자료구조 실습 보고서

[제 4주]

제출일 : 2018.3.28.수요일

학번 : 201604137

이름 : 김예지

[프로그램 설명서]

```
import java.io.*;

public class Test {

    public Test() {
    }
    public void run(String file) {

        int words = 0, chars = 0;

        try {

            // 한 줄 읽을 준비
            BufferedReader br = new BufferedReader(new FileReader(file));
            String line = br.readLine(); // 한 줄 읽는다
            while (line != null) {
                MyStringTokenizer mst = new MyStringTokenizer(line, " ");
                while (mst.hasMoreTokens()) {
                    ++words;
                    String word = mst.nextToken().toUpperCase();
                    chars += word.length();
                }
                line=br.readLine(); //그 다음줄을 읽어서 아무 내용이 없을을 읽어들인다.(==NULL)
            }

            br.close();

        } catch (Exception e) {
            System.out.println(e);
        }
        System.out.println("words:" + words);
        System.out.println("characters:" + chars);
    }
```

BufferReader 역할은

1.StreamReader가 변환한 문자데이터를 버퍼링해서 블록단위나 라인단위의 처리가 가능하도록 해줌

2.Stream을 받아 Buffer 형태로 그 값을 읽는 클래스. 즉 외부에서 입력받은 값을 InputStream이 받아 다시 Buffer에 담아 놓은 것.

한 줄 읽어 들인 후 line에 저장하고 hasMoreToken으로 다음 words.length() == index가 아닐 때 까지 확인하면서 단어의 개수를 세고 다음 토큰을 가져오는 것을 반복한다.

```java
public class MyStringTokenizer {

    private String[] words;
    private String line = "";
    private String token = "";
    private int index; //가져오는용 hasNext용

    public MyStringTokenizer(String line, String token) {
        this.line = line;
        this.token = token;//delimiter
        this.words = makeWords(line, token);
        this.index = 0;
    }

    //단어의 배열을 만들어 주는 함수
    public String[] makeWords(String line, String token) {
        String[] tmp = null;

        int splitIndex = 0; // 나중에 tmp에 추가할때 참조하기위한 인덱스. 첫번째는 tmp의0번째에 넣고..
        int splitLength = 1; // 스트링배열의 개수 띄어쓰기3개면 단어4개=> 배열도 4개만큼 만들어줌
        int beginIndex = 0; // 자를때 어디서부터 어디까지 잘라야하는지의 기준
        int lastIndex = 0; //저는 lastIndex대신 i를 사용하여 단어의 마지막 위치를 표시하였습니다.

        for (int i = 0; i < line.length(); i++) {
            if (line.charAt(i) == ' ')
                splitLength++;
        }

        tmp = new String[splitLength];

        for (int i = 0; i < line.length(); i++) {
            if (splitLength == (splitIndex + 1)) { // 맨 마지막부분에서의 예외처리
                tmp[splitIndex] = line.substring(beginIndex, line.length());
                break;
            } else if (line.charAt(i) == ' ') {
                tmp[splitIndex] = line.substring(beginIndex, i);
                beginIndex=i+1; //구분자 다음 글자를 가져오기 위해서 +1
                splitIndex++;
            }
        }
        return tmp;

    }

    public boolean hasMoreTokens() {
        if(words.length == index) return false;
        else return true;
     }

    public String nextToken() {
        return words[index++];
    }
```

int splitIndex : tmp배열에 단어를 하나씩 담을 때 참조하기 위한 인덱스 변수.

int splitLength : 띄어쓰기(구분자)로 단어의 개수를 구분해주는 변수.

int beginIndex : 단어를 자를 때 기준이 되는 변수.

makeWords() 매소드는 메모장에 있는 한 줄의 내용을 단어의 형태로 끊어서 배열을 만들어주는 매소드다.

line의 길이만큼 포문을 돌려서 띄어쓰기(구분자)를 발견할 때 마다 splitLength를 증가시켜 단어의 개수를 센다.

단어 개수만큼의 길이를 가진 tmp배열을 만들고, 구분자를 만나면 구분자 전까지의 단어를 잘라서 tmp에 차례대로 넣어준다.
[단어개수==인덱스 길이]가 되면 마지막 단어를 넣고 포문을 끝낸다.
(lastIndex를 따로 설정해주는 대신 i로 설정하여 beginIndex로 조율했음.)

token이 더 존재하는가?를 묻고 makeWords()로 뽑아낸 길이가 index길이와 같으면 다 뽑아낸 것이므로 false반환, 아니면 true이므로 nextToken으로 들어와서 index번째의 단어를 꺼내온다.
꺼내온 다음에는 그 다음배열을 가리켜야 하므로 index++해줌.

[실행 결과 분석]

1.프로그램을 실행시키게 되면 저장된 파일을 Test로 넘겨준다.
2.내용을 읽어 들여 line에 저장한 후 객체에 line과 구분자를 넘겨준다.
3.객체 내에 있는 makeWords매소드에서 단어들을 따로 구별하여 tmp 배열에 저장하고 그것을 리턴한다.
4.객체의 hasMoreTokens에 다음 token이 존재하는지 확인하며 없을때까지 단어의 개수, 형태소의 개수를 센다.