

3. 활동일지(주간)

기본정보	보고주차	3주차 (12월 03일 ~ 12월 09일)
	작성자	이강민, 장정원, 신상현, 이민제
	제출일	2025.12.08.
활동내용	GitHub(https://github.com/hjus36/java-vocab-project)	
	<p>1. GUI 기능 고도화 및 전체 기능 통합</p> <ul style="list-style-type: none"> ■ MainFrame 전체 기능 완성 (추가·삭제·목록·저장·불러오기·퀴즈·통계) ■ 단어장 선택 기능(전공/토익/JLPT)을 위한 JComboBox 추가 ■ 단어장 전환 시 자동 파일 로드 기능 구현 ■ 단어 추가 시 공백 입력·중복 단어에 대해 예외 처리 ■ 파일 불러오기 실패 시 메시지 팝업으로 안내하고 프로그램이 중단되지 않도록 처리 <p>2. 단어장 종류 분리 기능 구현</p> <ul style="list-style-type: none"> ■ BookType 열거형(enum) 정의 (MAJOR / TOEIC / JLPT) ■ 각 단어장에 대해 별도 txt 파일 저장 구조 완성 (major_vocab.txt, toeic_vocab.txt, jlpt_vocab.txt) ■ FileManager에서 BookType 기반 경로 처리 기능 구현 ■ word_data 폴더가 존재하지 않을 경우 자동 생성하도록 처리 <p>3. QuizFrame & QuizManager 기능 개선</p> <ul style="list-style-type: none"> ■ 퀴즈 실행 시 단어장이 비어있을 경우 예외 처리 ■ 정답 입력을 비워 두고 정답 확인 버튼을 누를 경우, 먼저 입력을 요구하는 기능 추가 ■ 퀴즈 UI 개선(문제·정답 입력창 배치, 메시지 출력 개선) <p>4. StatsFrame 구현</p> <ul style="list-style-type: none"> ■ 단어 수, 퀴즈 정답률, 시도 횟수 등 통계 출력 구현 ■ 단어장 변경 시 통계 자동 갱신 처리 	

활동내용		<p>1. 전공 · 토익 · JLPT 단어가 한 파일에 섞여 저장되는 문제</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>기존에는 모든 단어를 ‘Word.txt’ 파일에 저장하는 방식이였지만, 이로 인해 전공/토익/JLPT 단어가 한 파일에 섞여 저장되어, 단어장별로 단어를 구분해서 관리하거나 퀴즈 · 통계를 분리해서 사용하는 것이 어려웠다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// 단어장 종류(전공 / 토익 / JLPT)를 나타내는 enum public enum BookType { MAJOR(path: "word_data/major_vocab.txt"), TOEIC(path: "word_data/toeic_vocab.txt"), JLPT(path: "word_data/jlpt_vocab.txt"); private final String path; // 각 단어장에 대응되는 파일 경로 BookType(String path) { this.path = path; } // 실제 파일 경로 반환 public String getPath() { return path; } }</pre> <p>-> ‘BookType’ 열거형(enum)을 정의하여 단어장 종류를 구분하였다. 또한, ‘FileManager’에서 저장/불러오기 시 ‘BookType’을 인자로 전달받아, 선택된 단어장 타입에 따라 다른 경로를 사용하도록 코드 구조를 수정하였다.</p> </td></tr> <tr> <td style="text-align: center;">문제점 및 해결 방안</td><td> <p>2. JLPT 단어장의 일부 한자가 깨져 보이는 현상</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>인코딩 문제라고 판단하고, UTF-8/EUC-KR 등으로 다시 확인했으나, 파일 자체는 UTF-8로 정상 저장되어 있었다. 사용하던 “맑은 고딕” 폰트가 한자를 제대로 지원하지 못해 렌더링이 깨지는 폰트 문제로 판단하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// ===== 총량: 출력 영역 ===== outputArea = new JTextArea(); outputArea.setEditable(b: false); outputArea.setFont(new Font(name: "SANS_SERIF", Font.PLAIN, size: 13)); JScrollPane scrollPane = new JScrollPane(outputArea); scrollPane.setBorder(BorderFactory.createTitledBorder(title: "출력")); add(scrollPane, BorderLayout.CENTER);</pre> <p>-> “맑은 고딕”에서 자바가 기본 제공하는 “SANS_SERIF” 폰트로 변경하여 정상 출력을 확인하였다.</p> </td></tr> </table> </td></tr> <tr> <td></td><td> <p>3. 단어 추가 시 중복 단어가 정상 처리되지 않는 문제</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p> </td></tr> <tr> <td style="text-align: center;">다음 주 계획</td><td> <ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 </td></tr> </table> </td></tr> <tbody> <tr> <td></td><td> <p>3. 단어 추가 시 중복 단어가 정상 처리되지 않는 문제</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p> </td></tr> <tr> <td style="text-align: center;">다음 주 계획</td><td> <ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 </td></tr> </table> </td></tr> </tbody> </table>	원인 분석	기존에는 모든 단어를 ‘Word.txt’ 파일에 저장하는 방식이였지만, 이로 인해 전공/토익/JLPT 단어가 한 파일에 섞여 저장되어, 단어장별로 단어를 구분해서 관리하거나 퀴즈 · 통계를 분리해서 사용하는 것이 어려웠다.	해결 방안	<pre>// 단어장 종류(전공 / 토익 / JLPT)를 나타내는 enum public enum BookType { MAJOR(path: "word_data/major_vocab.txt"), TOEIC(path: "word_data/toeic_vocab.txt"), JLPT(path: "word_data/jlpt_vocab.txt"); private final String path; // 각 단어장에 대응되는 파일 경로 BookType(String path) { this.path = path; } // 실제 파일 경로 반환 public String getPath() { return path; } }</pre> <p>-> ‘BookType’ 열거형(enum)을 정의하여 단어장 종류를 구분하였다. 또한, ‘FileManager’에서 저장/불러오기 시 ‘BookType’을 인자로 전달받아, 선택된 단어장 타입에 따라 다른 경로를 사용하도록 코드 구조를 수정하였다.</p>	문제점 및 해결 방안	<p>2. JLPT 단어장의 일부 한자가 깨져 보이는 현상</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>인코딩 문제라고 판단하고, UTF-8/EUC-KR 등으로 다시 확인했으나, 파일 자체는 UTF-8로 정상 저장되어 있었다. 사용하던 “맑은 고딕” 폰트가 한자를 제대로 지원하지 못해 렌더링이 깨지는 폰트 문제로 판단하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// ===== 총량: 출력 영역 ===== outputArea = new JTextArea(); outputArea.setEditable(b: false); outputArea.setFont(new Font(name: "SANS_SERIF", Font.PLAIN, size: 13)); JScrollPane scrollPane = new JScrollPane(outputArea); scrollPane.setBorder(BorderFactory.createTitledBorder(title: "출력")); add(scrollPane, BorderLayout.CENTER);</pre> <p>-> “맑은 고딕”에서 자바가 기본 제공하는 “SANS_SERIF” 폰트로 변경하여 정상 출력을 확인하였다.</p> </td></tr> </table>	원인 분석	인코딩 문제라고 판단하고, UTF-8/EUC-KR 등으로 다시 확인했으나, 파일 자체는 UTF-8로 정상 저장되어 있었다. 사용하던 “맑은 고딕” 폰트가 한자를 제대로 지원하지 못해 렌더링이 깨지는 폰트 문제로 판단하였다.	해결 방안	<pre>// ===== 총량: 출력 영역 ===== outputArea = new JTextArea(); outputArea.setEditable(b: false); outputArea.setFont(new Font(name: "SANS_SERIF", Font.PLAIN, size: 13)); JScrollPane scrollPane = new JScrollPane(outputArea); scrollPane.setBorder(BorderFactory.createTitledBorder(title: "출력")); add(scrollPane, BorderLayout.CENTER);</pre> <p>-> “맑은 고딕”에서 자바가 기본 제공하는 “SANS_SERIF” 폰트로 변경하여 정상 출력을 확인하였다.</p>		<p>3. 단어 추가 시 중복 단어가 정상 처리되지 않는 문제</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p> </td></tr> <tr> <td style="text-align: center;">다음 주 계획</td><td> <ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 </td></tr> </table>	원인 분석	단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.	해결 방안	<pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p>	다음 주 계획	<ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 		<p>3. 단어 추가 시 중복 단어가 정상 처리되지 않는 문제</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p> </td></tr> <tr> <td style="text-align: center;">다음 주 계획</td><td> <ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 </td></tr> </table>	원인 분석	단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.	해결 방안	<pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p>	다음 주 계획	<ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비
원인 분석	기존에는 모든 단어를 ‘Word.txt’ 파일에 저장하는 방식이였지만, 이로 인해 전공/토익/JLPT 단어가 한 파일에 섞여 저장되어, 단어장별로 단어를 구분해서 관리하거나 퀴즈 · 통계를 분리해서 사용하는 것이 어려웠다.																											
해결 방안	<pre>// 단어장 종류(전공 / 토익 / JLPT)를 나타내는 enum public enum BookType { MAJOR(path: "word_data/major_vocab.txt"), TOEIC(path: "word_data/toeic_vocab.txt"), JLPT(path: "word_data/jlpt_vocab.txt"); private final String path; // 각 단어장에 대응되는 파일 경로 BookType(String path) { this.path = path; } // 실제 파일 경로 반환 public String getPath() { return path; } }</pre> <p>-> ‘BookType’ 열거형(enum)을 정의하여 단어장 종류를 구분하였다. 또한, ‘FileManager’에서 저장/불러오기 시 ‘BookType’을 인자로 전달받아, 선택된 단어장 타입에 따라 다른 경로를 사용하도록 코드 구조를 수정하였다.</p>																											
문제점 및 해결 방안	<p>2. JLPT 단어장의 일부 한자가 깨져 보이는 현상</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>인코딩 문제라고 판단하고, UTF-8/EUC-KR 등으로 다시 확인했으나, 파일 자체는 UTF-8로 정상 저장되어 있었다. 사용하던 “맑은 고딕” 폰트가 한자를 제대로 지원하지 못해 렌더링이 깨지는 폰트 문제로 판단하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// ===== 총량: 출력 영역 ===== outputArea = new JTextArea(); outputArea.setEditable(b: false); outputArea.setFont(new Font(name: "SANS_SERIF", Font.PLAIN, size: 13)); JScrollPane scrollPane = new JScrollPane(outputArea); scrollPane.setBorder(BorderFactory.createTitledBorder(title: "출력")); add(scrollPane, BorderLayout.CENTER);</pre> <p>-> “맑은 고딕”에서 자바가 기본 제공하는 “SANS_SERIF” 폰트로 변경하여 정상 출력을 확인하였다.</p> </td></tr> </table>	원인 분석	인코딩 문제라고 판단하고, UTF-8/EUC-KR 등으로 다시 확인했으나, 파일 자체는 UTF-8로 정상 저장되어 있었다. 사용하던 “맑은 고딕” 폰트가 한자를 제대로 지원하지 못해 렌더링이 깨지는 폰트 문제로 판단하였다.	해결 방안	<pre>// ===== 총량: 출력 영역 ===== outputArea = new JTextArea(); outputArea.setEditable(b: false); outputArea.setFont(new Font(name: "SANS_SERIF", Font.PLAIN, size: 13)); JScrollPane scrollPane = new JScrollPane(outputArea); scrollPane.setBorder(BorderFactory.createTitledBorder(title: "출력")); add(scrollPane, BorderLayout.CENTER);</pre> <p>-> “맑은 고딕”에서 자바가 기본 제공하는 “SANS_SERIF” 폰트로 변경하여 정상 출력을 확인하였다.</p>																							
원인 분석	인코딩 문제라고 판단하고, UTF-8/EUC-KR 등으로 다시 확인했으나, 파일 자체는 UTF-8로 정상 저장되어 있었다. 사용하던 “맑은 고딕” 폰트가 한자를 제대로 지원하지 못해 렌더링이 깨지는 폰트 문제로 판단하였다.																											
해결 방안	<pre>// ===== 총량: 출력 영역 ===== outputArea = new JTextArea(); outputArea.setEditable(b: false); outputArea.setFont(new Font(name: "SANS_SERIF", Font.PLAIN, size: 13)); JScrollPane scrollPane = new JScrollPane(outputArea); scrollPane.setBorder(BorderFactory.createTitledBorder(title: "출력")); add(scrollPane, BorderLayout.CENTER);</pre> <p>-> “맑은 고딕”에서 자바가 기본 제공하는 “SANS_SERIF” 폰트로 변경하여 정상 출력을 확인하였다.</p>																											
	<p>3. 단어 추가 시 중복 단어가 정상 처리되지 않는 문제</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p> </td></tr> <tr> <td style="text-align: center;">다음 주 계획</td><td> <ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 </td></tr> </table>	원인 분석	단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.	해결 방안	<pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p>	다음 주 계획	<ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 																					
원인 분석	단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.																											
해결 방안	<pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p>																											
다음 주 계획	<ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 																											
	<p>3. 단어 추가 시 중복 단어가 정상 처리되지 않는 문제</p> <table border="1"> <tr> <td style="text-align: center;">원인 분석</td><td>단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.</td></tr> <tr> <td style="text-align: center;">해결 방안</td><td> <pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p> </td></tr> <tr> <td style="text-align: center;">다음 주 계획</td><td> <ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 </td></tr> </table>	원인 분석	단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.	해결 방안	<pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p>	다음 주 계획	<ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 																					
원인 분석	단어 추가 기능에서 ‘Word’ 객체 비교가 제대로 이루어지지 않아, 같은 영어 단어를 여러 번 입력하더라도 중복 여부가 정확히 체크되지 않았다. 특히 대소문자 차이('Apple' vs 'apple')나 앞뒤 공백('apple' vs ' apple ')이 있는 경우, 서로 다른 단어로 인식하여 중복 단어가 계속 추가되는 문제가 발생하였다.																											
해결 방안	<pre>// 소문자로 통일 후 비교 @Override public boolean equals(Object obj) { if (this == obj) return true; if (!(obj instanceof Word)) return false; Word other = (Word) obj; if (english == null other.english == null) return false; return english.equalsIgnoreCase(other.english); } @Override public int hashCode() { return english == null ? 0 : english.toLowerCase().hashCode(); } // 단어 추가 private void addWord() { String eng = engField.getText().trim(); String kor = korField.getText().trim(); // 단어 삭제 private void deleteWord() { String eng = engField.getText().trim();</pre> <p>-> ‘Word’ 클래스에서 ‘equals()’와 ‘hashCode()’를 오버라이드하여, 영어 단어를 비교할 때 소문자로 통일 후에 비교하도록 수정하고, 입력 받은 단어에 대해 ‘trim()’을 적용해 앞뒤 공백을 제거한 후 처리하도록 하였다.</p>																											
다음 주 계획	<ol style="list-style-type: none"> 최종 테스트 확인 최종 보고서 및 발표 자료 준비 																											

	이름	계획 역할	수행 내용	달성도%
팀원 기여도 평가	이강민	<p>〈프로젝트 총괄 / GUI 개발〉</p> <ul style="list-style-type: none"> - 전체 프로그램 구조 설계 및 GUI 통합 - MainFrame 화면 구성 및 이벤트 처리 - QuizFrame 연동 및 GUI 흐름 제어 - GUI 사용자 입력 검증 및 예외 처리 	<ul style="list-style-type: none"> - MainFrame의 단어장 선택 (JComboBox) 기능 추가 및 자동 로드 기능 구현 - 추가 · 삭제 · 목록 · 저장 · 불러오기 · 퀴즈 · 통계 버튼 기능 완성 - JLPT(한자) 단어 깨짐 현상 해결 - StatsFrame 화면 전환 및 통계 표시 기능 보완 - 전체 GUI 흐름 안정화 및 오류 없는 동작 검증 	100%
	장정원	<p>〈데이터 로직 개발 / 기능 통합〉</p> <ul style="list-style-type: none"> - WordBook 클래스 구현 (단어 등록, 검색, 통계 기능) - Word 클래스 정의 및 데이터 구조 설계 - 중복 처리 및 통계 알고리즘 작성 - 기능별 단위 테스트 수행 	<ul style="list-style-type: none"> - WordBook 구조 정리 및 중복 단어 처리 개선>equals/hashCode 보완) - trim() 적용으로 공백 입력에 대한 예외 처리 - 전공/토익/JLPT 분리 구조 적용 후 전체 데이터 흐름 점검 - FileManager와 WordBook 연동 테스트 수행 - 단어 통계(개수) 및 퀴즈 데이터 반영 검증 	100%
	신상현	<p>〈퀴즈 기능 / 문서화 및 발표 자료〉</p> <ul style="list-style-type: none"> - QuizFrame 클래스 구현 (랜덤 문제 출제 및 정답 판별) - 퀴즈 결과 저장 및 통계 기능 보조 개발 - 전체 프로젝트 테스트 케이스 작성 - 프로젝트 보고서 작성 - 발표 자료 제작 및 시연 영상 준비 	<ul style="list-style-type: none"> - QuizManager의 정답률 · 문제 수 계산 로직 보완 - 퀴즈 실행 시 단어장이 비어 있을 경우 안내 메시지 출력 추가 - 퀴즈 UI 구성 및 정답 처리 메시지 흐름 개선 - StatsFrame에 반영될 퀴즈 데이터의 연동 확인 - 활동일지 및 문서 정리 보조 (문서 구조 점검, 설명 보완) 	100%
	이민재	<p>〈파일 입출력 및 예외 처리 담당〉</p> <ul style="list-style-type: none"> - FileManager 파일 저장/불러오기 개발 - 예외 처리 로직 작성 - 인코딩 문제 해결 	<ul style="list-style-type: none"> - BookType 기반의 단어장 분리 저장 구조 구현 (major_vocab.txt / toeic_vocab.txt / jlpt_vocab.txt) - word_data 폴더 자동 생성 처리 - 저장/불러오기 오류 (FileNotFoundException 등)에 대한 예외 메시지 보완 - UTF-8 인코딩 점검 및 파일 구조 정리 - GUI 전체 파일 입출력 흐름 연동 테스트 수행 	100%