JavaPrograming\src\final_exam\week14\moreLogical\MoreLogicalHangman.java

```java
package easyjava.final_exam.week14.moreLogical;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Random;
import java.util.Scanner;

/**
 * 더 논리적인 형태로 개선된 행맨 게임 구현 클래스.
 */
// 행맨 게임 만들기 (BufferedReader 활용)
// 주어진 단어를 문자 하나씩 추측해서 맞추는 행맨(hangman) 프로그램을 작성하시오.
// - 처음에는 단어에 포함된 문자의 개수만큼 빈칸이 나타나며, 사용자는 빈칸에 들어갈 문자를 하나씩 추측한다.
// - 추측한 문자가 맞으면 빈칸 대신에 맞춘 문자를 출력한다.
// - 프로그램에서 사용할 문자열은 12개의 단어로 구성된 D:\Temp\words.txt 파일에 있는 문자열 중 무작위로 선택한다.
// - 여섯 번을 초과해서 잘못된 추측을 하면 게임이 종료된다.
public class MoreLogicalHangman {

    private static final int MAX_TRIES = 6; // 최대 시도 횟수
    private static final String WORDS_FILE = "D:\\temp\\words.txt"; // 단어 목록 파일 경로

    /**
     * 메인 메소드: 게임을 실행하는 진입점
     *
     * @param args 프로그램 실행 시 전달할 명령행 인수들
     * @throws IOException 파일 읽기 오류가 발생할 경우
     */
    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        char playAgain;

        do {
            String secretWord = getRandomWordFromFile(); // 무작위 단어 선택
            if (secretWord == null) {
                System.out.println("단어를 불러오는 데 문제가 발생했습니다.");
                return; // 파일을 불러오는 데 문제가 있으면 게임을 종료합니다.
            }
            playGame(secretWord, scanner); // 게임 실행

            System.out.print("한 번 더 게임할래요 (y/n)?: ");
            playAgain = scanner.next().charAt(0); // 다시 할지 여부 입력

        } while (playAgain == 'y' || playAgain == 'Y'); // 'y'나 'Y'를 입력할 때까지 반복
    }

    /**
     * 게임 실행 메소드
     *
     * @param secretWord 선택된 비밀 단어
     * @param scanner    사용자 입력을 받기 위한 Scanner 객체
     */
    private static void playGame(String secretWord, Scanner scanner) {
        StringBuilder dashes = createDashes(secretWord.length()); // 빈칸으로 구성된 단어
        int triesLeft = MAX_TRIES; // 남은 시도 횟수
        StringBuilder guessedLetters = new StringBuilder(); // 추측한 문자열 저장
```

```
57
58          while (triesLeft > 0 && !isWordGuessed(secretWord, dashes.toString())) {
59              displayGameStatus(dashes.toString(), guessedLetters.toString(), triesLeft); // 게임 상태 출력
60              char guess = getValidGuess(scanner, guessedLetters); // 유효한 추측 입력 받기
61
62              if (secretWord.contains(String.valueOf(guess))) {
63                  updateDashes(secretWord, dashes, guess); // 추측이 맞으면 빈칸 업데이트
64                  System.out.println("정확한 추측입니다!");
65              } else {
66                  triesLeft--; // 추측이 틀리면 시도 횟수 감소
67                  System.out.println("추측을 잘못했습니다.");
68              }
69              guessedLetters.append(guess); // 추측 문자열에 추가
70          }
71
72          if (isWordGuessed(secretWord, dashes.toString())) {
73              System.out.println("축하합니다! 단어를 맞췄습니다: " + secretWord);
74          } else {
75              System.out.println("게임 오버! 정답은 " + secretWord + "입니다.");
76          }
77      }
78
79      /**
80       * 파일에서 무작위 단어 선택 메소드
81       *
82       * @return 선택된 무작위 단어
83       */
84      private static String getRandomWordFromFile() {
85          Random random = new Random();
86          int lineCount = 0;
87          String randomWord = null;
88
89          try (BufferedReader reader = new BufferedReader(new FileReader(WORDS_FILE))) {
90              // 파일의 총 라인 수 세기
91              while (reader.readLine() != null) {
92                  lineCount++;
93              }
94
95              // 무작위로 선택할 라인 번호 결정
96              int randomLine = random.nextInt(lineCount) + 1;
97
98              // 파일을 다시 열어서 선택된 라인의 단어를 찾음
99              reader.close();
100             try (BufferedReader newReader = new BufferedReader(new FileReader(WORDS_FILE))) {
101                 for (int i = 0; i < randomLine; i++) {
102                     randomWord = newReader.readLine();
103                 }
104             }
105
106         } catch (IOException e) {
107             e.printStackTrace();
108         }
109
110         return randomWord;
111     }
112
113     /**
114      * 빈칸으로 구성된 StringBuilder 생성 메소드
```

```
115         *
116         * @param length 생성할 빈칸의 길이
117         * @return 빈칸으로 구성된 StringBuilder 객체
118         */
119        private static StringBuilder createDashes(int length) {
120            StringBuilder dashes = new StringBuilder();
121            for (int i = 0; i < length; i++) {
122                dashes.append('-');
123            }
124            return dashes;
125        }
126
127        /**
128         * 단어가 맞게 추측되었는지 확인하는 메소드
129         *
130         * @param secretWord 비밀 단어
131         * @param dashes     빈칸으로 구성된 문자열
132         * @return 단어가 맞게 추측되었는지 여부
133         */
134        private static boolean isWordGuessed(String secretWord, String dashes) {
135            return secretWord.equals(dashes);
136        }
137
138        /**
139         * 빈칸에 추측한 문자를 업데이트하는 메소드
140         *
141         * @param secretWord 비밀 단어
142         * @param dashes     빈칸으로 구성된 문자열
143         * @param guess      추측한 문자
144         */
145        private static void updateDashes(String secretWord, StringBuilder dashes, char guess) {
146            for (int i = 0; i < secretWord.length(); i++) {
147                if (secretWord.charAt(i) == guess) {
148                    dashes.setCharAt(i, guess);
149                }
150            }
151        }
152
153        /**
154         * 게임 상태를 화면에 출력하는 메소드
155         *
156         * @param dashes         빈칸으로 구성된 문자열
157         * @param guessedLetters 추측한 문자열
158         * @param triesLeft      남은 시도 횟수
159         */
160        private static void displayGameStatus(String dashes, String guessedLetters, int
    triesLeft) {
161            System.out.println("추측할 단어입니다: " + dashes);
162            System.out.println("지금까지 추측한 내용입니다: " + guessedLetters);
163            System.out.println("남은 추측 횟수: " + triesLeft);
164        }
165
166        /**
167         * 사용자로부터 유효한 추측을 입력 받는 메소드
168         *
169         * @param scanner        입력을 받기 위한 Scanner 객체
170         * @param guessedLetters 추측한 문자열
171         * @return 유효한 추측 문자
172         */
173        private static char getValidGuess(Scanner scanner, StringBuilder guessedLetters) {
```

```
174         char guess;
175         while (true) {
176             System.out.print("추측한 문자를 입력하세요: ");
177             String input = scanner.next().toLowerCase();
178 
179             // 입력이 한 글자의 알파벳인지 확인
180             if (input.length() != 1 || !Character.isLetter(input.charAt(0))) {
181                 System.out.println("유효하지 않은 입력입니다. 한 글자 알파벳을 입력하세요.");
182                 continue;
183             }
184 
185             guess = input.charAt(0);
186             // 이미 추측한 문자인지 확인
187             if (guessedLetters.toString().contains(String.valueOf(guess))) {
188                 System.out.println("이미 추측한 문자입니다. 다른 문자를 입력하세요.");
189             } else {
190                 break;
191             }
192         }
193         return guess;
194     }
195 }
196 
```