

2019 소프트웨어프로젝트

Project 3

친구목록 프로그램 (*FriendsList*)



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. FriendListFile 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. Friend class
 - 3.2.3.1. Analyzation of the source code
 - 3.2.3.2. Point of view
 4. Program Results
 - 4.1. Problem of exception
 - 4.2. AutoTest
 5. Conclusion
 6. References
 7. Evaluation

1. Abstract

이 프로젝트에서는 카톡이나 전화 앱에 있는 친구목록 관리 프로그램을 작성했다. 단순히 입출력으로 보여주는 프로그램이 아닌, 친구 정보를 내부적으로 입력/저장 및 관리/출력하는 프로그램을 작성했다.
이는 향후 친구정보의 추가/수정/삭제로 확장이 용이하게 객체 지향 개념으로 설계하였다.

프로젝트의 차안점은 다음과 같다.

1. 파싱 기능을 구현하였다.
2. 옳지 않은 입력 형식의 예외처리 기능을 구현하였다.
3. 정규표현식을 사용하여 패턴을 만들어 예외처리를 하였다.
4. 객체 지향 개념으로 기능을 구현하였다.
5. contact 정보를 private으로 관리하였다.

이 paper에서 시행착오에 대한 해결점은 다음과 같다.

contact information을 저장하는 객체에 null을 반환하면 안된다. 따라서 exception이 아닌 경우만 객체를 생성함으로써 null 정보를 저장하는 오류를 해결하였다.

2. Introduction

이 프로젝트의 목적은 다음과 같다.

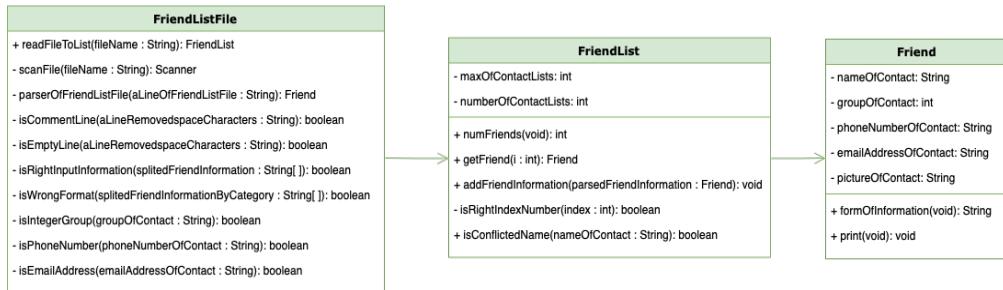
1. Java String 구조의 사용법을 연습한다.
2. File I/O의 사용을 연습한다.
3. 인터페이스 정의를 통한 프로그램의 배분/검사 등 분업화된 프로그래밍을 맛본다.

프로젝트의 내용은 다음과 같다.

1. 문제 정의에 입각해서 프로그램 구조를 설계한다. 즉, 사용할 클래스를 정의 한다.
 - 1.1. Friend class, FriendList class, FriendListFile class
2. 각 클래스에 따른 정보 기술 방법을 이해한다.
3. 정상 입력 파일에 대한 동작을 완성한다.
4. 비정상 입력 파일에 대응하도록 확장한다.
5. 자동 실행 검사 (AutoTest)이용해서 프로그램을 점검 한다.

3. Proposed Program

3.1. Class diagram



▲ Fig. 1. Class diagram

3.2. Analyzation of classes

3.2.1. FriendListFile class

```

1 import java.io.File;
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 FriendList readFileToList(String fileName) {
10         Scanner inputFile = scanFile(fileName);
11         if (inputFile != null) {
12             while (inputFile.hasNext()) {
13                 String aLineOfFriendListFile = inputFile.nextLine();
14                 Friend parsedFriendInformation = parseFriendListFile(aLineOfFriendListFile);
15                 if (parsedFriendInformation != null) {
16                     try {
17                         friendList.addFriendInformation(parsedFriendInformation);
18                     } catch (ArrayIndexOutOfBoundsException e) {
19                         System.out.println("The number of contacts is out of boundary.");
20                         System.out.println("The maximum number of contacts is 100.");
21                     }
22                 }
23             }
24             inputFile.close();
25             return friendList;
26         } else {
27             System.out.println("The file is empty.");
28             return friendList;
29         }
30     }
31
32     private Scanner scanFile(String fileName) {
33         try {
34             // make a file instance
35             File file = new File(fileName);
36             return new Scanner(file);
37         } catch (Exception e) {
38             // print error message
39             System.out.println(e.getMessage());

```

cs

▲ Fig. 2. Source code of the class FriendListFile 1

```

40     return null;
41 }
42 }
43 private Friend parseFriendListFile(String aLineOfFileFriendList) {
44     Friend parsedFriendInformation = null;
45     String originalLine = aLineOfFileFriendList;
46     // remove space characters
47     String aLineRemovedspaceCharacters = originalLine.replace(" ", "");
48     // start to parse the line if it is not an empty or a comment line
49     if (!isCommentLine(aLineRemovedspaceCharacters) && !isEmptyLine(aLineRemovedspaceCharacters)) {
50         String[] splitFriendInformationByCategory = aLineRemovedspaceCharacters.split(":");
51         if (isRightInputInformation(splitFriendInformationByCategory)) {
52             parsedFriendInformation = new Friend();
53             parsedFriendInformation.setGroupOfContact(Integer.parseInt(splitFriendInformationByCategory[1]));
54             parsedFriendInformation.setNameOfContact(splitFriendInformationByCategory[0]);
55             parsedFriendInformation.setPhoneNumberOfContact(splitFriendInformationByCategory[2]);
56             parsedFriendInformation.setEmailAddressOfContact(splitFriendInformationByCategory[3]);
57             if (splitFriendInformationByCategory.length == 5) {
58                 parsedFriendInformation.setPictureOfContact(splitFriendInformationByCategory[4]);
59             } else {
60                 if (aLineRemovedspaceCharacters.substring(aLineRemovedspaceCharacters.length() - 1,
61                     aLineRemovedspaceCharacters.length()) != ":") {
62                     System.out.println("The format of the input is wrong.");
63                 }
64             }
65         }
66         return parsedFriendInformation;
67     } else {
68         return null;
69     }
70 }
71 private boolean isCommentLine(String aLineRemovedspaceCharacters) {
72     if (aLineRemovedspaceCharacters.startsWith("//")) {
73         return true;
74     } else {
75         return false;
76     }
77 }
78 private boolean isEmptyLine(String aLineRemovedspaceCharacters) {
79     if (aLineRemovedspaceCharacters == null) {
80         return true;
81     } else {
82         return false;
83     }
84 }
85 private boolean isRightInputInformation(String[] splitFriendInformationByCategory) {
86     if (splitFriendInformationByCategory.length < 4 || splitFriendInformationByCategory.length > 5) {
87         System.out.println("At least one category is omitted.");
88         return false;
89     } else if (isWrongFormat(splitFriendInformationByCategory)) {
90         return false;
91     }
92     return true;
93 }

```

cs

▲ Fig. 3. Source code of the class FriendListFile 2

```

94     private boolean isWrongFormat(String[] splittedFriendInformationByCategory) {
95         if (friendList.isConflictedName(splittedFriendInformationByCategory[0]))
96             || !isIntegerGroup(splittedFriendInformationByCategory[1])
97             || !isPhoneNumber(splittedFriendInformationByCategory[2])
98             || !isEmailAddress(splittedFriendInformationByCategory[3])) {
99             return true;
100        } else {
101            return false;
102        }
103    }
104    private boolean isIntegerGroup(String groupOfContact) {
105        try {
106            Integer.parseInt(groupOfContact);
107        } catch (NumberFormatException e) {
108            System.out.println("The format of the group of the contact is not a integer.");
109            return false;
110        }
111        return true;
112    }
113    private boolean isPhoneNumber(String phoneNumberOfContact) {
114        if (Pattern.matches("^\\d{3}+\\d{4}+\\d{4}$", phoneNumberOfContact.trim())) {
115            return true;
116        } else {
117            return false;
118        }
119    }
120    private boolean isEmailAddress(String emailAddressOfContact) {
121        emailAddressOfContact = emailAddressOfContact.trim();
122        if (emailAddressOfContact == null) {
123            System.out.println("Email category is empty.");
124            return true;
125        } else if (Pattern.matches("^[a-zA-Z-]+([a-zA-Z-]+)*@[[:\w+\.\.]+\w+$",
126                emailAddressOfContact)) {
127            return true;
128        } else {
129            System.out.println("The format of the input of the email is wrong.");
130            return false;
131        }
132    }
133 }

```

▲ Fig. 4. Source code of the class FriendListFile 3

3.2.1.1. Analyzation of the source code

1-3 : 외부 클래스 추가

- 1 : File class
- 2 : Scanner class
- 3 : Pattern class

5-155 : FriendListFile class

- 7 : friendList 객체 생성
- 9-30 : public readFileToList method 정의
 - 10 : inputFile 객체 생성
 - 11: 빈 파일인지 검사

- 12-23 : 파일 한 줄씩 읽어오는 반복문
- 14 : 한 줄 파싱
- 16-21 : friendList 객체에 파싱된 friend contact 추가
 - 18-21 : friend contact 수가 maximum을 넘었을 경우 예외처리
- 24 : 파일 전체 파싱 끝난 후 파일 닫음
- 25 : 변경된 friendList 반환
- 26-29 : 빈 파일의 경우 예외처리
- 32-42 : private scanFile method 정의
 - 35 : filename의 이름을 가진 file 객체 생성
 - 36 : 성공시 readFileToList method의 inputFile 객체 생성
 - 37-41 : 실패시 예외처리
- 43-70 : private parseFriendListFile method 정의
 - 44 : 파싱된 contact 정보를 저장할 변수 선언
 - 47 : 공백 문자를 무시하기 위해 공백문자 제거
 - 48 : 빈줄이거나 “//”로 시작하는 comment를 제거하기 위한 조건문
 - 50 : 공백문자가 제거된 줄을 구분자(“:”)를 기준으로 항목을 잘라서 저장
 - 51-66 : contact 형식 검사
 - 50 : parsedFriendInformation 객체 생성 및 객체에 항목별로 저장
 - 60-63 : image 항목이 생략되었을 경우, 마지막 구분자(“:”)의 생략 예외처리
- 71-77 : private isCommentLine method 정의
 - 72 : “//”로 시작하는 경우 comment 처리
- 78-84 : private isEmptyLine method 정의
 - 79 : 빈줄
- 85-93 : private isRightInputInformation method 정의
 - 86-88 : 항목이 4개 혹은 5개가 아닐 경우
 - 89-91 : 입력된 형식이 잘못 되었을 경우
- 94-106 : private isWrongFormat method 정의
 - 95-98 : 같은 이름 충돌, group 항목 integer값, 전화번호 형식, 이메일주소 형식 체크
- 104-112 : private isIntegerGroup method 정의
 - 107-110 : 예외처리
- 113-119 : private isPhoneNumber method 정의
 - 114 : 정규표현식으로 전화번호 형식 조건문 표현
- 120-132 : private isEmailAddress method 정의
 - 122-124 : email 항목이 비었을 경우 예외처리
 - 125-127 : 정규표현식으로 email 주소 형식 조건문 표현

3.2.1.2. Point of view

1. 파일 입출력을 구현하였다.
2. 파싱 기능을 구현하였다.
3. 파일을 list로 바꾸는 기능을 구현하였다.
4. 옳지 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.

$$\begin{aligned} & ^{000-000(3개-4개)-000(3개-4개)}\$: ^{[0-9]\{3\}+\backslash\-[0-9]\{3,4\}+\backslash\-[0-9]\{3,4\}}\$ \end{aligned}$$

- 이메일 정규표현식을 설명하자면 다음과 같다.

^ID부+@+HOST부\$

1. ID부 표현 : [[_a-z0-9-]+(.[_a-z0-9-]+)*

2. @

3. HOST부 표현 : (?:\\w+\\.\\.)+\\w+

3.2.2. FriendList class

```
1 public class FriendList {  
2  
3     private int maxOfContactLists = 100;  
4     private int numberofContactLists = 0;  
5     private Friend[] contactInformation = new Friend[maxOfContactLists];  
6  
7     public int numFriends() {  
8  
9         return numberofContactLists;  
10    }  
11  
12    public Friend getFriend(int i) {  
13  
14        if (isRightIndexNumber(i)) {  
15            return contactInformation[i];  
16        }  
17        else {  
18            return null;  
19        }  
20    }  
21  
22    public void addFriendInformation(Friend parsedFriendInformation) {  
23        contactInformation[numberofContactLists++] = parsedFriendInformation;  
24    }  
25  
26    private boolean isRightIndexNumber(int index) {  
27        if (index >= 0 && index <= 100 && index <= numberofContactLists) {  
28            return true;  
29        }  
30        else {  
31            return false;  
32        }  
33    }  
34  
35    public boolean isConflictedName(String nameOfContact) {  
36        for (int i = 0; i < numberofContactLists; i++) {  
37            if (nameOfContact.equals(contactInformation[i].getNameOfContact())) {  
38                System.out.println("The name of the contact is conflicted.");  
39                return true;  
40            }  
41        }  
42        return false;  
43    }  
44}
```

cs

▲ Fig. 5. Source code of the class FriendList 1

3.2.2.1. Analyzation of the source code

1-48 : FriendList class

- 5 : contactInformation 배열 객체 생성
- 7-10 : public numFriends method 정의
- 12-20 : public getFriend method 정의
 - 14 : index의 형식이 옳은지 판단
- 22-24 : public addFriendInformation method 정의

- 23 : 파싱된 contact 정보를 contactInformation 배열 객체에 저장
- 26-33 : private isRightIndexNumber method 정의
 - 27 : index 범위 판단
- 35-43 : public isConflictedName method 정의
- 36-41 : 이전에 contactInformation 배열 객체에 저장된 이름에 같은 이름이 있는지 판단

3.2.2.2. Point of view

1. Friend class를 배열 객체로 생성하였다.
 - 1.1. contact 정보를 배열 객체에 저장하였다.
2. 배열 index 범위의 maximum을 100으로 설정하였다.
 - 2.1. 배열의 index 값 예외처리를 하였다.
3. 이름충돌 예외처리를 객체 지향 개념으로 구현하였다.

3.2.3. 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 setNameOfContact(String nameOfContact) {  
12        this.nameOfContact = nameOfContact;  
13    }  
14    public String getNameOfContact() {  
15        return nameOfContact;  
16    }  
17    public void setGroupOfContact(int groupOfContact) {  
18        this.groupOfContact = groupOfContact;  
19    }  
20    public int getGroupOfContact() {  
21        return groupOfContact;  
22    }  
23    public void setPhoneNumberOfContact(String phoneNumberOfContact) {  
24        this.phoneNumberOfContact = phoneNumberOfContact;  
25    }  
26    public String getPhoneNumberOfContact() {  
27        return phoneNumberOfContact;  
28    }  
29    public void setEmailAddressOfContact(String emailAddressOfContact) {  
30        this.emailAddressOfContact = emailAddressOfContact;  
31    }  
32    public String getEmailAddressOfContact() {  
33        return emailAddressOfContact;  
34    }  
35    public void setPictureOfContact(String pictureOfContact) {  
36        this.pictureOfContact = pictureOfContact;  
37    }  
38    public String getPictureOfContact() {  
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. 6. Source code of the class Friend 1

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 형식 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

4. Program Results

4.1. Problem of exception

```

1 package project3;
2
3 public class FriendList {
4
5     private int maxOfContactLists = 100;
6     private int numberOfContactLists = 0;
7     Friend[] contactInformation = new Friend[maxOfContactLists];
8
9     public int numFriends() {
10         return numberOfContactLists;
11     }
12
13     public Friend getFriend(int i) {
14
15         if (isRightIndexNumber(i)) {
16             return contactInformation[i];
17         } else {
18             return null;
19         }
20     }
21
22     public void printContactInformation() {
23
24         for (int i = 0; i < numberOfContactLists; i++) {
25             contactInformation[i].print();
26         }
27     }
28
29     public void addFriendInformation(Friend parsedFriendInformation) {
30
31         contactInformation[numberOfContactLists++] = parsedFriendInformation;
32     }
33
34     private boolean isRightIndexNumber(int index) {
35
36         if (index >= 0 && index <= 100 && index <= numberOfContactLists) {
37             return true;
38         }
39     }
40 }

```

Wed Apr 17 00:15:54 KST 2019 by 20185659
Case1: Normal Input
null:null:null
null:null:null
null:null:null
End of Case1
Case2: Check String Trimming
The result should be the same as the normal case above.
null:null:null
null:null:null
null:null:null
null:null:null
End of Case2
*** From now on, Test Reactions on Abnormal Inputs ***
Your Answer (your program's reaction) should be similar to the Correct Answer
Case3-0: Check ID Conflicts
Correct Answer: Name Conflict
Your Answer : null:null:null
null:null:null
null:null:null
null:null:null
End of Case3-0
Case3-1: Check Format Error ->
Correct Answer: Irregular input line , Skip the input line : Tom:2: 010-222-2222: tom@naver.com
Your Answer : null:null:null
null:null:null
null:null:null
null:null:null
End of Case3-1
Case3-2: Check Format Error ->
Correct Answer: Illegal value -- xxx : Skip the input line : Fred: xxx :010-1111-1111: fred@google.com :fred.gif
Your Answer : The format of the group of the contact is not a integer.
null:null:null
null:null:null
null:null:null
null:null:null
End of Case3-2
Case3-3: Check Format Error ->
Correct Answer: Illegal email address ; Skip the input line : Peter: 2 : 010-3333-333 : peter#daum.net : peter.jpg
Your Answer : The format of the input of the email is wrong.
null:null:null
null:null:null
Peter:2:010-3333-333:peter#daum.net:peter.jpg
End of Case3-3
Case4: Unknown Schedule File
Correct Answer: Unknown File
Your Answer : File name is unfound.
The file is empty.
End of Case4

▲ Fig. 7. Problem of exception in Eclipse 1

Wed Apr 17 00:15:54 KST 2019 by 20185659
Case1: Normal Input
null:null:null
null:null:null
null:null:null
End of Case1
Case2: Check String Trimming
The result should be the same as the normal case above.
null:null:null
null:null:null
null:null:null
null:null:null
End of Case2
*** From now on, Test Reactions on Abnormal Inputs ***
Your Answer (your program's reaction) should be similar to the Correct Answer
Case3-0: Check ID Conflicts
Correct Answer: Name Conflict
Your Answer : null:null:null
null:null:null
null:null:null
null:null:null
End of Case3-0
Case3-1: Check Format Error ->
Correct Answer: Irregular input line , Skip the input line : Tom:2: 010-222-2222: tom@naver.com
Your Answer : null:null:null
null:null:null
null:null:null
null:null:null
End of Case3-1
Case3-2: Check Format Error ->
Correct Answer: Illegal value -- xxx : Skip the input line : Fred: xxx :010-1111-1111: fred@google.com :fred.gif
Your Answer : The format of the group of the contact is not a integer.
null:null:null
null:null:null
null:null:null
null:null:null
End of Case3-2
Case3-3: Check Format Error ->
Correct Answer: Illegal email address ; Skip the input line : Peter: 2 : 010-3333-333 : peter#daum.net : peter.jpg
Your Answer : The format of the input of the email is wrong.
null:null:null
null:null:null
Peter:2:010-3333-333:peter#daum.net:peter.jpg
End of Case3-3
Case4: Unknown Schedule File
Correct Answer: Unknown File
Your Answer : File name is unfound.
The file is empty.
End of Case4

▲ Fig. 8. Problem of exception in AutoTest

AutoTest에서 exception 정보가 null로 저장이 되어 Eclipse에서 FriendList class의 printContactInformation method를 정의하여 확인해보았다. data파일에 잘못된 전화번호 형식을 가진 contact 하나를 추가해서 넣었다. 그 결과, exception으로 처리되는 contact가 null로 저장되어 출력됐다.

contact가 추가되는 addFriendInformation method가 호출되는 FriendListFile class를 분석해보니, 객체 생성 위치의 문제점이 있었다.

```

eclipse-workspace - project3/src/project3/FriendListFile.java - Eclipse IDE
45 }
46 }
47 }
48 private Friend parserOffFriendListFile(String aLineOffFriendListFile) {
49     Friend parsedFriendInformation = null;
50     String originalLine = aLineOffFriendListFile;
51     // remove space characters
52     String aLineRemovedSpaceCharacters = originalLine.replace(" ", "");
53     // start to parse the line if it is not empty or a comment line
54     if (!isCommentLine(aLineRemovedSpaceCharacters) && !isEmptyLine(aLineRemovedSpaceCharacters)) {
55         if (isRightInputInformation(aLineRemovedSpaceCharacters)) {
56             parsedFriendInformation = new Friend();
57             String[] splittedFriendInformationByCategory = aLineRemovedSpaceCharacters.split(":");
58             if (isRightInputInformation(splittedFriendInformationByCategory)) {
59                 parsedFriendInformation.setGroupOfContact(Integer.parseInt(splittedFriendInformationByCategory[0]));
60                 parsedFriendInformation.setNameOfContact(splittedFriendInformationByCategory[1]);
61                 parsedFriendInformation.setPhoneNumberOfContact(splittedFriendInformationByCategory[2]);
62                 parsedFriendInformation.setEmailAddressOfContact(splittedFriendInformationByCategory[3]);
63                 if (splittedFriendInformationByCategory.length == 5) {
64                     parsedFriendInformation.setPictureOfContact(splittedFriendInformationByCategory[4]);
65                 } else {
66                     if (aLineRemovedSpaceCharacters.substring(aLineRemovedSpaceCharacters.length() - 1,
67                         aLineRemovedSpaceCharacters.length()) != ":") {
68                         System.out.println("The format of the input is wrong.");
69                     }
70                 }
71             }
72             return parsedFriendInformation;
73         } else {
74             return null;
75         }
76     }
77 }
78 private boolean isCommentLine(String aLineRemovedSpaceCharacters) {
79     if (aLineRemovedSpaceCharacters.startsWith("//")) {
80         return true;
81     }

```

Console output:

```

<terminated> TestFriendListProgram [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Apr 17, 2019, 4:56:00 AM)
The format of the input of the phoneNumber is wrong.
Fred:1:010-1111-1111:fred@google.com:fred.gif
Tom:2:010-222-2222:tom@naver.com:tom.jpeg
Peter:2:010-333-333:peter@daum.net:peter.jpg
null:0:null:null>null

```

▲ Fig. 9. Problem of exception in Eclipse 2

Fig. 9. 을 보면, 56번째 line에서 객체를 생성한다. 즉, CommentLine^o나 EmptyLine^o 없으면 무조건 객체를 생성하게 된다. 따라서 또다른 exception을 처리해주는 **isRightInputInformation method**를 거치지 않아, **exception**의 경우 **null**인 객체를 반환하게 된다.

```

eclipse-workspace - project3/src/project3/FriendListFile.java - Eclipse IDE
45 }
46 }
47 }
48 private Friend parserOffFriendListFile(String aLineOffFriendListFile) {
49     Friend parsedFriendInformation = null;
50     String originalLine = aLineOffFriendListFile;
51     // remove space characters
52     String aLineRemovedSpaceCharacters = originalLine.replace(" ", "");
53     // start to parse the line if it is not empty or a comment line
54     if (!isCommentLine(aLineRemovedSpaceCharacters) && !isEmptyLine(aLineRemovedSpaceCharacters)) {
55         if (isRightInputInformation(aLineRemovedSpaceCharacters)) {
56             if (isRightInputInformation(splittedFriendInformationByCategory)) {
57                 parsedFriendInformation = new Friend();
58                 parsedFriendInformation.setGroupOfContact(Integer.parseInt(splittedFriendInformationByCategory[0]));
59                 parsedFriendInformation.setNameOfContact(splittedFriendInformationByCategory[1]);
60                 parsedFriendInformation.setPhoneNumberOfContact(splittedFriendInformationByCategory[2]);
61                 parsedFriendInformation.setEmailAddressOfContact(splittedFriendInformationByCategory[3]);
62                 if (splittedFriendInformationByCategory.length == 5) {
63                     parsedFriendInformation.setPictureOfContact(splittedFriendInformationByCategory[4]);
64                 } else {
65                     if (aLineRemovedSpaceCharacters.substring(aLineRemovedSpaceCharacters.length() - 1,
66                         aLineRemovedSpaceCharacters.length()) != ":") {
67                         System.out.println("The format of the input is wrong.");
68                     }
69                 }
70             }
71             return parsedFriendInformation;
72         } else {
73             return null;
74         }
75     }
76 }
77 }
78 private boolean isCommentLine(String aLineRemovedSpaceCharacters) {
79     if (aLineRemovedSpaceCharacters.startsWith("//")) {
80         return true;
81     }

```

Console output:

```

<terminated> TestFriendListProgram [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (Apr 17, 2019, 4:56:08 AM)
The format of the input of the phoneNumber is wrong.
Fred:1:010-1111-1111:fred@google.com:fred.gif
Tom:2:010-222-2222:tom@naver.com:tom.jpeg
Peter:2:010-333-333:peter@daum.net:peter.jpg

```

▲ Fig. 10. Program Result in Eclipse

따라서, **Fig. 9.**에서 58번째 line에서 객체를 생성하도록 수정했다. 그 결과, 의도한대로 exception^o null로 저장되지 않고 contact 저장을 건너뛰었다.

4.2. AutoTest

```
Wed Apr 17 05:05:34 KST 2019 by 20185659
Case1: Normal Input
Fred:0:010-1111-1111fred@google.com:fred.gif
Tom:2:010-222-2222tom@naver.com:tom.jpeg
Peter:2:010-3333-333:peter@daum.net:peter.jpg
End of Case1

Case2: Check String Trimming
The result should be the same as the normal case above.
Fred:0:010-1111-1111fred@google.com:fred.gif
Tom:2:010-222-2222tom@naver.com:tom.jpeg
Peter:2:010-3333-333:peter@daum.net:peter.jpg
End of Case2

*** From now on, Test Reactions on Abnormal Inputs ***
Your Answer (your program's reaction) should be similar to the Correct Answer

Case3-0: Check ID Conflicts
Correct Answer: Name Conflict
Your Answer : The name of the contact is conflicted.
Fred:0:010-1111-1111fred@google.com:fred.gif
Tom:2:010-222-2222tom@naver.com:tom.jpeg
End of Case3-0

Case3-1: Check Format Error ->
Correct Answer: Irregular input line : Skip the input line : Tom:2: 010-222-2222: tom@naver.com
Your Answer : The format of the input is wrong.
Fred:0:010-1111-1111fred@google.com:fred.gif
Tom:2:010-222-2222tom@naver.com:null
Peter:2:010-3333-333:peter@daum.net:peter.jpg
End of Case3-1

Case3-2: Check Format Error ->
Correct Answer: Illegal value -- xxx : Skip the input line : Fred: xxx:010-1111-1111: fred@google.com :fred.gif
Your Answer : The format of the group of the contact is not a integer.
Tom:2:010-222-2222tom@naver.com:tom.jpeg
Peter:2:010-3333-333:peter@daum.net:peter.jpg
End of Case3-2

Case3-3: Check Format Error ->
Correct Answer: Illegal email address : Skip the input line : Peter: 2 : 010-3333-333 : peter#daum.net : peter.jpg
Your Answer : The format of the input is wrong.
Fred:0:010-1111-1111fred@google.com:fred.gif
Tom:2:010-222-2222tom@naver.com:tom.jpeg
End of Case3-3

Case4: Unknown Schedule File
Correct Answer: Unknown File
Your Answer : File name is unfound.
The file is empty.
End of Case4
```

▲ Fig. 11. Program Result in AutoTest

5. Conclusion

이 프로젝트에서는 카톡이나 전화 앱에 있는 친구목록 관리 프로그램을 작성했다. 단순히 입출력으로 보여주는 프로그램이 아닌, 친구 정보를 내부적으로 입력/저장 및 관리/출력하는 프로그램을 작성했다. 이는 향후 친구정보의 추가/수정/삭제로 확장이 용이하게 객체 지향 개념으로 설계하였다.

프로젝트의 차안점은 다음과 같다.

1. 파싱 기능을 구현하였다.
2. 옳지 않은 입력 형식의 예외처리 기능을 구현하였다.
3. 정규표현식을 사용하여 패턴을 만들어 예외처리를 하였다.
4. 객체 지향 개념으로 기능을 구현하였다.
5. contact 정보를 private으로 관리하였다.

이 paper에서 시행착오에 대한 해결점은 다음과 같다.

contact information을 저장하는 객체에 **null**을 반환하면 안된다. 따라서 **exception**이 아닌 경우만 객체를 생성함으로써 **null** 정보를 저장하는 오류를 해결하였다.

6. References

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

7. Evaluation

평가 항목	학생 자체 평가 (리포트 해당 부분 표시 및 간단한 의견)	평가 (빈칸)	점수 (빈칸)
기본동작 - autotest 결과화면	<p>autoTest를 이용한 구현 검증</p> <ul style="list-style-type: none"> - 규정을 모두 준수하여 기본동작 검증 - autotest 결과 이미지 첨부 		
설계 사항 - 설계 착안점 - 사용한 클래스 - 시행 착오 및 해결점	<ul style="list-style-type: none"> - 설계 착안점을 각 class 코드 설명에 추가 - class 설계도 첨부 - Program results에 시행 착오와 해결점을 정리함 		
본인 인증	<ul style="list-style-type: none"> - github 로그 (https://github.com/AlliyHyeeseongKim/softwareProjectJava/tree/master/project3) - ellipse에서 Test class를 만들어 test한 이미지 첨부 - AutoTest 결과 이미지를 첨부 - 출처를 명확히 밝힘 		
기타	설계의 착안점을 정리하여 본인의 프로그램의 오류를 발견했고 이를 해결하는 과정을 잘 기술함		
총평/계	평가 항목을 모두 만족하였으며, 직접 프로그래밍하였음을 사진을 통해 입증함		