInformation method 정의
  - (contactInformation 안의 특정 Friend 정보를 수정하는 method)
- 37 - 41 : public printContactInformation method 정의
  - (contactInformation 안의 특정 Friend 정보를 콘솔에 출력하는 method)
- 43 - 48 : public addFriendInformation method 정의
  - (contactInformation 안에 파싱된 Friend 정보를 저장하는 method)
  - 40 : 파싱된 contact 정보를 contactInformation에 저장
- 50 - 57 : private isRightIndexNumber method 정의
  - (index가 contactInformation 안에서 유효한 숫자인지 검사하는 method)

- 59 - 67 : public isConflictedName method 정의  
(새로운 Friend의 이름과 contactInformation 안에 같은 이름이 있는지 검사하는 method)

### 3.2.2.2. Point of view

1. Friend class를 ArrayList로 생성하였다.
  - 1.1. ArrayList는 일반 배열에 비해 크기를 임의로 변화시킬 수 있어 메모리 낭비를 줄일 수 있다.
  - 1.2. List에 들어갈 type을 임의로 설정할 수 있다. Friend class를 type으로 설정했다.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with "project6" selected, containing "src" and "friendlist-norm.data". Inside "src", there are files: Friend.java, FriendList.java, FriendListFile.java, and TestFriendListProgram.java.
- Editor:** Displays the content of FriendList.java. A specific line of code is highlighted with a red box:

```
private ArrayList<Friend> contactInformation = new ArrayList<Friend>();
```
- Console:** Shows the output of a terminated Java application:

```
<terminated> TestFriendListProgram (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java
John:1:010-1111-1111;john@google.com;john.gif
Tom:2:010-2222-2222;tom@naver.com;tom.jpeg
Peter:2:010-3333-3333;peter@daum.net;peter.jpg
```

▲ Fig. 17. Eclipse of FriendList class

### 3.2.3. FriendListFile class

```
1 import java.io.*;
2 import java.util.Scanner;
3 import java.util.regex.Pattern;
4
5 public class FriendListFile {
6
7     private FriendList friendList = new FriendList();
8
9     public static String fileName;
10
11    public FriendList readFileToList(String fileName) {
12        FriendListFile.fileName = fileName;
13
14        Scanner inputFile = scanFile(fileName);
15
16        if (inputFile != null) {
17            while (inputFile.hasNext()) {
18                String aLineOfFriendListFile = inputFile.nextLine();
19                Friend parsedFriendInformation = parserOfFriendListFile(aLine
20 ofFriendListFile);
21                if (parsedFriendInformation != null) {
22                    try {
23                        friendList.addFriendInformation(parsedFriendInformati
24 on);
25                    } catch (ArrayIndexOutOfBoundsException e) {
26                        System.out.println("The number of contacts is out of
27 boundary.");
28                        System.out.println("The maximum number of contacts is
29 100.");
30                    }
31                }
32            }
33            inputFile.close();
34            return friendList;
35        } else {
36            System.out.println("The file is empty.");
37            return friendList;
38        }
39    }
40
41
42}
```

```

43     private Scanner scanFile(String fileName) {
44         try {
45             // make a file instance
46             File file = new File(fileName);
47             return new Scanner(file);
48         } catch (Exception e) {
49             // print error message
50             System.out.println("File name is unfound.");
51             return null;
52         }
53     }
54
55     private Friend parserOfFriendListFile(String aLineOfFriendListFile) {
56         Friend parsedFriendInformation = null;
57         String originalLine = aLineOfFriendListFile;
58         // remove space characters
59         String aLineRemovedspaceCharacters = originalLine.replace(" ", "");
60         // start to parse the line if it is not an empty or a comment line
61         if (!isCommentLine(aLineRemovedspaceCharacters) && !
62             isEmptyLine(aLineRemovedspaceCharacters)) {
63             String[] splitedFriendInformationByCategory = aLineRemovedspaceCh
64             aracters.split(":");
65             if (isRightInputInformation(splitedFriendInformationByCategory)) {
66                 parsedFriendInformation = new Friend();
67                 parsedFriendInformation.setGroup(Integer.parseInt(splitedFrie
68                 ndInformationByCategory[1]));
69                 parsedFriendInformation.setName(splitedFriendInformationByCat
70                 egory[0]);
71                 parsedFriendInformation.setPhoneNumber(splitedFriendInformati
72                 onByCategory[2]);
73                 parsedFriendInformation.setEmailAddress(splitedFriendInformati
74                 onByCategory[3]);
75                 if (splitedFriendInformationByCategory.length == 5) {
76                     parsedFriendInformation.setPicture(splitedFriendInformati
77                 onByCategory[4]);
78                 } else {
79                     if (aLineRemovedspaceCharacters.substring(aLineRemovedspa
80                     ceCharacters.length() - 1, aLineRemovedspaceCharacters.length()) != ":" ) {
81                         System.out.println("The format of the input is wrong.");
82                     }
83                 }
84             }
85             return parsedFriendInformation;
86         } else {

```

```

87         return null;
88     }
89 }
90
91 private boolean isCommentLine(String aLineRemovedspaceCharacters) {
92     if (aLineRemovedspaceCharacters.startsWith("//")) {
93         return true;
94     } else {
95         return false;
96     }
97 }
98
99 private boolean isEmptyLine(String aLineRemovedspaceCharacters) {
100    if (aLineRemovedspaceCharacters == null) {
101        return true;
102    } else {
103        return false;
104    }
105 }
106
107 private boolean isRightInputInformation(String[] splitedFriendInformationByCategory) {
108     if (splitedFriendInformationByCategory.length < 4 || splitedFriendInformationByCategory.length > 5) {
109         System.out.println("At least one category is omitted.");
110         return false;
111     } else if (isWrongFormat(splitedFriendInformationByCategory)) {
112         return false;
113     }
114     return true;
115 }
116
117
118 private boolean isWrongFormat(String[] splitedFriendInformationByCategory) {
119     if (friendList.isConflictedName(splitedFriendInformationByCategory[0])
120         || !isIntegerGroup(splitedFriendInformationByCategory[1])
121         || !isPhoneNumber(splitedFriendInformationByCategory[2])
122         || !isEmailAddress(splitedFriendInformationByCategory[3])) {
123         return true;
124     } else {
125         return false;
126     }
127 }
128
129
130 private boolean isIntegerGroup(String groupOfContact) {

```

```

131     try {
132         Integer.parseInt(groupOfContact);
133     } catch (NumberFormatException e) {
134         System.out.println("The format of the group of the contact is no
135 t a integer.");
136         return false;
137     }
138     return true;
139 }
140 private boolean isPhoneNumber(String phoneNumberOfContact) {
141     if (Pattern.matches("^\\d{3}\\d{4}[-]\\d{3}\\d{4}[-]\\d{3}$",
142     phoneNumberOfContact.trim())) {
143         return true;
144     } else {
145         System.out.println("The format of the input of the phoneNumber is wrong.");
146         return false;
147     }
148 }
149 private boolean isEmailAddress(String emailAddressOfContact) {
150     emailAddressOfContact = emailAddressOfContact.trim();
151
152     if (emailAddressOfContact == null) {
153         System.out.println("Email category is empty.");
154         return true;
155     } else if (Pattern.matches("^[a-zA-Z0-9]+([a-zA-Z0-9-]+)*@[?:\\w+\\.]+\\w+$",
156     emailAddressOfContact)) {
157         return true;
158     } else {
159         System.out.println("The format of the input of the email is wrong.");
160         return false;
161     }
162 }
163
164 public void saveNewFriendListFile(FriendList friendList) throws IOException {
165     FileWriter fileWriter = new FileWriter(fileName);
166     for (int i = 0; i < friendList.numFriends(); i++) {
167         fileWriter.write(friendList.getFriend(i).formOfInformation()
168 + "\n");
169     }
170     fileWriter.close();
171 }
172 }
```

▲ Fig. 18. Source code of FriendListFile class

### 3.2.3.1. Analyzation of the source code

1 - 3 : 외부 class 추가

5 - 172 : FriendListFile class

- 7 : FriendList class object, friendList 생성
- 9 : fileName static variable 정의
- 11 - 39 : public readFileDialog method 정의 (file을 읽어서 FriendList로 반환하는 method)
  - 14 : scanFile method를 호출
    - (fileName과 일치하는 정상 file이 존재하는지 검사하는 method)
  - 16 - 35 : 정상 file이 존재하는 경우 file 읽기
    - 19 - 20 : parserOffFriendListFile method 호출
      - (file을 한줄씩 파싱해서 Friend로 반환하는 method)
    - 21 - 31 : parsedFriendInformation을 추가하기 전에 index boundary 예외처리
    - 35 - 38 : 정상 file이 존재하지 않는 경우 예외처리
  - 43 - 53 : private scanFile method 정의
    - (fileName과 일치하는 정상 file이 존재하는지 검사하는 method)
- 55 - 89 : private parserOffFriendListFile method 정의
  - (file을 한줄씩 파싱해서 Friend로 반환하는 method)
  - 56 : 파싱된 contact 정보를 저장할 변수 선언
  - 59 : 공백문자를 무시하기 위해 공백문자 제거
  - 61 - 62 : 빈줄이거나 “//”로 시작하는 comment를 제거하기 위한 조건문
  - 63 - 64 : 공백문자가 제거된 줄을 구분자(“.”)를 기준으로 항목을 잘라서 저장
  - 65 - 88 : contact 형식 검사
    - 65 - 77 : parsedFriendInfromation 객체 생성 및 객체에 항목별로 저장
- 91 - 97 : private isCommentLine method 정의 (Comment인지 확인하는 method)
- 99 - 105 : private isEmptyLine method 정의 (빈줄인지 확인하는 method)
- 107 - 117 : private isRightInputInformation method 정의 (정상 입력 값인지 확인하는 method)
  - 109 - 113 : 항목이 4개 혹은 5개가 아닐 경우
  - 113 - 116 : 입력된 형식이 잘못 되었을 경우
- 119 - 128 : private isWrongFormat method 정의 (입력 값 형식 확인하는 method)
  - 120 - 123 : 같은 이름 충돌, group 항목 integer값, 전화번호 형식, 이메일주소 형식 체크
- 130 - 139 : private isIntegerGroup method 정의 (group 형식이 integer인지 확인하는 method)
  - 131 - 137 : 예외처리
- 140 - 148 : private isPhoneNumber method 정의
  - (phoneNumber가 정상 입력 값인지 확인하는 method)
  - 114 : 정규표현식으로 전화번호 형식 조건문 표현
- 149 - 162 : private isEmailAddress method 정의
  - (emailAddress가 정상 입력 값인지 확인하는 method)
  - 152 - 154 : email 항목이 비었을 경우 예외처리
  - 155 - 157 : 정규표현식으로 email 주소 형식 조건문 표현
- 164 - 171 : public saveNewFriendListFile method 정의 (friendList file을 다시 작성하는 method)

### 3.2.3.2. Point of view

1. 파일 입출력을 구현하였다.
2. 파싱 기능을 구현하였다.
3. 파일을 list로 바꾸는 기능을 구현하였다.
4. 옮기지 않은 입력 형식의 예외처리 기능을 구현하였다.
  - 4.1. 이름 충돌 (FriendList class에서 구현)

- 4.2. Group 형식
  - 4.3. 전화번호형식
  - 4.4. Email 형식
5. 복잡한 조건문을 피하기 위해 정규표현식 [1] 을 사용하였다.
- 전화번호 정규표현식을 설명하자면 다음과 같다.  
^000-000(3개-4개)-000(3개-4개)\$ : ^[0-9]{3}+\\-[0-9]{3,4}+\\-[0-9]{3,4}\$
  - 이메일 정규표현식을 설명하자면 다음과 같다.  
^ID부+@+HOST부\$
    - 1. ID부 표현 : [[\_a-z0-9]+(.[\_a-z0-9-]+)\*
    - 2. @
    - 3. HOST부 표현 : (?:\\w+\\.\\.)+\\w+

### 3.2.4. FriendListFrame class

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import java.io.IOException;
4 import java.util.ArrayList;
5 import javax.swing.*;
6
7 public class FriendListFrame extends JFrame {
8
9     FriendList friendList;
10    FriendListFile friendListFile = new FriendListFile();
11
12    private ArrayList<FriendListInformationPanel> friendListInformationPanelList = new Ar-
13 rayList<FriendListInformationPanel>();
14
15    private JPanel menuPanel = new JPanel();
16    private JPanel labelPanel = new JPanel();
17    private JPanel checkBoxPanel = new JPanel();
18    private JPanel friendListInformationPanel = new JPanel();
19
20    private JButton add = new JButton("Add");
21    private JButton delete = new JButton("Delete");
22    private JButton modify = new JButton("Modify");
23    private JButton save = new JButton("Save");
24
25    public JButton getAddButton() {
26        return add;
27    }
28    public JButton getDeleteButton() {
29        return delete;
30    }
31    public JButton getModifyButton() {
32        return modify;
33    }
34    public JButton getSaveButton() {
35        return save;
36    }
37
38    public FriendListFrame(FriendList friendList) {
39
40        this.friendList = friendList;
41
42        this.setLayout(new BorderLayout());
43        this.setSize(1180, 410);
```

```

44
45     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46     this.setTitle("친구 목록");
47     this.addFriendListFrameButtons();
48     this.addFriendListFrameLabels();
49     this.addFriendListInformations();
50     this.setVisible(true);
51 }
52
53 public void updateFriendListFrame(Object frameObject) {
54     new FriendListFrame(this.friendList);
55 }
56
57 private void addFriendListInformations() {
58
59     for (int i = 0; i < friendList.numFriends(); i++) {
60         Friend friendInformation = friendList.getFriend(i);
61         friendListInformationPanelList.add(new FriendListInformationPanel(friendInfor
62 mation));
63         friendListInformationPanel.add(friendListInformationPanelList.get(i));
64     }
65
66     this.add(friendListInformationPanel,BorderLayout.CENTER);
67 }
68
69 public void addFriendInformaiton(Friend friendInformation) {
70
71     friendList.addFriendInformation(friendInformation);
72 }
73
74 private void addFriendListFrameButtons() {
75
76     menuPanel.setLayout(new FlowLayout());
77     menuPanel.setPreferredSize(new Dimension(60, 240));
78
79     menuPanel.add(add, BorderLayout.EAST);
80     add.setPreferredSize(new Dimension(60, 60));
81     add.addActionListener(new ButtonsActionListener(this));
82     menuPanel.add(delete, BorderLayout.EAST);
83     delete.setPreferredSize(new Dimension(60, 60));
84     delete.addActionListener(new ButtonsActionListener(this));
85     menuPanel.add(modify, BorderLayout.EAST);
86     modify.setPreferredSize(new Dimension(60, 60));
87 }
```

```

88     modify.addActionListener(new ButtonsActionListener(this));
89     menuPanel.add(save, BorderLayout.EAST);
90     save.setPreferredSize(new Dimension(60, 60));
91     save.addActionListener(new ButtonsActionListener(this));
92
93     this.add(menuPanel, BorderLayout.EAST);
94
95 }
96
97 public void addFriendListFrameLabels() {
98
99     labelPanel.setLayout(new FlowLayout(FlowLayout.RIGHT, 130, 5));
100
101    JLabel name = new JLabel("이름");
102    labelPanel.add(name);
103    name.setPreferredSize(new Dimension(23, 15));
104    JLabel group = new JLabel("그룹");
105    labelPanel.add(group);
106    group.setPreferredSize(new Dimension(48, 15));
107    JLabel phoneNumber = new JLabel("전화번호");
108    labelPanel.add(phoneNumber);
109    phoneNumber.setPreferredSize(new Dimension(140, 15));
110    JLabel emailAddress = new JLabel("Email");
111    labelPanel.add(emailAddress);
112    emailAddress.setPreferredSize(new Dimension(90, 15));
113    JLabel picture = new JLabel("사진");
114    labelPanel.add(picture);
115    picture.setPreferredSize(new Dimension(65, 15));
116
117    this.add(labelPanel, BorderLayout.NORTH);
118
119 }
120
121 private int checkedFriendInformation() {
122
123     for(int i = 0; i < friendListInformationPanelList.size(); i++) {
124         if (friendListInformationPanelList.get(i).getCheckBox().isSelected()) {
125             return i;
126         }
127     }
128     return -1;
129 }
130
131 protected void deleteButtonAction() {

```

```

132
133     if (checkedFriendInformation() != -1) {
134         friendList.removeFriend(checkedFriendInformation());
135         this.dispose();
136         this.updateFriendListFrame(this);
137     }
138 }
139
140 protected void modifyButtonAction() {
141
142     if (checkedFriendInformation() != -1) {
143         friendList.modifyFriendInformation(checkedFriendInformation(),
144             Integer.parseInt(friendListInformationPanelList.get(checkedFriendI
145 nformation()).getgroup().getText()),
146             friendListInformationPanelList.get(checkedFriendInformation()).get
147 phoneNumber().getText(),
148             friendListInformationPanelList.get(checkedFriendInformation()).get
149 emailAddress().getText());
150         this.dispose();
151         this.updateFriendListFrame(this);
152     }
153 }
154
155 protected void saveButtonAction() throws IOException {
156
157     friendListFile.saveNewFriendListFile(this.friendList);
158 }
159 }
160
161 class ButtonsActionListener implements ActionListener {
162
163     FriendListFrame mainViewFrame;
164
165     public ButtonsActionListener(FriendListFrame mainViewFrame) {
166         this.mainViewFrame = mainViewFrame;
167     }
168
169     @Override
170     public void actionPerformed(ActionEvent e) {
171         if (e.getSource().equals(mainViewFrame.getAddButton()))
172             new AddFriendInformationFrame(mainViewFrame);
173         else if (e.getSource().equals(mainViewFrame.getDeleteButton()))
174             mainViewFrame.deleteButtonAction();
175         else if (e.getSource().equals(mainViewFrame.getModifyButton()))

```

```

176         mainViewFrame.modifyButtonAction();
177     else if (e.getSource().equals(mainViewFrame.getSaveButton()))
178     {
179         try {
180             mainViewFrame.saveButtonAction();
181         } catch (IOException e1) {
182             e1.printStackTrace();
183         }
184     }

```

▲ Fig. 19. Source code of FriendListFrame class

#### 2.2.4.1. Analyzation of the source code

1 - 4 : 외부 class 추가

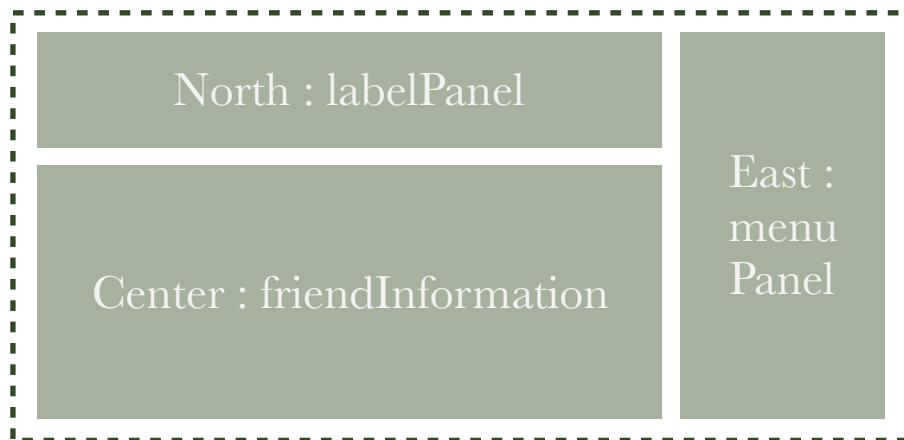
6 - 159 : FriendListFrame class (JFrame class 상속)

- 8 : FriendList 객체 선언 (addFriendListInformations method에서 객체 반환)
- 10 : FriendListFile 객체 생성
- 12 - 13 : FriendListInformationPanel class type ArrayList 선언
- 15 - 18 : JPanel 객체 생성
- 20 - 23 : JButton 객체 생성
- 25 - 36 : JButton getter method 정의
- 38 - 51 : FriendListFrame 객체 생성자 (파라미터 : friendList)
  - 40 : 생성자의 파라미터(friendList)를 FriendListFrame의 friendList에 저장
  - 42 : 레이아웃 설정 (border)
  - 43 : 화면 크기 설정
  - 45 : 화면 닫기 설정
  - 46 : 화면 title 설정
  - 47 : addFriendListFrameButtons method 호출  
(FriendListFrame에 buttons을 생성하는 method)
  - 48 : addFriendListFrameLabels method 호출  
(FriendListFrame에 label을 생성하는 method)
  - 49 : addFriendListInformations method 호출  
(FriendListFrame에 정보를 생성하는 method)
- 53 - 55 : updateFriendListFrame method 정의  
(Friend 정보가 update된 frame을 update하는 method)
- 57 - 67 : addFriendListInformations method 정의 (friendInformation panel을 생성하는 method)
  - 60 : friendList에서 friend 를 받아와 arrayList에 저장
  - 61 - 62 : FriendListInformationPanel 객체를 생성하여 arrayList에 저장
  - 63 : friendListInformationPanelList에 저장된 panel을 가져와 frame에 추가
- 69 - 73 : addFriendInformation method 정의  
(새로 추가된 Friend 정보를 friendList에 추가하는 method)
- 75 - 95 : addFriendListFrameButtons method 정의 (friendList frame에 button을 생성하는 method)
  - 77 : menuPanel 레이아웃 설정 (flow)
  - 78 : menuPanel 크기 설정
  - 80 - 82 : menuPanel에 add button 생성
  - 83 - 85 : menuPanel에 delete button 생성
  - 86 - 88 : menuPanel에 modify button 생성

- 89 - 91 : menuPanel에 save button 생성
- 93 : menuPanel을 friendListFrame에 생성 (위치는 border의 East)
- 97 - 119 : addFriendListFrameLabels method 정의 (friendList frame에 label을 생성하는 method)
  - 99 : labelPanel 레이아웃 설정 (flow)
  - 101 - 103 : labelPanel에 name label 생성
  - 104 - 106 : labelPanel에 group label 생성
  - 107 - 109 : labelPanel에 phoneNumber label 생성
  - 110 - 112 : labelPanel에 emailAddress label 생성
  - 113 - 115 : labelPanel에 picture label 생성
  - 117 : labelPanel을 friendListFrame에 생성 (위치는 border의 North)
- 121 - 129 : checkedFriendInformation method 정의 (체크된 체크박스의 index를 반환하는 method)
- 131 - 138 : deleteButtonAction method 정의
  - (delete button을 눌렀을 때의 action을 수행하는 method)
  - 133 - 137 : 체크박스가 체크됐을 때 수행
    - 134 : friendList에서 index의 Friend 정보를 삭제
    - 135 : 화면 종료
    - 136 : 화면 update
- 140 - 153 : modifyButtonAction method 정의
  - (modify button을 눌렀을 때의 action을 수행하는 method)
  - 142 - 152 : 체크박스가 체크됐을 때 수행
    - 143 - 149 : friendList 객체의 modifyFriendInformation method 호출
      - (friend 정보를 수정하는 method)
    - 150 : 화면 종료
    - 151 : 화면 update
- 155 - 158 : saveButtonAction method 정의
  - (save button을 눌렀을 때의 action을 수행하는 method)
  - 157 : friendListFile 객체의 saveFriendListFile method 호출
    - (data file을 새로 만들어 덮어쓰는 method)
- 161 - 185 : ButtonActionListener class (ActionListener class 상속)
  - 163 : FriendListFrame 객체 선언 (생성자에서 객체 생성)
  - 165 - 167 : ButtonsActionListener 생성자 정의
  - 169 : Overriding
  - 170 - 184 : actionPerformed method 정의 (action을 수행하는 method)
    - 171 - 172 : add button을 눌렀을 때 수행
      - 172 : AddFriendInformationFrame 생성
    - 173 - 174 : delete button을 눌렀을 때 수행
      - 174 : mainViewFrame 객체의 deleteButtonAction method 호출
        - (delete button을 눌렀을 때의 action을 수행하는 method)
    - 175 - 176 : modify button을 눌렀을 때 수행
      - 176 : mainViewFrame 객체의 modifyButtonAction method 호출
        - (modify button을 눌렀을 때의 action을 수행하는 method)
    - 177 - 182 : save button을 눌렀을 때 수행
      - 178 - 182 : try-catch로 예외처리
        - 179 : mainViewFrame 객체의 saveButtonAction method 호출
          - (save button을 눌렀을 때의 action을 수행하는 method)

#### 2.2.4.2. Point of view

1. **friendListFrame**의 레이아웃은 다음과 같다.



▲ Fig. 20. Layout of FriendListFrame

### 3.2.5. FriendListInformationPanel class

```
1 import java.awt.*;
2 import javax.swing.*;
3 import javax.swing.border.Border;
4
5 public class FriendListInformationPanel extends JPanel {
6
7     private JLabel name = new JLabel();
8     private JTextField group = new JTextField();
9     private JTextField phoneNumber = new JTextField();
10    private JTextField emailAddress = new JTextField();
11    private JLabel picture = new JLabel();
12    private JCheckBox checkBox = new JCheckBox();
13
14    public JTextField getGroup() {
15        return group;
16    }
17
18    public JTextField getPhoneNumber() {
19        return phoneNumber;
20    }
21
22    public JTextField getEmailAddress() {
23        return emailAddress;
24    }
25
26    public JCheckBox getCheckBox() {
27        return checkBox;
28    }
29
30    private JPanel checkBoxPanel = new JPanel();
31    private JPanel informationPanel = new JPanel();
32
33    public FriendListInformationPanel(Friend friend) {
34
35        this.setLayout(new BorderLayout());
36        checkBoxPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
37        informationPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 95, 5));
38
39        checkBoxPanel.add(checkBox);
40
41        name.setText(friend.getName());
42        informationPanel.add(name);
43        name.setPreferredSize(new Dimension(80, 20));
44        name.setHorizontalAlignment(JTextField.CENTER);
45        group.setText(friend.getGroup() + "");
```

cs

```

44     informationPanel.add(group);
45     group.setPreferredSize(new Dimension(30, 20));
46     group.setHorizontalAlignment(JTextField.CENTER);
47     phoneNumber.setText(friend.getPhoneNumber());
48     informationPanel.add(phoneNumber);
49     phoneNumber.setPreferredSize(new Dimension(160, 20));
50     phoneNumber.setHorizontalAlignment(JTextField.CENTER);
51     emailAddress.setText(friend.getEmailAddress());
52     informationPanel.add(emailAddress);
53     emailAddress.setPreferredSize(new Dimension(180, 20));
54     emailAddress.setHorizontalAlignment(JTextField.CENTER);
55     picture.setText(friend.getPicture());
56     informationPanel.add(picture);
57     picture.setPreferredSize(new Dimension(60, 20));
58     picture.setHorizontalAlignment(JTextField.CENTER);
59
60     this.add(checkBoxPanel, BorderLayout.WEST);
61     this.add(informationPanel, BorderLayout.CENTER);
62 }
63 }
```

cs

▲ Fig. 21. Source code of FriendInformationPanel class

### 3.2.5.1. Analyzation of the source code

1 - 3 : 외부 class 추가

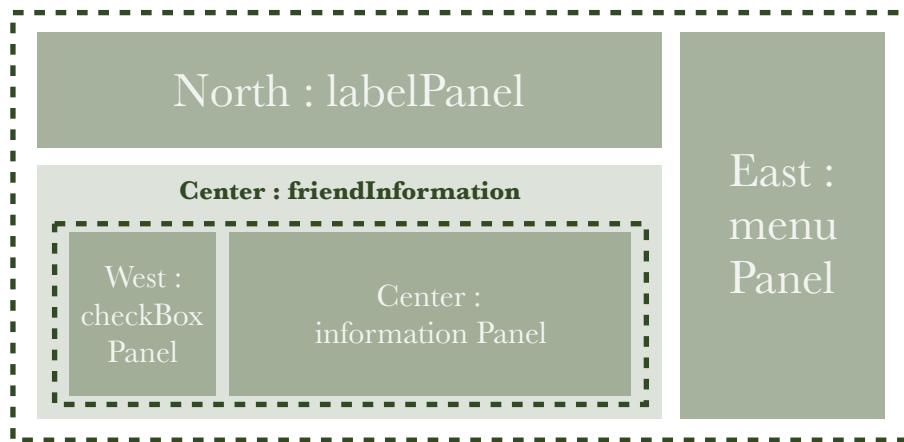
5 - 63 : FriendListInformationPanel class (JPanel class 상속)

- 7 : JLabel 객체 생성
- 8 - 10 : JTextField 객체 생성
- 11 : JLabel 객체 생성
- 12 : JCheckBox 객체 생성
- 14 - 25 : JTextField getter method 정의
- 27 - 28 : JPanel 객체 생성
- 30 - 62 : FriendListInformationPanel 객체 생성자(파라미터 : friend)
  - 32 : 전체 레이아웃 설정 (border)
  - 33 : checkBoxPanel 레이아웃 설정 (flow)
  - 34 : informationPanel 레이아웃 설정 (flow)
  - 36 : checkBoxPanel에 checkBox 생성
  - 39 - 42 : informationPanel에 name textField 생성 (textField Center 정렬)
  - 43 - 46 : informationPanel에 group textField 생성 (textField Center 정렬)
  - 47 - 50 : informationPanel에 phone number textField 생성 (textField Center 정렬)
  - 51 - 54 : informationPanel에 email address textField 생성 (textField Center 정렬)
  - 55 - 58 : informationPanel에 picture textField 생성 (textField Center 정렬)
  - 60 : checkBoxPanel을 friendListInformationPanel에 생성 (위치는 border의 West)

- 61 : informationPanel을 friendListInformationPanel에 생성 (위치는 border의 Center)

#### 2.2.5.2. Point of view

- 이를 **friendListFrame**에 적용하면 다음과 같다.



▲ Fig. 22. Layout of FriendListFrame with FriendListInformationPanel

- FriendListInformationPanel**의 레이아웃은 다음과 같다.



▲ Fig. 23. Layout of FriendListInformationPanel

### 3.2.6. AddFriendListInformationFrame class

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import java.io.IOException;
4
5 import javax.swing.*;
6
7 public class AddFriendInformationFrame extends JFrame {
8
9     private FriendListFrame mainViewFrame;
10
11    private JPanel menuPanel = new JPanel();
12    private JPanel labelPanel = new JPanel();
13    private JPanel inputTextFieldsPanel = new JPanel();
14
15    private JTextField name = new JTextField();
16    private JTextField group = new JTextField();
17    private JTextField phoneNumber = new JTextField();
18    private JTextField emailAddress = new JTextField();
19    private JTextField picture = new JTextField();
20
21    private JButton done = new JButton("Done");
22
23    public JTextField getNameTextField() {
24        return name;
25    }
26    public JTextField getGroupTextField() {
27        return group;
28    }
29    public JTextField getPhoneNumberTextField() {
30        return phoneNumber;
31    }
32    public JTextField getEmailAddressTextField() {
33        return emailAddress;
34    }
35    public JTextField getPictureTextField() {
36        return picture;
37    }
38
39    public AddFriendInformationFrame(FriendListFrame mainViewFrame) {
40
41        this.mainViewFrame = mainViewFrame;
42
43        this.setLayout(new BorderLayout());
```

cs

```
44     this.setSize(1020, 108);
45     this.setTitle("추가될 친구 정보");
46     this.addFriendInformationFrameButtons();
47     this.addFriendInformationFrameLabels();
48     this.addInputTextFields();
49     this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
50     this.setVisible(true);
51 }
52
53 private void addInputTextFields() {
54
55     inputTextFieldsPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 20));
56     inputTextFieldsPanel.setPreferredSize(new Dimension(960, 60));
57
58     inputTextFieldsPanel.add(name);
59     name.setPreferredSize(new Dimension(185, 20));
60     name.setHorizontalAlignment(JTextField.CENTER);
61     inputTextFieldsPanel.add(group);
62     group.setPreferredSize(new Dimension(185, 20));
63     group.setHorizontalAlignment(JTextField.CENTER);
64     inputTextFieldsPanel.add(phoneNumber);
65     phoneNumber.setPreferredSize(new Dimension(185, 20));
66     phoneNumber.setHorizontalAlignment(JTextField.CENTER);
67     inputTextFieldsPanel.add(emailAddress);
68     emailAddress.setPreferredSize(new Dimension(185, 20));
69     emailAddress.setHorizontalAlignment(JTextField.CENTER);
70     inputTextFieldsPanel.add(picture);
71     picture.setPreferredSize(new Dimension(185, 20));
72     picture.setHorizontalAlignment(JTextField.CENTER);
73
74     this.add(inputTextFieldsPanel, BorderLayout.CENTER);
75
76 }
77
78 private void addFriendInformationFrameButtons() {
79
80     menuPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
81     menuPanel.setPreferredSize(new Dimension(65, 60));
82
83     menuPanel.add(done);
84     done.setPreferredSize(new Dimension(60, 60));
85     done.addActionListener(new DoneActionListener(mainViewFrame, this));
86
87     this.add(menuPanel, BorderLayout.EAST);
```

cs

```
88     }
89
90     private void addFriendInformationFrameLabels() {
91
92         labelPanel.setLayout(new FlowLayout(FlowLayout.RIGHT,100,5));
93         labelPanel.setPreferredSize(new Dimension(960, 20));
94
95         JLabel name = new JLabel("이름");
96         name.setPreferredSize(new Dimension(90, 15));
97         labelPanel.add(name);
98         JLabel group = new JLabel("그룹");
99         group.setPreferredSize(new Dimension(85, 15));
100        labelPanel.add(group);
101        JLabel phoneNumber = new JLabel("전화번호");
102        phoneNumber.setPreferredSize(new Dimension(95, 15));
103        labelPanel.add(phoneNumber);
104        JLabel emailAddress = new JLabel("Email");
105        emailAddress.setPreferredSize(new Dimension(90, 15));
106        labelPanel.add(emailAddress);
107        JLabel picture = new JLabel("사진");
108        picture.setPreferredSize(new Dimension(75, 15));
109        labelPanel.add(picture);
110
111        this.add(labelPanel, BorderLayout.NORTH);
112    }
113}
114
115 class DoneActionListener implements ActionListener {
116
117     Friend newFriend = new Friend();
118     FriendList friendList = new FriendList();
119     FriendListFile friendListFile = new FriendListFile();
120     FriendListFrame mainViewFrame;
121     AddFriendInformationFrame addViewFrame;
122
123     public DoneActionListener(FriendListFrame mainViewFrame, AddFriendInformationFrame addViewFrame) {
124         this.mainViewFrame = mainViewFrame;
125         this.addViewFrame = addViewFrame;
126     }
127
128     @Override
129     public void actionPerformed(ActionEvent e) {
130         Friend inputInformation = new Friend();
```

cs

```

132
133     if (!
134         friendList.isConflictedName(addViewFrame.getNameTextField().getText())) {
135             inputInformation.setName(addViewFrame.getNameTextField().getText());
136             try {
137                 updateNewFriend();
138                 addViewFrame.dispose();
139                 mainViewFrame.dispose();
140                 mainViewFrame.updateFriendListFrame(mainViewFrame);
141             } catch (IOException e1) {
142                 e1.printStackTrace();
143             }
144         }
145     else
146         JOptionPane.showMessageDialog(null, "The name of the contact, " + addViewFrame.getNameTextField().getText() + ", is conflicted.");
147     }
148
149
150     private void updateNewFriend() throws IOException {
151         newFriend.setName(addViewFrame.getNameTextField().getText());
152         newFriend.setGroup(Integer.parseInt(addViewFrame.getGroupTextField().getText()));
153     };
154         newFriend.setPhoneNumber(addViewFrame.getPhoneNumberTextField().getText());
155         newFriend.setEmailAdress(addViewFrame.getEmailAddressTextField().getText());
156     ;
157         newFriend.setPicture(addViewFrame.getPictureTextField().getText());
158
159         mainViewFrame.addFriendInformation(newFriend);
160     }
161 }
```

CS

▲ Fig. 24. Source code of AddFriendListInformationFrame class

### 3.2.6.1. Analyzation of the source code

1 - 3 : 외부 class 추가

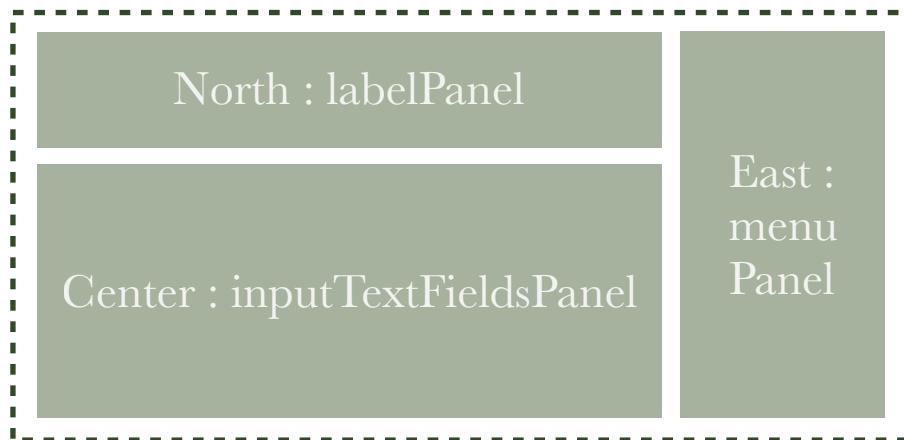
5 - 113 : AddFriendInformationFrame class (JPanel 상속)

- 9 : FriendListFrame 객체 선언 (생성자에서 객체 생성)
- 11 - 13 : JPanel 객체 생성
- 15 - 19 : JTextField 객체 생성
- 21 : JButton 객체 생성
- 23 - 37 : JTextField getter method 정의
- 19 - 51 : AddFriendInformationFrame 객체 생성자 (파라미터 : FriendListFrame)
  - 41 : FriendListFrame 객체 생성

- 43 : 레이아웃 설정 (border)
- 44 : 화면 크기 설정
- 45 : 화면 title 설정
- 46 : addFriendInformationFrameButtons method 호출  
(FriendInformationFrame에 button을 생성하는 method)
- 47 : addFriendInformationFrameLabels method 호출  
(FriendInformationFrame에 label을 생성하는 method)
- 48 : addInputTextFields method 호출  
(FriendInformationFrame에 textField를 생성하는 method)
- 49 : 화면 닫기 설정 (dispose)
- 53 - 76 : addInputTextFields method 정의  
(addFriendInformationFrame에 textField를 생성하는 method)
  - 55 : inputTextFieldsPanel 레이아웃 설정 (flow)
  - 58 - 60 : inputTextFieldsPanel에 name textPanel 생성 (textField Center 정렬)
  - 61 - 63 : inputTextFieldsPanel에 group textPanel 생성 (textField Center 정렬)
  - 64 - 66 : inputTextFieldsPanel에 phoneNumber textPanel 생성 (textField Center 정렬)
  - 67 - 69 : inputTextFieldsPanel에 emailAddress textPanel 생성 (textField Center 정렬)
  - 70 - 72 : inputTextFieldsPanel에 picture textPanel 생성 (textField Center 정렬)
  - 74 : inputTextFieldsPanel을 AddFriendInformationFrame에 생성 (위치는 border의 Center)
- 78 - 88 : addFriendInformationFrameButtons() method
  - 80 : menuPanel 레이아웃 설정 (flow)
  - 83 - 85 : menuPanel에 done button 생성
  - 87 : menuPanel을 addFriendInformationFrame에 생성 (위치는 border의 East)
- 90 - 112 : addFriendInformationFrameLabel() method
  - 92 : labelPanel 레이아웃 설정 (flow)
  - 95 - 97 : labelPanel에 name label 생성
  - 98 - 100 : labelPanel에 group label 생성
  - 101 - 103 : labelPanel에 phoneNumber label 생성
  - 104 - 106 : labelPanel에 emailAddress label 생성
  - 107 - 109 : labelPanel에 picture label 생성
  - 111 : labelPanel을 AddFriendInformationFrame에 생성 (위치는 border의 North)

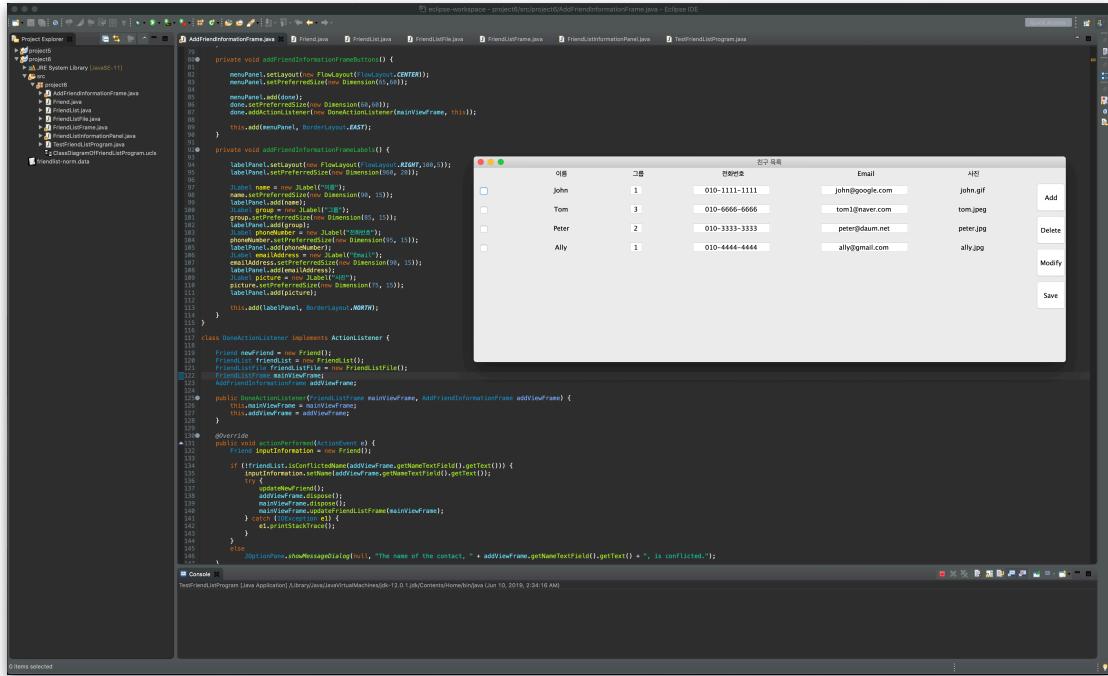
### 3.2.6.2. Point of view

#### 1. AddFriendListInformationFrame의 레이아웃은 다음과 같다.



▲ Fig. 25. Layout of AddFriendListInformationFrame

## 4. Program Results



▲ Fig. 26. Test in Eclipse 1

```

1 public class TestFriendListProgram {
2
3     public static void main(String[] args) {
4
5         FriendListFile friendListFile = new FriendListFile();
6         FriendList friendList = friendListFile.readFileToList("friendlist-norm.data");
7
8         friendList.printContactInformation();
9         new FriendListFrame(friendList);
10    }
11 }
```

▲ Fig. 27. Source code of the class TestFriendListProgram

TestFriendListProgram main class를 작성하여 프로그램을 친구 목록 관리 프로그램을 테스트 했다.

이때의 시행착오는 다음과 같다.

1. AddFriendInformationFrame의 “Done” button actionListener를 수행하는 과정에서 FriendListFrame Object가 넘겨지지 않았다.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - project6/src/project6/AddFriendInformationFrame.java - Eclipse IDE
- Left Sidebar:** Package Explorer showing projects project1 through project6.
- Code Editor:** The code for `AddFriendInformationFrame.java`. A specific line of code is highlighted with a red box:
 

```
    done.addActionListener(new DoneActionListener("FriendListFrame", his));
```
- Console Tab:** Displays the error message: `FriendListFrame cannot be resolved to a variable`.
- Status Bar:** Shows the file is Writable and the line count is 84 : 70.

▲ Fig. 28. Trial and error 1

이 시행착오는 public frame 혹은 object가 아닌 class를 넘겨주려고 시도했기 때문에 발생한 오류이다. 따라서 Fig. 29.와 같이 AddFrameInformationFrame class의 생성자에서 FriendListFrame 자신을 넘겨주도록 수정하였다.

The screenshot shows the Eclipse IDE interface with the following details:

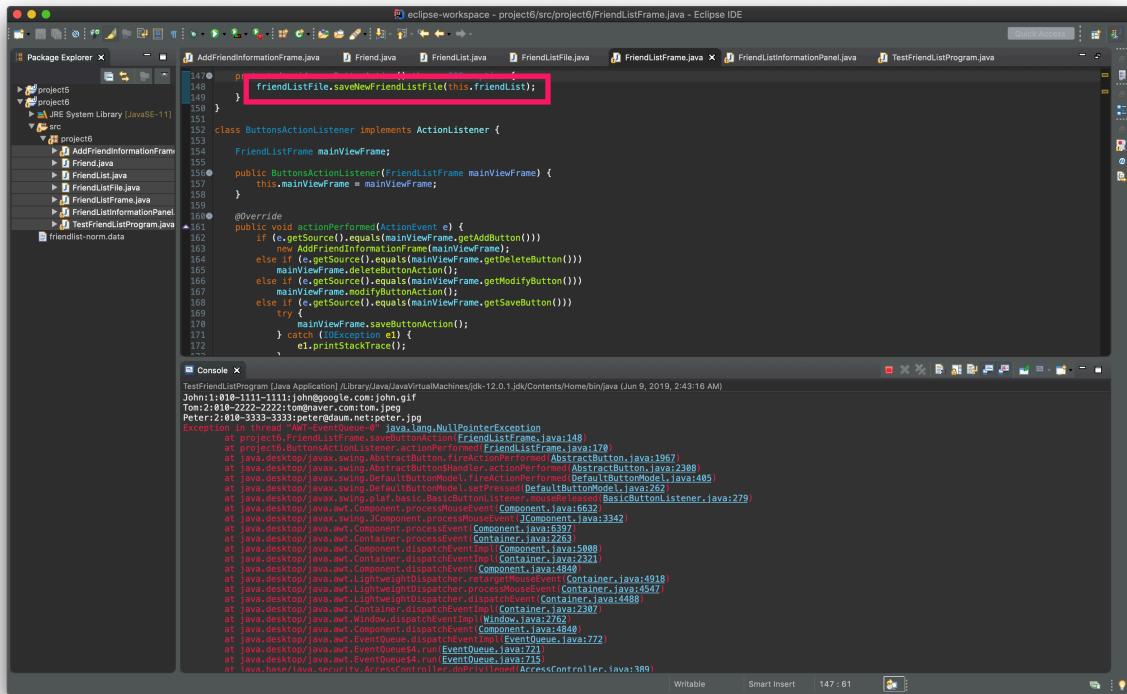
- Title Bar:** eclipse-workspace - project6/src/project6/AddFriendInformationFrame.java - Eclipse IDE
- Left Sidebar:** Package Explorer showing projects project1 through project6.
- Code Editor:** The code for `AddFriendInformationFrame.java`. A specific line of code is highlighted with a red box:
 

```
public AddFriendInformationFrame(FriendListFrame mainViewFrame) {
```
- Console Tab:** Displays the output of the program, showing three entries:
 

```
John:1:010-1111-1111:john@google.com:john.gif
Tom:2:010-2222-2222:tom@naver.com:tom.jpeg
Peter:2:010-3333-3333:peter@daum.net:peter.jpg
```
- Status Bar:** Shows the file is Writable and the line count is 133 : 9.

▲ Fig. 29. Solution of the trial and error 1

## 2. “Save” button을 눌렀을 때, Object 할당이 안 된 component를 호출한다.



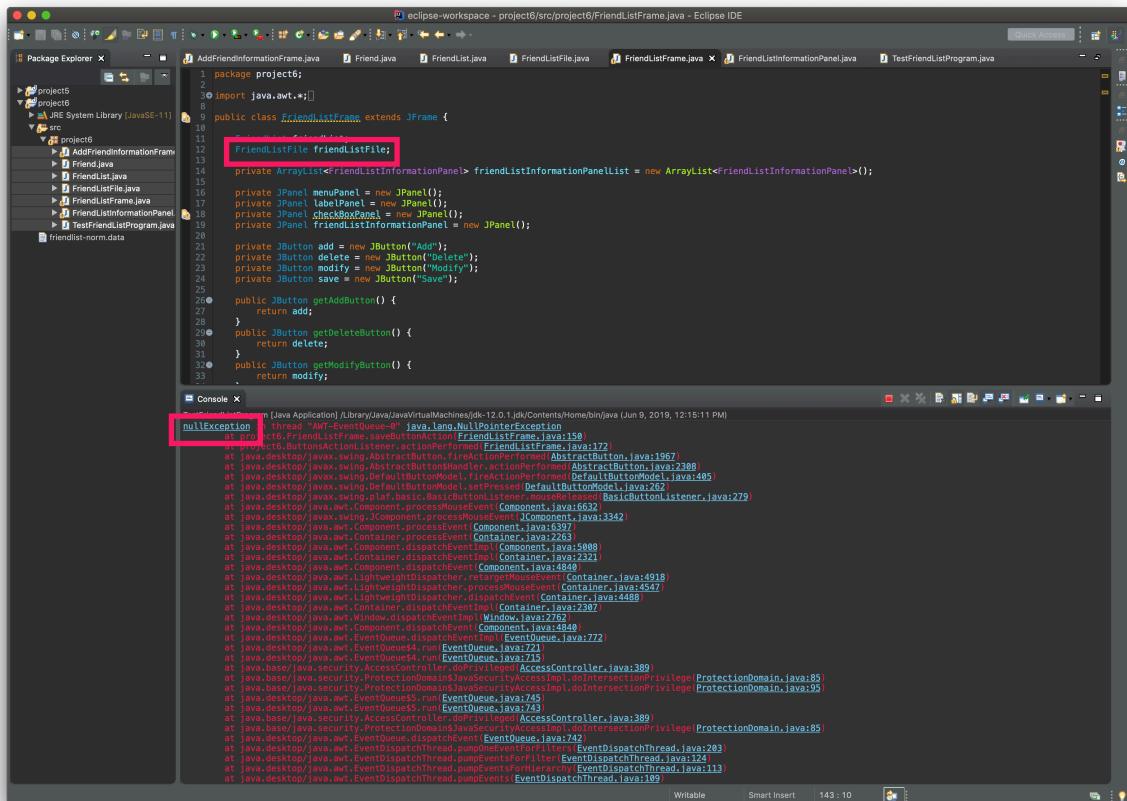
The screenshot shows the Eclipse IDE interface with the code editor open. The code is in `FriendListFrame.java`. A red box highlights the line `friendListFile.saveNewFriendListFile(this.friendList);`. The error message in the console is:

```

Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
at project6.ButtonsActionListener.actionPerformed(FriendListFrame.java:149)
at java.desktop/java.awt.AbstractButton.fireActionPerformed(AbstractButton.java:178)
at java.desktop/java.awt.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:238)
at java.desktop/java.awt.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
at java.desktop/java.awt.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
at java.desktop/java.awt.Component.processMouseEvent(Component.java:632)
at java.desktop/java.awt.Component.processEvent(Component.java:3342)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2008)
at java.desktop/java.awt.Container.dispatchEvent(Container.java:2321)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2321)
at java.desktop/java.awt.Container.dispatchEvent(Container.java:488)
at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Container.java:4918)
at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Container.java:4488)
at java.desktop/java.awt.Window.dispatchEventImpl(Window.java:2162)
at java.desktop/java.awt.Component.dispatchEvent(Component.java:4848)
at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:772)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:380)

```

▲ Fig. 30. Trial and error 2



The screenshot shows the Eclipse IDE interface with the code editor open. The code is in `FriendListFrame.java`. A red box highlights the line `nullException`. The error message in the console is:

```

nullException
at java.lang.Object.requireNonNull(Object.java:359)
at project6.ButtonsActionListener.actionPerformed(FriendListFrame.java:172)
at java.desktop/java.awt.AbstractButton.fireActionPerformed(AbstractButton.java:1967)
at java.desktop/java.awt.AbstractButtonHandler.actionPerformed(AbstractButton.java:2388)
at java.desktop/java.awt.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
at java.desktop/java.awt.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
at java.desktop/java.awt.Component.processMouseEvent(Component.java:632)
at java.desktop/java.awt.Component.processEvent(Component.java:3342)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2008)
at java.desktop/java.awt.Container.dispatchEvent(Container.java:2321)
at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2321)
at java.desktop/java.awt.Container.dispatchEvent(Container.java:488)
at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:772)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:380)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:95)
at java.desktop/java.awt.EventQueue$3.run(EventQueue.java:745)
at java.desktop/java.awt.EventQueue$3.run(EventQueue.java:743)
at java.base/java.security.AccessController.doPrivileged(AccessController.java:389)
at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:742)
at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:742)
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:124)
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:113)
at java.desktop/java.awt.EventQueue$EventDispatchThread.run(EventDispatchThread.java:199)

```

▲ Fig. 31. Reason of the trial and error 2 (1)

```

147     protected void saveButtonAction() throws IOException {
148         .....
149         friendListFile.saveNewFriendListFile(this, friendList);
150     }
151 }
152 class ButtonsActionListener implements ActionListener {
153     FriendListFrame mainViewFrame;
154     .....
155     public ButtonsActionListener(FriendListFrame mainViewFrame) {
156         this.mainViewFrame = mainViewFrame;
157     }
158     @Override
159     public void actionPerformed(ActionEvent e) {
160         if (e.getSource().equals(mainViewFrame.getAddButton()))
161             mainViewFrame.addButtonAction();
162         else if (e.getSource().equals(mainViewFrame.getDeleteButton()))
163             mainViewFrame.deleteButtonAction();
164         else if (e.getSource().equals(mainViewFrame.getModifyButton()))
165             mainViewFrame.modifyButtonAction();
166         else if (e.getSource().equals(mainViewFrame.getSaveButton()))
167             try {
168                 mainViewFrame.saveButtonAction();
169             } catch (IOException e1) {
170                 e1.printStackTrace();
171             }
172     }
173 }
174 }
175 }

```

Console output:

```

java.io.IOException: nullException
        at java.lang.Thread.run([Java Application] /Library/Java/JavaVirtualMachines/jdk-12.0.1.jdk/Contents/Home/bin/java [Jun 9, 2019, 12:15:11 PM]

```

▲ Fig. 32. Reason of the trial and error 2 (2)

이 시행착오는 Fig. 32.의 디버깅 과정에서 Fig. 31.와 같이 생성되지 않은 객체를 호출하려고 시도했기 때문에 발생한 오류이다. 따라서 Fig. 33.과 같이 객체를 생성하여 해결했다.

```

1 package project6;
2 import java.awt.*;
3 
4 public class FriendListFrame extends JFrame {
5     .....
6     friendListFile friendListFile = new FriendListFile();
7 
8     private ArrayList<FriendListInformationPanel> friendListInformationPanelList = new ArrayList<FriendListInformationPanel>();
9 
10    .....
11 }
12 
13 public JButton getAddButton() {
14     return add;
15 }
16 public JButton getDeleteButton() {
17     return delete;
18 }
19 public JButton getModifyButton() {
20     return modify;
21 }
22 public JButton getSaveButton() {
23     return save;
24 }
25 }
26 
27 public JButton getAddButton() {
28     return add;
29 }
30 public JButton getDeleteButton() {
31     return delete;
32 }
33 public JButton getModifyButton() {
34     return modify;
35 }
36 public JButton getSaveButton() {
37     return save;
38 }

```

Console output:

```

TestFriendListProgram [Java Application] /Library/Java/JavaVirtualMachines/jdk-12.0.1.jdk/Contents/Home/bin/java [Jun 9, 2019, 12:26:40 PM]
java.io.FileWriter@69d9e2d9

```

▲ Fig. 33. Solution of the trial and error 2

## 5. Conclusion

이 프로젝트에서는 프로젝트 5의 GUI에 이벤트 핸들링을 추가해서 친구목록 관리 프로그램을 완성했다.

이 프로젝트의 추진 방법은 다음과 같다.

### 1. 서로 독립적인 Class들 사이에서 자료를 공유하는 방법 연구

1.1. **JFrame**을 기반으로 하는 **Class** 복수개 생성

1.2. 그들 간의 정보 교환

(ex. “Add”의 경우, 입력창 **frame**에 입력한 내용이 메인 **frame**으로 전달되어야 한다)

1.3. 프로그램 구조의 틀을 유지하면서 정보 교환

### 2. 화면 내용 수정의 경우, 화면의 재정비 필요

(ex. “Delete” 작업)

### 3. 프로그램 내부에 있는 친구 DB(friend list)와 화면에 보여지는 내용은 항상 일치

### 4. 위의 두가지 문제를 동시에 해결

4.1. **JTable** 사용 불가

4.2. ex. 매 작업마다 두 내용을 동기화, 화면 갱신

이 프로젝트를 진행하면서 panel 생성과 레이아웃 설정과 관련한 시행착오를 통해 독립적인 Class들 사이에서 자료를 공유하는 방법을 이해할 수 있었다.

따라서 친구목록 관리 프로그램에 있어서 고찰점은 다음과 같다.

### 1. 객체 생성에 유의해야 한다.

## 6. References

[1] Vasili, “8 Regular Expressions You Should Know”, <https://code.tutsplus.com/tutorials/8-regular-expressions-you-should-know--net-6149>

[2] 남궁 성(2008). Java의 정석. 도우출판

## 7. Evaluation

평가 항목	학생 자체 평가 (리포트 해당 부분 표시 및 간단한 의견)	평가 (빈칸)	점수 (빈칸)
완성도 (동작 여부) - “초기” 동작 - “Add” 동작 - “Modify” 동작 - “Delete” 동작 - “Save” 동작 - 기타 비정상 동작 실험 (구현 한계 점검)	<ul style="list-style-type: none"> <li>- “초기”에 화면이 적절하게 생성된다.</li> <li>- “Add” 버튼 클릭시 추가 창이 적절하게 생성된다.</li> <li>- “Modify” 버튼 클릭시 수정사항이 변경된 화면이 적절하게 생성된다.</li> <li>- “Delete” 버튼 클릭시 선택된 정보가 삭제된 화면이 적절하게 생성된다.</li> <li>- “Save” 버튼 클릭시 수정된 data file이 생성되며 저장된다.</li> <li>- 비정상 동작은 배제한다는 조건에 따라 고려하지 않았다.</li> </ul>		
설계 노트 - 주요 결정사항 및 근거 - 한계/문제점 - 해결 방안 - 필수내용 : * 프로그램 구성(Class 구조) * member visibility	<ul style="list-style-type: none"> <li>- Class diagram 사진을 첨부하였다. (visibility를 diagram에 표현하였다.)</li> <li>- 설계 시 착안 사항을 모두 만족했다.</li> <li>- 시행 착오 내용을 사진을 첨부하여 설명하고 해결방안을 서술하였다.</li> </ul>		
리포트 - 평가자 시각으로 리포트 검토 - 위의 평가 요소들이 명확하게 기술되었는가?	<ul style="list-style-type: none"> <li>- 평가자 시각에서 소스코드와 구조, 분석이 명확하게 작성하였다.</li> <li>- 위의 평가 요소들이 명확하게 기술되었다.</li> <li>- github 로그 (<a href="https://github.com/AlliyHyeeseongKim/softwareProjectJava/tree/master/project6">https://github.com/AlliyHyeeseongKim/softwareProjectJava/tree/master/project6</a>)</li> </ul>		
총평 / 계	평가 항목을 모두 만족하였으며, 직접 프로그래밍하였음을 사진을 통해 입증하였다.		