# 객체지향언어2(7) 미니프로젝트

# 결과 보고서

## 다마고치 키우기 게임

2291031

모바일소프트웨어트랙

전아린

2023.12.12

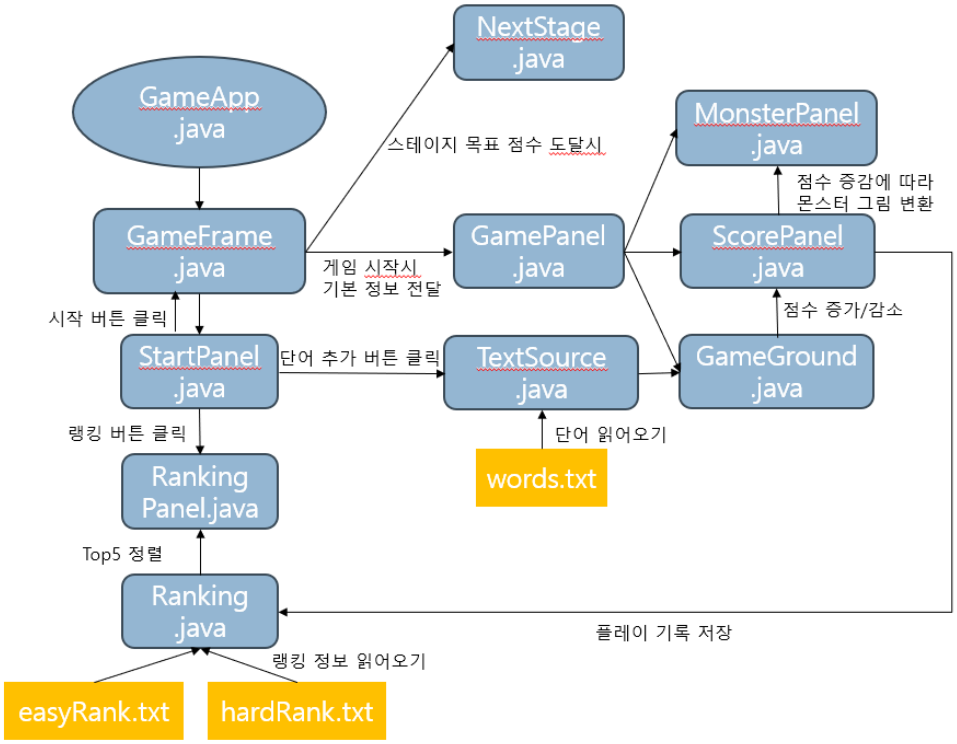
# 1. 작품 개요

게임을 실행하면 첫 시작 화면이 뜨고 배경음악이 재생된다. 처음 시작 화면에는 게임 시작 버튼, 단어장에 단어 추가 버튼, 랭킹 확인 버튼, 나가기 버튼이 있다. 단어 추가 버튼을 누르면 추가하고 싶은 단어를 입력할 수 있는 팝업창이 뜬다. 알파벳 외에 다른 글자를 입력하거나 이미 존재하는 단어일 경우 추가하지 않는다. 랭킹은 각 레벨별 top 5까지 출력한다. 만약 동일한 이름이 있을 경우 최고 점수 하나만 기록한다. 이름과 레벨을 입력하지 않고 게임 시작 버튼을 누를 시 입력하라는 메시지가 뜬다. 이름, 레벨 모두 입력하고 시작 버튼을 누르면 키우고 싶은 알을 고를 수 있는 화면으로 넘어간다. 키울 수 있는 알의 종류는 총 3가지이다. 알을 선택하면 게임 진행 화면으로 넘어간다. 화면 상단 툴바는 배경 음악을 조정할 수 있는 버튼들로 구성되어 있다. music/mute 버튼은 배경음악을 재생하거나 끌 수 있다. sound+/-버튼은 배경음을 높이거나 줄일 수 있다. exit 버튼을 누르면 게임을 종료한다.

단계는 총 2가지로 구성하였다. 쉬움 단계는 5자 이하의 단어가 나오고, 어려움 단계는 11자 이하의 단어가 나온다. 스테이지는 총 3개로 구성하였다. 스테이지 1은 최대 생명이 3개이고 0.5초 간격으로 단어가 떨어진다. 쉬움 단계의 경우 2.5초 간격, 어려움 단계는 2.2초 간격으로 단어가 생성된다. 스테이지 2는 최대 생명이 4개이고 0.4초 간격으로 단어가 떨어진다. 쉬움 단계의 경우 2.35초 간격, 어려움 단계는 2초 간격으로 단어가 생성된다. 스테이지 3은 최대 생명이 5개이고 0.35초 간격으로 단어가 떨어진다. 쉬움 단계의 경우 2.2초 간격, 어려움 단계는 1.8초 간격으로 단어가 생성된다. 단어를 맞췄을 경우 효과음이 재생된다. 60% 확률로 생성되는 사과 아이콘이 그려진 단어를 맞추면 +5점, 15% 확률로 생성되는 고기 아이콘은 +10점을 올려준다. 10% 확률로 만들어지는 시계 아이콘은 5초간 단어 생성과 단어 내려옴을 5초간 멈춘다. 10%로 생성되는 하트 아이콘은 현재 생명이 최대 생명 미만일 경우 생명을 +1하고, 5% 확률로 생성되는 컵케이크 아이콘은 10초 간 점수를 2배로 올려준다. 점수가 오르면 스테이지 1에서는 알이 좌우로 움직이고 스테이지 2 이상이면 몬스터가 웃으며 상하로 움직인다. 단어를 맞추지 못했을 경우 효과음과 함께 생명이 1 감소된다. 맞추지 못하면 스테이지 1에서는 알의 움직임을 멈추고, 스테이지 2 이상이면 몬스터가 눈물을 흘리고 움직임을 멈춘다. 스테이지 1은 100점, 2는 250점, 3은 400점에 도달할 경우 다음 스테이지로 넘어갈 수 있는 화면이 나온다. 스페이스바를 빠르게 눌러 바가 목표 지점까지 도달할 경우 다음 스테이지로 넘어갈 수 있는 버튼이 나온다. 버튼을 누르면 새로운 스테이지로 넘어갈 수 있다. 스테이지를 넘어갈 때마다 알->유년기->성년기 단계로 몬스터가 성장한다. 스테이지 3에서 Next Stage 버튼을 누르면 효과음과 함께 게임 클리어 화면이 출력된다. 게임 클리어 화면에서는 추가 스테이지로 넘어갈 수 있는 버튼, 게임 재시작 버튼, 게임을 종료하는 버튼이 존재한다. 추가 스테이지는 스테이지 3과 동일한 구성으로 이루어져 있다. 목숨이 0이 되면 효과음과 함께 게임 오버 화면이 나온다. 게임 오버 화면에서는 최종 점수와 게

임 재시작 버튼, 게임 종료 버튼이 존재한다. 게임 클리어 화면, 게임 오버 화면에서 재시작 버튼을 누르면 처음 화면으로 돌아간다.

## 2. 프로그램 구조





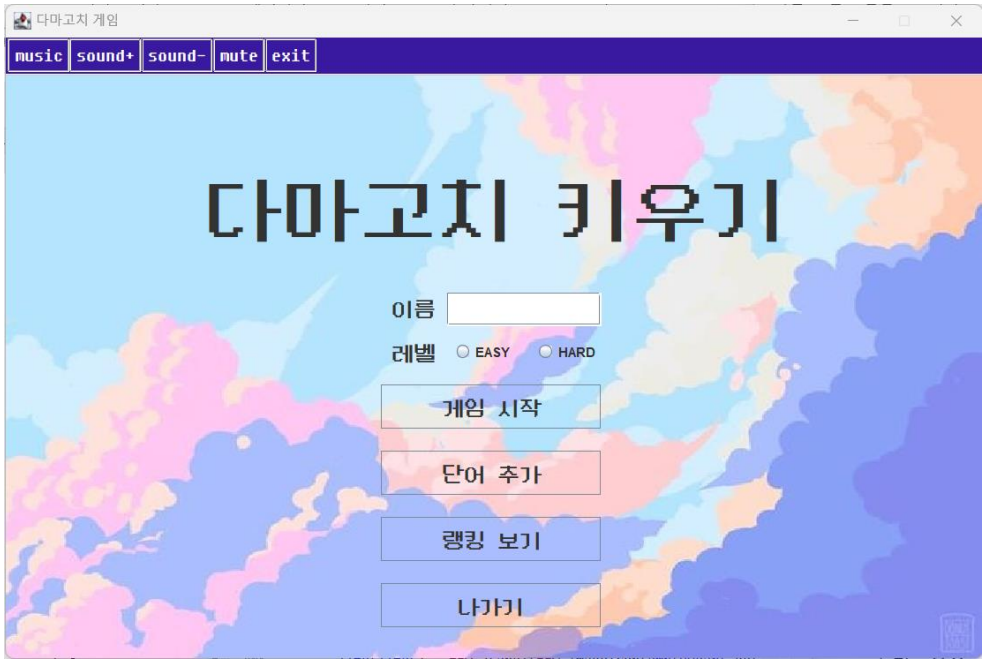
<스테이지2 이상부터 단어를 맞추면 happy, 맞추지 못하면 sad 이미지>

# 3. 프로그램 실행 과정

## 1. 시작 화면

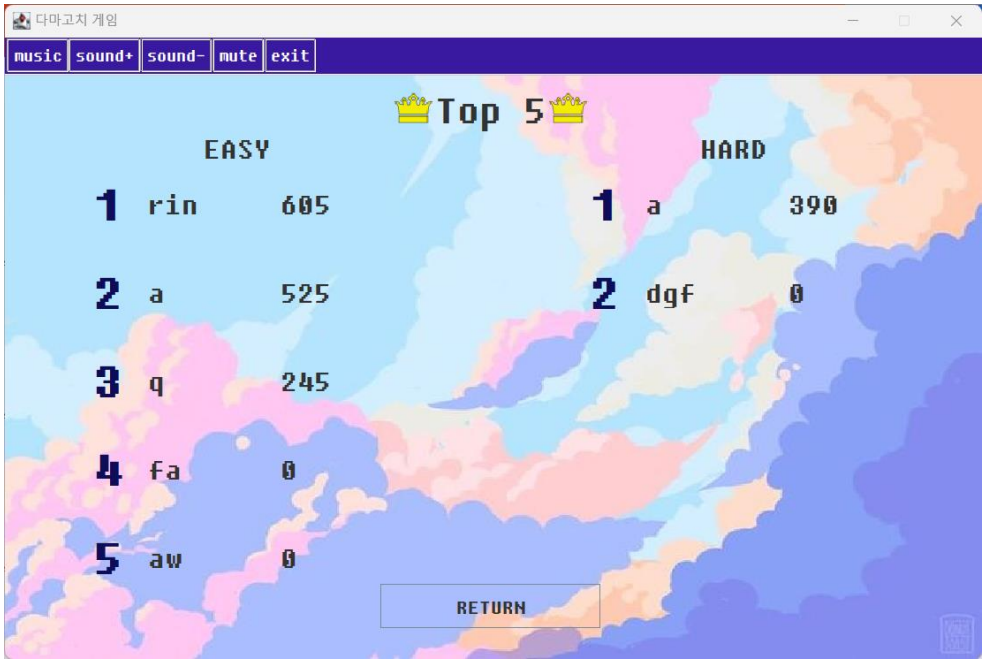
### 1-1) 시작 화면

: 왼쪽 상단의 exit 버튼이나 나가기 버튼을 누르면 프로그램을 종료한다. 프로그램 실행과 동시에 배경 사운드가 재생되고, music/mute 버튼을 눌러 배경 음악을 틀거나 끌 수 있다. sound+/- 버튼을 누르면 사운드 조정이 가능하다.



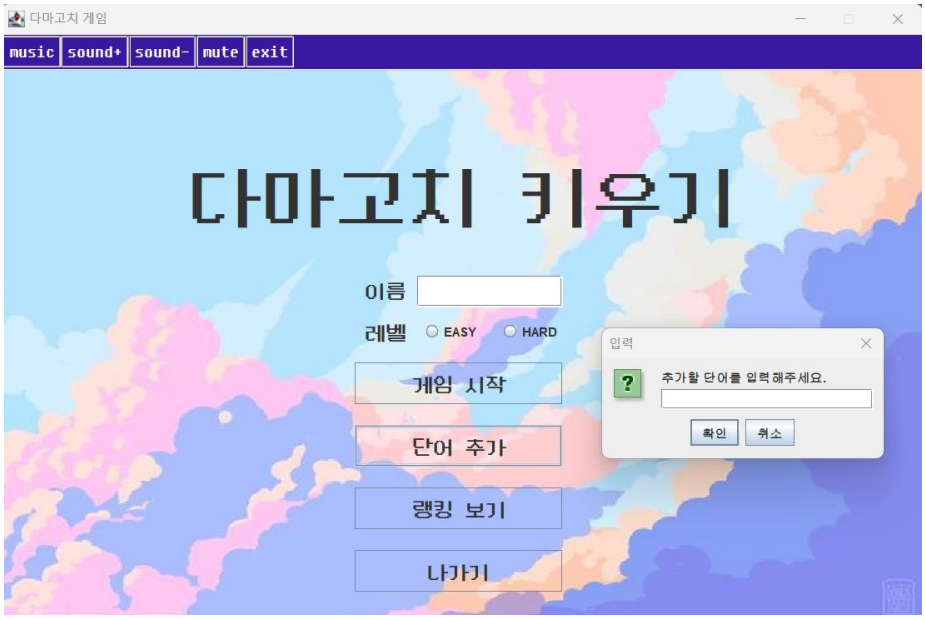

### 1-2) 랭킹

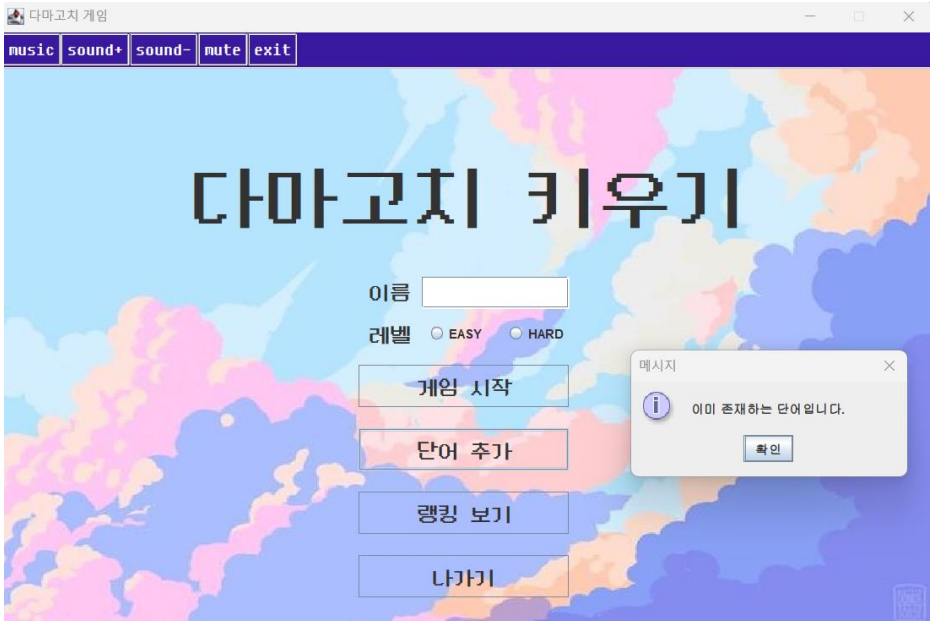
: 난이도별 top 5를 출력한다. 만약 같은 이름의 플레이어가 있을 시 플레이어의 최고 기록만 표시되게 한다. return 버튼을 누르면 시작 화면으로 돌아간다.




## 1-3) 단어 추가

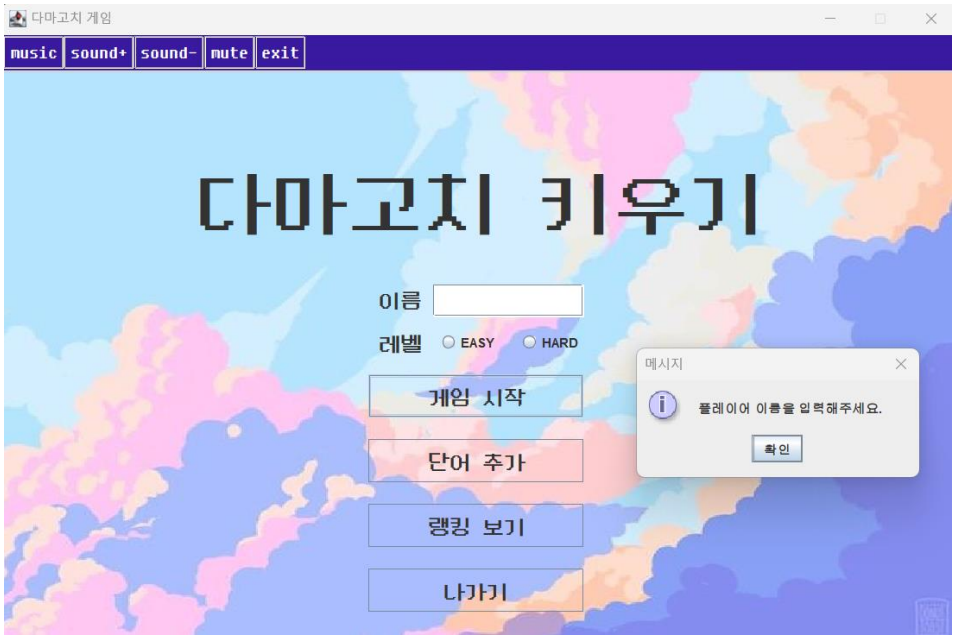
: 단어 추가 버튼을 누르면 단어장에 단어를 추가할 수 있다. 단어가 이미 단어장에 존재할 경우엔 추가하지 않는다.

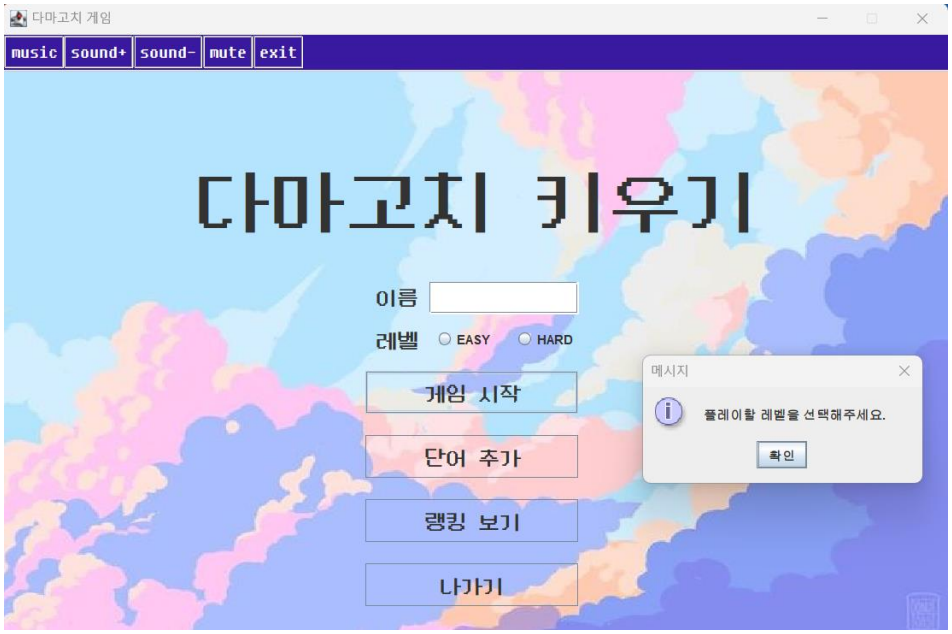


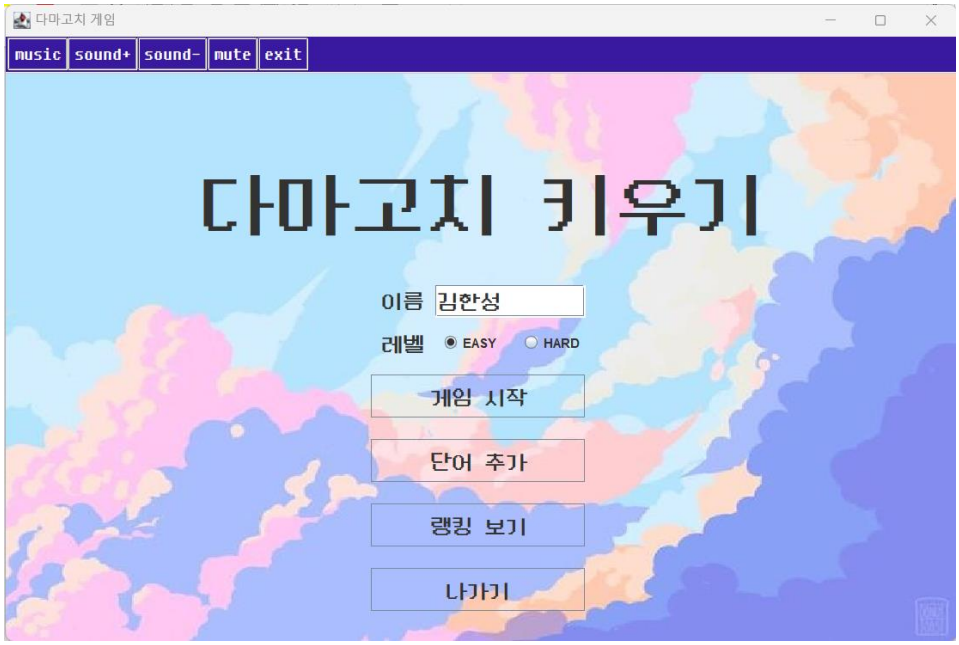



## 1-4) 게임 시작

: 이름과 플레이할 레벨을 선택하고 게임 시작 버튼을 누르면 게임을 시작할 수 있다. 만약 이름이나 레벨을 입력하지 않았을 시 입력하라는 메시지를 출력한다.





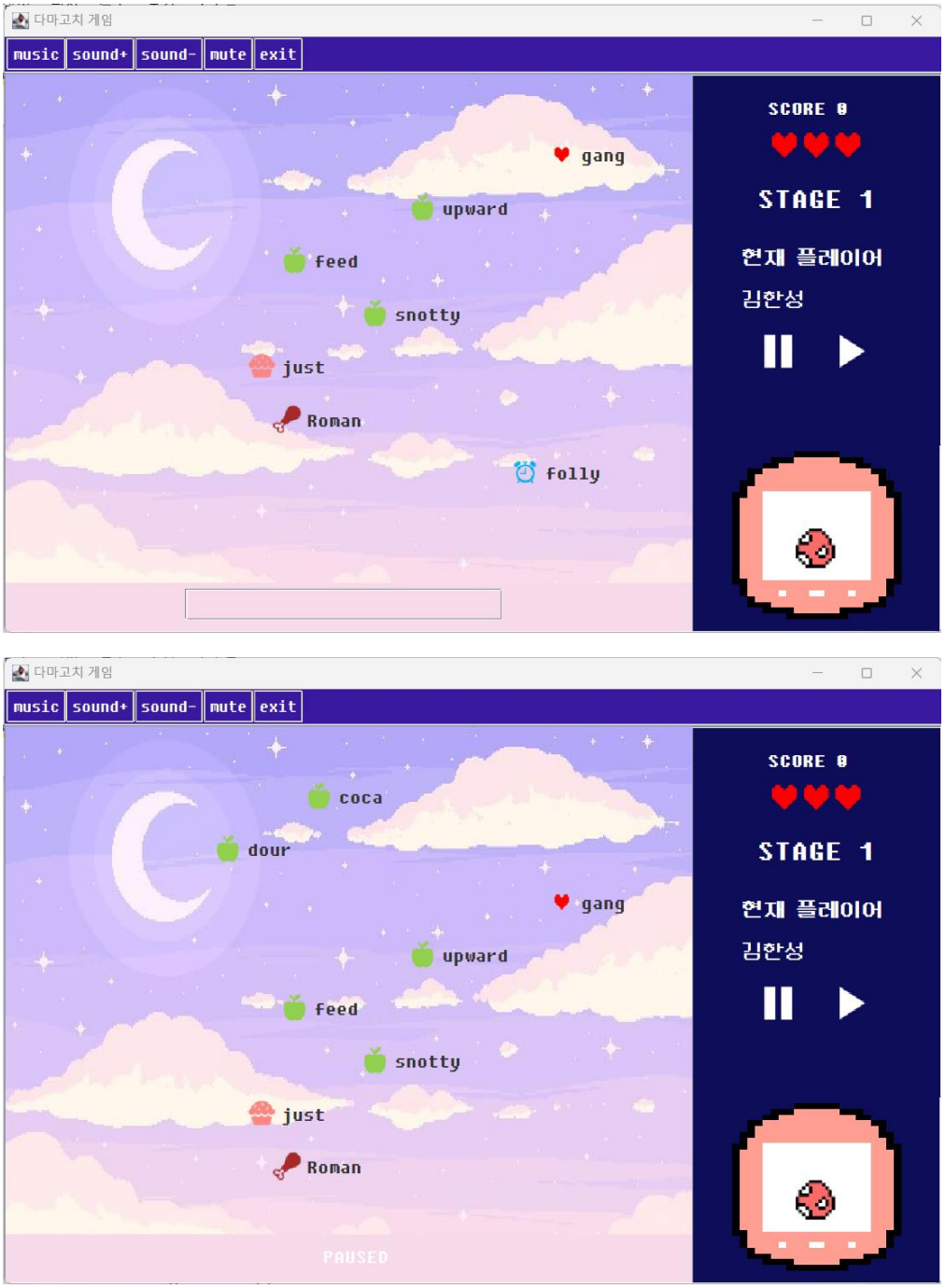

## 1-5) 알 선택 화면

: 사용자가 세개의 알 중 키우고 싶은 알을 하나 선택하면 본게임 화면으로 넘어간다.


## 2. 게임 플레이 화면

### 2-1) 스테이지 1

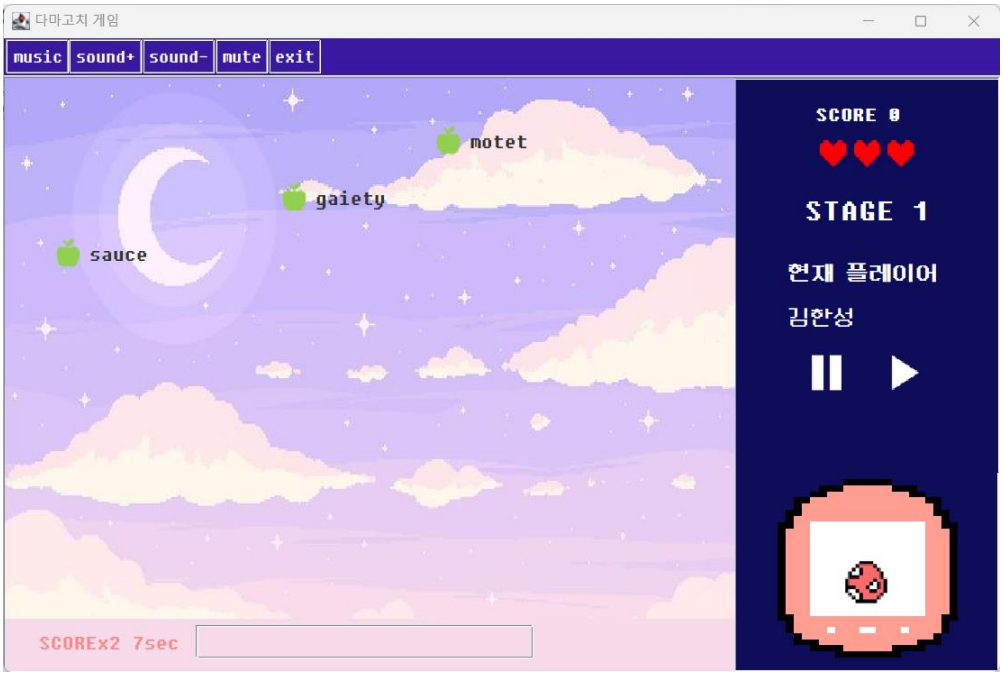
: 0.5초 간격으로 단어가 떨어진다. 쉬움 단계는 5자 이하의 단어, 어려움 단계는 11자 이하의 단어가 생성된다. 쉬움 단계일 경우 단어가 2.5초 간격으로 생성되고 어려움 단계일 경우 단어가 2.2초 간격으로 생성된다. 스테이지 1에서는 생명 3개가 주어진다. 오른쪽에 있는 pause 버튼을 누르면 게임이 중단되고 play 버튼을 누르면 다시 시작된다. 사과, 고기 아이콘 단어를 맞추면 점수가 각 5점/10점 증가한다. 시계 아이콘의 단어를 맞추면 단어 생성과 단어가 내려오는 걸 5초간 멈춘다. 하트 아이콘의 단어를 맞추면 생명이 1 증가하고, 생명이 이미 최대일 경우 증가하지 않는다. 컵케이크 아이콘의 단어를 맞출 시 10초간 점수 2배 이벤트가 발생한다. 100점에 도달할 경우 다음 스테이지로 넘어갈 수 있다. 스테이지 1에서는 점수가 올라갈 경우 알이 좌우로 움직인다.




<게임 정지 버튼을 눌렀을 때>

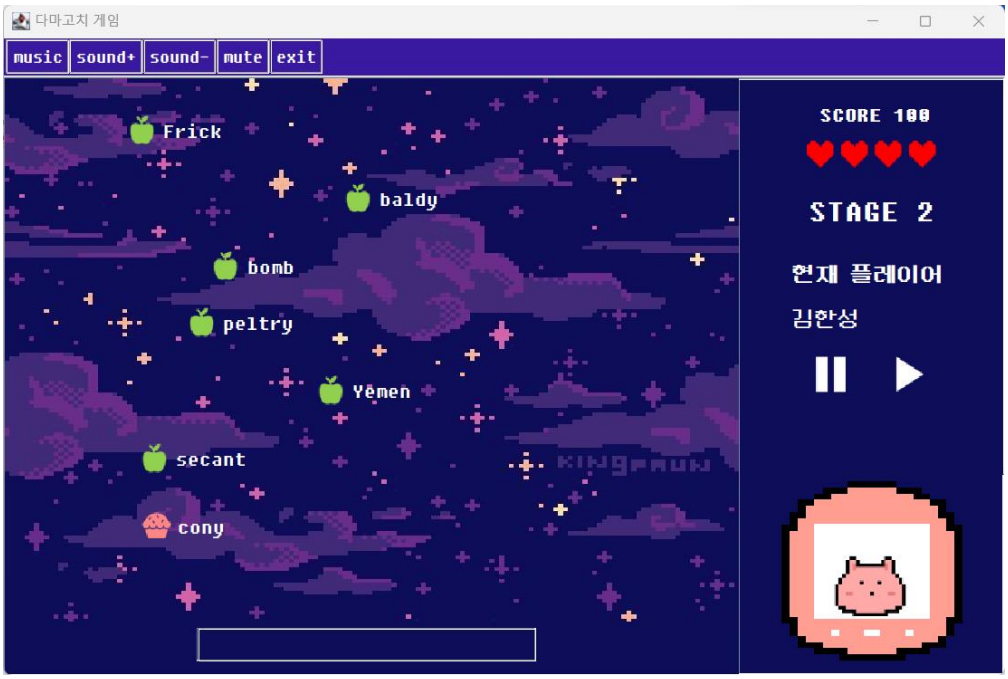
## 2-1-1) 컵케이크가 그려진 단어를 맞췄을 때

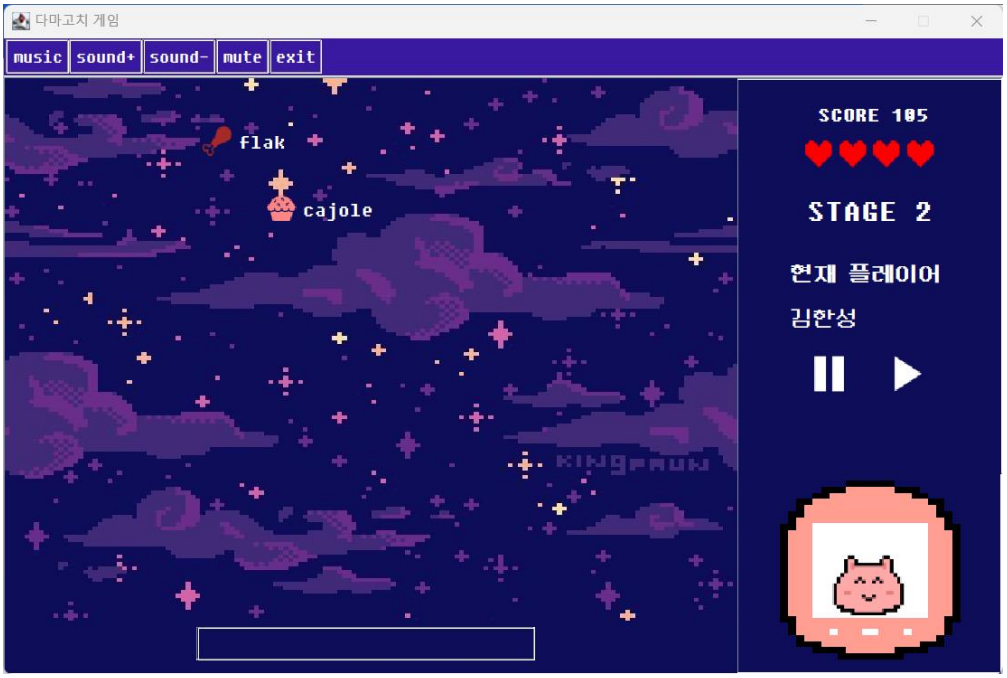
: 10초 동안 점수 2배 효과가 발생한다. 왼쪽 하단에 점수 2배 효과의 남은 시간이 표시된다.



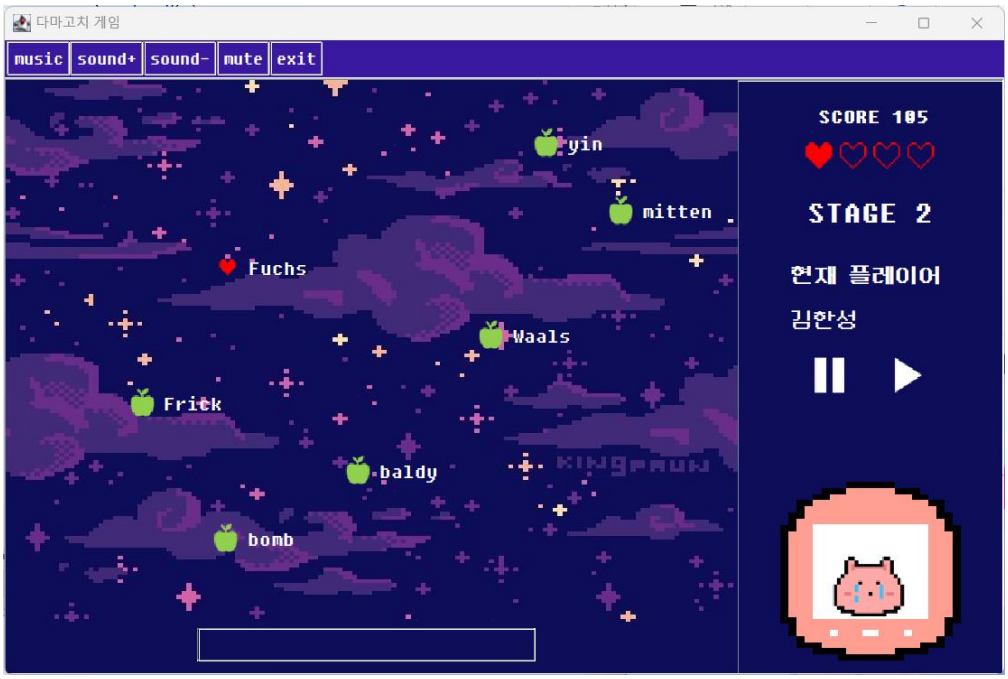

## 2-2) 스테이지 2

: 0.4초 간격으로 단어가 떨어진다. 쉬움 단계일 경우 단어가 2.35초 간격으로 생성되고 어려움 단계일 경우 단어가 2초 간격으로 생성된다. 스테이지 2에서는 생명 4개가 주어진다. 250점에 도달할 경우 다음 스테이지로 넘어갈 수 있다. 스테이지 2부터 생명이 깎일 경우 몬스터가 눈물을 흘린다. 점수를 얻을 경우에는 몬스터가 웃으며 상하로 움직인다.





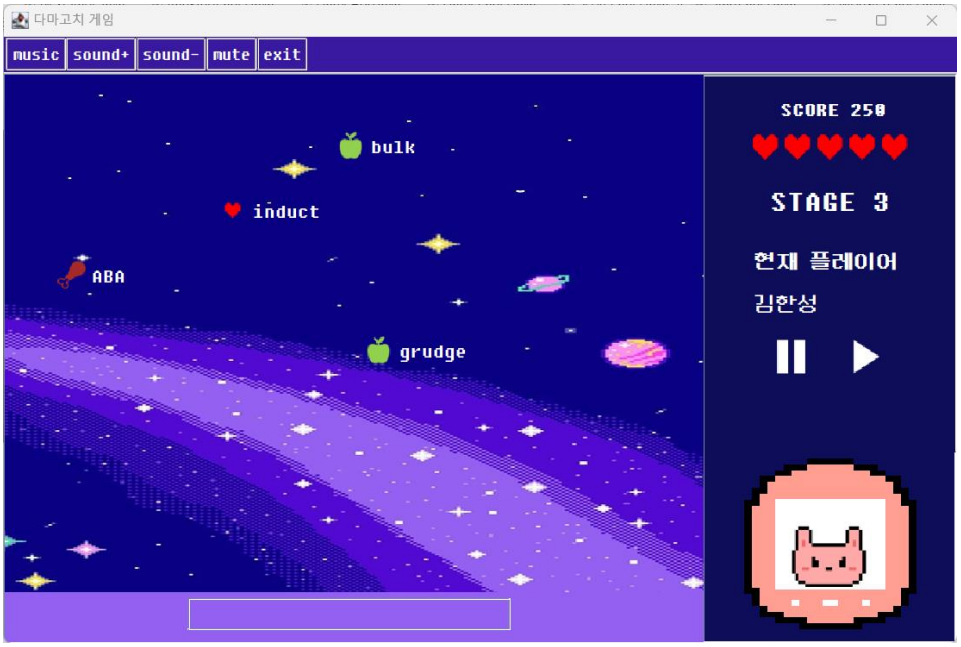

<점수를 얻었을 때>

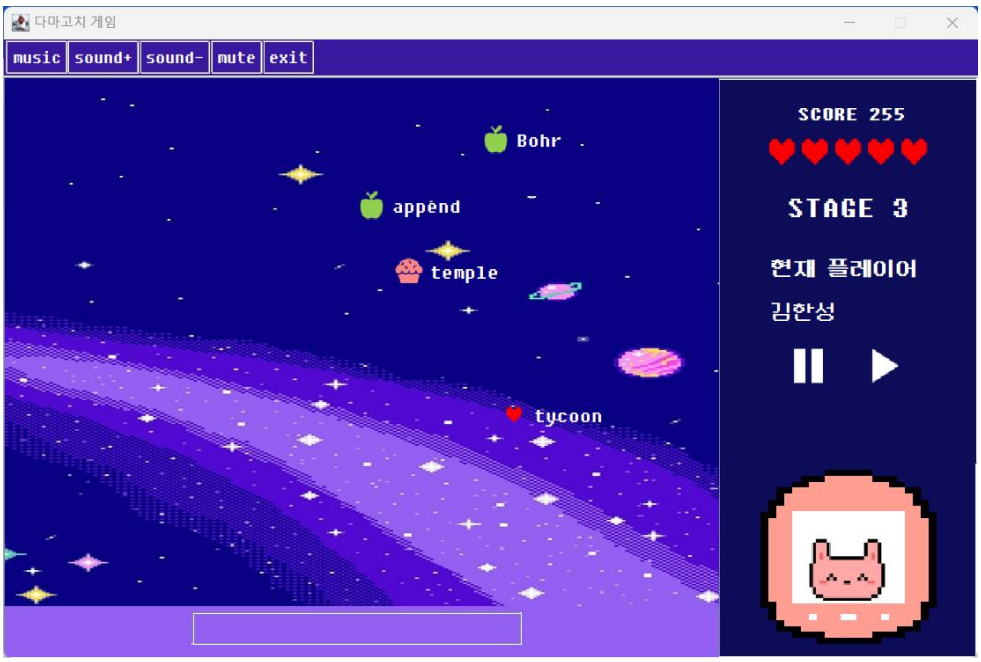



<생명이 깎였을 때>

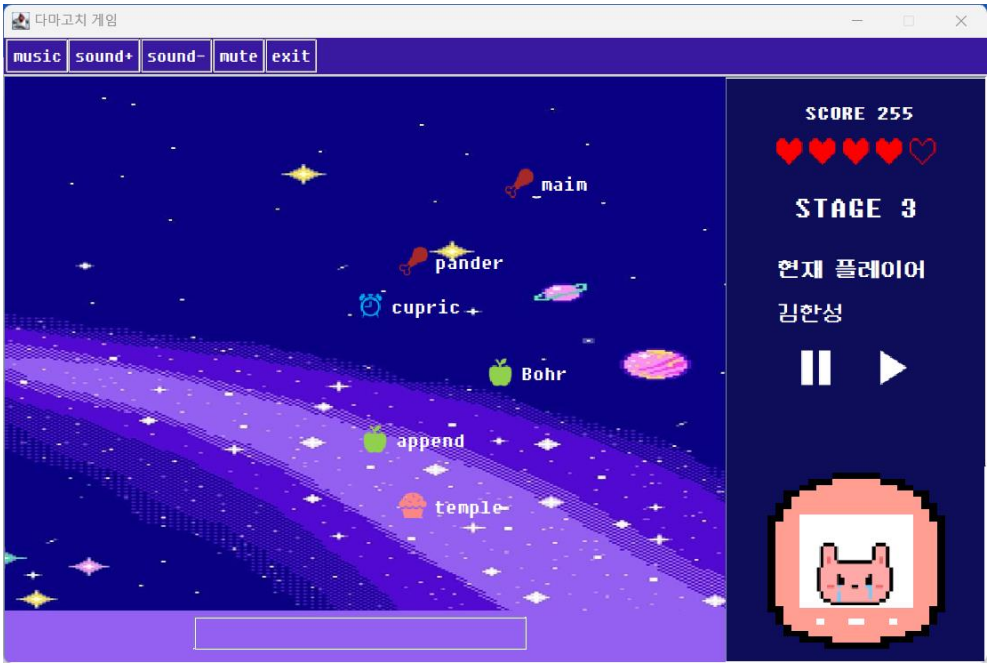
## 2-3) 스테이지 3

: 0.35초 간격으로 단어가 떨어진다. 쉬움 단계일 경우 단어가 2.2초 간격으로 생성되고 어려움 단계일 경우 단어가 1.8초 간격으로 생성된다. 스테이지 3에서는 생명 5개가 주어진다. 400점에 도달할 경우 다음 스테이지로 넘어갈 수 있다.


다마고치 게임
music
sound+
sound-
mute
exit
bulk
induct
ABA
grudge
SCORE 250
STAGE 3
현재 플레이어
김한성


다마고치 게임
music
sound+
sound-
mute
exit
Bohr
append
temple
tycoon
SCORE 255
STAGE 3
현재 플레이어
김한성

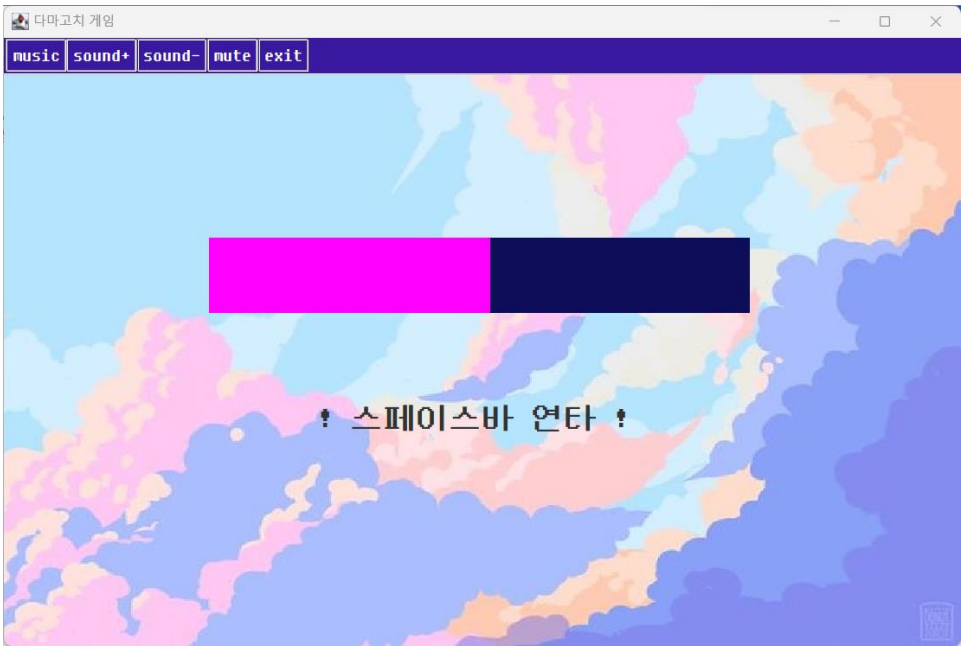
<점수를 얻었을 때>

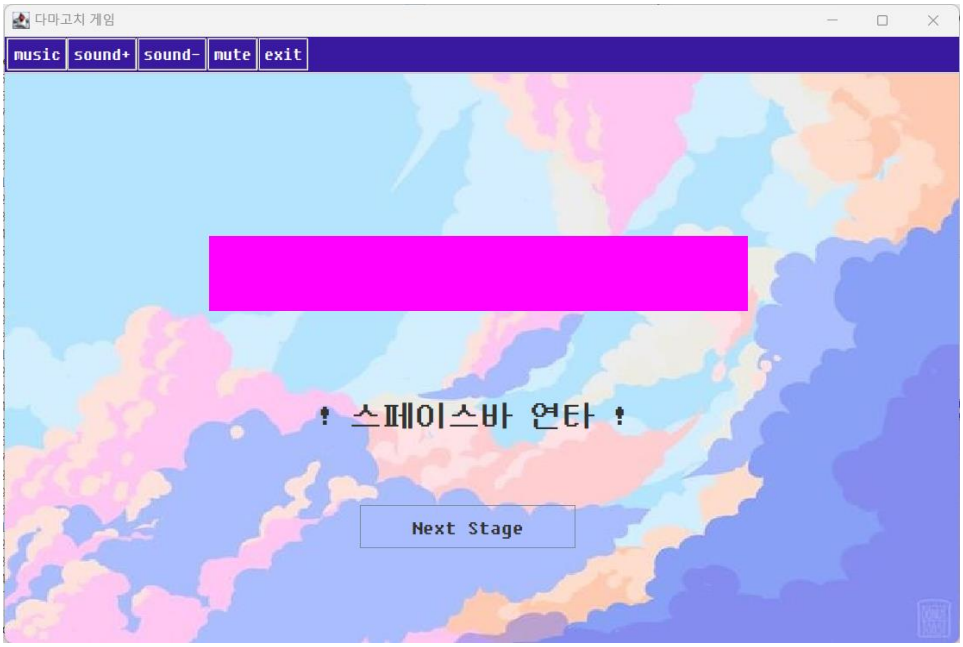



<생명이 깎였을 때>

## 2-4) 다음 스테이지로 넘어가기

: 스테이지1에서 100점, 스테이지2에서 250점, 스테이지3에서 400점에 도달할 경우 다음 스테이지로 넘어갈 수 있는 화면이 뜬다. 스페이스바를 연타하면 바가 채워지고, 바가 끝까지 다 채워지면 다음 스테이지로 넘어갈 수 있는 버튼이 나온다. 버튼을 클릭하면 스테이지 클리어 효과음과 함께 다음 스테이지가 나온다.





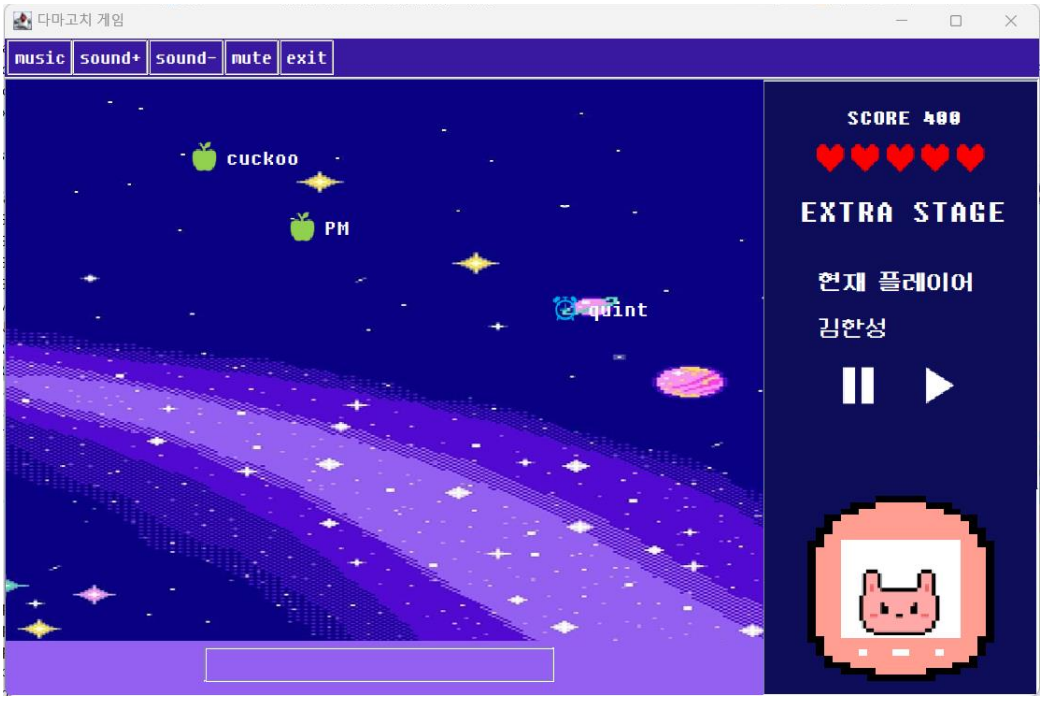

## 2-5) 게임 클리어

: 스테이지 3에서 Next Stage 버튼을 누르면 게임 클리어 효과음과 함께 게임 클리어 화면이 출력된다. EXTRA STAGE 버튼을 누를 경우 추가 스테이지가 실행되고, RESTART 버튼을 누르면 첫 시작 화면으로 돌아간다. EXIT 버튼을 누르면 프로그램을 종료한다.


## 2-6) 추가 스테이지

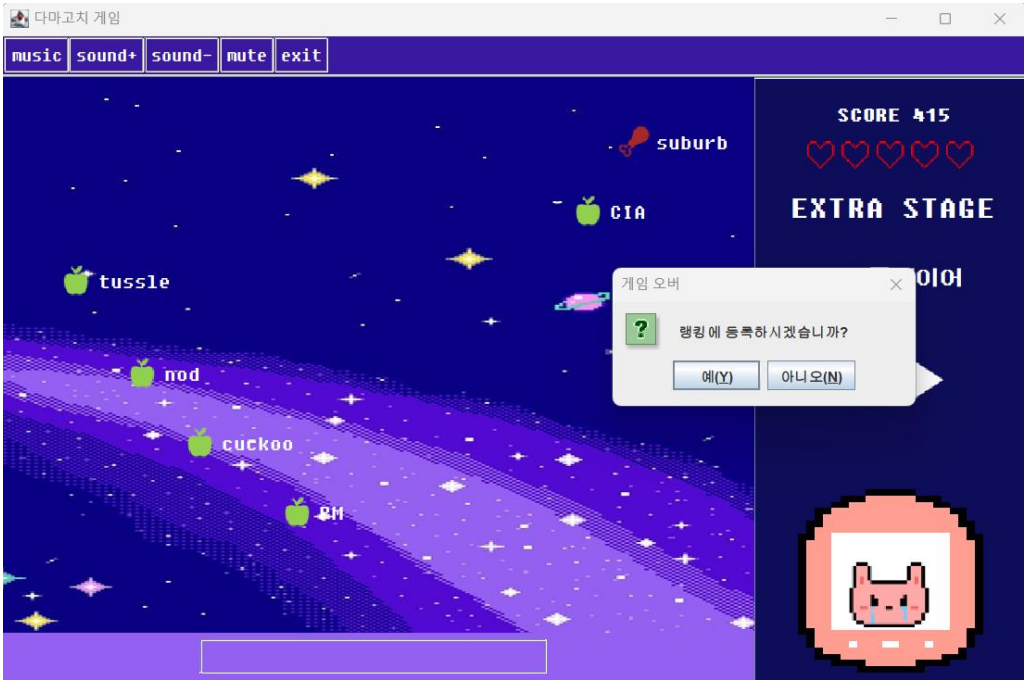
: EXTRA STAGE 버튼을 누르면 추가 스테이지로 넘어간다. 스테이지 3과 구성이 동일

하고, 게임 방식도 동일하다. 플레이어가 목숨을 다 잃을 때까지 실행된다.




## 2-7) 게임 오버

: 생명이 0이 되면 랭킹 등록 여부를 물어보는 메시지가 출력된다. 예를 누르면 랭킹 텍스트 파일에 기록된다. 예/아니오 중 하나를 누르면 게임오버 효과음과 함께 게임 오버 화면이 나온다. RESTART 버튼을 누르면 처음 시작화면으로 돌아가고, EXIT 버튼을 누르면 게임을 종료한다.




## 4. 프로그램 소스 코드

**GameApp.java**

```
public class GameApp {

    public static void main(String[] args) {
        new GameFrame();
    }

}
```

**GameFrame.java**

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.FloatControl;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;
import javax.swing.*;

public class GameFrame extends JFrame {
    private StartPanel startPanel;
```

```
    private GamePanel gamePanel;
    private int level, monsterIndex, stage;
    private String playerName;
    private Clip clip;
    private NextStage nextStage = new NextStage(this, stage);

    public GameFrame() {
        setTitle("다마고치 게임");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setBackground(new Color(56, 25, 160));
        toStartPanel();
        loadAudio();
        menu();
        setSize(900, 600);
        setResizable(false);
        setVisible(true);
    }

    // 게임 정보를 설정하는 메소드
    public void setGameInfo(int level, int stage, int monsterIndex,
String playerName) {
        this.level = level;
        this.stage = stage;
        this.monsterIndex = monsterIndex;
        this.playerName = playerName;
    }

    // 게임 정보(레벨, 스테이지, 몬스터, 플레이어 이름)를 리턴하는 메소드
    public int getLevel() {
        return level;
    }

    public int getStage() {
        return stage;
    }

    public int getMonster() {
        return monsterIndex;
    }

    public String getPlayerName() {
        return playerName;
    }

    // NextStage 패널을 보이게 하는 메소드
    public void toNextStage() {
        nextStage = new NextStage(this, stage);
        getContentPane().add(nextStage);
        remove(gamePanel);
        revalidate();
    }

    // GamePanel 을 보이게 하는 메소드
    public void toGamePanel() {
        // startPanel, nextStage 패널이 존재할 경우 지움
        if (startPanel != null) {
```

```
            remove(startPanel);
        }
        if (nextStage != null) {
            remove(nextStage);
        }
        gamePanel = new GamePanel(this, level, monsterIndex, playerName);
        getContentPane().add(gamePanel);
        revalidate();
    }

    // StartPanel을 보이게 하는 메소드
    public void toStartPanel() {
        // gamePanel, nextStage 패널이 존재할 경우 지움
        if (nextStage != null) {
            remove(nextStage);
        }
        if (gamePanel != null) {
            remove(gamePanel);
        }
        startPanel = new StartPanel(this);
        getContentPane().add(startPanel);
        revalidate();
        repaint();
    }

    private void menu() {
        JToolBar bar = new JToolBar("game");

        // 배경 음악을 재생하는 버튼
        JButton music = new JButton("music");
        music.setFont(new Font("DungGeunMo", Font.PLAIN, 18));
        music.setToolTipText("배경 음악을 틉니다.");
        music.setOpaque(false);
        music.setForeground(Color.white);
        music.setFocusPainted(false);
        music.setFocusable(false);
        music.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                clip.start();
            }
        });
        bar.add(music);

        // 배경 음악 소리를 키우는 버튼
        JButton soundUp = new JButton("sound+");
        soundUp.setFont(new Font("DungGeunMo", Font.PLAIN, 18));
        soundUp.setToolTipText("소리를 높입니다.");
        soundUp.setOpaque(false);
        soundUp.setForeground(Color.white);
        soundUp.setFocusPainted(false);
        soundUp.setFocusable(false);
        soundUp.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (clip != null &&
clip.isControlSupported(FloatControl.Type.MASTER_GAIN)) {
```

```
                    FloatControl gainControl = (FloatControl)
clip.getControl(FloatControl.Type.MASTER_GAIN);
                    float range = gainControl.getMaximum() -
gainControl.getMinimum();
                    float gain = (range * 10 / 100) +
gainControl.getValue(); // 볼륨 10% 증가
                    gain = Math.min(gain, gainControl.getMaximum());
                    gainControl.setValue(gain);
                }
            }
        });
        bar.add(soundUp);

        // 배경 음악 소리를 줄이는 버튼
        JButton soundDown = new JButton("sound-");
        soundDown.setFont(new Font("DungGeunMo", Font.PLAIN, 18));
        soundDown.setToolTipText("소리를 줄입니다.");
        soundDown.setOpaque(false);
        soundDown.setForeground(Color.white);
        soundDown.setFocusPainted(false);
        soundDown.setFocusable(false);
        soundDown.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (clip != null &&
clip.isControlSupported(FloatControl.Type.MASTER_GAIN)) {
                    FloatControl gainControl = (FloatControl)
clip.getControl(FloatControl.Type.MASTER_GAIN);
                    float range = gainControl.getMaximum() -
gainControl.getMinimum();
                    float gain = gainControl.getValue() - (range * 5 /
100); // 볼륨 10% 감소
                    gain = Math.max(gain, gainControl.getMinimum());
                    gainControl.setValue(gain);
                }
            }
        });
        bar.add(soundDown);

        // 배경 음악을 멈추는 버튼
        JButton mute = new JButton("mute");
        mute.setFont(new Font("DungGeunMo", Font.PLAIN, 18));
        mute.setToolTipText("배경 음악을 음소거합니다.");
        mute.setOpaque(false);
        mute.setForeground(Color.white);
        mute.setFocusPainted(false);
        mute.setFocusable(false);
        mute.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                clip.stop();
            }
        });
        bar.add(mute);

        // 게임 종료 버튼
        JButton exit = new JButton("exit");
```

```
        exit.setFont(new Font("DungGeunMo", Font.PLAIN, 18));
        exit.setToolTipText("게임을 종료합니다.");
        exit.setOpaque(false);
        exit.setForeground(Color.white);
        exit.setFocusPainted(false);
        exit.setFocusable(false);
        exit.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int choose = JOptionPane.showConfirmDialog(null, "정말
종료하시겠습니까?", "게임 종료", JOptionPane.YES_NO_OPTION);
                if (choose == JOptionPane.YES_OPTION)
                    System.exit(0);
            }
        });
        bar.add(exit);

        // 화면 상단에 고정
        bar.setFloatable(false);
        bar.setOpaque(false);
        add(bar, BorderLayout.NORTH);
    }

    // 사운드 무한 재생 메소드
    private void loadAudio() {
        try {
            clip = AudioSystem.getClip();
            File audioFile = new File("audio/startMusic.wav");
            AudioInputStream audioStream =
AudioSystem.getAudioInputStream(audioFile);
            clip.open(audioStream);
            clip.loop(Clip.LOOP_CONTINUOUSLY);
        } catch (LineUnavailableException e) {
            e.printStackTrace();
        } catch (UnsupportedAudioFileException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // 사운드 중단 메소드
    public void muteAudio() {
        clip.stop();
    }
}
```

**GameGround.java**

```
import java.awt.*;
import java.awt.event.*;
import java.util.Vector;
import javax.swing.*;
public class GameGround extends JPanel {
    private ScorePanel scorePanel;
    private TextSource textSource = null;
    private JTextField textInput = new JTextField(20);
```

```
    private Vector<JLabel> label = new Vector<JLabel>();
    private ImageIcon iconImage[] = new ImageIcon[5];
    private ImageIcon icon;
    private Image backgroundImg;
    private int x;
    private int wordIndex = 0;
    private int level;
    private int stage = 1;
    public GameThread gameth;
    public WordThread wordth;
    private DoubleScoreThread dst;
    private GameFrame gameFrame;
    // 단어 2배 이벤트 발생 시 true
    private boolean isDouble = false;
    public GameGround(ScorePanel scorePanel, GameFrame gameFrame, int
level, int stage) {
        textSource = new TextSource(this);
        this.scorePanel = scorePanel;
        this.gameFrame = gameFrame;
        this.level = level;
        this.stage = stage;
        setLayout(null);
        for (int i = 0; i < iconImage.length; i++) {
            iconImage[i] = new ImageIcon((i + 1) + ".png");
        }
        newWord(level);
        runGame();
    }
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (stage == 1) {
            icon = new ImageIcon("background/gameBackground1.jpg");
            g.setColor(new Color(248, 219, 234));
        } else if (stage == 2) {
            icon = new ImageIcon("background/gameBackground2.jpg");
            g.setColor(new Color(14, 13, 89));
        } else {
            icon = new ImageIcon("background/gameBackground3.jpg");
            g.setColor(new Color(148, 96, 241));
        }
        backgroundImg = icon.getImage();
        g.drawImage(backgroundImg, 0, 0, this.getWidth(),
this.getHeight(), this);
        g.fillRect(0, 480, this.getWidth(), this.getHeight());
    }
    // 단어를 생성하는 메소드
    public void newWord(int level) {
        label.add(new JLabel());
        String word = textSource.next();
        // 쉬움 단계일 경우 단어 길이 5자 이하로 제한
        if (level == 1) {
            while (true) {
                if (word.length() > 6) {
                    word = textSource.next();
                } else
                    break;
```

```
            }
        }
        // 어려움 단계일 경우 단어 길이 11자 이하로 제한
        else {
            while (true) {
                if (word.length() > 12) {
                    word = textSource.next();
                } else
                    break;
            }
        }
        int r = (int) (Math.random() * 100) + 1;
        int imageIndex = -1;
        // 60% 확률로 사과 아이콘
        if (r <= 60)
            imageIndex = 0;
        // 15% 확률로 고기 아이콘
        else if (r <= 75)
            imageIndex = 1;
        // 10% 확률로 시계 아이콘
        else if (r <= 85)
            imageIndex = 2;
        // 10% 확률로 생명 아이콘
        else if (r <= 95)
            imageIndex = 3;
        // 5% 확률로 컵케이크 아이콘
        else
            imageIndex = 4;
        Image image =
iconImage[imageIndex].getImage().getScaledInstance(30, 30,
Image.SCALE_SMOOTH);
        ImageIcon icon = new ImageIcon(image);
        if (level == 1) {
            x = (int) (Math.random() * 500) + 40;
        } else {
            x = (int) (Math.random() * 450) + 40;
        }
        // 랜덤한 단어를 만들고
        // 아이템 이미지를 아이콘에 달기
        JLabel wordLabel = label.get(wordIndex);
        wordLabel.setText(word);
        wordLabel.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        wordLabel.setSize(200, 30);
        wordLabel.setLocation(x, 20);
        if (stage >= 2) {
            wordLabel.setForeground(Color.white);
        }
        wordLabel.setName(Integer.toString(imageIndex));
        wordLabel.setIcon(icon);
        add(wordLabel);
        wordIndex++;
    }
    // 단어 게임 진행 메소드
    private void runGame() {
        // 단어를 입력할 텍스트 필드
        if (stage >= 2) {
            textInput.setForeground(Color.white);
```

```
        }
        textInput.setSize(300, 30);
        textInput.setLocation(170, 485);
        textInput.setBackground(new Color(14, 13, 89));
        textInput.setFont(new Font("DungGeunMo", Font.PLAIN, 18));
        textInput.setFocusable(true);
        textInput.setOpaque(false);
        add(textInput);
        textInput.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                boolean found = false;
                JTextField tf = (JTextField) e.getSource();
                String text = tf.getText();
                for (int i = 0; i < label.size(); i++) {
                    // 화면에 있는 단어와 입력한 단어가 일치할 경우
                    if (text.equals(label.get(i).getText())) {
                        boolean stageClear = false;
                        int score = 0;
                        String imageIndex = label.get(i).getName();
                        // 입력한 단어의 아이콘에 따라 결정
                        switch (imageIndex) {
                        // 사과 아이콘
                        case "0":
                            if (isDouble)
                                score = 10;
                            else
                                score = 5;
                            break;
                        // 고기 아이콘
                        case "1":
                            if (isDouble)
                                score = 30;
                            else
                                score = 15;
                            break;
                        // 시계 아이콘
                        case "2":
                            gameth.stopRain(false);
                            wordth.stopWord(false);
                            break;
                        // 생명 아이콘
                        case "3":
                            scorePanel.addLife();
                            break;
                        // 컵케이크 아이콘
                        case "4":
                            dst.doubleScore();
                            break;
                        }
                        stageClear = scorePanel.increase(score, stage);
                        // 단어 숨기기
                        label.get(i).setVisible(false);
                        found = true;
                        // 스테이지를 클리어했을 경우 스레드를 멈춤
                        if (stageClear) {
                            gameth.interrupt();
```

```
                            wordth.interrupt();
                            gameFrame.toNextStage();
                        }
                    }
                }
                // 화면에 있는 단어와 입력한 단어가 일치하지 않을 경우
                if (!found) {
                    boolean isOver;
                    isOver = scorePanel.decrease(stage);
                    if (isOver) {
                        gameth.interrupt();
                        wordth.interrupt();
                    }
                }
                tf.setText("");
            }
        });
        // 스레드 작동 시작
        gameth = new GameThread(scorePanel, stage);
        gameth.start();
        wordth = new WordThread(stage);
        wordth.start();
        dst = new DoubleScoreThread();
        dst.start();
    }
    // 단어를 밑으로 내리는 스레드
    class GameThread extends Thread {
        private boolean stopFlag = false;
        private ScorePanel scorePanel;
        private int stage;
        private boolean pause = false;
        private JLabel pauseInfo;
        public GameThread(ScorePanel scorePanel, int stage) {
            this.scorePanel = scorePanel;
            this.stage = stage;
        }
        // 내려오는 단어를 멈추는 메소드
        public void stopRain(boolean pause) {
            this.pause = pause;
            stopFlag = true;
            // pause가 true일 경우 게임을 일시정지 시킴
            if (pause == true) {
                remove(textInput);
                pauseInfo = new JLabel("PAUSED");
                pauseInfo.setFont(new Font("DungGeunMo", Font.PLAIN,
20));
                pauseInfo.setSize(80, 20);
                pauseInfo.setLocation(300, 490);
                pauseInfo.setForeground(Color.WHITE);
                add(pauseInfo);
                repaint();
            }
        }
        // 단어가 다시 내려오게 만드는 메소드
        synchronized public void resumeRain() {
            stopFlag = false;
            if (pause) {
```

```
                add(textInput);
                remove(pauseInfo);
                pause = false;
                repaint();
            }
        }
        // 내려오는 단어 5초간 멈추는 메소드
        synchronized private void checkWait() {
            if (stopFlag && !pause) {
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e) {
                }
                resumeRain();
            }
        }
        @Override // 스레드 무한루프
        public void run() {
            while (true) {
                checkWait();
                if (!stopFlag) {
                    for (int i = 0; i < label.size(); i++) {
                        // 단어가 바닥까지 내려왔을 경우
                        if (label.get(i).getY() > 420 &&
label.get(i).isVisible()) {
                            label.get(i).setText("");
                            label.get(i).setVisible(false);
                            boolean isOver = scorePanel.decrease(stage);
                            // 게임오버일 경우 스레드를 멈춤
                            if (isOver) {
                                wordth.interrupt();
                                gameth.interrupt();
                            }
                        } else {

label.get(i).setLocation(label.get(i).getX(), label.get(i).getY() + 10);
                        }
                    }
                }
                try {
                    // 스테이지 1일 경우 0.5초 간격
                    if (stage == 1) {
                        Thread.sleep(500);
                    }
                    // 스테이지 2일 경우 0.4초 간격
                    else if (stage == 2) {
                        Thread.sleep(400);
                    }
                    // 스테이지 3일 경우 0.35초 간격
                    else {
                        Thread.sleep(350);
                    }
                } catch (InterruptedException e) {
                    stopFlag = true;
                    return;
                }
            }
```

```
        }
    }
    // 단어를 새로 만드는 스레드
    class WordThread extends Thread {
        private boolean stopFlag = false;
        private int stage;
        private boolean pause = false;
        public WordThread(int stage) {
            this.stage = stage;
        }
        // 단어 생성을 멈추는 메소드
        public void stopWord(boolean pause) {
            this.pause = pause;
            stopFlag = true;
        }
        // 단어 생성을 다시 시작하는 메소드
        synchronized public void resumeWord() {
            stopFlag = false;
            if (pause)
                pause = false;
        }
        // 단어 생성 5초간 멈추는 메소드
        synchronized private void checkWait() {
            if (stopFlag && !pause) {
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e) {
                }
                resumeWord();
            }
        }
        @Override
        public void run() {
            try {
                while (true) {
                    checkWait();
                    if (!stopFlag) {
                        // 쉬움 단계
                        if (level == 1) {
                            // 스테이지 1일 경우 2.5초 간격
                            if (stage == 1) {
                                Thread.sleep(2500);
                            }
                            // 스테이지 2일 경우 2.35초 간격
                            else if (stage == 2) {
                                Thread.sleep(2350);
                            }
                            // 스테이지 3일 경우 2.2초 간격
                            else {
                                Thread.sleep(2200);
                            }
                        }
                        // 어려움 단계
                        else {
                            // 스테이지 1일 경우 2.2초 간격
                            if (stage == 1) {
                                Thread.sleep(2200);
```

```
                                }
                                // 스테이지 2일 경우 2초 간격
                                else if (stage == 2) {
                                     Thread.sleep(2000);
                                }
                                // 스테이지 3일 경우 1.8초 간격
                                else {
                                     Thread.sleep(1800);
                                }
                           }

                           // 새 단어 생성
                           synchronized (label) {
                              newWord(level);
                           }
                      }
                 }
            } catch (InterruptedException e) {
                 stopFlag = true;
                 return;
            }
       }
  }
  // 10초동안 점수 2배 스레드
  class DoubleScoreThread extends Thread {
       private boolean stopFlag = true;
       private int count = 0;
       private JLabel leftTime = new JLabel();
       // 스레드를 wait 상태로 만드는 메소드
       synchronized public void stopDoubleScore() {
            stopFlag = true;
            leftTime.setVisible(false);
            isDouble = false;
            try {
                 wait();
            } catch (InterruptedException e) {
            }
       }
       // 스레드를 Runnable 상태로 만드는 메소드
       synchronized public void doubleScore() {
            stopFlag = false;
            leftTime.setVisible(true);
            isDouble = true;
            count += 10;
            notify();
       }
       // 10초간 점수를 2배로 처리하게 만드는 메소드
       synchronized private void checkWait() {
            if (!stopFlag) {
                 try {
                      remove(leftTime);
                      // 점수 2배 효과의 남은 시간을 알려줌
                      leftTime.setText("SCOREx2 " +
Integer.toString(count) + "sec");
                      leftTime.setForeground(new Color(253, 135, 135));
                      leftTime.setFont(new Font("DungGeunMo", Font.PLAIN,
20));
```

```
                    leftTime.setSize(200, 20);
                    leftTime.setLocation(30, 490);
                    add(leftTime);
                    revalidate();
                    repaint();
                    count--;
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                }
            }
        }
        @Override
        public void run() {
            while (true) {
                checkWait();
                // 10초가 지나면 스레드를 wait 상태로 바꿈
                if (count == 0) {
                    stopDoubleScore();
                }
            }
        }
    }
}
```

**GamePanel.java**

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class GamePanel extends JPanel {
    public ScorePanel scorePanel;
    private GameFrame gameFrame;
    private MonsterPanel monsterPanel;
    public GameGround gameGround;
    private JSplitPane hPane = new JSplitPane();
    private JSplitPane vPane = new JSplitPane();
    public GamePanel(GameFrame gameFrame, String playerName) {
        this.gameFrame = gameFrame;
        setLayout(new BorderLayout());
        splitPanel();
    }
    // GamePanel 영역 나누기
    private void splitPanel() {
        hPane.setOrientation(JSplitPane.HORIZONTAL_SPLIT);
        hPane.setDividerLocation(650);
        hPane.setDividerSize(0);
        add(hPane);
        vPane.setOrientation(JSplitPane.VERTICAL_SPLIT);
        vPane.setDividerLocation(350);
        // GameFrame으로부터 정보를 얻어와 각 패널 생성
        monsterPanel = new MonsterPanel(gameFrame.getStage(),
gameFrame.getMonster());
        scorePanel = new ScorePanel(gameFrame.getLevel(),
gameFrame.getStage(), gameFrame.getPlayerName(), monsterPanel, this);
        gameGround = new GameGround(scorePanel, gameFrame,
gameFrame.getLevel(), gameFrame.getStage());
        // 오른쪽 위 점수 패널 / 오른쪽 아래 몬스터 패널
```

```
        // 왼쪽 본게임 패널
        vPane.setTopComponent(scorePanel);
        vPane.setBottomComponent(monsterPanel);
        hPane.setRightComponent(vPane);
        hPane.setLeftComponent(gameGround);
        vPane.setDividerSize(0);
    }
    // 게임 오버 화면
    public void gameOver(int score) {
        setLayout(null);
        remove(hPane);
        remove(vPane);
        JLabel ending = new JLabel("Game Over!");
        ending.setFont(new Font("DungGeunMo", Font.PLAIN, 60));
        ending.setSize(400, 60);
        ending.setLocation(300, 100);
        add(ending);
        // 최종 점수 출력
        JLabel gameOverScore = new JLabel("SCORE " +
Integer.toString(score));
        gameOverScore.setFont(new Font("DungGeunMo", Font.PLAIN, 45));
        gameOverScore.setSize(300, 45);
        gameOverScore.setLocation(350, 200);
        add(gameOverScore);
        // 게임 재시작 버튼
        JButton restartButton = new JButton("RESTART");
        restartButton.setLocation(340, 400);
        restartButton.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        restartButton.setOpaque(false);
        restartButton.setBackground(new Color(0, 0, 0, 0));
        restartButton.setSize(200, 40);
        restartButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                gameFrame.toStartPanel();
            }
        });
        add(restartButton);
        // 게임 종료 버튼
        JButton exitButton = new JButton("EXIT");
        exitButton.setLocation(340, 460);
        exitButton.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        exitButton.setOpaque(false);
        exitButton.setBackground(new Color(0, 0, 0, 0));
        exitButton.setSize(200, 40);
        exitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        add(exitButton);
        revalidate();
        repaint();
    }
    @Override
    public void paintComponent(Graphics g) {
```

```
        super.paintComponent(g);
        ImageIcon icon = new
ImageIcon("background/startBackground.jpg");
        Image backgroundImg = icon.getImage();
        g.drawImage(backgroundImg, 0, 0, this.getWidth(),
this.getHeight(), this);
    }
}
```

**MonsterPanel.java**

```
import java.awt.*;
import javax.swing.*;
public class MonsterPanel extends JPanel {
    private ImageIcon background = new ImageIcon("dama.png");
    private Image backgroundImg = background.getImage();
    private ImageIcon babyImg[] = new ImageIcon[3];
    private ImageIcon adultImg[] = new ImageIcon[3];
    private ImageIcon monster;
    private JLabel monsterImg;
    private int stage;
    public MonsterThread mt;
    public MonsterPanel(int stage, int monsterIndex) {
        setLayout(null);
        this.stage = stage;
        // 스테이지 1일 경우 플레이어가 선택한 알 사진 저장
        if (stage == 1) {
            monster = new ImageIcon("monster/egg" + (monsterIndex + 1) +
".png");
            monsterImg = new JLabel(new
ImageIcon(monster.getImage().getScaledInstance(100, 100,
Image.SCALE_SMOOTH)));
        }
        // 스테이지 2일 경우 선택한 알의 유년기 사진 저장
        else if (stage == 2) {
            babyImg[0] = new ImageIcon("monster/baby" + (monsterIndex +
1) + ".png");
            babyImg[1] = new ImageIcon("monster/happybaby" +
(monsterIndex + 1) + ".png");
            babyImg[2] = new ImageIcon("monster/sadbaby" + (monsterIndex
+ 1) + ".png");
            monsterImg = new JLabel(
                    new
ImageIcon(babyImg[0].getImage().getScaledInstance(100, 100,
Image.SCALE_SMOOTH)));
        }
        // 스테이지 3 또는 추가 스테이지일 경우
        // 선택한 알의 성년기 사진 저장
        else {
            adultImg[0] = new ImageIcon("monster/adult" + (monsterIndex
+ 1) + ".png");
            adultImg[1] = new ImageIcon("monster/happyadult" +
(monsterIndex + 1) + ".png");
            adultImg[2] = new ImageIcon("monster/sadadult" +
(monsterIndex + 1) + ".png");
            monsterImg = new JLabel(
```

```
                    new
ImageIcon(adultImg[0].getImage().getScaledInstance(100, 100,
Image.SCALE_SMOOTH)));
        }
        monsterImg.setLocation(65, 50);
        monsterImg.setSize(100, 100);
        add(monsterImg);
        mt = new MonsterThread(stage);
        mt.start();
    }
    // 몬스터의 이미지를 바꾸는 메세지
    public void changeImg(int index) {
        // 스테이지 2일 경우 유년기 사진 바꾸기
        if (stage == 2) {
            monsterImg
                    .setIcon(new
ImageIcon(babyImg[index].getImage().getScaledInstance(100, 100,
Image.SCALE_SMOOTH)));
        }
        // 스테이지 3 또는 추가 스테이지일 경우 성년기 사진 바꾸기
        else if (stage >= 3) {
            monsterImg
                    .setIcon(new
ImageIcon(adultImg[index].getImage().getScaledInstance(100, 100,
Image.SCALE_SMOOTH)));
        }
        // 인덱스가 1일 경우(맞았을 경우)
        // 몬스터가 움직이게 함
        if (index == 1) {
            mt.startMove();
        }
        // 인덱스가 0이거나(점수 증가 x) 2일 경우(틀렸을 경우)
        // 몬스터 움직임을 멈춤
        else if (index == 0 || index == 2) {
            mt.stopMove();
        }
        monsterImg.repaint();
    }
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        setBackground(new Color(14, 13, 89));
        g.drawImage(backgroundImg, 0, 0, this.getWidth(),
this.getHeight(), this);
        g.setColor(Color.white);
        g.fillRect(65, 43, 103, 84);
    }
    // 몬스터의 움직임을 구현하는 메소드
    class MonsterThread extends Thread {
        private boolean stopFlag = true;
        private int stage;
        private int count;
        public MonsterThread(int stage) {
            this.stage = stage;
        }
        // 몬스터가 움직이게 하는 메소드
        synchronized public void startMove() {
```

```
            stopFlag = false;
            notify();
        }
        // 몬스터의 움직임을 멈추는 메소드
        public void stopMove() {
            stopFlag = true;
        }
        // stopFlag가 true일 경우
        // 스레드를 wait 상태로 만드는 메소드
        synchronized private void checkWait() {
            if (stopFlag) {
                try {
                    count = 0;
                    if (stage == 1) {
                        monsterImg.setLocation(65, monsterImg.getY());
                    } else {
                        monsterImg.setLocation(monsterImg.getX(), 50);
                    }
                    wait();
                } catch (InterruptedException e) {
                    return;
                }
            }
        }
        @Override
        public void run() {
            count = 0;
            while (true) {
                checkWait();
                if (!stopFlag) {
                    // 스테이지 1일 경우 좌우로 이동
                    if (stage == 1) {
                        if (count % 2 == 0) {
                            monsterImg.setLocation(60,
monsterImg.getY());
                        } else {
                            monsterImg.setLocation(70,
monsterImg.getY());
                        }
                    }
                    // 스테이지 2 이상일 경우 위아래로 이동
                    else if (stage >= 2) {
                        if (count % 2 == 0) {
                            monsterImg.setLocation(monsterImg.getX(),
monsterImg.getY() - 10);
                        } else {
                            monsterImg.setLocation(monsterImg.getX(),
monsterImg.getY() + 10);
                        }
                    }
                    count++;
                    try {
                        sleep(300);
                    } catch (InterruptedException e) {
                        stopFlag = true;
                        return;
                    }
```

```
                }
            }
        }
    }
}
```

**NextStage.java**

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

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;
import javax.swing.*;

public class NextStage extends JPanel {
    private BarLabel bar = new BarLabel();
    private GameFrame gameFrame;
    private Color purple = new Color(14, 13, 89);
    private NextStage nextStage = this;
    private int stage;
    private BarThread bt = new BarThread(bar);
    private JButton nextButton = new JButton("Next Stage");
    private int maxBarSize = 500;
    private Clip clip;

    public NextStage(GameFrame gameFrame, int stage) {
        this.gameFrame = gameFrame;
        this.stage = stage;
        setLayout(null);
        setBackground(new Color(255, 171, 228));
        bar.setOpaque(true);
        bar.setBackground(purple);
        bar.setLocation(190, 150);
        bar.setSize(500, 70);
        add(bar);

        JLabel info = new JLabel("! 스페이스바 연타 !");
        info.setFont(new Font("DungGeunMo", Font.PLAIN, 30));
        info.setSize(300, 30);
        info.setLocation(290, 300);
        add(info);

        // 스페이스바를 누르면 fill() 메소드 실행
        setFocusable(true);
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                int key = e.getKeyCode();
                if (key == KeyEvent.VK_SPACE) {
                    bar.fill();
                }
            }
        });
```

```
        // 누르면 다음 스테이지로 넘어가는 버튼
        nextButton.setLocation(330, 400);
        nextButton.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        nextButton.setOpaque(false);
        nextButton.setBackground(null);
        nextButton.setSize(200, 40);
        nextButton.setVisible(false);
        nextButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // 스테이지 3 에서 넘어가는 경우 게임 클리어 화면 출력
                if (stage == 3) {
                    loadAudio("audio/gameClear.wav");
                    gameClear();
                } else {
                    // 게임 정보 재설정(현 stage+1) 후 다음 스테이지로 넘어감
                    gameFrame.setGameInfo(gameFrame.getLevel(),
gameFrame.getStage() + 1, gameFrame.getMonster(),
                            gameFrame.getPlayerName());
                    gameFrame.toGamePanel();
                }
            }
        });
        add(nextButton);

        // 바 스레드 실행
        bt.start();
        setVisible(true);
    }

    // 게임 클리어 메소드
    public void gameClear() {
        removeAll();
        JLabel gameClear = new JLabel("GAME CLEAR !");
        gameClear.setFont(new Font("DungGeunMo", Font.PLAIN, 40));
        gameClear.setSize(260, 40);
        gameClear.setLocation(310, 100);
        add(gameClear);

        // 행복한 유년기 몬스터와 성년기 몬스터 출력
        int monsterIndex = gameFrame.getMonster();
        ImageIcon babyIcon = new ImageIcon("monster/happybaby" +
(monsterIndex + 1) + ".png");
        ImageIcon adultIcon = new ImageIcon("monster/happyadult" +
(monsterIndex + 1) + ".png");
        JLabel babyImg = new JLabel(new
ImageIcon(babyIcon.getImage().getScaledInstance(200, 200,
Image.SCALE_SMOOTH)));
        JLabel adultImg = new JLabel(
                new ImageIcon(adultIcon.getImage().getScaledInstance(200,
200, Image.SCALE_SMOOTH)));

        adultImg.setSize(200, 200);
        adultImg.setLocation(400, 240);
        add(babyImg);
```

```
        babyImg.setSize(200, 200);
        babyImg.setLocation(240, 150);
        add(adultImg);

        // 추가 스테이지로 넘어갈 수 있는 버튼
        JButton extraStageButton = new JButton("EXTRA STAGE");
        extraStageButton.setLocation(110, 410);
        extraStageButton.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        extraStageButton.setOpaque(false);
        extraStageButton.setBackground(new Color(0, 0, 0, 0));
        extraStageButton.setSize(200, 40);
        extraStageButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // 게임 정보 재설정(현 stage+1) 후 추가 스테이지로 넘어감
                gameFrame.setGameInfo(gameFrame.getLevel(),
gameFrame.getStage() + 1, gameFrame.getMonster(),
                        gameFrame.getPlayerName());
                gameFrame.toGamePanel();
            }
        });
        add(extraStageButton);

        // 게임을 재시작할 수 있는 버튼
        JButton restartButton = new JButton("RESTART");
        restartButton.setLocation(330, 410);
        restartButton.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        restartButton.setOpaque(false);
        restartButton.setBackground(new Color(0, 0, 0, 0));
        restartButton.setSize(200, 40);
        restartButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                gameFrame.toStartPanel();
            }
        });
        add(restartButton);

        // 나가기 버튼
        JButton exitButton = new JButton("EXIT");
        exitButton.setLocation(550, 410);
        exitButton.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        exitButton.setOpaque(false);
        exitButton.setBackground(new Color(0, 0, 0, 0));
        exitButton.setSize(200, 40);
        exitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        add(exitButton);

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

```
    class BarLabel extends JLabel {
        int barSize = 0;

        // 바 채우기 메소드
        synchronized public void fill() {
            barSize += 30;
            repaint();
            notify();
        }

        // 바 비우기 메소드
        synchronized public void consume() {
            // 바 크기가 0 이면 스레드를 wait 상태로 바꿈
            if (barSize == 0) {
                try {
                    wait();
                } catch (InterruptedException e) {
                }
            }
            barSize -= 10;
            repaint();
            notify();
        }

        @Override // 현재 사이즈에 맞게 바 그리기
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.MAGENTA);
            int width = (int) (((double) (this.getWidth())) / maxBarSize *
barSize);
            if (width == 0)
                return;
            g.fillRect(0, 0, width, 70);
        }

        // 바의 현재 사이즈를 리턴하는 메소드
        public int getBarSize() {
            return barSize;
        }
    }

    class BarThread extends Thread {
        private BarLabel bar;

        public BarThread(BarLabel bar) {
            this.bar = bar;
        }

        // 바 스레드를 wait 상태로 바꾸는 메소드
        synchronized public void stopBar() {
            try {
                wait();
            } catch (InterruptedException e) {
            }
        }

        @Override
```

```
        public void run() {
            while (true) {
                try {
                    // 현재 바의 크기가 최대 바 사이즈와 같거나 클 경우
                    // 스레드를 멈추고 다음 스테이지로 넘어가는 버튼을 보이게
함
                    if (bar.getBarSize() >= maxBarSize) {
                        loadAudio("audio/stageClear.wav");
                        nextButton.setVisible(true);
                        revalidate();
                        repaint();
                        stopBar();
                        this.interrupt();
                        return;
                    }
                    // 0.1초 간격으로 바 크기를 줄임
                    sleep(100);
                    bar.consume();
                } catch (InterruptedException e) {
                    return;
                }
            }
        }
    }

    // 사운드 재생 메소드
    private void loadAudio(String fileName) {
        try {
            clip = AudioSystem.getClip();
            File audioFile = new File(fileName);
            AudioInputStream audioStream =
AudioSystem.getAudioInputStream(audioFile);
            clip.open(audioStream);
            clip.start();
        } catch (LineUnavailableException e) {
            e.printStackTrace();
        } catch (UnsupportedAudioFileException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        ImageIcon background = new
ImageIcon("background/startBackground.jpg");
        Image backgroundImg = background.getImage();
        g.drawImage(backgroundImg, 0, 0, this.getWidth(),
this.getHeight(), this);
    }
}
```

**Ranking.java**

```
import java.io.*;
```

```java
import java.util.*;

import javax.swing.JOptionPane;

public class Ranking {
    private FileWriter fout;
    private Vector<String> easyNames = new Vector<String>();
    private Vector<String> easyScores = new Vector<String>();
    private Vector<String> hardNames = new Vector<String>();
    private Vector<String> hardScores = new Vector<String>();
    private Vector<String> easyRank = new Vector<String>();
    private Vector<String> hardRank = new Vector<String>();
    private Scanner easyList;
    private Scanner hardList;

    // 텍스트 파일에 저장되어 있는 정보를 벡터에 저장하기
    public Ranking() {
        try {
            easyList = new Scanner(new FileReader("easyRank.txt"));
            while (easyList.hasNext()) {
                easyNames.add(easyList.nextLine());
                easyScores.add(easyList.nextLine());
            }
            hardList = new Scanner(new FileReader("hardRank.txt"));
            while (hardList.hasNext()) {
                hardNames.add(hardList.nextLine());
                hardScores.add(hardList.nextLine());
            }
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(null, "파일 없음");
            return;
        }
        rank();
    }

    // 텍스트 파일에 랭킹 정보 추가하기
    public void addRanking(int level, String playerName, int score) {
        if (level == 1) {
            try {
                fout = new FileWriter("easyRank.txt", true);
                fout.write(playerName + "\n" + score + "\n");
                fout.close();
            } catch (IOException e) {
                JOptionPane.showMessageDialog(null, "파일 없음");
            }
        } else {
            try {
                fout = new FileWriter("hardRank.txt", true);
                fout.write(playerName + "\n" + score + "\n");
                fout.close();
            } catch (IOException e) {
                JOptionPane.showMessageDialog(null, "파일 없음");
            }
        }
    }

    // top 5 정렬
```

```
    private void rank() {
        int max, maxIndex, count = 0;
        boolean alreadyExists;
        easyRank.clear();
        hardRank.clear();
        while (true) {
            alreadyExists = false;
            max = 0;
            maxIndex = 0;
            // 등록된 기록이 5개 미만일 경우
            if (easyNames.size() == 0)
                break;
            else if (count == 5)
                break;
            for (int i = 0; i < easyNames.size(); i++) {
                int score = Integer.parseInt(easyScores.get(i));
                if (max < score) {
                    max = score;
                    maxIndex = i;
                }
            }

            // 같은 이름이 랭킹에 존재하는지 확인
            for (int i = 0; i < easyRank.size(); i++) {
                String info = easyRank.get(i);
                StringTokenizer st = new StringTokenizer(info, ",");
                String rankName = st.nextToken();
                if (easyNames.get(maxIndex).equals(rankName)) {
                    alreadyExists = true;
                }
            }

            // 같은 이름이 존재하지 않을 경우에만 top 5 랭킹에 추가
            if (!alreadyExists) {
                count++;
                easyRank.add(easyNames.get(maxIndex) + "," +
Integer.toString(max));
            }

            easyNames.remove(maxIndex);
            easyScores.remove(maxIndex);
        }
        count = 0;
        while (true) {
            alreadyExists = false;
            max = 0;
            maxIndex = 0;
            // 등록된 기록이 5개 미만일 경우
            if (hardNames.size() == 0)
                break;
            else if (count == 5)
                break;
            for (int i = 0; i < hardNames.size(); i++) {
                int score = Integer.parseInt(hardScores.get(i));
                if (max < score) {
                    max = score;
                    maxIndex = i;
```

```
                }
            }

            // 같은 이름이 랭킹에 존재하는지 확인
            for (int i = 0; i < hardRank.size(); i++) {
                String info = hardRank.get(i);
                StringTokenizer st = new StringTokenizer(info, ",");
                String rankName = st.nextToken();
                if (hardNames.get(maxIndex).equals(rankName)) {
                    alreadyExists = true;
                }
            }

            // 같은 이름이 존재하지 않을 경우에만 top 5 랭킹에 추가
            if (!alreadyExists) {
                count++;
                hardRank.add(hardNames.get(maxIndex) + "," +
Integer.toString(max));
            }

            hardNames.remove(maxIndex);
            hardScores.remove(maxIndex);
        }

    }

    // top 5 정보를 저장한 벡터 리턴하기
    public Vector<String> getRank(int level) {
        if (level == 1)
            return easyRank;
        else
            return hardRank;
    }
}
```

## RankingPanel.java

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;

import javax.swing.*;

public class RankingPanel extends JPanel {
    private Ranking r = new Ranking();
    private Color purple = new Color(14, 13, 89);
    private StartPanel startPanel;

    // 각 레벨의 top 5 플레이어 출력
    public RankingPanel(StartPanel startPanel) {
        this.startPanel = startPanel;
        setLayout(null);

        JLabel ranking = new JLabel("Top 5");
        ranking.setFont(new Font("DungGeunMo", Font.PLAIN, 40));
        ranking.setLocation(390, 10);
        ranking.setSize(200, 40);
```

```
        add(ranking);

        ImageIcon crownIcon = new ImageIcon("crown.png");
        JLabel crown1 = new JLabel(new
ImageIcon(crownIcon.getImage().getScaledInstance(40, 40,
Image.SCALE_SMOOTH)));
        crown1.setSize(40, 40);
        crown1.setLocation(350, 10);
        add(crown1);
        JLabel crown2 = new JLabel(new
ImageIcon(crownIcon.getImage().getScaledInstance(40, 40,
Image.SCALE_SMOOTH)));
        crown2.setSize(40, 40);
        crown2.setLocation(490, 10);
        add(crown2);

        JLabel easy = new JLabel("EASY");
        easy.setFont(new Font("DungGeunMo", Font.PLAIN, 30));
        easy.setSize(100, 30);
        easy.setLocation(180, 50);
        add(easy);

        JLabel hard = new JLabel("HARD");
        hard.setFont(new Font("DungGeunMo", Font.PLAIN, 30));
        hard.setSize(100, 30);
        hard.setLocation(630, 50);
        add(hard);

        // top 5 정렬한 벡터 리턴 받기
        Vector<String> easyRank = r.getRank(1);
        Vector<String> hardRank = r.getRank(2);

        for (int i = 0; i < easyRank.size(); i++) {
            JLabel num = new JLabel(Integer.toString(i + 1));
            num.setFont(new Font("DungGeunMo", Font.BOLD, 50));
            num.setForeground(purple);
            num.setSize(30, 50);
            num.setLocation(80, 90 + (80 * i));
            add(num);

            // 문자열에서 이름과 점수 분리 후 출력
            String info = easyRank.get(i);
            StringTokenizer st = new StringTokenizer(info, ",");
            JLabel name = new JLabel(st.nextToken());
            name.setSize(100, 30);
            name.setFont(new Font("DungGeunMo", Font.PLAIN, 30));
            name.setLocation(130, 100 + (80 * i));
            add(name);

            JLabel score = new JLabel(st.nextToken());
            score.setSize(100, 30);
            score.setFont(new Font("DungGeunMo", Font.PLAIN, 30));
            score.setLocation(250, 100 + (80 * i));
            add(score);
        }

        for (int i = 0; i < hardRank.size(); i++) {
```

```
            JLabel num = new JLabel(Integer.toString(i + 1));
            num.setFont(new Font("DungGeunMo", Font.BOLD, 50));
            num.setForeground(purple);
            num.setSize(30, 50);
            num.setLocation(530, 90 + (80 * i));
            add(num);

            // 문자열에서 이름과 점수 분리 후 출력
            String info = hardRank.get(i);
            StringTokenizer st = new StringTokenizer(info, ",");
            JLabel name = new JLabel(st.nextToken());
            name.setSize(100, 30);
            name.setFont(new Font("DungGeunMo", Font.PLAIN, 30));
            name.setLocation(580, 100 + (80 * i));
            add(name);

            JLabel score = new JLabel(st.nextToken());
            score.setSize(100, 30);
            score.setFont(new Font("DungGeunMo", Font.PLAIN, 30));
            score.setLocation(710, 100 + (80 * i));
            add(score);
        }

        // 누르면 다시 시작 화면으로 돌아가는 버튼
        JButton returnButton = new JButton("RETURN");
        returnButton.setLocation(340, 460);
        returnButton.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        returnButton.setOpaque(false);
        returnButton.setBackground(new Color(0, 0, 0, 0));
        returnButton.setSize(200, 40);
        returnButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                startPanel.setVisible(true);
            }
        });
        add(returnButton);

    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        ImageIcon backgroundIcon = new
ImageIcon("background/startBackground.jpg");
        Image backgroundImg = backgroundIcon.getImage();
        g.drawImage(backgroundImg, 0, 0, this.getWidth(),
this.getHeight(), this);
    }
}
```

## ScorePanel.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
```

```java
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;
import javax.swing.*;

public class ScorePanel extends JPanel {
    private Ranking rank = new Ranking();
    private MonsterPanel monsterPanel;
    private GamePanel gamePanel;
    private int score;
    private JLabel life[];
    private JLabel scoreText = new JLabel("SCORE ");
    private JLabel scoreLabel, player, playerInfo, stageInfo;
    private int lifes, maxLife;
    private ImageIcon heart = new ImageIcon("heart.png");
    private ImageIcon emptyHeart = new ImageIcon("emptyHeart.png");
    private ImageIcon background = new ImageIcon("cloud.png");
    private String playerName;
    private int level, stage;
    private Color purple = new Color(14, 13, 89);
    private Clip clip;

    public ScorePanel(int level, int stage, String playerName,
MonsterPanel monsterPanel, GamePanel gamePanel) {
        this.level = level;
        this.stage = stage;
        this.playerName = playerName;
        this.monsterPanel = monsterPanel;
        this.gamePanel = gamePanel;
        setLayout(null);
        setBackground(purple);

        // 각 스테이지에 맞게 생명, 점수 초기화
        if (stage == 1) {
            lifes = 3;
            maxLife = 3;
            score = 0;
        } else if (stage == 2) {
            lifes = 4;
            maxLife = 4;
            score = 100;
        } else if (stage == 3) {
            lifes = 5;
            maxLife = 5;
            score = 250;
        } else {
            lifes = 5;
            maxLife = 5;
            score = 400;
        }
        life = new JLabel[lifes];

        // 점수
        scoreLabel = new JLabel(Integer.toString(score));
        scoreLabel.setFont(new Font("DungGeunMo", Font.BOLD, 20));
```

```
        scoreLabel.setLocation(135, 20);
        scoreLabel.setSize(100, 20);
        scoreLabel.setForeground(Color.white);
        add(scoreLabel);

        scoreText.setFont(new Font("DungGeunMo", Font.BOLD, 20));
        scoreText.setLocation(70, 20);
        scoreText.setSize(100, 20);
        scoreText.setForeground(Color.white);
        add(scoreText);

        // 현재 스테이지 정보 출력
        if (stage > 3) {
            stageInfo = new JLabel("EXTRA STAGE");
            stageInfo.setFont(new Font("DungGeunMo", Font.BOLD, 30));
            stageInfo.setSize(250, 20);
            stageInfo.setLocation(30, 100);
        } else {
            stageInfo = new JLabel("STAGE " + Integer.toString(stage));
            stageInfo.setFont(new Font("DungGeunMo", Font.BOLD, 30));
            stageInfo.setSize(150, 30);
            stageInfo.setLocation(60, 100);
        }
        stageInfo.setForeground(Color.white);
        add(stageInfo);

        // 현재 플레이어 이름 출력
        playerInfo = new JLabel("현재 플레이어");
        playerInfo.setFont(new Font("DungGeunMo", Font.BOLD, 20));
        playerInfo.setSize(200, 20);
        playerInfo.setForeground(Color.white);
        playerInfo.setLocation(45, 160);
        add(playerInfo);

        player = new JLabel(playerName);
        player.setFont(new Font("DungGeunMo", Font.PLAIN, 20));
        player.setSize(140, 20);
        player.setForeground(Color.white);
        player.setLocation(45, 200);
        add(player);

        // 생명 이미지 설정
        for (int i = 0; i < life.length; i++) {
            life[i] = new JLabel(new
ImageIcon(heart.getImage().getScaledInstance(50, 50,
Image.SCALE_SMOOTH)));
        }

        // 게임 일시정지 버튼
        Image pauseImage = new
ImageIcon("pause.png").getImage().getScaledInstance(40, 40,
Image.SCALE_SMOOTH);
        Image pausePressed = new
ImageIcon("pausePressed.png").getImage().getScaledInstance(40, 40,
Image.SCALE_SMOOTH);
        JButton pauseButton = new JButton(new ImageIcon(pauseImage));
        pauseButton.setPressedIcon(new ImageIcon(pausePressed));
```

```
        pauseButton.setContentAreaFilled(false);
        pauseButton.setSize(40, 40);
        pauseButton.setLocation(60, 240);
        pauseButton.setOpaque(false);
        pauseButton.setBackground(null);
        pauseButton.setForeground(Color.white);
        pauseButton.setBorderPainted(false);
        pauseButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                gamePanel.gameGround.gameth.stopRain(true);
                gamePanel.gameGround.wordth.stopWord(true);
            }
        });
        add(pauseButton);

        // 게임 재시작 버튼
        Image playImage = new
ImageIcon("play.png").getImage().getScaledInstance(40, 40,
Image.SCALE_SMOOTH);
        Image playPressed = new
ImageIcon("playPressed.png").getImage().getScaledInstance(40, 40,
Image.SCALE_SMOOTH);
        JButton playButton = new JButton(new ImageIcon(playImage));
        playButton.setPressedIcon(new ImageIcon(playPressed));
        playButton.setContentAreaFilled(false);
        playButton.setSize(40, 40);
        playButton.setLocation(130, 240);
        playButton.setOpaque(false);
        playButton.setBackground(null);
        playButton.setForeground(Color.white);
        playButton.setBorderPainted(false);
        playButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                gamePanel.gameGround.gameth.resumeRain();
                gamePanel.gameGround.wordth.resumeWord();
            }
        });
        add(playButton);
    }

    @Override // 생명 출력
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        for (int i = 0; i < life.length; i++) {
            if (stage == 1) {
                life[i].setLocation(70 + i * 30, 50);
            } else if (stage == 2) {
                life[i].setLocation(55 + i * 30, 50);
            } else {
                life[i].setLocation(40 + i * 30, 50);
            }
            life[i].setSize(30, 30);
            add(life[i]);
        }
    }
```

```
    // 점수를 증가시키는 메소드
    public boolean increase(int score, int stage) {
        this.score += score;
        scoreLabel.setText(Integer.toString(this.score));
        loadAudio("audio/increase.wav");
        // 몬스터의 이미지를 바꿈
        if (score == 0) {
            monsterPanel.changeImg(0);
        } else {
            monsterPanel.changeImg(1);
        }
        // 각 스테이지마다 목표 점수에 도달할 경우 true 리턴
        if (this.score >= 100 && stage == 1) {
            monsterPanel.mt.interrupt();
            return true;
        } else if (this.score >= 250 && stage == 2) {
            monsterPanel.mt.interrupt();
            return true;
        } else if (this.score >= 400 && stage == 3) {
            monsterPanel.mt.interrupt();
            return true;
        }
        // 목표 점수에 도달하지 못했을 경우 false 리턴
        return false;
    }

    // 생명을 감소시키는 메소드
    public boolean decrease(int stage) {
        // 생명을 감소시키고 남은 생명을 다시 그림
        // 생명이 감소하면 빈 하트 출력
        for (int i = 0; i < lifes; i++) {
            remove(life[i]);
        }
        lifes--;
        life[lifes] = new JLabel(new
ImageIcon(emptyHeart.getImage().getScaledInstance(50, 50,
Image.SCALE_SMOOTH)));
        repaint();
        monsterPanel.changeImg(2);
        loadAudio("audio/decrease.wav");
        // 생명이 0이 될 경우 true 리턴
        if (lifes == 0) {
            gamePanel.gameGround.gameth.interrupt();
            gamePanel.gameGround.wordth.interrupt();
            int choose = JOptionPane.showConfirmDialog(null, "랭킹에
등록하시겠습니까?", "게임 오버", JOptionPane.YES_NO_OPTION);
            if (choose == JOptionPane.YES_OPTION)
                rank.addRanking(level, playerName, score);
            gamePanel.gameOver(score);
            loadAudio("audio/gameOver.wav");
            return true;
        }
        // 생명이 1 이상일 경우 false 리턴
        return false;
    }
```

```
    // 오디오 재생
    private void loadAudio(String fileName) {
        try {
            clip = AudioSystem.getClip();
            File audioFile = new File(fileName);
            AudioInputStream audioStream =
AudioSystem.getAudioInputStream(audioFile);
            clip.open(audioStream);
            clip.start();
        } catch (LineUnavailableException e) {
            e.printStackTrace();
        } catch (UnsupportedAudioFileException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // 생명 1 추가
    public void addLife() {
        // 이미 생명이 가득 차있을 경우 리턴
        if (lifes == maxLife) {
            return;
        }
        // 생명을 증가시키고 남은 생명을 다시 그림
        life[lifes] = new JLabel(new
ImageIcon(heart.getImage().getScaledInstance(50, 50,
Image.SCALE_SMOOTH)));
        lifes++;
        repaint();
    }

}
```

**StartPanel.java**

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.sound.sampled.*;
import javax.swing.*;
public class StartPanel extends JPanel {
    private JLabel title = new JLabel("다마고치 키우기");
    private JButton startButton = new JButton("게임 시작");
    private JButton exitButton = new JButton("나가기");
    private JButton rankingButton = new JButton("랭킹 보기");
    private JButton addButton = new JButton("단어 추가");
    private JTextField player = new JTextField();
    private String playerName = "";
    private ImageIcon icon = new
ImageIcon("background/startBackground.jpg");
    private Image backgroundImg = icon.getImage();
    private TextSource textSource = new TextSource(this);
    private int selectedLevel = -1;
    private RankingPanel rp = new RankingPanel(this);
    private GameFrame gameFrame;
```

```
    private Font descriptionFont = new Font("DungGeunMo", Font.PLAIN,
20);
    // 생성자
    public StartPanel(GameFrame gameFrame) {
        setLayout(null);
        this.gameFrame = gameFrame;
        setBackground(new Color(255, 171, 228));
        title.setFont(new Font("DungGeunMo", Font.PLAIN, 70));
        title.setLocation(180, 70);
        title.setSize(600, 100);
        add(title);
        buttons();
        getInfo();
    }
    @Override // 배경화면 그리기
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(backgroundImg, 0, 0, this.getWidth(),
this.getHeight(), this);
    }
    // 사용자 이름과 게임 난이도를 입력 받는 메소드
    private void getInfo() {
        JLabel name = new JLabel("이름");
        name.setFont(descriptionFont);
        name.setLocation(350, 200);
        name.setSize(100, 20);
        add(name);
        // 이름을 입력받는 텍스트 필드
        player.setFont(descriptionFont);
        player.setLocation(400, name.getY() - 3);
        player.setSize(140, 30);
        add(player);
        JLabel level = new JLabel("레벨");
        level.setFont(descriptionFont);
        level.setLocation(350, 225);
        level.setSize(100, 50);
        add(level);
        // 게임 난이도 선택
        // easy는 1, hard는 2로 저장
        ButtonGroup levels = new ButtonGroup();
        JRadioButton easy = new JRadioButton("EASY");
        JRadioButton hard = new JRadioButton("HARD");
        easy.setOpaque(false);
        easy.setBackground(null);
        easy.setLocation(level.getX() + 55, level.getY() + 15);
        easy.setSize(70, 20);
        easy.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                selectedLevel = 1;
            }
        });
        levels.add(easy);
        hard.setOpaque(false);
        hard.setBackground(null);
        hard.setLocation(easy.getX() + 75, level.getY() + 15);
        hard.setSize(70, 20);
```

```
        hard.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                selectedLevel = 2;
            }
        });
        levels.add(hard);
        add(easy);
        add(hard);
    }
    // 키울 알을 선택하는 메소드
    public void chooseEgg() {
        removeAll();
        JLabel monsterDes = new JLabel("키울 알을 골라주세요!");
        monsterDes.setFont(new Font("DungGeunMo", Font.PLAIN, 40));
        monsterDes.setSize(500, 100);
        monsterDes.setLocation(230, 80);
        add(monsterDes);
        JButton buttons[] = new JButton[3];
        for (int i = 0; i < buttons.length; i++) {
            String path = "monster/egg" + (i + 1) + ".png";
            ImageIcon ic = new ImageIcon(path);
            buttons[i] = new JButton(new
ImageIcon(ic.getImage().getScaledInstance(200, 200,
Image.SCALE_SMOOTH)));
            buttons[i].setBounds(270 + i * 120, 200, 20, 15);
            buttons[i].setSize(100, 100);
            buttons[i].setOpaque(false);
            buttons[i].setBackground(null);
            add(buttons[i]);
            int index = i;
            buttons[i].addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    removeAll();
                    gameFrame.setGameInfo(selectedLevel, 1, index,
playerName);
                    gameFrame.toGamePanel();
                }
            });
        }
        revalidate();
        repaint();
    }
    private void buttons() {
        // 게임 시작 버튼
        startButton.setOpaque(false);
        startButton.setBackground(null);
        startButton.setFont(descriptionFont);
        startButton.setLocation(340, 280);
        startButton.setSize(200, 40);
        add(startButton);
        startButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                playerName = player.getText();
                // 이름, 레벨 미선택
                if (selectedLevel == -1 && playerName.equals("")) {
```

```
                    JOptionPane.showMessageDialog(null, "플레이어 이름을
입력해주세요.");
                    JOptionPane.showMessageDialog(null, "플레이할 레벨을
선택해주세요.");
                }
                // 레벨 미선택
                else if (selectedLevel == -1) {
                    JOptionPane.showMessageDialog(null, "플레이할 레벨을
선택해주세요.");
                }
                // 이름 미입력
                else if (playerName.equals("")) {
                    JOptionPane.showMessageDialog(null, "플레이어 이름을
입력해주세요.");
                }
                // 이름과 레벨 모두 선택했을 경우 알 선택으로 넘어감
                else {
                    chooseEgg();
                }
            }
        });
        // 단어 추가 버튼
        addButton.setLocation(340, 340);
        addButton.setSize(200, 40);
        addButton.setOpaque(false);
        addButton.setBackground(null);
        addButton.setFont(descriptionFont);
        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String newWord = JOptionPane.showInputDialog("추가할
단어를 입력해주세요.");
                if (newWord == null)
                    return;
                newWord.trim();
                // 이미 텍스트 파일에 단어가 존재할 경우
                if (textSource.search(newWord)) {
                    JOptionPane.showMessageDialog(null, "이미 존재하는
단어입니다.");
                }
                // 알파벳 외에 다른 글자를 작성했거나 아무 단어도 작성하지
않았을 때
                else if (newWord.equals("") || !newWord.matches("[a-zA-
Z]+")) {
                    JOptionPane.showMessageDialog(null, "영어 단어를
입력해주세요.", "단어 추가 오류", JOptionPane.ERROR_MESSAGE);
                    return;
                }
                // 올바르게 단어를 입력했을 경우 텍스트 파일에 추가
                else {
                    textSource.add(newWord);
                }
            }
        });
        add(addButton);
        // Top 5를 확인하는 버튼
```

```
        rankingButton.setOpaque(false);
        rankingButton.setBackground(null);
        rankingButton.setFont(descriptionFont);
        rankingButton.setLocation(340, 400);
        rankingButton.setSize(200, 40);
        rankingButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                getParent().add(rp);
                rp.setVisible(true);
                setVisible(false);
            }
        });
        add(rankingButton);
        // 나가기 버튼
        exitButton.setOpaque(false);
        exitButton.setBackground(null);
        exitButton.setFont(descriptionFont);
        exitButton.setLocation(340, 460);
        exitButton.setSize(200, 40);
        add(exitButton);
        exitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
    }
}
```

**TextSource.java**

```
import java.awt.*;
import java.io.*;
import java.util.*;

import javax.swing.JOptionPane;

public class TextSource {
    private Vector<String> wordVector = new Vector<String>(30000);
    private FileWriter fout;

    // 텍스트 파일로부터 단어를 읽어와 벡터에 저장
    public TextSource(Component parent) {
        try {
            Scanner scanner = new Scanner(new FileReader("words.txt"));
            while (scanner.hasNext()) {
                String word = scanner.nextLine();
                wordVector.add(word);
            }
            scanner.close();
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(null, "파일 없음");
            System.exit(0);
        }
    }

    // 벡터에서 랜덤 인덱스의 단어를 리턴하는 메소드
```

```
    public String next() {
        int n = wordVector.size();
        int index = (int) (Math.random() * n);
        return wordVector.get(index);
    }

    // 이미 존재하는 단어인지 찾는 메소드
    // 존재하면 true, 없으면 false 리턴
    public boolean search(String word) {
        for (int i = 0; i < wordVector.size(); i++) {
            if (wordVector.get(i).equals(word))
                return true;
        }
        return false;
    }

    // 텍스트 파일에 단어를 추가하는 메소드
    public void add(String word) {
        try {
            fout = new FileWriter("words.txt", true);
            fout.write(word);
            fout.write("\n");
            fout.close();
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "파일 없음");
        }

        wordVector.add(word);
    }
}
```

## 5. 결론

이번 미니프로젝트 제작에서 화면 전환이 가장 복잡하고 어려웠다. 처음에 버튼을 누르면 시작 화면에서 게임 화면으로 넘어가는 것이 계속 안 돼서 많이 힘들었다. 하지만 toGamePanel 메소드를 만들어 화면 전환을 구현하고 내가 추가하고 싶은 기능을 하나씩 추가해 보니 프로젝트 제작이 점점 즐거워졌다. 미니프로젝트를 진행하면서 자바의 스레드 구현을 제대로 연습할 수 있었다. 스레드 작동 방식을 비롯한 스레드에 대한 추가적인 정보는 다음 학기 운영체제 시간에 열심히 배워보고 싶다.