PL Assignment #2 : Recognizing Tokens

과제물 부과일 : 2014-03-25(화)

Program Upload 마감일 : 2014-03-31(월) 23:59:59

문제

다양한 형태의 identifier, integer number(**음수 포함**) 들로 이루어진 text file을 입력 받아, 각 요소를 인식하여 출력하는 program을 작성하시오. Input file name은 as02.txt이다.

예를 들어 as02.txt file의 내용이 아래와 같다면,

```
banana   267  h  cat  -3789  7   y2010
```

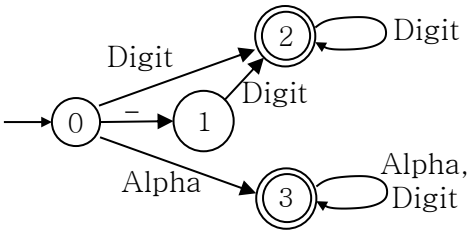
출력은 아래와 같아야 한다.

```
id: banana
int: 267
id: h
id: cat
int: -3789
int: 7
id: y2010
```

Regular Expression

```
id:     Alpha[Alpha|Digit]*
int:    Digit+ | "-" Digit+
Alpha:  [A-Z] | [a-z]
Digit:  [0-9]
```

mDFA



Programming

Token 표현하기

```
	public enum TokenType{
		ID(3), INT(2);

		private final int finalState;

		TokenType(int finalState) {
			this.finalState = finalState;
		}
	}
```

Data Type

```
	public static class Token {
		public final TokenType type;
		public final String lexme;

		Token(TokenType type, String lexme) {
			this.type = type;
			this.lexme = lexme;
		}

		@Override
		public String toString() {
			return String.format("[%s: %s]", type.toString(), lexme);
		}
	}
```

Programming-TM

Control state transition of transition diagram by TM(transition matrix).

"Array of accept(final) states"를 사용

```
	private int transM[][];
	private boolean accept[];
	private String source;
	private int pos;

	public Scanner(String source) {
		this.accept = new boolean[]{ false, false, true,
true };// Final states: 2,3
		this.transM = new int[4][128];
		this.source = source == null ? "" : source;
		init_TM();
	}

	private void init_TM() {
	     transM[4][128] = { {...}, {...}, {...}, {...} };
	     // values of entries:  -1, 0, 1, 2, 3 : next state
	     // TransM[0]['0'] = 2, ..., TransM[0]['9'] = 2,
	     // TransM[0]['-'] = 1,
	     // TransM[0]['a'] = 3, ..., TransM[0]['z'] = 3,
	     // TransM[1]['0'] = 2, ..., TransM[1]['9'] = 2,
```

```
            // TransM[2]['0'] = 2, ..., TransM[1]['9'] = 2,
            // TransM[3]['A'] = 3, ..., TransM[3]['Z'] = 3,
            // TransM[3]['a'] = 3, ..., TransM[3]['z'] = 3,
            // TransM[3]['0'] = 3, ..., TransM[3]['9'] = 3,
            // ...
            //    The values of the other entries are all -1.
      }

      private Token nextToken(){
            char c = Character.SPACE_SEPARATOR;
            StringBuffer sb = new StringBuffer();
            Token result = null;

            int StateOld = 0, StateNew;
            boolean acceptState = false;


            while( pos < source.length() ){
                  if(!Character.isWhitespace( c =
source.charAt(pos)))
                        break;
                  pos++;
            }//의미 없는 공백문자 무시

            if( pos >= source.length() )
                  return null; //input data가 더 이상 없을 경우

            while (!acceptState) {
                  StateNew = transM[StateOld][c]; //입력문자로 새로운 상태 판별

                  if (StateNew == -1) {      // 입력된 문자의
상태(StateNew)가 reject일때 이전문자(StateOld)의 상태로 reject, accept판별
                        if (accept[StateOld]) {
                              acceptState = true; break;
                        } // accept
                        else { acceptState = false; break; } //
reject
                  }

                  sb.append(c);
                  StateOld = StateNew;

                  pos++;
                  if(pos < source.length())
                        c = source.charAt(pos);
                  else
                        c = 0; // Null Character
            }

            if(acceptState){
                  for (TokenType t : TokenType.values()){
                        if(t.finalState == StateOld){
                              result = new Token(t, sb.toString());
                              break;
                        }
                  }
            }else
                  System.out.println(String.format("acceptState
error %s\n", sb.toString()));
```

```
        return result;
    }

    public List<Token> tokenize() {
        //Token 리스트반환, nextToken()이용..

        return …
    }

    public static void main(String[] args){
        //txt file to String

        String source = …
        Scanner s = new Scanner(source);
        List<Token> tokens = s.tokenize();

        //print
        …
    }
```

유의 사항

- 입력 data는 프로그램을 제대로 검증할 수 있는 data로 구성되어야 한다.
- 반드시 transition matrix를 사용하여야 한다.