자바 과제4

리포트

2017.11.27 과제4

정수경

소프트웨어공학과

2017156037

**<1번>**

**[문제]**

**첨부한 words.txt 파일은 한 라인에 하나의 영어 단어가 들어가 있다. 이 파일을**

**모두 읽어 ArrayList<String>에 저장하고, 영어 단어를 입력받고 ArrayList를 검색하여**

**그 단어로 시작하는 모든 단어를 출력하는 프로그램을 작성하라.**

**[전략]**

문제1에서 해결해야 할 것 :

1. 파일을 찾아 읽은 후 // 8단원 파일 입출력 개념
2. 배열에 저장하고 // 7단원 ArrayList 컬렉션 클래스 개념 사용
3. 배열에서 사용자가 입력한 단어로 시작하는 단어 찾기

**[코드구조]**

public class WordSearch {

//멤버1 : readFile() 파일을 읽는 메소드

private void readFile() {}

//멤버2 : wordQuery() 단어 검색을 시행하는 메소드

private void wordQuery() {}

//main

public static void main(String[] args) {

//클래스의 객체 생성

WordSearch ws = new WordSearch();

//메소드 호출해서 실행

ws.readFile(); //1. words.txt 파일 읽기

ws.wordQuery();//2. 단어 검색하기

}

}

**[코드 짜면서 고민했던 부분]**

**(Idea1) 파일 읽어 들이기**

//<1단계> 읽기 전용 바이트 스트림 생성 후 파일 찾아 연결

**FileInputStream fin = null;**

**fin = new FileInputStream("c:\\Temp\\words.txt");**

* FileInputStream 클래스를 사용하는 이유 : 읽기 전용 바이트 스트림(텍스트, 바이너리 형식 모두 접근 가능)

//<2단계>

**Scanner fileScanner = new Scanner(fin);**

* 스캐너 클래스에 FileInputStream 객체를 인자로 주고 읽어들이게 함.

**(Idea2) 파일에 저장되어 있는 한 줄 당 한 단어 읽기**

**wordArray.add(fileScanner.nextLine());**

* Scanner 클래스의 nextLine() 메소드는 \n을 포함하는 한 라인을 읽고 \n을 제외한 나머지 문자열 리턴
* 파일에 한 줄 당 한 단어가 저장되어 있으므로 Scanner 클래스의 nextLine() 메소드 사용

**(Idea3) 입력한 단어로 시작하는 단어 찾기**

//입력 받은 문자열의 길이만큼 배열 안에 있는 단어들을 자르기

**int n=findword.length();**

**String cutword = arrword.substring(0, n);**

* 변수n의 용도는? 사용자가 입력한 단어의 길이를 알아내서 나중에 배열에 저장된 단어들을 순차 검색할 때, 앞에서부터 이 길이만큼 자르려고 한다.
* 그 이후에는 잘라진 단어와 사용자가 입력한 찾는 단어의 문자열을 비교한다.

**[소스코드]**

import java.io.\*; //입출력 클래스 사용하기 위해 패키지 임포트

import java.util.\*; //스캐너 클래스와 컬렉션 클래스 사용하기 위해 패키지 임포트

**//주의!! words.txt 파일을 저장한 위치 : c드라이브의 Temp 디렉토리**

/\* <문제1 : 해결해야 할 것>

\* 파일을 찾아 읽은 후,

\* 배열에 저장하고,

\* 배열에서 사용자가 입력한 단어로 시작하는 단어 찾기

\*/

public class WordSearch {

//컬렉션 클래스(ArrayList)의 객체 생성 : 파일에서 읽은 데이터를 저장할 배열

ArrayList <String> wordArray = new ArrayList<String>(); //ArrayList 생성

//멤버1 : readFile() 파일을 읽는 메소드

private void readFile() {

**FileInputStream fin = null;**

try {

//<1단계> 읽기 전용 바이트 스트림 생성 후 파일 찾아 연결

**fin = new FileInputStream("c:\\Temp\\words.txt");** //파일이 있는 위치 : c드라이브의 Temp 디렉토리

/\*FileInputStream 클래스를 사용하는 이유 :

읽기 전용 바이트 스트림(텍스트, 바이너리 형식 모두 접근 가능)

\*/

**Scanner fileScanner = new Scanner(fin);**

// 스캐너 클래스에 FileInputStream 객체를 인자로 주고 읽어들이게 함.

.

while(fileScanner.hasNext()) { //그 다음에 읽을 것이 남아있으면 true 반환

//<2단계> 파일에서 읽은 내용 arrayList <String> 배열에 한 줄씩 저장하기

**wordArray.add(fileScanner.nextLine());**

/\*(설명)

한 줄 당 한 단어가 저장되어 있으므로 Scanner 클래스의 nextLine() 메소드 사용

\n을 포함하는 한 라인을 읽고 \n을 제외한 나머지 문자열 리턴

\*/

}

fileScanner.close(); //스캐너 객체 반납

}

catch(IOException e) {

System.out.println("words.txt 파일 읽는 중 오류 발생");

}

}

//멤버2 : wordQuery() 단어 검색을 시행하는 메소드

private void wordQuery() {

Scanner scanner = new Scanner(System.in);

while(true) {

//검색할 단어 입력

System.out.print("단어>> ");

String findword = scanner.nextLine(); //사용자가 입력 시 공백을 허용할 것이므로

**int n=findword.length();** //n : 사용자가 입력한 단어의 길이 -> 나중에 배열에 저장된 단어를 이 길이만큼 자르기 위해

int count=0; //count : 단어 찾지 못한 경우 판별용 -> 배열을 끝까지 다 돌았는데 사용자가 입력한 단어를 찾지 못할 경우 count는 그대로 0 유지

//<3단계> arrayList 배열에서 단어 검색 (wordArray)

for(int i=0;i<wordArray.size();i++) { //배열의 크기만큼 반복

String arrword = wordArray.get(i); //배열에 저장된 단어 가져오기 -> get(i) 메소드로

//(1)배열에 저장된 단어들 중에 사용자가입력한 단어보다 단어 길이가 짧은 경우

if(arrword.length() < n)

continue; //검색 대상에서 제외되므로 i++로 이동

//(2)입력 받은 문자열의 길이만큼 배열 안에 있는 단어들을 자르기

**String cutword = arrword.substring(0, n);**

if(findword.equals(cutword)) {//비교해서 같으면

System.out.println(arrword); //단어 앞에서 끝까지 출력

count++;

}

//(3)배열을 다 돌았는데 찾지 못했을 경우

if(i==wordArray.size()-1 && count==0) //맨 마지막 인덱스일 때 count의 상태가 0이라면..

System.out.println("발견할 수 없습니다..");

}

/\*<Idea>

\* 입력 받은 문자열의 길이만큼 배열 안에 있는 단어들을 잘라서 앞부분 비교 -> String 클래스의 substring() 메소드 사용

\* (경우의 수)

\* -> 배열에 저장된 단어들 중에 사용자가입력한 단어보다 단어 길이가 짧은 경우 : 검색 대상에서 제외

\* -> 배열을 다 돌았는데 찾지 못했을 경우 :

\*/

//무한 반복문 종료조건

if(findword.equals("exit")) {

System.out.println("종료합니다...");

break; //while문 벗어나는 경우 그 다음에 시행할 명령문 없으므로 프로그램 완전히 끝남

}//end if

}//end while

scanner.close(); //스캐너 객체 반납

}//end wordQuery()

//main

public static void main(String[] args) {

//클래스의 객체 생성

WordSearch ws = new WordSearch();

//메소드 호출해서 실행

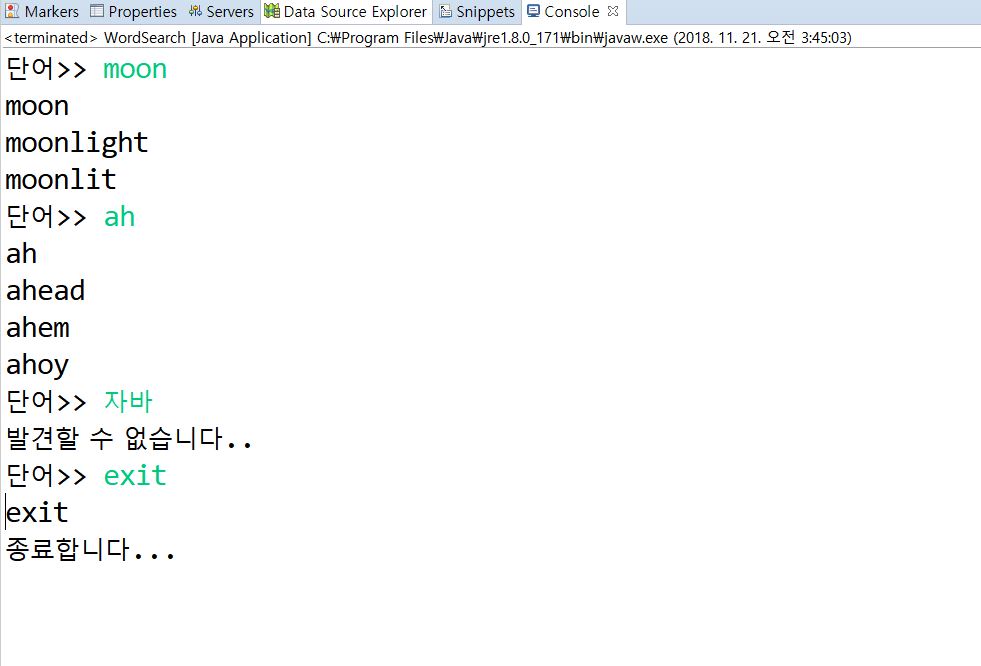
ws.readFile(); //1. words.txt 파일 읽기

ws.wordQuery();//2. 단어 검색하기

}

}

**[실행화면]**



**<2번>**

**[문제]**

**문자열을 파일에서 한 줄씩 입력 받아 저장한 후 입력 순서대로와 역순으로 출력하는 프로그램을 작성하라.**

**입력 순으로 출력하는 프로그램은 ArrayList를  사용하고, 역순으로 출력하는 프로그램은 LinkedList 사용하여 스택을 구현하여 작성하라.**

**[전략]**

문제에서 해결해야 할 것 :

1. 파일에서 한 줄씩 입력 받아

2. 배열에 저장 후,

3. 순서대로 출력 -> ArrayList 컬렉션 클래스 사용하자!

4. 역순으로 출력 -> push(), pop() 메소드를 내가 정의한 Stack클래스 안에 구현하자.

//참고로 Lifo라는 이름의 클래스로 Stack의 자료구조를 구현하였다.

-> LinkedList 컬렉션 클래스 사용하자.

**[코드구조 설명]**

//Fifo : 순서대로 출력하기 위한 클래스

Class Fifo{

//(1) readFile() 구현 : 파일을 읽는 메소드

private void readFile() {}

//(2) fifoPrint() 구현 : fifo() 파일에 있는 순서대로 출력하는 메소드

private void fifoPrint() {}

//(3) main

public static void main(String[] args) {

//객체 생성

Fifo fifo = new Fifo();

fifo.readFile(); //1. 파일 읽기

fifo.fifoPrint(); //2. 파일에 있는 순서대로 출력(fifo)

//객체 생성

Lifo lifo = new Lifo();

lifo.lifoPrint();

//3. 파일에 저장되어 있는 역순으로 출력(lifo)

}

}

//Lifo : 역순으로 출력하기 위한 클래스

class Lifo {

//(1) 스택의 push() 구현

public void push() {

}

//(2) 스택의 pop() 구현

public void pop() {

}

//(3) lifoPrint() 구현 : 화면에 출력 <- 메소드 실행순서 push 그다음 pop..

public void lifoPrint() {

push();

pop();

}

}

**[소스코드]**

import java.io.\*; //파일 입출력,

import java.util.\*;

**//주의!! p2-sample.txt의 위치 : c드라이브의 Temp 디렉토리에 위치**

/\*<문제2 : 해결해야 할 것>

\* 파일에서 한 줄씩 입력 받아

\* 배열에 저장 후,

\* 순서대로 출력 -> ArrayList 컬렉션 클래스 사용

\* 역순으로 출력 -> LinkedList 컬렉션 클래스 사용해서 스택(push, pop) 구현

\*/

//Fifo : 순서대로 출력하기 위한 클래스

public class Fifo {

//컬렉션 클래스(ArrayList)의 객체 생성 : 파일에서 읽은 데이터를 저장할 fifo 출력용 배열

static ArrayList <String> wordArray = new ArrayList<String>(); //ArrayList 생성

//멤버1 : readFile() 파일을 읽는 메소드

private void readFile() {

FileInputStream fin = null;

try {

//<1단계> 읽기 전용 바이트 스트림 생성 후 파일 찾아 연결

fin = new FileInputStream("c:\\Temp\\p2-sample.txt"); //파일이 있는 위치 : c드라이브의 Temp 디렉토리

/\*FileInputStream 클래스를 사용하는 이유 :

읽기 전용 바이트 스트림(텍스트, 바이너리 형식 모두 접근 가능)

\*/

Scanner fileScanner = new Scanner(fin);

//스캐너 클래스에 파일을 객체로 주고 읽어들이게 함.

/\*Scanner 클래스를 사용하는 이유 : \*/

while(fileScanner.hasNext()) { //그 다음에 읽을 것이 남아있으면 true 반환

//<2단계> 파일에서 읽은 내용 arrayList <String> 배열에 한 줄씩 저장하기

wordArray.add(fileScanner.nextLine());

/\*(설명)

한 줄 당 한 단어가 저장되어 있으므로 Scanner 클래스의 nextLine() 메소드 사용

\n을 포함하는 한 라인을 읽고 \n을 제외한 나머지 문자열 리턴

\*/

}

fileScanner.close(); //스캐너 객체 반납

}

catch(IOException e) {

System.out.println("p2-sample.txt 파일 읽는 중 오류 발생");

}

}

//멤버2 : fifo() 파일에 있는 순서대로 출력하는 메소드

private void fifoPrint() {

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

System.out.println(wordArray.get(i)); //배열에 저장된 단어 가져오기 -> get(i) 메소드로

}

}

//main

public static void main(String[] args) {

//객체 생성

Fifo fifo = new Fifo();

fifo.readFile(); //1. 파일 읽기

fifo.fifoPrint(); //2. 파일에 있는 순서대로 출력(fifo)

//객체 생성

Lifo lifo = new Lifo();

lifo.lifoPrint();

//3. 파일에 저장되어 있는 역순으로 출력(lifo)

}

}

//Lifo : 역순으로 출력하기 위한 클래스

class Lifo {

int index=0;

LinkedList <String> wordLinked = new LinkedList<String> ();

/\*LinkedList의 객체 생성 : 파일에서 읽은 데이터를 저장할 lifo 출력용 배열

-> 동시에 스택에 요소를 저장할 공간 배열로 사용

\*/

//(1) 스택의 push() 구현

public void push() {

for(int j=0;j<Fifo.wordArray.size();j++) {

wordLinked.add(Fifo.wordArray.get(j));

index++;

}

}

//(2) 스택의 pop() 구현

public void pop() {

while(true) {

index--; //index 현재 상태는 배열의 총 갯수이다. 뒤에서 배열의 맨 끝 요소를 출력하면 개수-1이므로 하나 감소

System.out.println(wordLinked.get(index));

if(index==0) //

break;

}

}

//화면에 출력 <- 메소드 실행순서 push 그다음 pop..

public void lifoPrint() {

push();

pop();

}

}

**[실행결과]**

