[7반]

객체지향언어2 미니 프로젝트

2023.12.12 2291001 한지운

객체지향언어2 미니 프로젝트 결과 보고서

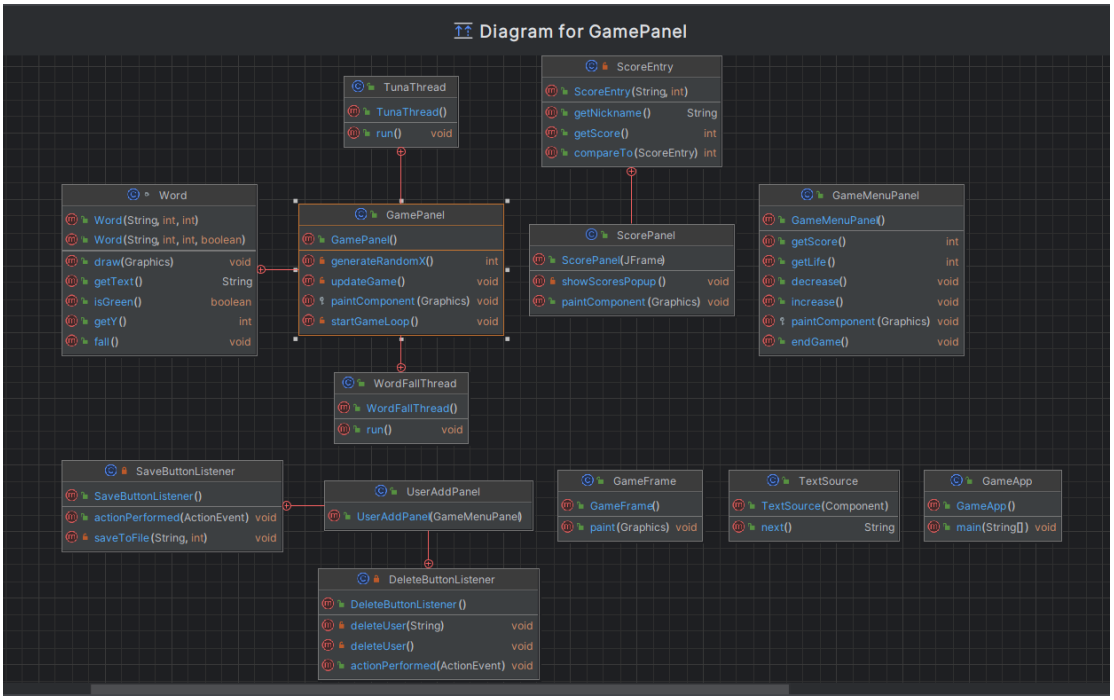
<Help Tuna!>

2291001 7반 한지운

1. 작품 개요

 참치는 평생 동안 헤엄치는 것을 단 한 번도 멈추지 않는다. 호흡하는 방식에 의해 멈추는 즉시 참치는 죽기 때문이다. 이 게임에서는 물살을 따라 헤엄치는 참치를 방해하는 바다 쓰레기가 흘러오는데, 이때 바다 쓰레기에 적힌 단어들을 없앰으로써 참치가 멈추지 않고 헤엄칠 수 있도록 돕는다. 중간마다 20%의 확률로 초록색으로 된 단어가 나오는데, 초록색 단어를 놓칠 때마다 목숨이 1개씩 줄어든다. 총 5개의 목숨이 모두 소모되면 게임이 종료된다. 또한, 함께 떨어지는 검정색 단어들은 점수를 10점씩 증가시킨다. 우측 상단 패널에는 남은 목숨, 현재 점수 등이 출력된다. 우측 하단 패널에는 이전 플레이어들의 점수 기록에 이어 현재 플레이어의 점수를 기록할 수 있고, 점수판에 있는 이전 플레이어를 삭제할 수도 있다.

2. 프로그램 구조

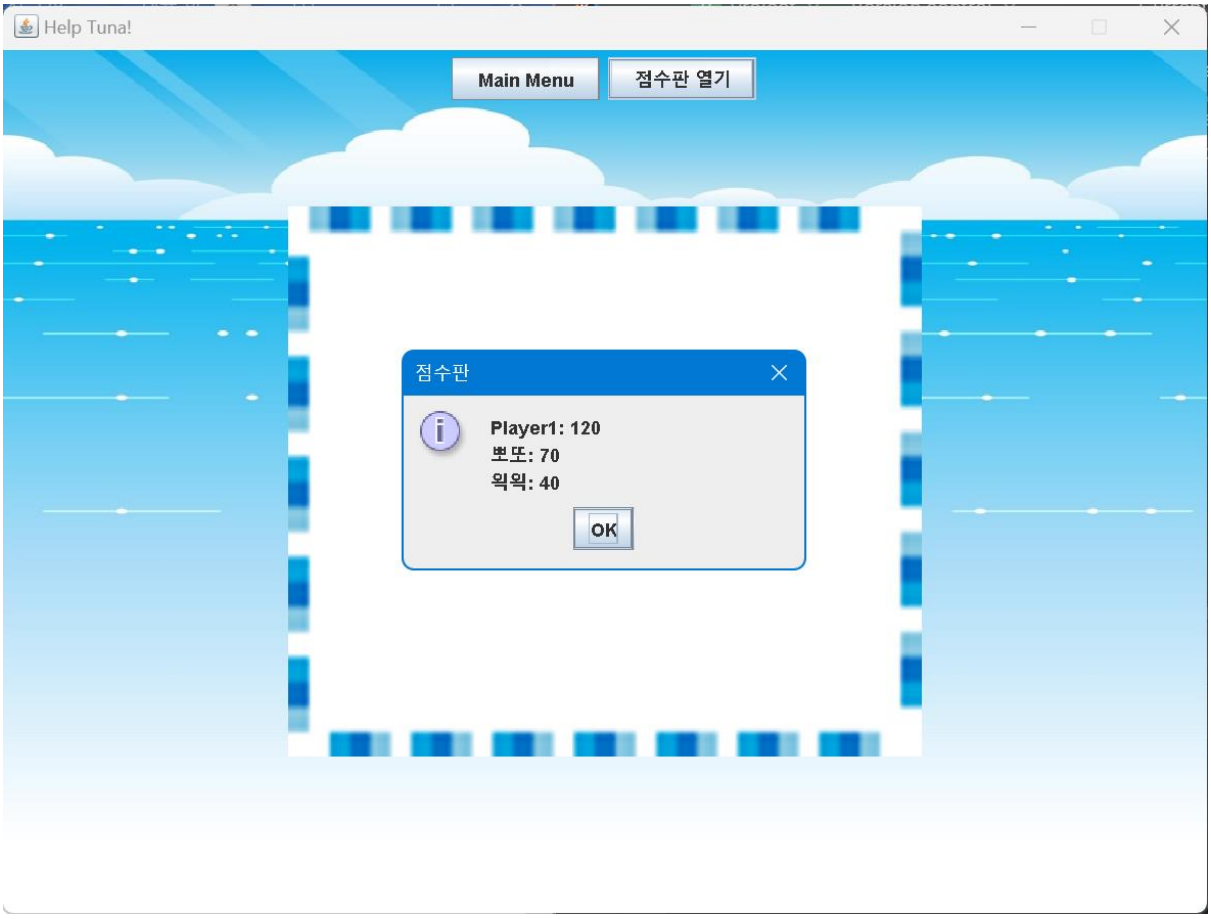


3. 프로그램 실행 과정

게임 실행

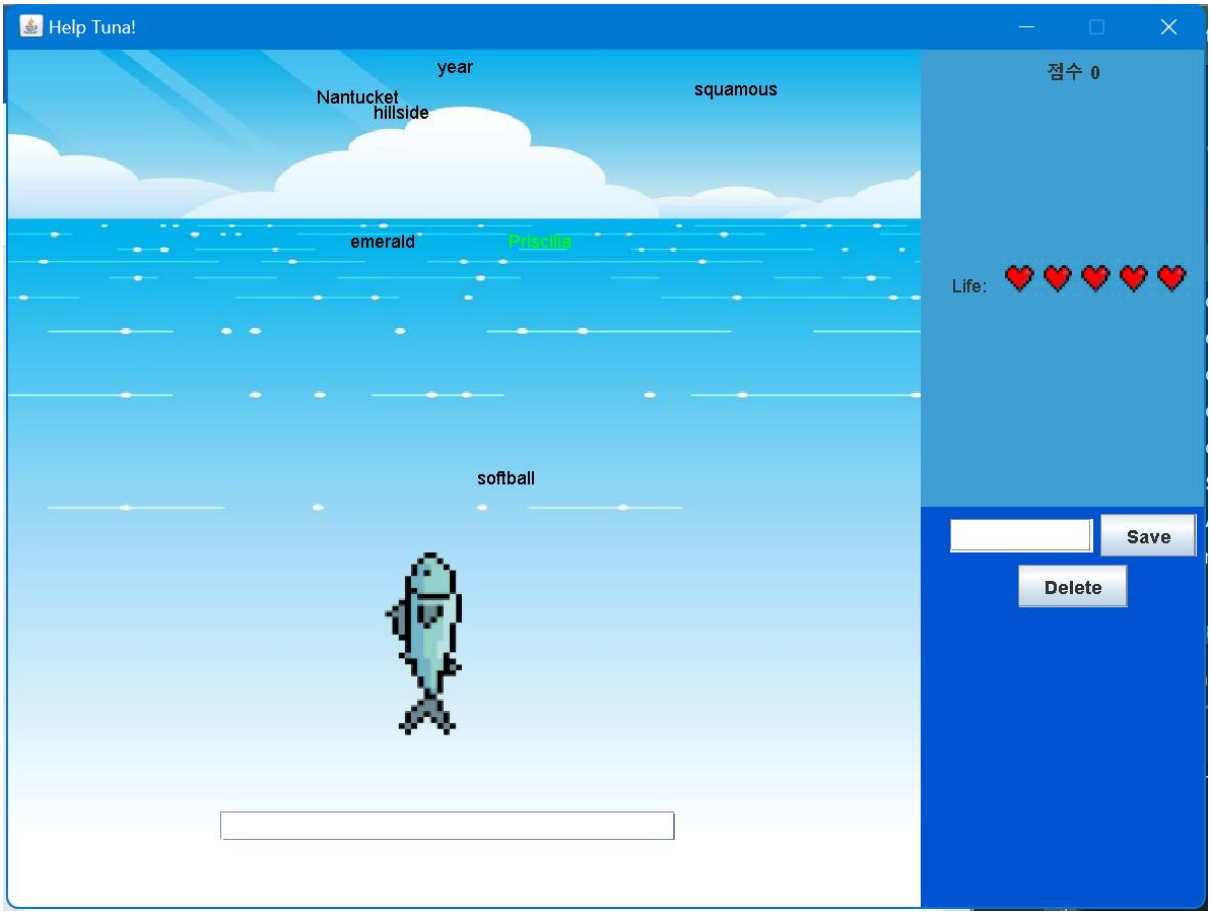

Help Tuna!
HELPTuna!
Game Start
Score
@2291001 한지운
참치를 도와라! 초록색 단어를 5번 놓치면 참치가 죽는다. 검정색 단어를 입력해 점수를 획득해라!

Score 버튼 클릭 시,

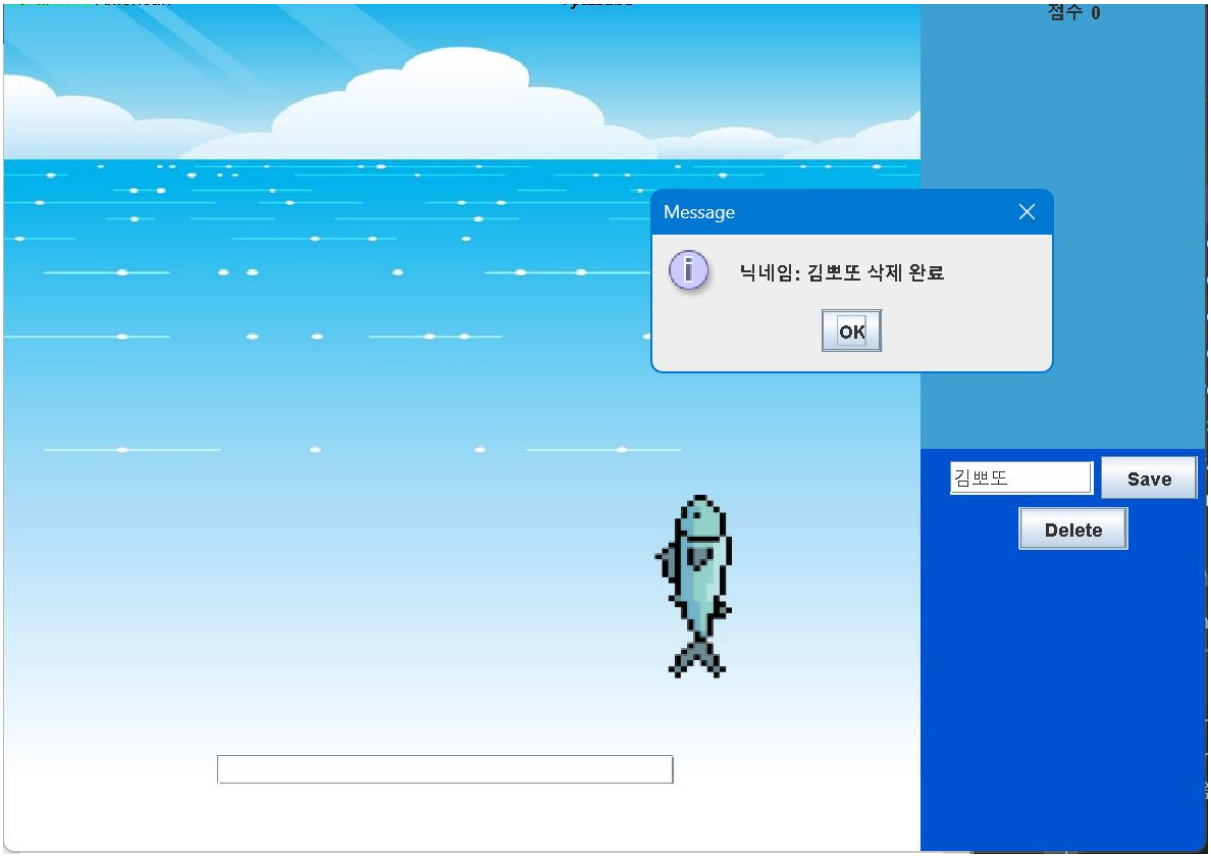


Main menu 버튼 클릭 시 다시 메인 메뉴로 돌아감

Game Start 버튼 클릭 시 게임 진행



목숨5개 모두 소모하면 플레이어 닉네임과 점수 등록/삭제



4. 프로그램 소스 코드

[GameApp.java]

```java
public class GameApp {
    public static void main(String[] args) {
        new GameFrame();
    }
}
```

[GameFrame.java]

```java
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;

import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

// 메인 게임 프레임
public class GameFrame extends JFrame {
    private ImageIcon bgImg = new ImageIcon("main.png"); // 배경 이미지
```

```
    private Image img = bgImg.getImage();
    private ImageIcon tunaImg = new ImageIcon("tuna90.png"); // 참치
이미지
    private Image tuna = tunaImg.getImage().getScaledInstance(120,
50, Image.SCALE_SMOOTH);
    private ImageIcon titleImg = new ImageIcon("Title.png"); // title
이미지
    private Image title = titleImg.getImage().getScaledInstance(400,
250, Image.SCALE_SMOOTH);
    private ImageIcon menuIcon = new ImageIcon("menu.png"); // 메뉴
이미지
    private Image menuImg =
menuIcon.getImage().getScaledInstance(300, 200, Image.SCALE_SMOOTH);
// 이미지 크기 조절


    public GameFrame() {
        setTitle("Help Tuna!");
        setSize(800, 600);
        setLayout(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton btn1 = new JButton("Game Start");
        JButton btn2 = new JButton("Score");

        // 버튼 크기와 위치 설정
        btn1.setSize(100, 40);
        btn2.setSize(100, 40);
        btn1.setLocation(getWidth() / 2 - 46, getHeight() / 2 - 28);
        btn2.setLocation(getWidth() / 2 - 46, getHeight() / 2 + 15);
        add(btn1);
        add(btn2);

        btn1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                GamePanel gamePanel = new GamePanel();
                setContentPane(gamePanel);
                revalidate(); // 프레임을 다시 그리도록 갱신
            }
        });

        btn2.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                ScorePanel scorePanel = new ScorePanel(GameFrame.this);
                setContentPane(scorePanel);
                revalidate(); // 프레임을 다시 그리도록 갱신
            }
        });

        setResizable(false); // 크기 변경 금지
        setVisible(true);
    }
```

```
    @Override
    public void paint(Graphics g) {

        // 배경 이미지 그리기
        if (bgImg != null) {
            g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
        }

        // 메뉴 이미지 그리기
        int menuX = getWidth() / 2 - 140;
        int menuY = getHeight() / 2 - 60;
        g.drawImage(menuImg, menuX, menuY, this);

        // 참치 그리기
        g.drawImage(tuna, menuX + 90, menuY - 40, this);

        // title 그리기
        g.drawImage(title, menuX - 35, menuY - 200, this);

    }
}
```

[GameMenuPanel.java]

```
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;

public class GameMenuPanel extends JPanel {
    private int score = 0;
    private int life = 5;
    private JLabel scoreLabel = new JLabel(Integer.toString(score));
    private ImageIcon lifeImg = new ImageIcon("life.png"); // life
이미지
    private Image img = lifeImg.getImage();

    public GameMenuPanel() {
        setBackground(new Color(63, 159, 210)); // 배경색 설정

        add(new JLabel("점수"));
        add(scoreLabel);
    }

    public int getLife() {
        return life;
    }

    public int getScore() {
        return score;
    }

    public void increase() {
```

```
        score += 10;
        scoreLabel.setText(Integer.toString(score));
    }

    public void decrease() {
        this.life--; // life 감소
        scoreLabel.setText(Integer.toString(score));
        repaint(); // life 감소 시 다시 그리기
        if(this.life == 0) {
            endGame();
        }
    }

    public void endGame() {
        // Game Over
        System.out.println("Game Over");
    }


    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        if (lifeImg != null) {
            int imgWidth = img.getWidth(this);
            int imgHeight = img.getHeight(this);

            for (int i = 0; i < life; i++) {
                // life 이미지 우측에서부터 순서대로 출력
                int x = getWidth() - (i + 1) * imgWidth - 23;
                int y = (getHeight() - imgHeight) / 2;
                g.drawImage(img, x, y, imgWidth, imgHeight, this);

                // "Life:" 라벨 중앙에 출력
                int labelX = 20;
                int labelY = y + imgHeight - 4;
                g.drawString("Life:", labelX, labelY);
            }
        }
    }

}
```

[GamePanel.java]

```
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import java.awt.Color;
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.concurrent.CopyOnWriteArrayList;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

public class GamePanel extends JPanel {
    private ImageIcon bgImg = new ImageIcon("bg.jpg"); // 배경 이미지
    private Image img = bgImg.getImage();
    private ImageIcon tunaImg = new ImageIcon("tuna.png"); // 참치
이미지
    private Image tuna = tunaImg.getImage().getScaledInstance(50,
120, Image.SCALE_SMOOTH); // 이미지 크기 조절
    private int tunaX = 0, tunaY = 330; // 참치의 X, Y 좌표
    private boolean movingRight = true;
    private JTextField textInput = new JTextField(20);
    private TextSource textSource = null;
    private CopyOnWriteArrayList<Word> fallingWords = new
CopyOnWriteArrayList<>(); // 떨어지는 단어 목록
    // CopyOnWriteArrayList 는 복사본을 사용하여 동기화 문제 해결(여러
스레드가 동시에 컬렉션 수정 시)
    private JLabel label = new JLabel(""); // 단어 출력용 레이블


    private GameMenuPanel gameMenuPanel = null; // 게임 메뉴 패널
    private UserAddPanel userAddPanel = null; // 사용자 추가 패널
    WordFallThread wThread;
    TunaThread tThread;

    public GamePanel() {
        setLayout(null);

        textInput.setSize(300, 20); // 단어 입력창
        textInput.setLocation(140, 500);
        add(textInput);

        label.setSize(100, 20); // 단어 label 부착
        label.setLocation(10, 10);
        add(label);

        // 게임 메뉴 패널 생성
        gameMenuPanel = new GameMenuPanel();
        gameMenuPanel.setBounds(600, 0, 200, 300);
        add(gameMenuPanel);

        // 사용자 추가 패널 생성
        userAddPanel = new UserAddPanel(gameMenuPanel);
        userAddPanel.setBounds(600, 300, 200, 300);
        add(userAddPanel);

        // Enter 키 입력 이벤트 처리
        textInput.addActionListener(new ActionListener() {
```

```
            @Override
            public void actionPerformed(ActionEvent e) {
                JTextField tf = (JTextField) e.getSource();
                String inputText = tf.getText();

                for (Word word : fallingWords) {
                    if (inputText.equals(word.getText())) {
                        gameMenuPanel.increase(); // 점수 증가
                        tf.setText(""); // 입력 창 비우기
                        fallingWords.remove(word); // 맞춘 단어는 목록에서 제거

                        break; // 맞은 단어를 찾음
                    }
                }
            }
        });

        textSource = new TextSource(this);

        // 단어 떨어뜨리는 스레드 시작
        wThread = new WordFallThread();
        wThread.start();

        // 참치 움직이는 스레드 시작
        tThread = new TunaThread();
        tThread.start();

        // 게임 루프 시작
        startGameLoop();
    }

    // 게임 루프 역할
    private void startGameLoop() {
        // 스케줄된 작업을 처리할 쓰레드 제공
        ScheduledExecutorService executorService = Executors.newSingleThreadScheduledExecutor();
        // scheduleAtFixedRate 메서드를 호출하여 주기적인 작업을 실행
        executorService.scheduleAtFixedRate(new Runnable() {
            @Override
            public void run() {
                updateGame(); // 게임 상태 업데이트
                repaint();    // 화면 갱신
            }
        }, 0, 16, TimeUnit.MILLISECONDS);
        // 0 은 초기 딜레이. 작업을 즉시 시작하도록 설정
        // 16 >> 두 실행 간의 시간 간격 밀리초 단위. => 60frame/sec
        // TimeUnit.MILLISECONDS 는 시간 간격의 단위를 밀리초로 설정
    }

    private void updateGame() {
        // 새로운 단어를 일정 간격으로 생성
```

```
        if (Math.random() < 0.01) {
            boolean isGreen = Math.random() < 0.2; // 20% 확률로 초록색
단어 생성
            Word newWord = new Word(textSource.next(),
generateRandomX(), 10, isGreen);
            fallingWords.add(newWord);
        }

        // 패널을 벗어난 단어들 제거
        fallingWords.removeIf(word -> word.getY() > getHeight()); //
리스트의 각 요소(word)에 대해 조건 검사


        // 레이블 위치 업데이트
        label.setLocation(label.getX(), label.getY() + 10);

        // 물고기 위치 업데이트
        if (movingRight) {
            tunaX += 5; // 물고기를 오른쪽으로 이동
            if (tunaX >= getWidth() - gameMenuPanel.getWidth() - 100)
{
                tunaX = getWidth() - gameMenuPanel.getWidth() - 100; //
오른쪽 끝에 도달하면 고정
                movingRight = false; // 왼쪽으로 이동하도록 변경
            }
        } else {
            tunaX -= 5; // 물고기를 왼쪽으로 이동
            if (tunaX <= 0) {
                tunaX = 0; // 왼쪽 끝에 도달하면 고정
                movingRight = true; // 오른쪽으로 이동하도록 변경
            }
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // 배경 이미지 그리기
        if (bgImg != null) {
            g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
        }

        // 떨어지는 단어들 그리기
        for (Word word : fallingWords) {
            word.draw(g);
        }

        // 참치 그리기
        g.drawImage(tuna, tunaX, tunaY, this);
    }
```

```
    // X 좌표를 랜덤하게 생성하는 메서드
    private int generateRandomX() {
        int panelWidth = getWidth() - 100;  // 패널 내부에서 랜덤한 위치
        int maxX = Math.min(panelWidth, getWidth() -
gameMenuPanel.getWidth() - 100);

        return (int) (Math.random() * maxX);
    }

    // 단어를 떨어뜨리는 스레드
    public class WordFallThread extends Thread {
        private static final int BASE_FALL_SPEED = 200; // 기본 떨어지는
속도 (200ms)
        private int fallSpeed = BASE_FALL_SPEED;
        private long lastSpeedChangeTime =
System.currentTimeMillis(); // 현재 시스템 시간을 밀리초로 반환하는 메서드
        private boolean stopThread = false; // 스레드 중지 여부


        public void run() {
            while (!stopThread) {
                try {
                    sleep(fallSpeed);

                    // 20 초마다 속도를 높임
                    long elapsedTimeSinceLastChange =
System.currentTimeMillis() - lastSpeedChangeTime;
                    if (elapsedTimeSinceLastChange > 20000) { //
20 초마다

                        fallSpeed /= 2; // 속도를 높임
                        lastSpeedChangeTime =
System.currentTimeMillis();
                    }

                    // 떨어지는 단어들의 위치 업데이트 및 패널 아래로 떨어진 경우
제거
                    Iterator<Word> iterator = fallingWords.iterator();
                    while (iterator.hasNext()) {
                        Word word = iterator.next();
                        if (word != null) {
                            word.fall();
                            if (word.getY() > getHeight()) {
                                // 패널 아래로 떨어진 경우 목록에서 제거
                                fallingWords.remove(word);
                                if (word.isGreen()) {
                                    gameMenuPanel.decrease(); // 초록색
단어가 떨어진 경우 life 감소


                                    if(gameMenuPanel.getLife() == 0) { //
life == 0 일 경우 스레드 중지
                                        stopThread = true;
```

```
                                    wThread.interrupt();
                                }
                            }
                        }
                    }
                }
            } catch (InterruptedException e) {
                return;
            }
        }
    }

    // 참치를 좌우로 패널 따라 이동하게 하는 스레드
    public class TunaThread extends Thread {
        private int tunaSpeed = 3; // 참치의 이동 속도
        private int minX = 0; // 참치의 최소 x 좌표
        private int maxX; // 참치의 최대 x 좌표
        private boolean stopThread = false; // 스레드 중지 여부

        public TunaThread() {
            // 최대 x 좌표 계산
            maxX = getWidth() - gameMenuPanel.getWidth() -
tunaImg.getIconWidth() - 100;
        }

        public void run() {
            while (!stopThread) {
                try {
                    Thread.sleep(50); // 참치의 움직이는 속도 조절

                    // 참치 이동 방향에 따라 좌표 업데이트
                    if (tunaX >= maxX) {
                        tunaSpeed = -3; // 오른쪽 끝에 도달하면 왼쪽으로 이동
                    } else if (tunaX <= minX) {
                        tunaSpeed = 3; // 왼쪽 끝에 도달하면 오른쪽으로 이동
                    }

                    tunaX += tunaSpeed;
                    repaint(); // 화면 갱신

                    if(gameMenuPanel.getLife() == 0) { // life == 0 일
경우 스레드 중지
                        stopThread = true;
                        tThread.interrupt();
                    }
                } catch (InterruptedException e) {
                    return;
                }
            }
        }
```

```
    }

    class Word {
        private String text;
        private int x, y;
        private boolean isGreen; // 초록색 여부
        private static final int FALL_SPEED = 5; // 떨어지는 속도

        public Word(String text, int x, int y) {
            this(text, x, y, false); // 초록색 여부 기본값은 false
        }

        public Word(String text, int x, int y, boolean isGreen) {
            this.text = text;
            this.x = x;
            this.y = y;
            this.isGreen = isGreen;
        }

        public String getText() {
            return text;
        }

        public boolean isGreen() {
            return isGreen;
        }

        public void fall() {
            y += FALL_SPEED;
            if (y > getHeight()) {
                // 패널 아래로 떨어진 경우 목록에서 제거
                fallingWords.remove(this);
            }
        }

        public void draw(Graphics g) {
            if (isGreen) {
                g.setColor(Color.GREEN); // 초록색 설정
            } else {
                g.setColor(Color.BLACK); // 기본색 설정
            }
            g.drawString(text, x, y);
        }

        public int getY() {
            return y;
        }
    }
}
```

[ScorePanel.java]

```
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
```

```java
import javax.swing.JOptionPane;
import javax.swing.JPanel;

import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class ScorePanel extends JPanel {
    private ImageIcon bgImg = new ImageIcon("board.png"); // 배경
이미지
    private Image img = bgImg.getImage();
    private JButton menuBtn;
    private JFrame frame;

    public ScorePanel(JFrame frame) {
        this.frame = frame;
        setLayout(new FlowLayout());

        menuBtn = new JButton("Main Menu");
        add(menuBtn);


        JButton scoresBtn = new JButton("점수판 열기");
        add(scoresBtn);

        setVisible(true);

        menuBtn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    // 메인 화면으로 돌아가기 위해 GameFrame 다시 열기
                    frame.setContentPane(new GameFrame());
                    frame.revalidate();
                    frame.repaint();
                } catch (IllegalArgumentException ignored) {
                    // IllegalArgumentException 발생시 무시
                }
            }
        });

        scoresBtn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // 팝업창으로 점수 출력
                showScoresPopup();
            }
        });
    }
```

```
    private void showScoresPopup() {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader("scores.txt"));
            List<ScoreEntry> scoreEntries = new ArrayList<>();
            String line;

            while ((line = reader.readLine()) != null) {
                String[] parts = line.split(":");
                if (parts.length == 2) {
                    String nickname = parts[0].trim();
                    int score = Integer.parseInt(parts[1].trim());
                    scoreEntries.add(new ScoreEntry(nickname, score));
                }
            }
            reader.close();

            // 점수가 높은 순서대로 정렬
            Collections.sort(scoreEntries,
Collections.reverseOrder());

            // 팝업창에 출력할 문자열 생성
            StringBuilder popupText = new StringBuilder("<html>");
            for (ScoreEntry entry : scoreEntries) {
                popupText.append(entry.getNickname()).append(":
").append(entry.getScore()).append("<br>");
            }
            popupText.append("</html>");

            // 팝업창 생성 및 설정
            JOptionPane.showMessageDialog(frame, popupText, "점수판",
JOptionPane.INFORMATION_MESSAGE);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        // 배경 이미지 그리기
        if (bgImg != null) {
            g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
        }
    }

    private static class ScoreEntry implements Comparable<ScoreEntry>
{
        private String nickname;
        private int score;

        public ScoreEntry(String nickname, int score) {
            this.nickname = nickname;
            this.score = score;
        }
```

```
        public String getNickname() {
            return nickname;
        }

        public int getScore() {
            return score;
        }

        @Override
        public int compareTo(ScoreEntry o) {
            return Integer.compare(this.score, o.score);
        }
    }
}
```

[TextSource.java]

```
import java.awt.Component;

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

public class TextSource {
    private Vector<String> wordVector = new Vector<String>(30000);
    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(); // 한 프로그램 안에서는 close 한번만.
        }
        catch(FileNotFoundException e) {
            System.out.println("파일 없어요.");
            System.exit(0);
        }
    }

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

[UserAddPanel.java]

```
import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
```

```java
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

public class UserAddPanel extends JPanel {
    private JTextField wordInput = new JTextField(10);
    private GameMenuPanel gameMenuPanel; // GamePanel에 접근하기 위한 참조


    public UserAddPanel(GameMenuPanel gameMenuPanel) {
        this.gameMenuPanel = gameMenuPanel;

        setBackground(new Color(0, 83, 208));

        JButton saveButton = new JButton("Save");
        saveButton.addActionListener(new SaveButtonListener());
        add(wordInput);
        add(saveButton);

        JButton deleteButton = new JButton("Delete");
        deleteButton.addActionListener(new DeleteButtonListener());
        add(deleteButton);
    }

    // Save 버튼 클릭 리스너
    private class SaveButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String nickName = wordInput.getText();
            int score = gameMenuPanel.getScore();

            if (!nickName.isEmpty()) {
                saveToFile(nickName, score);
                System.out.println("점수가 다음 닉네임으로 저장: " + nickName);
            } else {
                System.out.println("닉네임을 입력하세요.");
            }
        }

        // 파일에 저장하는 메서드
        private void saveToFile(String nickName, int score) {
            // BufferedWriter: 텍스트 데이터를 파일에 쓰기 위한 클래스. 버퍼링을 통해 입출력 효율을 높임
            try (BufferedWriter writer = new BufferedWriter(new FileWriter("scores.txt", true))) { // 이어쓰기 모드로 연 뒤, FileWriter로 래핑
```

```
                // 파일에 닉네임과 점수 저장
                writer.write(nickName + ": " + score);
                writer.newLine();
            } catch (IOException ex) {
                System.err.println("파일 입출력 예외 발생");
            }
        }
    }

    // Delete 버튼 클릭 리스너
    private class DeleteButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            deleteUser();
        }

        // 사용자 삭제 메서드
        private void deleteUser() {
            String nickname = wordInput.getText();
            int score = gameMenuPanel.getScore();

            if (!nickname.isEmpty()) {
                JOptionPane.showMessageDialog(UserAddPanel.this,
"닉네임: " + nickname + " 삭제 완료");

                // 사용자 삭제 메서드 호출
                deleteUser(nickname);
            } else {
                JOptionPane.showMessageDialog(UserAddPanel.this,
"닉네임을 입력하세요.");
            }
        }

        // 사용자 삭제 메서드 (파일에서 삭제)
        private void deleteUser(String nickname) {
            try {
                // 파일 읽기
                List<String> lines =
Files.readAllLines(Paths.get("scores.txt"));

                // 삭제할 사용자 찾기
                for (int i = 0; i < lines.size(); i++) {
                    if (lines.get(i).startsWith(nickname)) {
                        // 사용자 찾으면 삭제
                        lines.remove(i);
                        break;
                    }
                }

                // 파일 쓰기
                Files.write(Paths.get("scores.txt"), lines);

            } catch (IOException e) {
```

```
                System.err.println("파일 입출력 예외 발생");
            }
        }
    }
}
```

5. 결론

프로젝트를 수행하며 스레드에 관한 코드 구현에 가장 많은 시간을 소모했다. 직접 스레드 예제 코드를 타이핑해보며 공부할 때는 단어 하나만을 일정 시간으로 떨어뜨리는 스레드를 다루었는데, 이를 변형시켜 떨어지는 단어마다 isGreen을 추가하여 녹색인지 아닌지 검사 유무에 따라 이벤트 처리를 다르게 하고, 확률을 사용하여 랜덤하게 단어의 색상을 결정하는 기능을 구현해냈다. 또한, 떨어뜨리는 단어를 20초마다 속도를 올려주려 변형시키기도 했고, 이 과정에서 따로 시간을 재는 스레드를 만들지 않아도 시스템 자체의 시간을 밀리초로 반환하는 메서드에 관한 내용을 알게 되었다. 스레드 활용의 심화 부분을 익혀나가며 모르던 메서드, 이전에 내용을 넘어갔던 람다식 등을 다시 한번 찾아보며 많은 공부가 됨을 체감하였다. 또한 클래스 다이어그램이라는 것을 찾아보며 인텔리제이에서 제공하는 기능을 직접 사용해보았다. 생각보다 좋은 도구들이 많았는데 그동안 사용해보지 못했던 것이 아쉬움과 동시에 이렇게 사용해보며 이후에 자주 이용할 수 있을 것이라는 생각이 들었다. 단어를 떨어뜨리는 스레드를 구현하며 여러 스레드가 동시에 fallingWords를 공유하며(리스트 수정할 때) 동기화 문제를 맞닥뜨렸다. 이 부분에서 기존 사용했던 스레드로 해결하고자 하여 많은 시간이 소모되었다. 그러나 해결 방법을 검색하던 와중, CopyOnWriteArrayList라는 컬렉션 클래스를 알게 되었다. 수정 작업이 발생할 때마다 원본 리스트의 복사본을 만들어 작업을 수행하고, 작업이 끝난 후, 원본 리스트를 해당 복사본으로 교체하는 특수한 컬렉션 클래스를 알게 되었다. 일반적으로 'synchronized' 키워드를 사용해 메서드나 블록을 동기화할 수 있지만, 이 클래스는 읽기 작업이 빈번하게 발생하고 쓰기 작업이 상대적으로 적은 상황에서 효과적임을 알게 되며(보통 읽기 작업이 많아서 동기화 오버헤드가 큰 상황에 유리), 직접 사용해보는 경험도 소중하게 느껴졌다. 특수한 컬렉션보다는 일반적인 컬렉션 스레드를 사용하는 것이 맞다고 생각하였지만, 도저히 해결되지 않는 부분에서는 라이브러리에서 제공하는 생소한 기능들을 사용해보는 경험도 중요하다고 생각하게 되었다. 주로 이번 프로젝트를 통해 여러 개의 스레드를 생성, 중지시키는 것에 대해 확실히 학습하게 되었고, 여러 이벤트 처리를 해주면서 직접 동작시켜보며 다중 스레드 사용의 필요성을 몸소 체감하게 되었다. 제출 이후에도 배경음악 추가, 한/영 선택 등의 기능을 구현해 봐야겠다고 생각했다. 많은 경험과 공부가 된 프로젝트였다.