

## 객체지향언어2[8]

- 미니 프로젝트 결과 보고서 -

2171344 강민서

## 1. 작품 개요

### 1) 시작 화면 (StartFrame)

기본적인 화면 구성은 메뉴바, 배경레이블, 버튼이다. 메뉴바에 포함되는 것은 단어 파일 설정, 소리 설정이다. 메뉴바에 포함되는 것은 단어 파일 관련, 랭킹 관련, 소리 설정이다. 시작 화면에서 엔터를 누르면 플레이어의 정보를 입력받는 다이얼로그 창이 나타난다. 플레이어의 이름, 프로필 사진, 난이도, 단어 파일을 설정할 수 있다. 이후, 확인 버튼을 누르면 게임이 시작된다.

### 2) 게임 진행 화면 (GameFrame)

게임이 시작되면 3초 동안 단어를 타이핑 하기위한 준비 시간을 준다. 게임 시간은 총 1분(시연 영상에서는 시간 관계 상 45초로 진행하였습니다)이다. 1분이 지나면 자동으로 게임이 끝난다. 게임이 종료된다면 게임 종료 팝업 다이얼로그가 뜨고 점수 확인 화면으로 넘어갈 수 있다.

화면은 크게 3개의 부분(GamePanel, ProfileAndScorePanel, SettingPanel)으로 구성된다.

GamePanel 배경은 화면은 유저가 선택한 프로필 사진(캐릭터)에 따라 다르다. 단어들이 각각 다른 속도로 상단으로부터 하단으로 내려온다. 패널 상단에는 남은 게임 시간을 보여주는 타임레이블이 존재한다. 패널 하단에는 텍스트를 입력할 수 있는 영역이 존재하고, 유저들은 해당 영역에서 단어를 타이핑하여 Enter 키를 눌러 점수를 얻거나 잃을 수 있다. 단어를 맞추면 단어가 맞췄다는 것의 오디오가 출력하고, 단어가 틀렸으면 단어가 틀림의 오디오를 출력한다. 난이도에 따라 단어가 떨어지는 속도를 다르게 한다. 난이도가 낮다면 단어가 떨어지는 속도가 느리고, 얻는 점수도 낮다. 난이도에 따라 단어가 떨어지는 속도의 범위가 다르다.







































ProfileAndScorePanel에서는 유저의 이름, 유저가 선택한 프로필 사진, 점수를 보여준다. 유저가 단어를 맞췄는지 틀렸는지에 따라 프로필 사진의 표정이 바뀌도록 한다.

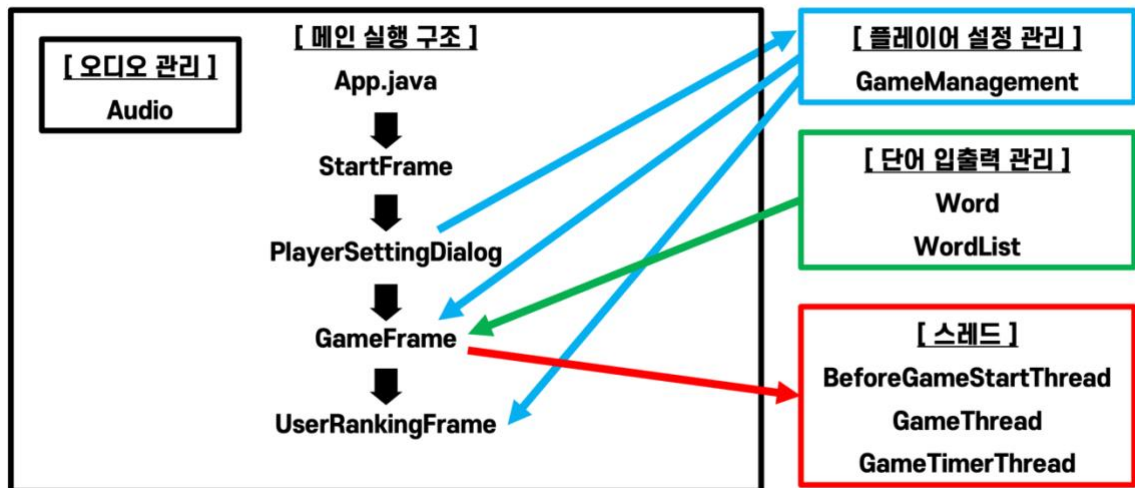
SettingPanel에는 게임 중지/재개 버튼, 오디오 중지 버튼, 오디오 재개 버튼이 존재한다. 게임 중지/재개 버튼을 이용해 게임 스레드와 오디오를 제어할 수 있다. 게임 중지 버튼이 눌리면 버튼의 텍스트를 바꾸고 오디오 버튼을 비활성화 한다.

### 3) 점수 확인 화면 (UserRankingFrame)

화면은 크게 2개의 부분(PlayResultPanel, RankingPanel)로 구성된다. PlayResultPanel에서는 플레이어의 정보와 최종 점수를 보여준다. RankingPanel에서는 난이도에 따른 랭킹을 확인할 수 있다. Enter를 누르면 프로그램이 종료된다.

## 2. 프로그램 구조

- ▼  src
  - ▼  (default package)
    - >  App.java
    - >  Audio.java
    - >  BeforeGameStartThread.java
    - >  GameFrame.java
    - >  GameManagement.java
    - >  GamePanel.java
    - >  GameThread.java
    - >  GameTimerThread.java
    - >  PlayerSettingDialog.java
    - >  PlayResultPanel.java
    - >  ProfileAndScorePanel.java
    - >  RankingPanel.java
    - >  SettingPanel.java
    - >  StartFrame.java
    - >  StartFrameCenterPanel.java
    - >  StartFrameNorthPanel.java
    - >  StartFrameSouthPanel.java
    - >  UserRankingFrame.java
    - >  Word.java
    - >  WordList.java
  - ▼  audio
    -  beforeGameStartAudio.wav
    -  correct.wav
    -  gameBackgroundAudio.wav
    -  gameEnded.wav
    -  mainStartAudio.wav
    -  wrong.wav
  - ▼  image
    - >  background
    - >  expression
    - >  main
    - >  resultProfile
    - >  settingProfile
  - ▼  txt
    -  toEICwords.txt
    -  words.txt



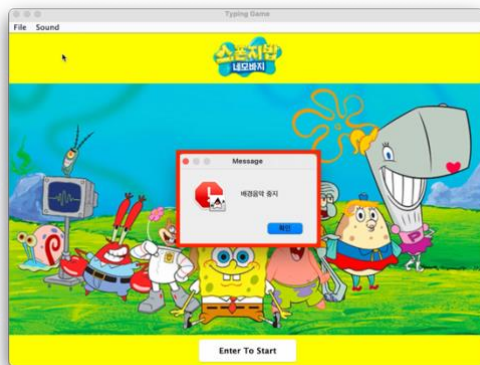
### 3. 프로그램 실행 과정

#### 1) 시작 프레임

Enter To Start 버튼에 포커스가 맞춰져 있다. 엔터를 누르면 플레이어 설정이 가능하다.

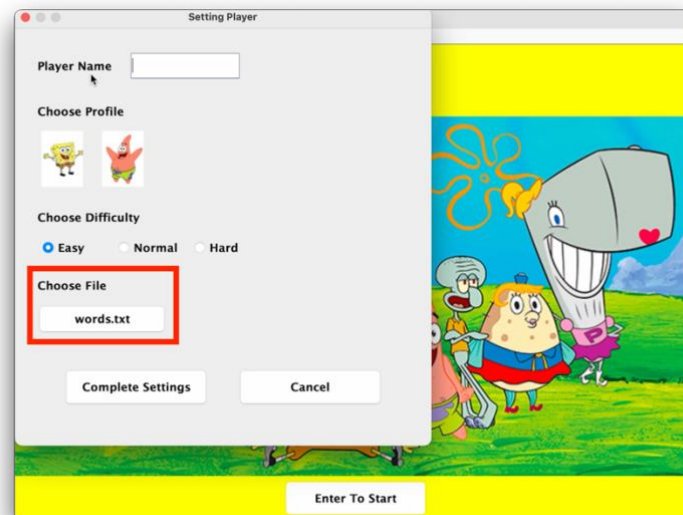


메뉴바에서 사용할 txt 파일 설정 및 배경 음악 제어가 가능하다. 파일이 선택되거나 소리 관련 설정이 변경된다면 팝업이 뜬다.

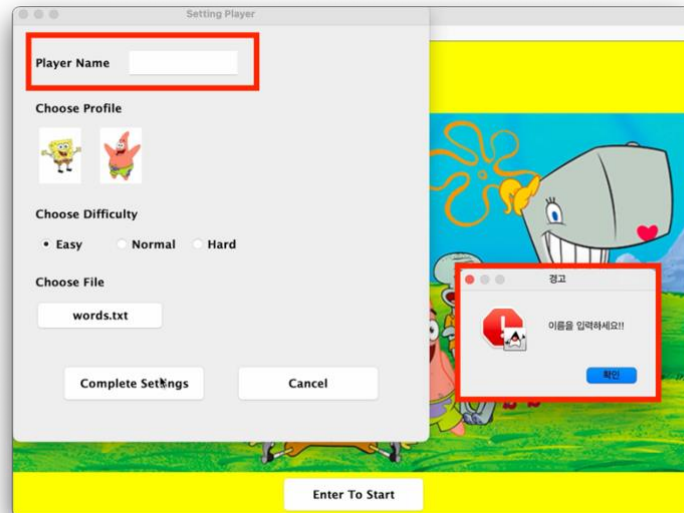


## 2) 플레이어 세팅 다이얼로그

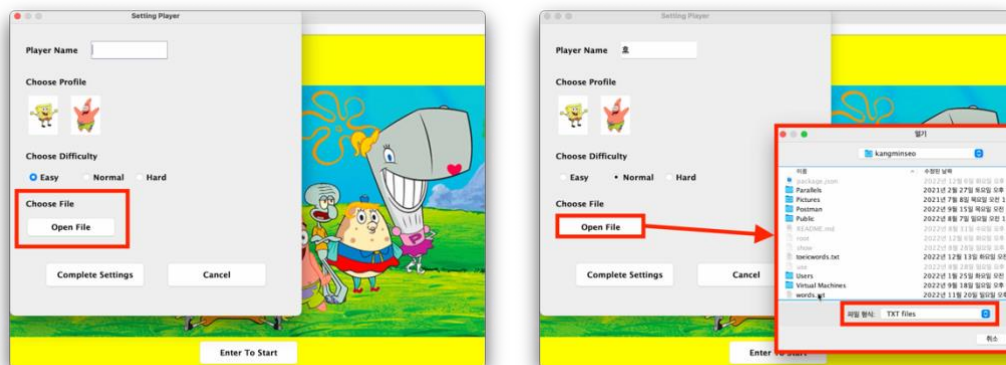
시작 프레임에서 파일을 선택한 경우 파일 선택 부분이 이미 설정 되어있다.



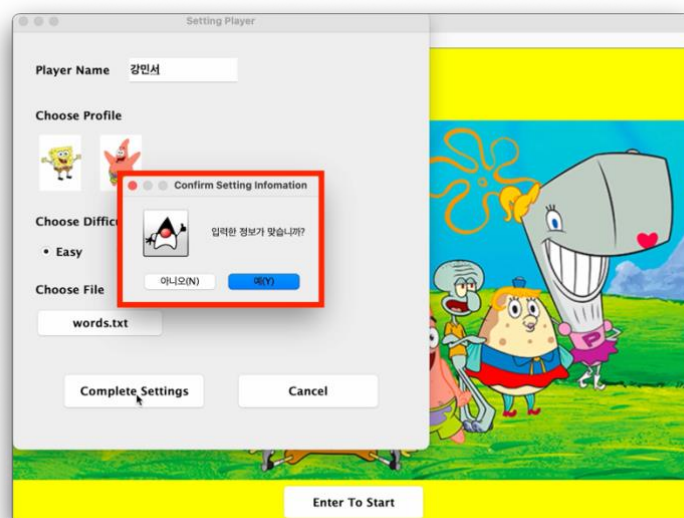
이름을 입력하지 않거나 파일을 선택하지 않으면 게임 시작이 불가능하다.



시작 프레임에서 파일을 선택하지 않은 경우 Open File 버튼을 눌러 파일을 설정할 수 있다.



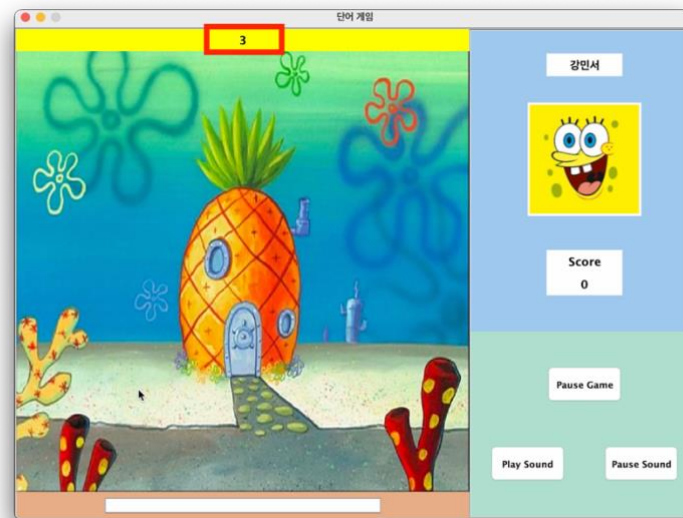
세팅을 완료하면 팝업 메시지를 띄우고 게임 프레임으로 넘어간다.



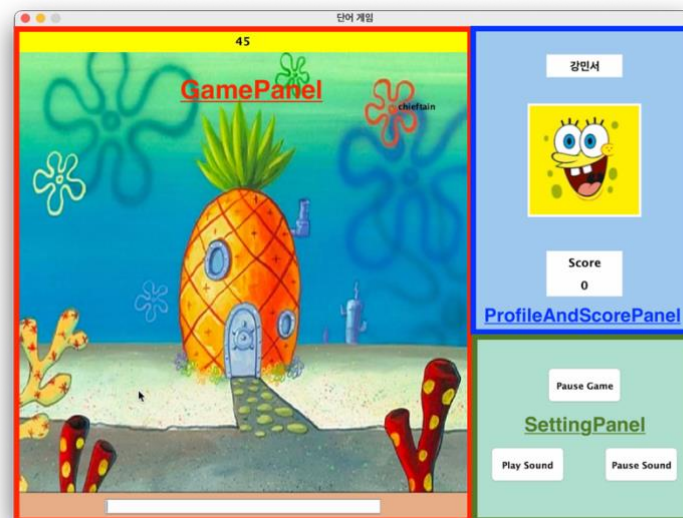


### 3) 게임 프레임

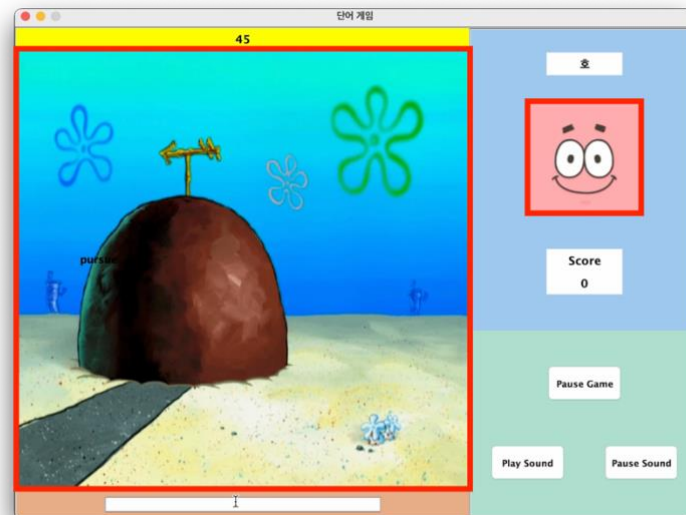
3초 카운트 다운 후 게임이 진행된다.



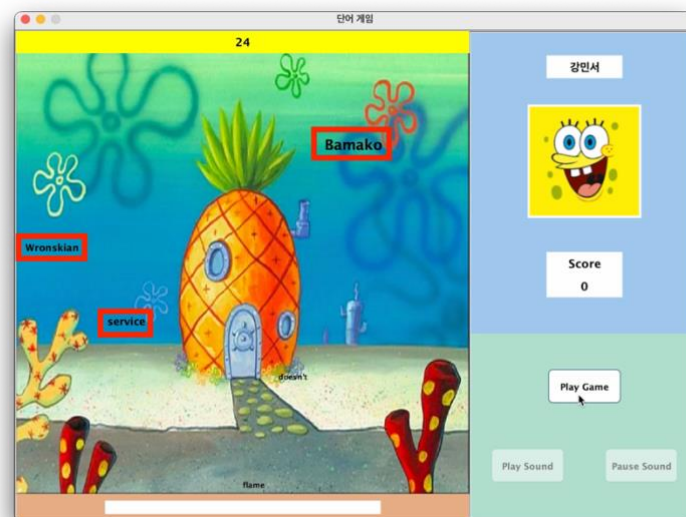
패널 구성은 다음과 같다.



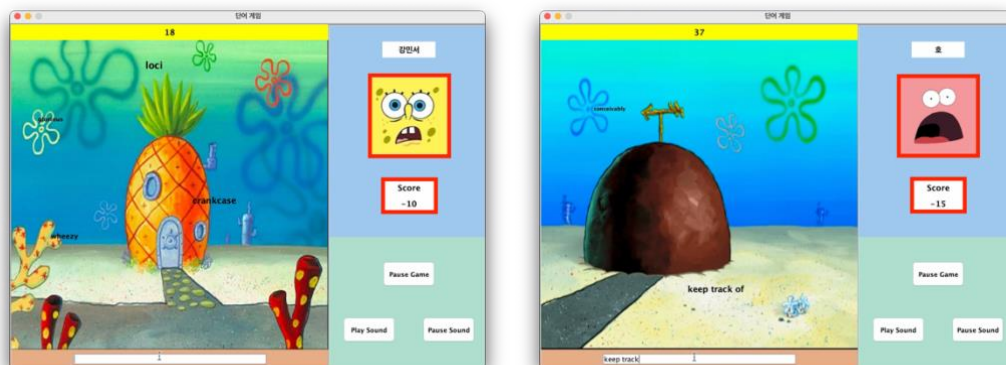
프로필에서 똥이를 선택했다면 게임 패널 배경화면과 프로필 사진이 프로필에서 스폰지밥을 선택했을 때와 다르다.



생성되는 단어의 크기는 모두 다르다.

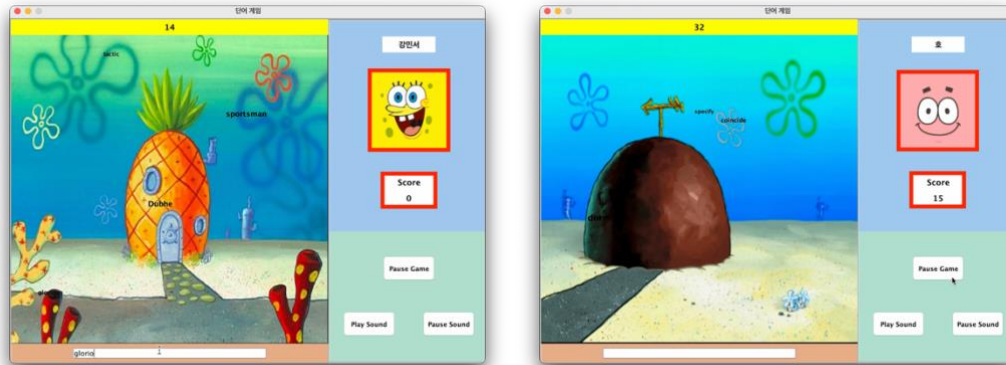


단어를 틀리면 표정이 변하고 점수가 내려간다.

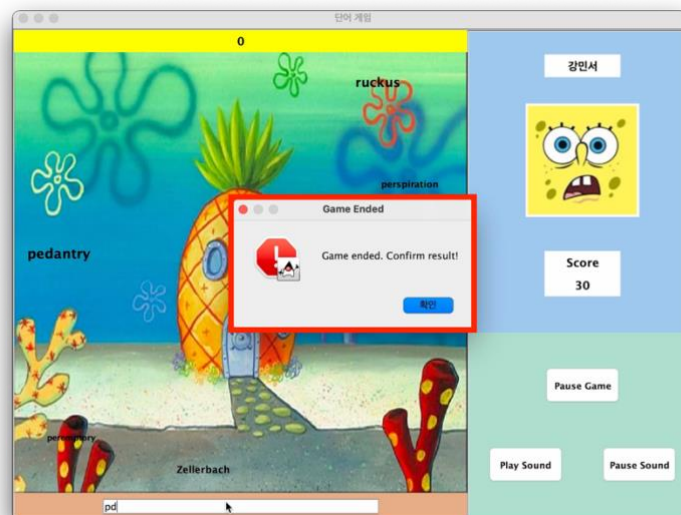




단어를 맞추면 표정이 변하고 점수가 올라간다. 스폰지밥을 프로필로 선택한 경우, 난이도가 Easy로 설정되어 있어 점수가 10점 올라간다. 뽕이를 선택한 경우, 난이도를 Normal로 설정했기 때문에 점수가 15점 올라간다.

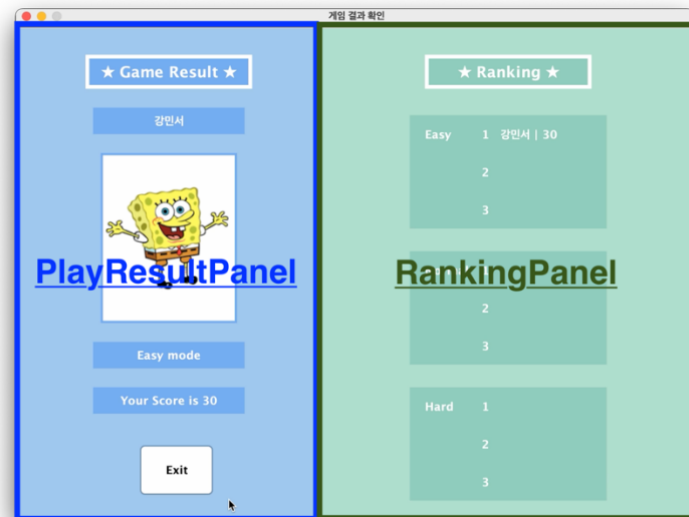


게임이 끝나면 팝업이 뜬다.



#### 4) 게임 결과 및 랭킹 확인 화면

플레이어의 프로필, 게임의 결과와 랭킹을 확인할 수 있다.



#### 4. 프로그램 소스 코드

App.java

```
public class App {

    public static void main(String[] args) {

        StartFrame startFrame = new StartFrame(); // startFrame 실행

    }

    public static void run() {

        GameFrame gameFrame = new GameFrame(); // gameFrame 실행

    }

}
```

Audio.java

```
import java.io.File;
import java.io.IOException;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class Audio {
```

```

private Clip startFrameClip;
private Clip beforeGameStartClip;
private Clip gameBackgroundClip;
private Clip correctClip;
private Clip wrongClip;
private Clip gameEndedClip;

// 생성자
public Audio() {

    startFrameClip = getClip("audio/mainStartAudio.wav");
    beforeGameStartClip = getClip("audio/beforeGameStartAudio.wav");
    gameBackgroundClip = getClip("audio/gameBackgroundAudio.wav");
    correctClip = getClip("audio/correct.wav");
    wrongClip = getClip("audio/wrong.wav");
    gameEndedClip = getClip("audio/gameEnded.wav");
}

// 오디오 클립 가져오기
private Clip getClip(String filePath) {

    Clip clip = null;

    try {
        clip = AudioSystem.getClip();
        File audioFile = new File(filePath);
        AudioInputStream audioStream = AudioSystem.getAudioInputStream(audioFile);
        clip.open(audioStream);
    }
    catch (LineUnavailableException e) { e.printStackTrace(); }
    catch (UnsupportedAudioFileException e) { e.printStackTrace(); }
    catch (IOException e) { e.printStackTrace(); }

    return clip;
}

// 오디오 재생
public void playAudio(String name) {

    switch(name) {
        case "startFrame":
            startFrameClip.start();
            break;

```

```

        case "beforeGameStart":
            beforeGameStartClip.start();
            break;
        case "gameBackground":
            gameBackgroundClip.start();
            break;
        case "correct":
            correctClip.start();
            break;
        case "wrong":
            wrongClip.start();
            break;
        case "gameEnded":
            gameEndedClip.start();
            break;
    }
}

```

// 오디오 정지

```

public void stopAudio(String name) {
    switch(name) {
        case "startFrame":
            startFrameClip.stop();
            break;
        case "beforeGameStart":
            beforeGameStartClip.stop();
            break;
        case "gameBackground":
            gameBackgroundClip.stop();
            break;
        case "correct":
            correctClip.stop();
            break;
        case "wrong":
            wrongClip.stop();
            break;
        case "gameEnded":
            gameEndedClip.stop();
            break;
    }
}

```

// 오디오 종료

```

    public void closeAudio(String name) {
        switch(name) {
            case "startFrame":
                startFrameClip.close();
                break;

            case "beforeGameStart":
                beforeGameStartClip.close();
                break;

            case "gameBackground":
                gameBackgroundClip.close();
                break;

            case "correct":
                correctClip.close();
                break;

            case "wrong":
                wrongClip.close();
                break;

            case "gameEnded":
                gameEndedClip.close();
                break;

        }
    }
}

```

## BeforeGameStartThread.java

```

import javax.swing.JLabel;

class BeforeGameStartThread extends Thread {

    // 생성자에게 전달 위한 레퍼런스 선언
    private GamePanel gamePanel = null;
    private JLabel timeLabel = null;
    private Audio audio = null;

    // 게임 시작 전 3초 카운트 다운을 위한 변수
    private int count = 3;

    // 생성자
    public BeforeGameStartThread(GamePanel gamePanel, JLabel timeLabel, Audio audio) {

        this.gamePanel = gamePanel;
    }
}

```

```

        this.timeLabel = timeLabel;
        this.audio = audio;

    }

    // 스레드 코드
    @Override
    public void run() {

        while(true) {

            try { // 3초 카운트 다운
                sleep(1000);
                count--;
                timeLabel.setText(Integer.toString(count));
                System.out.println("실행");
                if (count == 0) {
                    timeLabel.setText("게임 시작");
                    interrupt();
                }
            }
            catch (InterruptedException e) { // interrupt 받으면
                gamePanel.startGame(); // 게임 시작
                gamePanel.startGameTimerThread(); // 게임 타이머 스레드 시작
                audio.playAudio("gameBackground"); // 게임 배경 음악 시작
                return;
            }
        }
    }
}

```

GameFrame.java

```

import java.awt.BorderLayout;
import java.awt.Container;
import javax.swing.JFrame;
import javax.swing.JSplitPane;

public class GameFrame extends JFrame {

    private Container contentPane;
}

```



```

public Audio audio = new Audio();

private WordList wordList = new WordList();

// 패널 부착을 위한 패널 생성
private ProfileAndScorePanel profileAndScorePanel = new ProfileAndScorePanel();
private GamePanel gamePanel = new GamePanel(this, wordList, profileAndScorePanel, audio);
private SettingPanel settingPanel = new SettingPanel(audio, gamePanel);

// 생성자
public GameFrame() {

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setTitle("단어 게임");
    setSize(900, 680);

    contentPane = getContentPane();

    makeSplitPane();

    setVisible(true);
    setResizable(false);

    gamePanel.startCountDown(); // 3초 카운트 다운 시작
    audio.playAudio("beforeGameStart"); // 카운트 다운 음악 시작

}

// 스플릿팬 만들기 (영역 만들고 패널 부착)
private void makeSplitPane() {

    JSplitPane hPane = new JSplitPane();
    hPane.setOrientation(JSplitPane.HORIZONTAL_SPLIT); // 수평으로 분할
    hPane.setDividerLocation(600); // 디바이더 초기 위치 설정
    hPane.setDividerSize(0); // 스플릿팬 선 안보이게
    contentPane.add(hPane, BorderLayout.CENTER); // ContentPane 불러서 가운데에 부착

    JSplitPane vPane = new JSplitPane();
    vPane.setOrientation(JSplitPane.VERTICAL_SPLIT); // 수직으로 분할
    vPane.setDividerLocation(400); // 디바이더 초기 위치 설정
    vPane.setDividerSize(0); // 스플릿팬 선 안보이게
    hPane.setRightComponent(vPane); // vPane을 hPane 오른쪽에 부착
    hPane.setLeftComponent(gamePanel); // gamePanel을 hPane 왼쪽에 부착

```

```

        vPane.setTopComponent(profileAndScorePanel); // profileAndScorePanel을 vPane 위쪽에 부착
        vPane.setBottomComponent(settingPanel); // settingPanel을 vPane 아래쪽에 부착

    }

}

```

## GameManagement.java

```

import javax.swing.ImageIcon;

public class GameManagement {

    // 정보 저장하는 static 변수
    public static String name = null;
    public static String profile = "SpongebobSquarepants"; // 초기 설정
    public static ImageIcon profileImage = new ImageIcon("image/resultProfile/SpongebobSquarepantsprofile.png");
    public static String difficulty = "Easy"; // 초기 설정
    public static String pathName = null;
    public static String fileName = null;
    public static int score = 0;

}

```

## GamePanel.java

```

import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Iterator;
import java.util.Vector;

public class GamePanel extends JPanel {

    private static int MAX_WORDS = 30;
    public static int fontRandSize = 10;
    private final int LABEL_WIDTH = 150;
    private final int LABEL_HEIGHT = 40;
}

```

```

private ImageIcon icon;
private Font defaultFont = new Font("Jokerman", Font.BOLD, 15);
private JTextField inputField = new JTextField(MAX_WORDS);
private JLabel timeLabel = new JLabel("3");

// 생성자에게 전달 위한 레퍼런스 선언
private GameFrame gameFrame = null;
private WordList wordList = null;
private ProfileAndScorePanel profileAndScorePanel = null;
private Audio audio = null;

private MainPlayPanel MainPlayPanel = new MainPlayPanel();

// 단어 저장을 위한 벡터 생성
private Vector<Word> currentWords = new Vector<>(MAX_WORDS);

// 스레드 실행을 위한 레퍼런스 선언
private BeforeGameStartThread beforeGameStartThread = null;
private GameTimerThread gameTimerThread = null;
private GameThread gameThread = null;

// 랭킹 확인 프레임을 생성을 위한 레퍼런스 선언
private UserRankingFrame userRankingFrame;

// 생성자
public GamePanel(GameFrame gameFrame, WordList wordList, ProfileAndScorePanel profileAndScorePanel, Audio
audio) {

    this.gameFrame = gameFrame;
    this.wordList = wordList;
    this.profileAndScorePanel = profileAndScorePanel;
    this.audio = audio;

    setLayout(new BorderLayout());

    makeMainPlayPanel(); // 플레이 영역 만들기
    makeTimePanel(); // 시간 영역 만들기
    makeInputPanel(); // 입력 영역 만들기
    addActionListener(); // 액션 리스너 등록

}

// 플레이 영역을 가운데에 부착

```

```

private void makeMainPlayPanel() { add(MainPlayPanel, BorderLayout.CENTER); }

// 시간 영역을 위쪽에 부착
private void makeTimePanel() {

    JPanel timePanel = new JPanel();
    timePanel.setBackground(Color.YELLOW);
    timeLabel.setFont(defaultFont);
    timePanel.add(timeLabel);
    add(timePanel, BorderLayout.NORTH);

}

// 입력 영역을 아래쪽에 부착
private void makeInputPanel() {

    JPanel inputPanel = new JPanel();

    Color backgroundColor = new Color(221, 176, 141);
    inputPanel.setBackground(backgroundColor);

    inputPanel.add(inputField, BorderLayout.SOUTH);
    add(inputPanel, BorderLayout.SOUTH);

}

// 액션 리스너 등록
private void addStartActionListener() {

    inputField.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            JTextField textField = (JTextField) e.getSource(); // 입력받은 것 전달받음
            boolean isCorrect = false;
            // 비교 시작
            for (Word word : currentWords) {
                if (textField.getText().equals(word.getName())) { // 단어가 일치한다면
                    isCorrect = true;
                    profileAndScorePanel.increase(); // 점수 증가
                    word.setY(MainPlayPanel.getHeight()); // 단어를 아래에 배치
                    profileAndScorePanel.setNormalProfileImage(); // 표정 바꾸기
                    Audio playcorrect = new Audio(); // 효과음을 나타내기 위해 오디오 객체 생성
                    playcorrect.playAudio("correct"); // 효과음 발생
                }
            }
        }
    });
}

```

```

        break;
    }
}

if (!isCorrect) { // 단어가 일치하지 않는다면
    profileAndScorePanel.decrease(); // 점수 감소
    profileAndScorePanel.setSadProfileImage(); // 표정 바꾸기
    Audio playwrong = new Audio(); // 효과음을 나타내기 위해 오디오 객체 생성
    playwrong.playAudio("wrong"); // 효과음 발생

}

textField.setText(""); // 텍스트 필드 초기화
}

});

}

// 3초 카운트 다운 메소드
public void startCountDown() {
    if (beforeGameStartThread == null) {
        beforeGameStartThread = new BeforeGameStartThread(this, timeLabel, audio);
        beforeGameStartThread.start();
    }
}

// 게임 시작 메소드
public void startGame() {
    if (gameThread == null) {
        gameThread = new GameThread(this);
        gameThread.start();
    }
}

// 게임 타이머 시작 메소드
public void startGameTimerThread() {
    if (gameTimerThread == null) {
        gameTimerThread = new GameTimerThread(this, timeLabel, audio);
        gameTimerThread.start();
    }
}

// 게임 종료 메소드
public void gameEnd() {

    if (gameThread != null) {

```

```

        gameThread.interrupt(); // 게임 스레드 종료
        this.removeAll(); // 화면 정지
        audio.closeAudio("gameBackground");
        JOptionPane.showMessageDialog(gameFrame, "Game ended. Confirm result!", "Game Ended",
JOptionPane.ERROR_MESSAGE); // 팝업 띄우기
        gameFrame.dispose(); // gameFrame 종료
        GameManagement.score = profileAndScorePanel.getScore();
        userRankingFrame = new UserRankingFrame(profileAndScorePanel); // 랭킹 확인 프레임 띄우기
        audio.playAudio("gameEnded"); // 게임 종료 배경 음악 시작
    }

}

// 게임 중지 메소드
public void stopGame() {

    if(gameThread.getStopFlag() == false) {
        gameThread.stopGame();
        inputField.setEditable(false);
    }

    if(gameTimerThread.getStopFlag() == false) {
        gameTimerThread.stopTimer();
    }

}

// 게임 재개 메소드
public void resumeGame() {

    if(gameThread.getStopFlag() == true) {
        gameThread.resumeGame();
        inputField.setEditable(true);
    }

    if(gameTimerThread.getStopFlag() == true) {
        gameTimerThread.resumeTimer();
    }

}

// 단어 추가 메소드
public void addWord() {
    int x = (int) (Math.random() * (MainPlayPanel.getWidth() - LABEL_WIDTH / 2));
    Word word = new Word(wordList.getWord(), x, 0, Math.random() / 20 + 0.1);
    currentWords.add(word);
}

```



```

        addLabel(word);
    }

    // 레이블을 패널에 부착하는 메소드
    private void addLabel(Word word) {

        JLabel label = word.getLabel();
        fontRandSize = (int)(Math.random()*10 + 10); // 글자 크기 랜덤
        label.setFont(new Font("Jokerman", Font.BOLD, fontRandSize));
        label.setSize(LABEL_WIDTH, LABEL_HEIGHT);
        label.setLocation(getX(), 0);
        MainPlayPanel.add(label);

    }

    // 단어 세팅 메소드
    public void setWords() {

        Iterator<Word> iterator = currentWords.iterator(); // currentWords의 요소를 순차 검색할 Iterator 객체 리턴
        while (iterator.hasNext()) { // 방문할 요소가 남아있다면
            Word word = iterator.next(); // iterator가 가리키는 요소 리턴
            word.setY(word.getY() + word.getSpeed()); // Y 좌표 설정
            JLabel label = word.getLabel(); // 단어 받아오기
            label.setLocation((int) (word.getX()), (int) word.getY()); // 레이블 위치 지정
            if (label.getY() >= MainPlayPanel.getHeight()) { // 영역을 벗어나면
                MainPlayPanel.remove(label); // 패널에서 레이블 제거
                iterator.remove(); // iterator에서 제거
            }
        }

    }

}

class MainPlayPanel extends JPanel {

    // 게임 배경 화면 그리는 메소드
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(getBackgroundImage(), 0, 0, getWidth(), getHeight(), this);
    }

    // 생성자
    public MainPlayPanel() {}
}

```

```

// 게임 배경 화면 받아오는 메소드
public Image getBackgroundImage() {

    if(GameManagement.profile.equals("SpongebobSquarepants")) {

        icon = new ImageIcon("image/background/SpongebobBackground.png");
        return icon.getImage();

    }
    else {

        icon = new ImageIcon("image/background/PatrickStarBackground.png");
        return icon.getImage();

    }
}

}

```

## GameThread.java

```

// 게임 스레드
public class GameThread extends Thread {

    private GamePanel gamePanel = null;

    // 게임 일시 중지 확인 Flag 관련 변수 및 메소드
    private boolean stopFlag = false;
    public boolean getStopFlag() { return stopFlag; }
    public void stopGame() { stopFlag = true; }

    // 게임 이어서 하는 메소드
    // 스레드 깨우기
    synchronized public void resumeGame() {

        stopFlag = false;
        this.notify();

    }

    // 대기 상태 걸어놓는 메소드
    // 스레드 대기
    synchronized private void waitFlag() {

        try { this.wait(); }
        catch (InterruptedException e) { }
    }
}

```

```

    }

    // 생성자
    public GameThread(GamePanel gamePanel) {

        super("GameThread");
        this.gamePanel = gamePanel;

    }

    // 스레드 코드
    @Override
    public void run() {
        int count = 0;

        while (true) {
            try {
                if (stopFlag == true) waitFlag();
                if (count % 1000 == 0) gamePanel.addWord();
                gamePanel.setWords();
                count++;
                sleep(1);
            } catch (InterruptedException e) { return; }
        }
    }
}

```

## GameTimerThread.java

```

import javax.swing.JLabel;

// 게임 타이머 스레드
public class GameTimerThread extends Thread{

    // 생성자에게 전달 위한 레퍼런스 선언
    private Audio audio = null;
    private GamePanel gamePanel = null;
    private JLabel timeLabel = null;

    // 게임 시작 후 45초 카운트 다운을 위한 변수
    private int count = 45;

    // 게임 일시 중지 확인 Flag 관련 변수 및 메소드

```

```

private boolean stopFlag = false;

public boolean getStopFlag() { return stopFlag; }

public void stopTimer() { stopFlag = true; }

// 타이머 이어서 동작하는 메소드
// 스레드 깨우기
synchronized public void resumeTimer() {

    stopFlag = false;
    this.notify();

}

// 대기 상태 걸어놓는 메소드
// 스레드 대기
synchronized private void waitFlag() {

    try { this.wait(); }
    catch (InterruptedException e) { }

}

// 생성자
public GameTimerThread(GamePanel gamePanel, JLabel timeLabel, Audio audio) {

    this.gamePanel = gamePanel;
    this.timeLabel = timeLabel;
    this.audio = audio;

}

// 스레드 코드
@Override
public void run() {

    timeLabel.setText(Integer.toString(count));
    while(true) {
        try {
            sleep(1000);
            if (stopFlag == true) waitFlag();
            count--;
            timeLabel.setText(Integer.toString(count));
            if (count == -1) {
                timeLabel.setText("게임 종료");
            }
        }
    }
}

```

```

        gamePanel.gameEnd();
        interrupt();
    }
} catch (InterruptedException e) { return; }
}
}
}

```

## PlayerSettingDialog.java

```

import javax.swing.*.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.*.*;
import java.awt.event.*;

public class PlayerSettingDialog extends JDialog {

    // 생성자에게 전달 위한 레퍼런스 선언
    StartFrame startFrame = null;
    public Audio audio = null;

    private Container contentPane;
    private Font defaultFont = new Font("Jokerman", Font.BOLD, 15);

    // 유저의 정보 저장하는 변수 (GameManagemet의 static 변수에 저장)
    private String name = null;
    private String profile = "SpongebobSquarepants"; // 초기 설정
    private String difficulty = "Easy"; // 초기 설정
    private String pathName = null;
    private String fileName = null;

    // Player Name 부분
    private JLabel playerNameLabel = new JLabel("Player Name");
    private JTextField inputPlayerNameField = new JTextField(20); // Player Name 입력받는 필드

    // Choose Profile 부분
    private JLabel chooseProfileLabel = new JLabel("Choose Profile");
    private String profileArray [] = { "image/settingProfile/SpongebobSquarepants.png",
    "image/settingProfile/PatrickStar.png" };

    private ImageIcon profileImage [] = new ImageIcon[2];
    ButtonGroup profileButtonGroup = new ButtonGroup();
    JRadioButton profileButtonComponents[] = new JRadioButton[2];

```

```

// Choose Difficulty 부분

private JLabel difficultyLabel = new JLabel("Choose Difficulty");
private String difficultyArray [] = { "Easy", "Normal", "Hard" };
private ButtonGroup difficultyButtonGroup = new ButtonGroup();
private JRadioButton difficultyButtonComponents[] = new JRadioButton[3];


// Choose File 부분

private JLabel fileLabel = new JLabel("Choose File");
private JButton OpenFileButton = new JButton("Open File");
private JFileChooser chooser; // JFileChooser 레퍼런스 변수 선언


// Complete Settings와 Cancel 부분

private JButton completeSettingsButton = new JButton("Complete Settings");
private JButton cancelButton = new JButton("Cancel");


// 생성자

public PlayerSettingDialog(StartFrame startFrame, Audio audio) {

    super(startFrame, "Setting Player", true);
    this.startFrame = startFrame;
    this.audio = audio;

    contentPane = getContentPane();
    contentPane.setLayout(null);

    setSizeAndLocation(); // 컴포넌트 부착을 위한 컴포넌트 크기 및 위치 설정
    setAllFont(); // 폰트 설정
    makeAllComponents(); // 모든 컴포넌트들 부착
    addActionListener(); // 필요한 액션리스너들 부착

    setSize(550, 580);
    setVisible(false);
    setResizable(false); // 창 크기 변경 불가능하게

    if (GameManagement.fileName != null) {
        OpenFileButton.setText(GameManagement.fileName);
        fileName = GameManagement.fileName;
    }

}

private void setSizeAndLocation() {

// PlayerName 부분

```



```

        playerNameLabel.setSize(100, 40);
        playerNameLabel.setLocation(30, 30);
        inputPlayerNameField.setSize(150, 40);
        inputPlayerNameField.setLocation(150, 30);

        // Choose Profile 부분
        chooseProfileLabel.setSize(115, 40);
        chooseProfileLabel.setLocation(30, 90);

        for (int i = 0; i < profileButtonComponents.length; i++) {
            profileImage[i] = new ImageIcon(profileArray[i]);
            profileButtonComponents[i] = new JRadioButton(profileImage[i]);
            profileButtonComponents[i].setSize(60, 75);
            profileButtonComponents[i].setLocation(30+(i*80), 135);
        }
        profileButtonComponents[0].setSelected(true); // 스폰지밥 Default로 선택

        // Choose Difficulty 부분
        difficultyLabel.setSize(145, 40);
        difficultyLabel.setLocation(30, 230);

        for (int i = 0; i < difficultyButtonComponents.length; i++) {
            difficultyButtonComponents[i] = new JRadioButton(difficultyArray[i]);
            difficultyButtonComponents[i].setSize(100, 40);
            difficultyButtonComponents[i].setLocation(30+(i*100), 270);
        }
        difficultyButtonComponents[0].setSelected(true); // Easy 모드 Default로 선택

        // Choose File 부분
        fileLabel.setSize(145, 40);
        fileLabel.setLocation(30, 320);

        OpenFileButton.setSize(170, 40);
        OpenFileButton.setLocation(30, 365);

        // Complete Settings와 Cancel 부분
        completeSettingsButton.setSize(190, 50);
        completeSettingsButton.setLocation(65, 450);
        cancelButton.setSize(190, 50);
        cancelButton.setLocation(550/2+20, 450);
    }

    private void setAllFont() {

```

```

        playerNameLabel.setFont(defaultFont);
        inputPlayerNameField.setFont(defaultFont);
        chooseProfileLabel.setFont(defaultFont);
        difficultyLabel.setFont(defaultFont);
        for (int i = 0; i < difficultyButtonComponents.length; i++) {
            difficultyButtonComponents[i].setFont(defaultFont);
        }
        fileLabel.setFont(defaultFont);
        OpenFileButton.setFont(defaultFont);
        completeSettingsButton.setFont(defaultFont);
        cancelButton.setFont(defaultFont);
    }

    private void makeAllComponents() {

        contentPane.add(playerNameLabel);
        contentPane.add(inputPlayerNameField);
        contentPane.add(chooseProfileLabel);
        for (int i = 0; i < profileArray.length; i++) {
            profileButtonGroup.add(profileButtonComponents[i]);
            contentPane.add(profileButtonComponents[i]);
        }
        contentPane.add(difficultyLabel);
        for (int i = 0; i < difficultyButtonComponents.length; i++) {
            difficultyButtonGroup.add(difficultyButtonComponents[i]);
            contentPane.add(difficultyButtonComponents[i]);
        }
        contentPane.add(fileLabel);
        contentPane.add(OpenFileButton);
        contentPane.add(completeSettingsButton);
        contentPane.add(cancelButton);
    }

    private void addAllActionListener() {

        // 버튼 클릭 시 파일 다이얼로그 오픈
        OpenFileButton.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {

```

수행된 경우

"경고", JOptionPane.*WARNING\_MESSAGE*);

완전경로명

이름

ImageIcon("image/resultProfile/SpongebobSquarepantsprofile.png");

ImageIcon("image/resultProfile/PatrickStarprofile.png");

```
chooser = new JFileChooser(); // 파일 다이얼로그 생성
FileNameExtensionFilter filter = new FileNameExtensionFilter(
    "TXT files", // 파일 이름난에 출력될 문자열
    "txt"); // 파일 필터로 사용되는 확장자, *.txt만 나열됨

chooser.setFileFilter(filter); // 파일 다이얼로그에 파일 필터 설정

int ret = chooser.showOpenDialog(null); // 파일 열기 다이얼로그 출력
if (ret != JFileChooser.APPROVE_OPTION) { // 파일 선택이 정상적으로

    JOptionPane.showMessageDialog(null, "파일을 선택하세요",
        "경고", JOptionPane.WARNING_MESSAGE);

    return;
}

// 파일 선택이 정상적으로 수행된 경우
pathName = chooser.getSelectedFile().getPath(); // 사용자가 선택한 파일

fileName = chooser.getSelectedFile().getName(); // 사용자가 선택한 파일

OpenFileButton.setText(fileName); // 버튼 텍스트를 파일 이름으로 설정

}

});

// 프로필 선택 버튼에 리스너 등록
for (int i = 0; i < profileButtonComponents.length; i++) {
    profileButtonComponents[i].addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent e) {

            if (e.getStateChange() == ItemEvent.DESELECTED) return;
            if (profileButtonComponents[0].isSelected()) {
                profile = "SpongebobSquarepants";
                GameManagement.profile = profile;
                GameManagement.profileImage = new

ImageIcon("image/resultProfile/SpongebobSquarepantsprofile.png");

            }
            else if (profileButtonComponents[1].isSelected()) {
                profile = "PatrickStar";
                GameManagement.profile = profile;
                GameManagement.profileImage = new

ImageIcon("image/resultProfile/PatrickStarprofile.png");

            }
        }
    });
}
```

```

    });

}

// 난이도 선택 버튼에 리스너 등록
for (int i = 0; i < difficultyButtonComponents.length; i++) {
    difficultyButtonComponents[i].addItemListener(new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent e) {

            if (difficultyButtonComponents[0].isSelected()) {
                difficulty = "Easy";
                GameManagement.difficulty = difficulty;
            }
            else if (difficultyButtonComponents[1].isSelected()) {
                difficulty = "Normal";
                GameManagement.difficulty = difficulty;
            }
            else if (difficultyButtonComponents[2].isSelected()) {
                difficulty = "Hard";
                GameManagement.difficulty = difficulty;
            }
        }
    });
}

cancelButton.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) { dispose(); }

});

// Complete Settings 버튼에 리스너 등록
// 게임을 시작하시겠습니까? 다이얼로그 출력하고 YES NO 띄우고,
// YES이면 게임 화면으로 넘어가기, NO이면 이전 상태로 돌아가기
completeSettingsButton.addActionListener(new ActionListener() {

    // confirm Dialog 출력
    @Override
    public void actionPerformed(ActionEvent e) {

        name = inputPlayerNameField.getText();
    }
}

```

```

        if (name.equals("") || name == null) // 이름이 입력되지 않은 경우 경고
            JOptionPane.showMessageDialog(null, "이름을 입력하세요!!",
"경고", JOptionPane.ERROR_MESSAGE);

        else if (fileName == null) // 파일이 선택되지 않은 경우 경고
            JOptionPane.showMessageDialog(null, "파일을 선택하세요!!",
"경고", JOptionPane.ERROR_MESSAGE);

        else { // 이름이 입력되었고 파일이 선택되었다면

            int confirmResult = JOptionPane.showConfirmDialog(contentPane,
"입력한 정보가 맞습니까?", "Confirm Setting Infomation",
JOptionPane.YES_NO_OPTION);

            if (confirmResult == JOptionPane.YES_OPTION) {

                GameManagement.name = name;
                GameManagement.pathName = pathName;
                GameManagement.fileName = fileName;

                if (GameManagement.fileName.equals("words.txt"))
                    GameManagement.fileName = ("txt" +
fileName);

                else if
(GameManagement.fileName.equals("toeicwords.txt"))

                    GameManagement.fileName = ("txt" +
fileName);

                audio.stopAudio("startFrame"); // 시작 음악 정지
                dispose(); // PlayerSettingDialog 종료
                startFrame.setVisible(false); // StartFrame 안보이게
                startFrame.dispose(); // StartFrame 종료
                App.run(); // 게임 화면으로 넘어가기

            }

        }

    }

});

}

}

```

```

import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.LineBorder;

public class PlayResultPanel extends JPanel {

    // 생성자에게 전달 위한 레퍼런스 선언
    private UserRankingFrame userRankingFrame = null;
    private ProfileAndScorePanel profileAndScorePanel = null;

    private Font defaultFont = new Font("Jokerman", Font.BOLD, 15);
    private LineBorder defaultLineBorder = new LineBorder(Color.WHITE, 5);

    private JLabel gameResultTextLabel = new JLabel(" ★ Game Result ★ ");
    private JLabel nameTextLabel = new JLabel(GameManagement.name);
    private JLabel profileImageLabel = new JLabel(GameManagement.profileImage);
    private JLabel difficultyLabel = new JLabel(GameManagement.difficulty + " mode");
    private JLabel scoreLabel;
    private JButton exitButton = new JButton("Exit");

    // 생성자
    public PlayResultPanel(UserRankingFrame userRankingFrame, ProfileAndScorePanel profileAndScorePanel) {

        this.profileAndScorePanel = profileAndScorePanel;
        this.userRankingFrame = userRankingFrame;

        Color backgroundColor = new Color(168, 200, 236);
        setBackground(backgroundColor);

        setLayout(null);

        makeTextLabels(); // 텍스트 레이블 부착
        makeProfileImageLabel(); // 이미지 레이블 부착
        makeExitButton(); // 종료 버튼 부착

    }

```



```
private void makeTextLabels() {

    // Game Result 텍스트
    gameResultTextLabel.setFont(new Font("Jokerman", Font.BOLD, 20));
    gameResultTextLabel.setForeground(Color.WHITE);
    gameResultTextLabel.setSize(220, 45);
    gameResultTextLabel.setLocation(90, 35);
    gameResultTextLabel.setOpaque(true);
    gameResultTextLabel.setBackground(new Color(129, 172, 236));
    gameResultTextLabel.setHorizontalAlignment(JLabel.CENTER);
    gameResultTextLabel.setBorder(defaultLineBorder);
    add(gameResultTextLabel);

    // name 텍스트
    nameTextLabel.setFont(defaultFont);
    nameTextLabel.setForeground(Color.WHITE);
    nameTextLabel.setSize(200, 35);
    nameTextLabel.setLocation(100, 105);
    nameTextLabel.setOpaque(true);
    nameTextLabel.setBackground(new Color(129, 172, 236));
    nameTextLabel.setHorizontalAlignment(JLabel.CENTER);
    add(nameTextLabel);

    // difficulty 텍스트
    difficultyLabel.setFont(defaultFont);
    difficultyLabel.setForeground(Color.WHITE);
    difficultyLabel.setSize(200, 35);
    difficultyLabel.setLocation(100, 415);
    difficultyLabel.setOpaque(true);
    difficultyLabel.setBackground(new Color(129, 172, 236));
    difficultyLabel.setHorizontalAlignment(JLabel.CENTER);
    add(difficultyLabel);

    // score 텍스트
    String score = Integer.toString(profileAndScorePanel.getScore());
    scoreLabel = new JLabel("Your Score is " + score);
    scoreLabel.setFont(defaultFont);
    scoreLabel.setForeground(Color.WHITE);
    scoreLabel.setSize(200, 35);
    scoreLabel.setLocation(100, 475);
    scoreLabel.setOpaque(true);
    scoreLabel.setBackground(new Color(129, 172, 236));
    scoreLabel.setHorizontalAlignment(JLabel.CENTER);
```

```

        add(scoreLabel);

    }

    private void makeProfileImageLabel() {

        profileImageLabel.setSize(180, 225);
        profileImageLabel.setLocation(110, 165);
        profileImageLabel.setBorder(new LineBorder(new Color(129, 172, 236), 3));
        add(profileImageLabel);

    }

    private void makeExitButton() {

        exitButton.setFont(defaultFont);
        exitButton.setSize(100, 70);
        exitButton.setLocation(160, 550);

        // 버튼에 이벤트 리스너 등록
        exitButton.addKeyListener(new ExitListener());

        // 포커스 강제 설정
        exitButton.setFocusable(true);
        exitButton.requestFocus();

        add(exitButton);

    }

    class ExitListener extends KeyAdapter {

        @Override
        public void keyPressed(KeyEvent e) {

            if (e.getKeyChar() == '\n') { // 엔터를 받으면
                System.exit(0); // 프로그램 종료
            }

        }

    }

}

```

## ProfileAndScorePanel.java

```
import java.awt.Color;
import java.awt.Font;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.LineBorder;

public class ProfileAndScorePanel extends JPanel {

    private Font defaultFont = new Font("Jokerman", Font.BOLD, 15);

    public static int score = 0;

    private JLabel nameLabel = new JLabel(GameManagement.name);
    private JLabel scoreLabel = new JLabel(Integer.toString(score));

    private ImageIcon profileImage;
    private JLabel profileLabel;

    private ImageIcon normalProfileImage;
    private ImageIcon sadProfileImage;

    // 생성자
    public ProfileAndScorePanel() {

        setOpaque(true);
        Color backgroundColor = new Color(168, 200, 236);
        setBackground(backgroundColor);

        setLayout(null);

        makeNameLabel(); // 이름 레이블 부착
        makeProfileLabel(); // 프로필 레이블 부착
        makeScoreLabel(); // 점수 레이블 부착

    }

    public void makeNameLabel() {
```

```

        nameLabel.setFont(defaultFont);
        nameLabel.setSize(100, 30);
        nameLabel.setLocation(100,30);
        nameLabel.setHorizontalAlignment(JLabel.CENTER);
        nameLabel.setOpaque(true);
        nameLabel.setBackground(Color.WHITE);
        this.add(nameLabel);
    }

    public void makeProfileLabel() {

        // 프로필에 따라 나타나는 이미지가 다르다
        if(GameManagement.profile.equals("SpongebobSquarepants")) {
            profileImage = new ImageIcon("image/expression/SpongebobNormal.png");
            // 원래 this.normalProfileImage = new ImageIcon("image/expression/SpongebobNormal.png");
            this.normalProfileImage = profileImage;
            this.sadProfileImage = new ImageIcon("image/expression/SpongebobSad.png");
            profileLabel = new JLabel(profileImage);
            profileLabel.setSize(150, 150);
            profileLabel.setLocation(75 ,95);
            profileLabel.setBorder(new LineBorder(Color.WHITE, 3));
            this.add(profileLabel);
        }
        else if(GameManagement.profile.equals("PatrickStar")) {
            profileImage = new ImageIcon("image/expression/PatrickStarNormal.png");
            this.normalProfileImage = new ImageIcon("image/expression/PatrickStarNormal.png");
            this.sadProfileImage = new ImageIcon("image/expression/PatrickStarSad.png");
            profileLabel = new JLabel(profileImage);
            profileLabel.setSize(150, 150);
            profileLabel.setLocation(75 ,95);
            profileLabel.setBorder(new LineBorder(Color.WHITE, 3));
            this.add(profileLabel);
        }
    }

    public void makeScoreLabel() {

        JLabel scoreLabelText = new JLabel("Score");
        scoreLabelText.setFont(defaultFont);
        scoreLabelText.setSize(100, 30);
        scoreLabelText.setLocation(100 ,290);
    }

```

```

        scoreLabelText.setHorizontalAlignment(JLabel.CENTER);
        scoreLabelText.setOpaque(true);
        scoreLabelText.setBackground(Color.WHITE);
        this.add(scoreLabelText);

        scoreLabel.setFont(defaultFont);
        scoreLabel.setSize(100, 30);
        scoreLabel.setLocation(100 ,320);
        scoreLabel.setHorizontalAlignment(JLabel.CENTER);
        scoreLabel.setOpaque(true);
        scoreLabel.setBackground(Color.WHITE);
        this.add(scoreLabel);

    }

    // 단어가 일치할 때 호출되는 메소드
    public void setNormalProfileImage() { profileLabel.setIcon(normalProfileImage); }

    // 단어가 일치하지 않을 때 호출되는 메소드
    public void setSadProfileImage() { profileLabel.setIcon(sadProfileImage); }

    // 점수 증가 메소드
    public void increase() {

        // 난이도에 따라 증가하는 점수가 다름
        if (GameManagement.difficulty.equals("Normal")) {
            score += 15;
            scoreLabel.setText(Integer.toString(getScore()));
        }
        else if (GameManagement.difficulty.equals("Hard")) {
            score += 20;
            this.scoreLabel.setText(Integer.toString(getScore()));
        }
        else {
            score += 10;
            scoreLabel.setText(Integer.toString(getScore()));
        }
    }

    // 점수 감소 메소드
    public void decrease() {

        // 난이도에 따라 감소하는 점수가 다름

```

```

        if (GameManagement.difficulty.equals("Normal")) {
            score -= 15;
            scoreLabel.setText(Integer.toString(getScore()));
        }
        else if (GameManagement.difficulty.equals("Hard")) {
            score -= 20;
            scoreLabel.setText(Integer.toString(getScore()));
        }
        else {
            score -= 10;
            scoreLabel.setText(Integer.toString(getScore()));
        }
    }

    // 점수 리턴 메소드
    public int getScore() { return score; }
}

```

## RankingPanel.java

```

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;

import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.LineBorder;

public class RankingPanel extends JPanel {

    private final int RECT_WIDTH = 260;
    private final int RECT_HEIGHT = 150;

    private Font defaultFont = new Font("Jokerman", Font.BOLD, 15); // 기본 폰트 설정
    private LineBorder defaultLineBorder = new LineBorder(Color.WHITE, 5);
    private String difficulty [] = { "Easy", "Normal", "Hard" };
    private JLabel rankingTextLabel = new JLabel(" ★ Ranking ★ ");
    private JLabel difficultyLabel [] = new JLabel[3];
    private JLabel numberLabel [] = new JLabel[3];
    private JLabel nameScoreLabel = null;

    // 생성자

```

```

public RankingPanel() {

    Color backgroundColor = new Color(184, 221, 207);
    setBackground(backgroundColor);

    setLayout(null);

    makeBasicLabels();
    addNameScore();

}

// 사각형 그리는 메소드
public void paintComponent(Graphics g) {

    super.paintComponent(g);
    g.setColor(new Color(157, 203, 189));
    for (int i = 0; i < 3; i++) {
        g.fillRect(120, 115 + (180*i), RECT_WIDTH, RECT_HEIGHT);
    }

}

private void makeBasicLabels() {

    // Ranking 텍스트
    rankingTextLabel.setFont(new Font("Jokerman", Font.BOLD, 20));
    rankingTextLabel.setForeground(Color.WHITE);
    rankingTextLabel.setSize(220, 45);
    rankingTextLabel.setLocation(140, 35);
    rankingTextLabel.setOpaque(true);
    rankingTextLabel.setBackground(new Color(157, 203, 189));
    rankingTextLabel.setHorizontalAlignment(JLabel.CENTER);
    rankingTextLabel.setBorder(defaultLineBorder);
    add(rankingTextLabel);

    // 난이도 텍스트
    for (int i = 0; i < difficultyLabel.length; i++) {
        difficultyLabel[i] = new JLabel(difficulty[i]);
        difficultyLabel[i].setFont(defaultFont);
        difficultyLabel[i].setForeground(Color.WHITE);
        difficultyLabel[i].setSize(65, 50);
        difficultyLabel[i].setLocation(140, 115 + (180*i));
        add(difficultyLabel[i]);
    }
}

```

```

    }

    // 숫자 텍스트
    for (int i = 0; i < numberLabel.length; i++) {
        for (int j = 0; j < numberLabel.length; j++) {
            numberLabel[i][j] = new JLabel(Integer.toString(i+1));
            numberLabel[i][j].setFont(defaultFont);
            numberLabel[i][j].setForeground(Color.WHITE);
            numberLabel[i][j].setSize(15, 50);
            numberLabel[i][j].setLocation(215, 115 + (50*i) + (180*j));
            add(numberLabel[i][j]);
        }
    }

    private void addNameScore() {

        nameScoreLabel = new JLabel(GameManagement.name + " | " + GameManagement.score);
        nameScoreLabel.setFont(defaultFont);
        nameScoreLabel.setForeground(Color.WHITE);
        nameScoreLabel.setSize(100, 50);

        if (GameManagement.difficulty.equals("Easy"))
            nameScoreLabel.setLocation(240, 115); // 140, 115 + (180*i)
        else if (GameManagement.difficulty.equals("Normal"))
            nameScoreLabel.setLocation(240, 295);
        else if (GameManagement.difficulty.equals("Normal"))
            nameScoreLabel.setLocation(240, 475);

        add(nameScoreLabel);
    }
}

```

SettingPanel.java

```

import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JPanel;

```



```

public class SettingPanel extends JPanel {

    // 생성자에게 전달 위한 레퍼런스 선언
    private Audio audio = null;
    private GamePanel gamePanel = null;

    private Font defaultFont = new Font("Jokerman", Font.BOLD, 12);

    private JButton pausePlayButton = new JButton("Pause Game");
    private JButton playSoundButton = new JButton("Play Sound");
    private JButton pauseSoundButton = new JButton("Pause Sound");

    // 생성자
    public SettingPanel(Audio audio, GamePanel gamePanel) {

        this.audio = audio;
        this.gamePanel = gamePanel;

        Color backgroundColor = new Color(184, 221, 207);
        setBackground(backgroundColor);

        setLayout(null);

        // 버튼들 부착
        makePausePlayButton();
        makePlaySoundButton();
        makePauseSoundButton();

    }

    public void makePausePlayButton() {

        pausePlayButton.setFont(defaultFont);
        pausePlayButton.setSize(100, 50);
        pausePlayButton.setLocation(100, 45);
        pausePlayButton.addActionListener(new GamePausePlayListener());
        this.add(pausePlayButton);

    }

    public void makePlaySoundButton() {

        playSoundButton.setFont(defaultFont);
        playSoundButton.setSize(100, 50);
    }
}

```

```

        playSoundButton.setLocation(25, 150);
        playSoundButton.addActionListener(new AudioListener());
        this.add(playSoundButton);
    }

    public void makePauseSoundButton() {

        pauseSoundButton.setFont(defaultFont);
        pauseSoundButton.setSize(100, 50);
        pauseSoundButton.setLocation(175, 150);
        pauseSoundButton.addActionListener(new AudioListener());
        this.add(pauseSoundButton);
    }

    private class GamePausePlayListener implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent e) { // 버튼이 클릭되면

            JButton button = (JButton)e.getSource();

            if(button.getText().equals("Pause Game")) { // 게임 중지시키기
                button.setText("Play Game"); // 텍스트 바꾸기
                audio.stopAudio("gameBackground"); // 오디오 중지
                gamePanel.stopGame(); // 게임스레드, 타이머스레드 중지
                playSoundButton.setEnabled(false); // playSoundButton 비활성화
                pauseSoundButton.setEnabled(true); // pauseSoundButton 활성화
            }
            else { // 게임 이어서하기
                button.setText("Pause Game"); // 텍스트 바꾸기
                audio.playAudio("gameBackground"); // 오디오 시작
                gamePanel.resumeGame(); // 게임스레드, 타이머스레드 다시 시작
                playSoundButton.setEnabled(true); // playSoundButton 활성화
                pauseSoundButton.setEnabled(false); // pauseSoundButton 비활성화
            }
        }
    }

    private class AudioListener implements ActionListener {

        @Override

```

```

        public void actionPerformed(ActionEvent e) {
            switch(e.getActionCommand()) {
                case "Play Sound":
                    audio.playAudio("gameBackground"); // 오디오 시작
                    break;
                case "Pause Sound":
                    audio.stopAudio("gameBackground"); // 오디오 중지
                    break;
            }
        }
    }
}

```

StartFrame.java

```

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;

public class StartFrame extends JFrame {

    // 생성자에게 전달 위한 레퍼런스 선언
    private StartFrameNorthPanel startFrameNorthPanel = null;
    private StartFrameCenterPanel startFrameCenterPanel = null;
    private StartFrameSouthPanel startFrameSouthPanel = null;

    private Container contentPane;
    public Audio audio = new Audio();

    // 처음 실행되는 것인지 확인하는 변수
    private static boolean isFirstExecute = true;

    // 생성자
    public StartFrame() {

        if (isFirstExecute) { // 처음 실행 된다면 StartFrame 출력

```

```

        setTitle("Typing Game"); // Title 설정
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 창 닫히면 프로그램 종료

        contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());

        makeMenu(); // 메뉴바 부착
        makePanels(); // 패널들 부착

        setSize(900, 680);
        setVisible(true);
        setResizable(false);

        audio.playAudio("startFrame");

        isFirstExecute = false;
    }
}

private void makeMenu() {

    JMenuBar menuBar = new JMenuBar();
    this.setJMenuBar(menuBar);

    // 메뉴
    JMenu fileMenu = new JMenu("File");
    JMenu soundMenu = new JMenu("Sound");

    // 메뉴를 메뉴바에 부착
    menuBar.add(fileMenu);
    menuBar.add(soundMenu);

    // 메뉴 아이템
    JMenuItem wordsItem = new JMenuItem("words.txt");
    JMenuItem toicwordsItem = new JMenuItem("toicwords.txt");

    JMenuItem soundOnItem = new JMenuItem("sound on");
    JMenuItem soundOffItem = new JMenuItem("sound off");

    // 메뉴 아이템을 메뉴에 부착
    fileMenu.add(wordsItem);
    fileMenu.addSeparator();

```

```

fileMenu.add(toeicwordsItem);
soundMenu.add(soundOnItem);
soundMenu.addSeparator();
soundMenu.add(soundOffItem);

wordsItem.addActionListener(new ItemActionListener());
toeicwordsItem.addActionListener(new ItemActionListener());
soundOnItem.addActionListener(new ItemActionListener());
soundOffItem.addActionListener(new ItemActionListener());

}

class ItemActionListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {

        String cmd = e.getActionCommand(); // 메뉴 아이템의 문자열 리턴

        switch(cmd) {
            case "words.txt":
                GameManagement.fileName = ("words.txt");
                JOptionPane.showMessageDialog(contentPane, GameManagement.fileName
+ " 선택됨",

                "Message", JOptionPane.ERROR_MESSAGE);
                break;
            case "toeicwords.txt":
                GameManagement.fileName = ("toeicwords.txt");
                JOptionPane.showMessageDialog(contentPane, GameManagement.fileName
+ " 선택됨",

                "Message", JOptionPane.ERROR_MESSAGE);
                break;
            case "sound on":
                audio.playAudio("startFrame");
                JOptionPane.showMessageDialog(contentPane, "배경음악 시작",
                "Message", JOptionPane.ERROR_MESSAGE);
                break;
            case "sound off":
                audio.stopAudio("startFrame");
                JOptionPane.showMessageDialog(contentPane, "배경음악 중지",
                "Message", JOptionPane.ERROR_MESSAGE);
                break;
        }
    }
}

```

```

    }

}

private void makePanels() {

    startFrameNorthPanel = new StartFrameNorthPanel();
    startFrameCenterPanel = new StartFrameCenterPanel();
    startFrameSouthPanel = new StartFrameSouthPanel(this, audio);

    contentPane.add(startFrameNorthPanel, BorderLayout.NORTH);
    contentPane.add(startFrameCenterPanel, BorderLayout.CENTER);
    contentPane.add(startFrameSouthPanel, BorderLayout.SOUTH);

}
}

```

#### StartFrameCenterPanel.java

```

import java.awt.Color;
import java.awt.FlowLayout;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class StartFrameCenterPanel extends JPanel {

    // 생성자
    public StartFrameCenterPanel() {

        setLayout(new FlowLayout());
        setBackground(Color.YELLOW);

        ImageIcon mainImage = new ImageIcon("image/main/mainImage.png");
        JLabel mainImageLabel = new JLabel(mainImage);
        add(mainImageLabel);

    }

}

```

#### StartFrameNorthPanel.java

```

import java.awt.Color;

```

```

import java.awt.FlowLayout;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class StartFrameNorthPanel extends JPanel {

    // 생성자
    public StartFrameNorthPanel() {

        setLayout(new FlowLayout());
        setBackground(Color.YELLOW);

        ImageIcon mainTextImage = new ImageIcon("image/main/spongebobMainText.png");
        JLabel mainTextImageLabel = new JLabel(mainTextImage);
        add(mainTextImageLabel);

    }
}

```

StartFrameSouthPanel.java

```

import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import javax.swing.JButton;
import javax.swing.JPanel;

public class StartFrameSouthPanel extends JPanel {

    // 생성자에게 전달 위한 레퍼런스 선언
    private StartFrame startFrame;
    private Audio audio;

    // playersettingdialog 실행을 위한 레퍼런스 변수 선언
    private PlayerSettingDialog playerSettingDialog;

    JButton startButton = new JButton("Enter To Start"); // 버튼 생성
}

```

// 생성자

```
public StartFrameSouthPanel(StartFrame startFrame, Audio audio) {
```

```
    this.startFrame = startFrame;
```

```
    this.audio = audio;
```

```
    setLayout(new FlowLayout());
```

```
    setBackground(Color. YELLOW);
```

```
    makeStartButton();
```

} // 생성자 끝

```
private void makeStartButton() {
```

```
    startButton.setPreferredSize(new Dimension(190, 50));
```

```
    startButton.setFont(new Font("Jokerman", Font. BOLD, 15));
```

// 포커스 강제 설정

```
    startButton.setFocusable(true);
```

```
    startButton.requestFocus();
```

```
    startButton.addKeyListener(new KeyPressedListener());
```

```
    add(startButton);
```

```
}
```

```
private class KeyPressedListener extends KeyAdapter {
```

```
    @Override
```

```
    public void keyPressed(KeyEvent e) {
```

```
        if (e.getKeyChar() == '\n') { // 엔터를 받으면
```

```
            playersettingdialog = new PlayerSettingDialog(startFrame, audio); // 모달
```

다이얼로그 생성

```
            playersettingdialog.setVisible(true); // 모달 다이얼로그 보이게
```

```
        }
```

```
    }
```

```
}
```



```
}
```

UserRankingFrame.java

```
import java.awt.BorderLayout;
import java.awt.Container;

import javax.swing.JFrame;
import javax.swing.JSplitPane;

public class UserRankingFrame extends JFrame {

    // 생성자에게 전달 위한 레퍼런스 선언
    private ProfileAndScorePanel profileAndScorePanel = null;

    // PlayResultPanel과 RankingPanel 사용하기 위해 레퍼런스 변수 선언
    private PlayResultPanel playResultPanel;
    private RankingPanel rankingPanel;

    private Container contentPane;

    public UserRankingFrame(ProfileAndScorePanel profileAndScorePanel) {

        this.profileAndScorePanel = profileAndScorePanel;

        playResultPanel = new PlayResultPanel(this, profileAndScorePanel);
        rankingPanel = new RankingPanel();

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setTitle("게임 결과 확인");
        setSize(900, 680);

        contentPane = getContentPane();
        makeSplitPane();

        setVisible(true);
        setResizable(false);

    }

    // 스플릿팬 만들기
    private void makeSplitPane() {
```

```

        JSplitPane hPane = new JSplitPane();
        hPane.setOrientation(JSplitPane.HORIZONTAL_SPLIT); // 수평으로 분할
        hPane.setDividerLocation(400); // 디바이더 초기 위치 설정
        hPane.setDividerSize(0); // 스플릿팬 선 안보이게
        contentPane.add(hPane, BorderLayout.CENTER); // ContentPane 불러서 가운데에 부착

        hPane.setLeftComponent(playResultPanel); // playResultPanel을 hPane 왼쪽에 부착
        hPane.setRightComponent(rankingPanel); // rankingPanel을 hPane 오른쪽에 부착

    }
}

```

## Word.java

```

import javax.swing.JLabel;

public class Word {

    // 멤버 변수 선언
    private String name;
    private double x;
    private double y;
    private JLabel label;
    private double speed;

    // 생성자
    public Word(String word, int x, int y, double speed) {

        this.name = word;
        this.x = x;
        this.y = y;

        label = new JLabel(word);

        // 게임 난이도에 따라 단어가 떨어지는 속도 다름
        if (GameManagement.difficulty.equals("Normal"))
            this.speed = speed + Math.random() * 0.3;
        else if (GameManagement.difficulty.equals("Hard"))
            this.speed = speed + (Math.random() * 0.5 + 0.2);
        else
            this.speed = speed;
    }
}

```

```

    public String getName() { return name; }

    public double getX() { return x; }

    public double getY() { return y; }

    public double getSpeed() { return speed; }

    public JLabel getLabel() { return label; }

    public void setY(double y) { this.y = y; }

}

```

WordList.java

```

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;
import java.util.Vector;

public class WordList {

    private Vector<String> wordVector = new Vector<String>();

    public WordList() {

        // GameManagement.fileName가 없으면 catch로 이동해서 처리함
        try {

            Scanner scanner = new Scanner(new FileReader(GameManagement.fileName));
            // 성공적으로 읽었다면
            while(scanner.hasNext()) { // 파일 끝까지
                String word = scanner.nextLine(); // \n 빼고 리턴해줌
                wordVector.add(word);
            }
            scanner.close();
        } catch (FileNotFoundException e) { e.printStackTrace(); }

    }

    public String getWord() { // wordVector 중 하나를 random하게 리턴

        int index = (int)(Math.random()*wordVector.size());
    }
}

```

```
        return wordVector.get(index);  
  
    }  
}
```

## 5. 결론

자바를 공부하며 객체지향언어의 특징을 이해할 수 있었다. 미니 프로젝트는 나의 프로그래밍 실력을 향상하기에 좋은 기회였다고 생각한다. 자바에 대해 내가 무엇을 알고, 무엇을 모르는지 확인할 수 있는 가장 확실한 방법이었다. 스레드 부분이 꽤 어려웠지만 수업 시간에 실습으로 진행한 코드들이 많은 도움이 되었고, 모르는 것은 계속 책을 읽어보는 반복 학습을 통해 스레드와 가까워졌다. 주제는 교수님이 정해 주셨지만, 그 위에 무엇을 엮는가에 따라 결과물이 천차만별이 될 수 있었을 것이다. 내가 제안서에서 작성한 내용들을 대부분 구현했다. 아이디어, 프로그램 구조, UI 디자인 등을 하나부터 열까지 내가 해낸 것이 자랑스롭다. 하지만 막상 코드를 완성하니 스파게티처럼 얽혀 있는 느낌이 든다. 어떻게 하면 클래스 서로의 관계를 더 잘 드러내고 프로그램 구조를 간단하게 할 수 있을지 공부하고 이번 겨울방학에는 이 프로그램을 단순화 시킬 것이다.