

12210795 김준호 컴퓨터공학과

프로젝트 주제 : Fandom Diary App (덕질 일기 앱)

개발 환경 : M1 MacOS, Eclipse - Java 17

프로젝트 구조 : app, frame, lib, listener, method, thread 총 6개의 패키지로 구성

main 함수 진입 파일 : FandomDiaryApp.java

디렉토리 구조

-diaries(일기 .txt 파일을 저장)

-images(일기의 이미지 .jpg 또는 .png 파일을 저장. 이후, 일기랑 함께 출력됨)

-public(프로그램 속 버튼 이미지, default 이미지, default 음악을 저장하는 폴더)

(1) 프로그램 목표

: “나의 아이돌(가수)의 덕질 일상을 기록하자!”

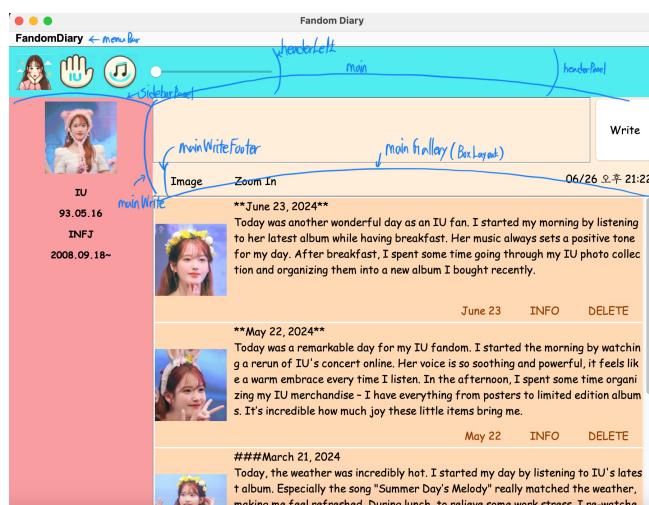
(2) 프로그램 사용 대상

-평소에 아이돌 사진을 많이 모으고 아이돌의 덕질 일상을 기록하는 사람

-내 스마트폰 갤러리에 아이돌 사진이 분류없이 저장되어 있고 한 곳에 모아서 보관하고 싶은 사람 누구나

(3) 프로그램 동작 과정

기본적으로 main함수가 존재하는 FandomDiaryApp.java 파일을 통해 프로그램이 시작된다.



src – method – Diary.java : 실제로 일기 작성시 읽기, 쓰기 관여하는 클래스

1. writeDiary()

정해진 형식에 맞춰서 내 프로젝트 ‘diaries’ 폴더에 텍스트를 txt 파일로 저장합니다.

```
public static void writeDiary(String fileNameFormatted, String userInput) {
    try {
        String filePath = "diaries/" + fileNameFormatted + ".txt";
        FileWriter fw = new FileWriter(filePath);
        BufferedWriter writer = new BufferedWriter(fw);
        if(userInput == null) {
            writer.write("");
        } else {
            writer.write(userInput);
        }
        writer.flush(); writer.close(); fw.close();
    } catch (IOException err) {
        err.printStackTrace();
    }
}
```

2. writeImage()

정해진 형식에 맞춰서 내 프로젝트 ‘images’ 폴더에 선택한 이미지를 저장합니다.

단, 이미지가 null(선택되지 않았으면)이면 default image를 복사해 저장합니다.

```
public static void writeImage(String fileNameFormatted, String srcPath) {
    if (srcPath != null) { //if image file is selected
        //get file extension
        int dotIndex = srcPath.lastIndexOf(".");
        String srcExtension = srcPath.substring(dotIndex);
        //set destPath
        String destPath = "images/" + fileNameFormatted + srcExtension;
        //copy Image
        Diary.copyImage(srcPath, destPath);
    } else { //if image file is not selected
        String destPath = "images/" + fileNameFormatted + ".jpg";
        Diary.copyImage("public/default_image.jpg", destPath);
    }
}
```

3. copyFile()

copyFile() 메서드는 이미지 또는 음악을 복사해 내 프로젝트 폴더에 복사할 때 사용되는 메서드입니다.

```
public static void copyFile(String srcPath, String destPath) {
    try {
        FileInputStream fis = new FileInputStream(srcPath);
        BufferedInputStream bis = new BufferedInputStream(fis);

        FileOutputStream fos = new FileOutputStream(destPath);
        BufferedOutputStream bos = new BufferedOutputStream(fos);

        byte[] buffer = new byte[1024 * 4]; //4KB buffer
        int c;
        while ((c = bis.read(buffer)) != -1) {
            bos.write(buffer, 0, c);
        }

        bos.flush(); bos.close(); fos.close(); bis.close(); fis.close();
    } catch (IOException err) {
        err.printStackTrace();
    }
}
```

4. `editTextFromFile()`, `editImageFromFile()`, `getTextFromFile()` : 일기의 ‘INFO’ 버튼을 클릭 후 일기의 텍스트 또는 이미지를 수정할 때 관여하는 메서드입니다. 이때, `getTextFromFile()` 메서드는 ‘INFO’ 버튼을 클릭할 때 ‘InfoAppDialog’의 `TextArea`에 잘려진 일기의 텍스트를 읽어올 때 사용되는 메서드입니다.

```

public static void editTextFromFile(Vector<String> diariesPath, String editedText, int currentPostIndex) {
    try {
        String filePath = diariesPath.get(currentPostIndex);
        FileWriter fw = new FileWriter(filePath);
        BufferedWriter writer = new BufferedWriter(fw);
        writer.write(editedText);
        writer.flush(); writer.close(); fw.close();
    } catch (IOException err) {
        err.printStackTrace();
    }
}

public static void editImageFromFile(Vector<String> imagesPath, String currentImagePath, int currentPostIndex) {
    Diary.copyFile(currentImagePath, imagesPath.get(currentPostIndex));
}

public static String getTextFromFile(Vector<String> diariesPath, int currentPostIndex) {
    StringBuffer sb = new StringBuffer();
    try {
        FileInputStream fis = new FileInputStream(diariesPath.get(currentPostIndex));
        InputStreamReader isr = new InputStreamReader(fis);
        BufferedReader br = new BufferedReader(isr);

        String line = null;
        while ((line = br.readLine()) != null) {
            sb.append(line + "\n");
        }
        br.close();
        isr.close();
        fis.close();
    } catch (IOException err) {
        err.printStackTrace();
    }
    return sb.toString();
}

```

src – method – FileDiary.java : 내부 동작 과정에 일기의 읽고 쓰기를 원활하게 도와주기 위해 만들어진 클래스

1. `getFilePath()`

‘diaries’, ‘images’ 폴더의 파일명을 읽어서 각 경로를 각각의 빅터에 저장하는 메서드

```

public static void getFilePath(Vector<String> diariesPath, Vector<String> imagesPath) {
    File dir = new File("diaries");
    File[] subFiles = dir.listFiles();
    for (File subFile : subFiles) {
        if (subFile.getName().equals(".DS_Store") || diariesPath.contains("diaries/" + subFile.getName())) {
            continue;
        }
        diariesPath.add("diaries/" + subFile.getName());
    }
    // https://reakwon.tistory.com/153
    Collections.sort(diariesPath);

    dir = new File("images");
    subFiles = dir.listFiles();
    for (File subFile : subFiles) {
        if (subFile.getName().equals(".DS_Store") || imagesPath.contains("images/" + subFile.getName())) {
            continue;
        }
        imagesPath.add("images/" + subFile.getName());
    }
    Collections.sort(imagesPath);
}

```

2. addTextToVector()

‘mainGallery’ 패널에 출력할 일기의 JTextArea를 만들고 벡터에 추가하는 클래스

```
// Make the text of the post JTextArea and save it as a vector.
public static void addTextsToVector(Vector<String> diariesPath, Vector<JTextArea> diariesJTextArea, int postIndex) {
    for(; postIndex < diariesPath.size(); postIndex++) {
        StringBuffer sb = new StringBuffer();
        try {
            FileInputStream fis = new FileInputStream(diariesPath.get(postIndex));
            InputStreamReader isr = new InputStreamReader(fis);
            BufferedReader br = new BufferedReader(isr);

            String line = null;
            while ((line = br.readLine()) != null) {
                sb.append(line + "\n");
            }
            br.close();
            isr.close();
            fis.close();
        } catch (IOException err) {
            err.printStackTrace();
        }
        int sbIndex = (sb.length() > 430) ? 430 : sb.length();
        JTextArea diary = new JTextArea(6, 20);
        diary.setText(sb.substring(0, sbIndex));
        diary.setBackground(new Color(255, 218, 185));
        diary.setFont(new Font("Comic Sans MS", Font.PLAIN, 15));
        diary.setRows(6);
        diary.setColumns(10);
        diary.setLineWrap(true);
        diary.setEditable(false);
        diary.setPreferredSize(new Dimension(100, 50));

        diariesJTextArea.add(diary);
    }
}
```

3. addImagesToVector()

‘mainGallery’ 패널에 출력할 일기의 Image를 만들고 썸네일의 크기를 조정해 벡터로 추가하는 클래스

```
public static void addImagesToVector(Vector<String> imagesPath, Vector<ButtonFilledWithImage> imagesBtns, int postIndex) {
    for(; postIndex < imagesPath.size(); postIndex++) {
        ButtonFilledWithImage temp = new ButtonFilledWithImage(imagesPath.get(postIndex), 100, 100);
        imagesBtns.add(temp);
    }
}
```

4. deleteTextAndImage()

일기를 ‘DELETE’하면 동작하는 메서드

```
public static void deleteTextAndImage(Vector<String> diariesPath, Vector<String> imagesPath, Vector<JTextArea> diariesJText
File deleteText = new File(diariesPath.get(deleteIndex));
File deleteImage = new File(imagesPath.get(deleteIndex));
diariesPath.remove(deleteIndex);
imagesPath.remove(deleteIndex);
diariesJTextArea.remove(deleteIndex);
imagesBtns.remove(deleteIndex);

if(deleteText.delete() && deleteImage.delete()) {
    System.out.println("Delete is success!");
}
else {
    System.out.println("Failed for delete.");
}
```

5. getFilePathSidebar(), addSideBarToVector(), editSideBar() : 사이드바의 텍스트를 읽고 내부에서 처리 용이하도록 가공하고 텍스트 수정이 필요할 때 사용되는 메서드

```
public static void getFilePathSidebar(Vector<String> sbLabelsPath) {
    File dir = new File("public/sidebar");
    File[] subFiles = dir.listFiles();
    for (File subfile : subFiles) {
        if (subfile.getName().equals(".DS_Store") || sbLabelsPath.contains("public/sidebar/" + subfile.getName())) { // If
            continue;
        }
        sbLabelsPath.add("public/sidebar/" + subfile.getName());
    }
    // https://reakwon.tistory.com/153
    Collections.sort(sbLabelsPath);
}
```

```

public static void addSideBarToVector(Vector<String> sbLabelsPath, Vector<JLabel> sbLabels) {
    for(int i = 0; i < sbLabelsPath.size(); i++) {
        StringBuffer sb = new StringBuffer();
        try {
            FileInputStream fis = new FileInputStream(sbLabelsPath.get(i));
            InputStreamReader isr = new InputStreamReader(fis);
            BufferedReader br = new BufferedReader(isr);

            String line = null;
            while ((line = br.readLine()) != null) {
                sb.append(line);
            }
            br.close();
            isr.close();
            fis.close();
        } catch (IOException err) {
            err.printStackTrace();
        }
        JLabel sbLabel = new JLabel(sb.toString());
        sbLabel.setFont(new Font("Comic Sans MS", Font.BOLD, 13));
        sbLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
        sbLabels.add(sbLabel);
    }
}

public static void editSideBar(String beforeText, String afterText, Vector<String> sbLabelsPath) {
    for(String sbLabelPath : sbLabelsPath) {
        StringBuffer sb = new StringBuffer();
        try {
            FileInputStream fis = new FileInputStream(sbLabelPath);
            InputStreamReader isr = new InputStreamReader(fis);
            BufferedReader br = new BufferedReader(isr);

            String line = null;
            while ((line = br.readLine()) != null) {
                sb.append(line);
            }
            br.close();
            isr.close();
            fis.close();
        } catch (IOException err) {
            err.printStackTrace();
        }

        // If you have found the index for the current label...
        if(sb.toString().equals(beforeText)) {
            try {
                FileWriter fw = new FileWriter(sbLabelPath);
                BufferedWriter writer = new BufferedWriter(fw);
                writer.write(afterText);
                writer.flush(); writer.close(); fw.close();
            } catch (IOException err) {
                err.printStackTrace();
            }
        }
    }
}

```

src – lib

1. ButtonFilledWithImage.java

버튼에 이미지가 여백 없이 설정되도록 도와주는 클래스

2. NonBorderButton.java

버튼의 경계를 없애고, 버튼이 클릭될 때 눌러지는 효과를 제거한 버튼 클래스

3. ResizedImageIcon.java

이미지가 width, height 입력에 맞게 크기가 조정되는 이미지 아이콘 클래스

src – listener – EditLabelListener.java

: 사이드바의 JLabel이 클릭되면 호출되는 마우스 리스너 클래스

src – thread – MusicThread.java

: 현재 음악의 위치에 맞게 사이드바가 움직이도록 만든 스레드 클래스

src – frame – DiaryEditFrame.java

: mainGallery 패널의 일기의 ‘INFO’ 버튼이 클릭되면 호출되는 JFrame 클래스

1. diaryImageButton 리스너 설명

이미지 버튼 클릭시 파일 창이 출력.

2. editDiary()

수정된 텍스트와 이미지를 실제로 파일을 수정해 반영하는 메서드

```
diaryImageButton.addActionListener(new ActionListener() {
    private JFileChooser chooser = new JFileChooser();
    private FileNameExtensionFilter imageFilter = new FileNameExtensionFilter("JPG & PNG", "jpg", "png");

    @Override
    public void actionPerformed(ActionEvent e) {
        chooser.setFileFilter(imageFilter);
        int result = chooser.showOpenDialog(null);
        if (result != JFileChooser.APPROVE_OPTION) {
            JOptionPane.showMessageDialog(null, "You didn't choose a image.", "Notice",
                JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        isSelectedImage = true;
        currentImagePath = chooser.getSelectedFile().getPath();
        diaryMainWriteArea.setFocusable(true);
        diaryMainWriteArea.requestFocus();
    }
});

setSize(600, 800);
setLocationRelativeTo(frame);
diaryMainWriteArea.setFocusable(true);
diaryMainWriteArea.requestFocus();
setVisible(true);
}

public void editDiary() {
    currentText = diaryMainWriteArea.getText();
    Diary.editTextFromFile(diariesPath, currentText, currentPostIndex);
    if (isSelectedImage) {
        Diary.editImageFromFile(imagesPath, currentImagePath, currentPostIndex);
    }
}
```

src – frame – DiaryFrame.java

: mainWrite 패널의 ‘ZOOM IN’ 버튼 클릭시 출력하는 JFrame 창이다.

1. diaryHeaderBtns[0] 리스너 설명

: ‘EXIT’ 버튼에 해당하는데, 창이 종료되어도 FandomDiaryApp의 JTextArea에서 이어쓰기 할 수 있도록 설계하였다.

2. diaryMainWriteArea 리스너 설명

현재 JTextArea에서 글자가 입력될 때마다 현재 총 글자수를 numOfCharsLabel에 출력되도록 설정하고 있다.

3. diaryHeaderBtns[1] 리스너 설명

‘WRITE’ 버튼에 해당하는데 writeDiary() 메서드를 사용하여 일기 파일을 작성 되도록 하고 있다.

```
// Listener
diaryHeaderBtns[0].addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        mainWriteArea.setText(userInput);
        DiaryFrame.this.dispose();
    }
});

diaryMainWriteArea.addKeyListener(new KeyAdapter() {
    @Override
    public void keyReleased(KeyEvent e) {
        numOfCharsLabel.setText(Integer.toString(diaryMainWriteArea.getText().length()));
        userInput = diaryMainWriteArea.getText();
    }
});

diaryHeaderBtns[1].addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        isClickedWrite = true;
        writeDiary();
        DiaryFrame.this.dispose();
    }
});
```

4. writeDiary() : 입력된 텍스트와 이미지를 실제로 작성하는 연산을 수행한다.

```
private void writeDiary() {
    String fileNameFormatted = now.format(DateTimeFormatter.ofPattern("MMdd_HHmm_ss_SS"));

    // Write Text and Image
    Diary.writeDiary(fileNameFormatted, userInput);
    Diary.writeImage(fileNameFormatted, srcPath);

    FileDiary.getFilePath(diariesPath, imagesPath);
    FileDiary.addTextsToVector(diariesPath, diariesJLabel, postIndex);
    FileDiary.addImagesToVector(imagesPath, imagesBtns, postIndex);

    userInput = "";
    srcPath = null;
}
```

src – frame – EditLabelDialog.java

: 사이드바 패널의 JLabel을 마우스로 클릭하면 뜨는ダイ얼로그 클래스이다.

1. dialogMainTa 리스너 설명

엔터키가 입력되면 텍스트 수정 여부를 판단한다.

이때, 입력된 텍스트 길이가 25자를 넘어서면 사이드바 패널의 영역을 넘어가기 때문에 글자수 제한을 걸어서 수행되도록 하고 있다.

```
dialogMainTa.addKeyListener(new KeyAdapter() {
    @Override
    public void keyReleased(KeyEvent e) {
        int keyCode = e.getKeyCode();
        if(keyCode == KeyEvent.VK_ENTER) {
            if(dialogMainTa.getText().length() >= 25) {
                JOptionPane.showMessageDialog(null, "It cannot exceed 25 characters.", "WARNING", JOptionPane.ERROR_MESSAGE);
            }
            else {
                // It does not include rewritten characters.
                String currentText = dialogMainTa.getText();
                int textSize = currentText.length();
                myLabel.setText(currentText.substring(0, textSize - 1));
                dispose();
            }
        }
    }
});
```

src – frame – InfoAppDialog.java

: 메뉴바의 ‘About FandomDiary...’ 클릭시 뜨는ダイ얼로그 창이다.

src – frame – SettingDialog.java

: 메뉴바의 ‘Settings’ 클릭시 뜨는ダイ얼로그 창이다. 이ダイ얼로그는 기본 이미지와 기본 음악을 설정 할 수 있도록 만들어진ダイ얼로그이다.

1. btn1, btn2 리스너 설명

각 버튼은 이미지 첨부 버튼과 음악 첨부 버튼을 의미한다.

기본 이미지와 기본 음악을 설정하기 위해서는 이미지와 음악 모두 첨부가 될 때만 가능하도록 코드를 작성하였다.

```
btn1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        chooserImage.setFileFilter(imageFilter);
        int result = chooserImage.showOpenDialog(null);
        if (result != JFileChooser.APPROVE_OPTION) {
            JOptionPane.showMessageDialog(null, "You didn't choose image.", "Notice",
                JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        srcImagePath = chooserImage.getSelectedFile().getPath();
        tf1.setText(srcImagePath);
        isSelectedImage = true;
    }
});
btn2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        chooserMusic.setFileFilter(musicFilter);
        int result = chooserMusic.showOpenDialog(null);
        if (result != JFileChooser.APPROVE_OPTION) {
            JOptionPane.showMessageDialog(null, "You didn't choose music.", "Notice",
                JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        srcMusicPath = chooserMusic.getSelectedFile().getPath();
        tf2.setText(srcMusicPath);
        isSelectedMusic = true;
    }
});
```

2. saveBtn 리스너 설명

‘Save’ 버튼을 의미한다. 이미지와 음악 모두 선택이 될 때만 변경 사항 저장을 하도록 하였다.

```
saveButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if(isSelectedImage == true && isSelectedMusic == true) {
            int result = JOptionPane.showConfirmDialog(null, "Do you want to save your changes?", "Really save?",
                JOptionPane.YES_NO_OPTION);
            if(result == JOptionPane.NO_OPTION) {
                return;
            }
            JOptionPane.showMessageDialog(null, "The music update is applied at the next run.", "INFORMATION", JOptionPane
                .DISPOSE_ON_CLOSE);
            Diary.copyFile(srcImagePath, "public/default_image.jpg");
            Diary.copyFile(srcMusicPath, "public/default_music.wav");
            isSucceed= true;
        } else {
            JOptionPane.showMessageDialog(null, "Please select an image and music file.", "WARNING", JOptionPane.ERROR
                .MESSAGE);
        }
    }
});
```

src – app – FandomDiary.java

: 이 프로그램의 main 함수가 존재하는 가장 기초가 되는 클래스이다.

1. FandomDiaryApp() 생성자

코드의 가독성을 위해, 각 패널 생성은 각 메서드를 통해 구현이 된다.

```
public FandomDiaryApp() {
    setTitle("Fandom Diary");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    createMenuBar();
    createHeaderPanel();
    createMainPanel();
    createSideBarPanel();

    setSize(900, 900);

    mainWriteArea.setFocusable(true);
    mainWriteArea.requestFocus();
    setLocationRelativeTo(null);
    setVisible(true);
}
```

2. createMenuBar() 메서드 속 menuItems[2] 리스너 설명

menuItems[2]는 ‘Quite’ 버튼을 의미한다.

프로그램 종료 조건은 현재 사용자가 텍스트를 작성한게 없거나 이미지를 선택하지 않았으면 정상 종료를 수행하게 된다.

```
menuItems[2].addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (isText || hasImage) {
            JOptionPane.showMessageDialog(FandomDiaryApp.this, "isTyping or hasImage!", "Quite",
                JOptionPane.ERROR_MESSAGE);
        } else {
            System.exit(0);
        }
    }
});
```

3. createHeaderPanel() 속 slider 리스너 설명

슬라이더는 현재 음악이 재생되고 있고 만약에 사용자가 슬라이더를 움직이고 있다면, 실시간으로 슬라이더가 위치한 값을 기반으로 음악의 현재 위치를 조정할 수 있도록 하고 있다. 참고로, 슬라이더의 범위는 0부터 100까지이다.

```
slider.addChangeListener(new ChangeListener() {
    @Override
    public void stateChanged(ChangeEvent e) {
        if (slider.getValueIsAdjusting() && clip != null && clip.isOpen() && clip.isRunning()) {
            int p = (int) (slider.getValue() * clip.getFrameLength() / 100.0);
            clip.setFramePosition(p);
        }
    }
});  
th.start();
```

4. createMainPanel() 속 writeZoomIn 리스너 설명

‘ZOOM IN’ 버튼 클릭하면 이벤트가 발생한다.

새로운 DiaryFrame 객체를 생성하여 큰 화면에서 일기를 작성하도록 도와준다. 이때, 이전에 JTextArea에 작성된 텍스트와 선택된 이미지가 있다면, 새로운 DiaryFrame에서 이어서 작성할 수 있도록 DiaryFrame 생성자의 인자로 전달하고 있다.

DiaryFrame의 창이 종료되었다면, 해당 DiaryFrame의 창에서 ‘EXIT’의 의한 종료인지, ‘WRITE’의 의한 종료인지 if 문으로 구별한 다음, 각 상황에 맞게 동작하도록 하고 있다. 만약, ‘EXIT’의 의한 종료였다면, 다시 JTextArea에서 이어서 작성할 수 있도록 도와주고 있고, 만약 ‘WRITE’의 의한 종료였다면, JTextArea 등의 여러 변수를 초기화하는 코드가 작성되어 있다.

```
writeZoomIn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        currentDiaryFrame = new DiaryFrame(FandomDiaryApp.this, "WRITE", mainWriteArea, diariesPath, imagePath,
                diariesJTextArea, imagesBttns, now, postIndex, srcPath);
        currentDiaryFrame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosed(WindowEvent we) {
                if (currentDiaryFrame.isClickedWrite()) {
                    // Initialize to default value
                    userInput = "";
                    srcPath = null;
                    isText = false;
                    hasImage = false;
                    mainWriteArea.setText(userInput);
                    // Update the current time.
                    now = LocalDateTime.now();
                    formattedNow = now.format(DateTimeFormatter.ofPattern("MM/dd a HH:mm"));
                    mainTimeLabel.setText(formattedNow);
                    // Update the current post.
                    postIndex = updatePosts(postIndexList, diariesJTextArea, diariesPath, imagesBttns, postPanel,
                            postIndex);
                } else {
                    userInput = currentDiaryFrame.getUserInput();
                    srcPath = currentDiaryFrame.getSrcPath();
                    isText = currentDiaryFrame.isTyping();
                    hasImage = currentDiaryFrame.hasImage();
                }
                repaint();
                revalidate();
            }
        });
    }
});
```

5. createSideBarPanel() 메서드 속 sbLabel 리스너 설명

sbLabel는 사이드바 패널 속 JLabel의 일부를 말하는 것이다.

해당 sbLabel를 클릭하게 되면 editLabelDialog 창이 뜨게 된다.

이때, editLabelDialog 창이 종료되면 편집 전 텍스트와 창이 종료된 후 텍스트를 비교해서 변경사항이 있다면 FileDiary 클래스의 editSideBar 메서드를 사용하여 반영도록 설계하였다.

```
// Add Listener to SideBarLabels
for (int i = 0; i < sbLabels.size(); i++) {
    JLabel sbLabel = sbLabels.get(i);
    sbLabel.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseReleased(MouseEvent e) {
            EditLabelDialog editLabelDialog = new EditLabelDialog(FandomDiaryApp.this, sbLabel);
            editLabelDialog.setVisible(true);
            editLabelDialog.addWindowListener(new WindowAdapter() {
                @Override
                public void windowClosed(WindowEvent we) {
                    String beforeText = editLabelDialog.getBeforeText();
                    String afterText = sbLabel.getText();
                    if(!beforeText.equals(afterText)) {
                        FileDiary.editSideBar(beforeText, afterText, sbLabelsPath);
                    }
                }
            });
            repaint();
            revalidate();
        }
    });
}
```

6. public int updatePosts() 메서드 설명

: 이 메서드는 일기가 만약에 추가 되었으면 mainGallery 패널에 일기를 추가하는 기능을 하는 메서드이다.

6 -1. infoBtn 리스너, currentDiaryEditFrame 리스너

infoBtn은 이미 작성된 일기를 수정할 때 사용되는 버튼이다.

이 리스너가 호출되면 DiaryEditFrame 창이 켜지면서 일기의 텍스트 또는 이미지를 수정할 수 있게 된다.

이때, 중요한 점은 mainGallery에 출력된 일기는 일기의 텍스트가 잘려서 나오지만, DiaryEditFrame 창에서는 전체 텍스트가 출력 되도록 설계하였다.

또한, currentDiaryEditFrame 리스너는 해당 프레임이 종료되면 일기의 텍스트 또는 이미지의 변경사항을 바로 mainGallery에 반영하도록 설계를 하였다.

```
// You can edit the post.
infoBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JPanel parentPanel = (JPanel) deleteBtn.getParent().getParent().getParent();
        currentPostIndex = -1;
        for (int i = 0; i < postIndexList.size(); i++) {
            if (postIndexList.get(i) == parentPanel) {
                currentPostIndex = i;
                break;
            }
        }
        if (currentPostIndex == -1) {
            return;
        }
        JTextArea currentPostTextArea = diariesJTextArea.get(currentPostIndex);
        String currentText = Diary.getTextFromFile(diariesPath, currentPostIndex);
        String currentImagePath = imagesPath.get(currentPostIndex);
        DiaryEditFrame currentDiaryEditFrame = new DiaryEditFrame(FandomDiaryApp.this, "WRITE", currentImagePath, currentPostIndex, diariesPath, imagesPath);
        currentDiaryEditFrame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosed(WindowEvent we) {
                JTextArea tempTextArea = diariesJTextArea.get(currentPostIndex);
                tempTextArea.setText(currentDiaryEditFrame.getCurrentText());
                ButtonFilledWithImage tempImageButton = imagesBtns.get(currentPostIndex);
                tempImageButton.setIcon(new ResizedImageIcon(currentImagePath, 100, 100));
                repaint();
                revalidate();
            }
        });
    }
});
```

6 – 2. deleteBtn 리스너 설명

deleteBtn은 각 일기 패널의 아래 있는 ‘DELETE’ 버튼을 의미한다.

우선, 어떤 게시글이 ‘DELETE’ 되는지 게시글의 번호를 찾기 위해 for 반복문을 수행하여 currentPostIndex에 저장하게 된다.

게시글의 번호를 찾았으면, postPanel 벡터와 postIndexList 링크드리스트에서 삭제하고, 실제로 폴더내 텍스트와 이미지 삭제 연산을 위해 FileDiary.deleteTextAndImage() 메서드를 호출하여 삭제한다.

```
// You can delete your post.
deleteBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int result = JOptionPane.showConfirmDialog(null, "Are you sure you want to delete it?",
            JOptionPane.YES_NO_OPTION);
        if (result == JOptionPane.NO_OPTION) {
            return;
        }
        JPanel parentPanel = (JPanel) deleteBtn.getParent().getParent().getParent();
        int currentPostIndex = -1;
        for (int i = 0; i < postIndexList.size(); i++) {
            if (postIndexList.get(i) == parentPanel) {
                currentPostIndex = i;
                break;
            }
        }
        if (currentPostIndex == -1) {
            return;
        }
        postPanel.remove(postIndexList.get(currentPostIndex));
        postIndexList.remove(currentPostIndex);
        postIndex--;
        FileDiary.deleteTextAndImage(diariesPath, imagesPath, diariesJTextArea, imagesBtns,
            currentPostIndex);

        repaint();
        revalidate();
    }
});
```

7. private class MusicButtonListener 설명

: 헤더 패널의 3개의 버튼에 이벤트를 정의하는 클래스이다.

```
private class MusicButtonListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        JButton temp = (JButton) e.getSource();
        if (temp == headerBtns[0]) {
            clip.start();
        } else if (temp == headerBtns[1]) {
            clip.stop();
        } else if (temp == headerBtns[2]) {
            clip setFramePosition(0);
            clip.start();
        }
    }
}
```