システムソフトウェア特論 課題#1

学籍番号:1831013 泉 祐太

2020 年 2 月 14 日

1　今回の課題

このレポートでは、演習 1-2 の (a)〜(d) の 4 問を扱う。以下の節で (a)〜(d) の各々について詳細に述べていく。実行例は 2 章で示す。

1.1　課題 1-2(a)

課題は Toknizer に、1 つ戻る back() を追加するものであった。

今回の実装は、「ひとまず Toknizer の位置の直前のトークンを別の変数に持たせて、back() が呼ばれたらそちらを表示する」ことで戻る機能とする方針で行っている。Toknizer の変更部分を載せる。

```
boolean flag = false;
public String prevTok() { return prev; }
public void fwd() {
  if(flag == false && !eof && sc.hasNext()) {
    flag = true;
    tok = sc.next();
  }
  else if(!eof && sc.hasNext()) {
    prev = tok;
    tok = sc.next();
  }
  else { eof = true; tok = "$"; }
}
```

flag は Toknizer 生成時に実行される fwd() とそれ以外を区別するために定義してある。初回以外の fwd() で直前の状態を変数 prev に持たせ、main クラス内で b を入力することで back()(ここでは prevTok() という名前で実装してある) を呼び、prev を表示させる。

この方針では、直前の状態を別変数に保持するだけなので、Toknizer が直前のトークンに移動するわけではない。従って、連続で back() を実行できないようになっているのが問題点である。List に Toknizer の持っているトークンの内容を逐次追加することで、Toknizer の通った部分全てを保持すれば自由に前後への行き

来が可能となると考えられる。

1.2 課題 1-2(b)

課題は、トークン文字列 s を次に Toknizer に読ませるような機能 push(s) を追加するものであった。

今回の実装は、「push(s) が呼ばれたら s を一旦他の変数に持たせておき、次の fwd() が呼ばれたときに代わりに s を表示させるようにする」方針で行っている。Toknizer の変更部分を載せる。

```
public void push(String sstr) { pushed = sstr; }
  public void fwd() {
    if(pushed != null){
      if(prev != null){ prev = tok; }
      tok = pushed;
      pushed = null;
    } else if(!flag && !eof && sc.hasNext()) {
      flag = true;
      tok = sc.next();
    } else if(!eof && sc.hasNext()) {
      prev = tok;
      tok = sc.next();
    }
      else { eof = true; tok = "$"; }
}
```

Toknizer での push メソッド自体は、文字列 s を別の変数 (ここでは pushed) に保存するだけである。pushed が null であるときは追加する文字列がないので通常通りの fwd() を実行するが、pushed に中身があるときは sc.next() を取得する代わりに pushed を次のトークンとするようになっている。1 つめの else if 節は課題 (a) の変更点であるので、(b) での変更点は一番最初の if 節のみである。

1.3 課題 1-2(c)

課題は、toknizer を新しいファイル s に切り替える open(s) を追加するものであった。

今回の実装は、「main 関数内で、open(s) が呼ばれたときに新しい toknizer を生成する」方針で行っている。toknizer は変更がないので、今回は main クラスの変更点のみを載せる。

```
if(line.equals("o")) {
  System.out.print("Next file->:");
  String nfile = sc.nextLine();
  tok = new Toknizer(nfile);
}
```

main クラス内で o を入力したときに、新しいファイル名を変数 nfile で受けて、nfile を読む新しい

toknizer に tok を置き換えることで open(s) を実現している。

1.4 課題 1-2(d)

好きな機能を toknizer に追加せよという課題だったので、「n 個先のトークンまでジャンプする」機能を追加した。

今回の実装は、「for 文で n 回 fwd() を繰り返す、n は一旦 String 型で受けるので int 型に変換する」方針で行った。main クラス内でこの機能を実装したので、main クラスの変更点のみを載せる。

```
if(line.equals("j")) {
  System.out.print("Input number->:");
  String skip = sc.nextLine();
  int skips = Integer.parseInt(skip);
  for(int i=0; i < skips; i++){
    tok.fwd();
    if(tok.isEof()){ break; }
  }
  System.out.println(tok.curTok());
}
```

変数 skip で飛ばしたい行数を一旦 String 型として受けて、メソッド parseInt で整数型に変換、これを for 文内の繰り返し回数としている。n の値が大きいときには、途中で EOF に到達することが考えられるので、一応 EOF に到達した場合にはその場で for 文で脱出することにしている。最後に n 個先のトークンを表示して終了である。

2　実行例

Sam12d.java に上の (a)〜(d) の全ての内容を実装し、Sample.java(これは”Hello, World”を表示する) を toknizer に与えて実行した。実行例は以下の通りである。

```
E:\programs\java>java Sam12d Sample.java
> p
class
> f
Sample{
> f
public
> b
Sample{
> p
public
> f
static
> n
input->:new
> p
static
> f
new
> f
void
> o
Next file->:Sam12a.java
> p
import
> q
```

なお、プログラム本体はメールで添付する。

3　アンケート

A1.
プログラミング自体は言語によって好き嫌いがあるが、使用する言語に制限がなければ好きである。
研究室で Python を扱っている都合上、使い慣れている言語に対してはプログラムを組んでいて楽しいと思う部分が強いが、初めて触る言語であったり、苦手なタイプの言語でのプログラミングを強制される場合は習得までの時間であったり、そもそもの苦手意識によって楽しくなくなる部分があるかなと感じている。個人的には関数型言語は合わないように思っている。
A2.
様々なプログラミング言語が存在することに対しては、各々の言語に強み弱みが存在するので各々が他の言語を補うという観点から重要なのではと考えている。UI を作成可能であることを知っている言語は java しかないし、Python はかなり幅広い問題に対して対応できるが実行速度がいまいちで、C++ に書き換えられるならその方がよいなど、複数言語があるからには各々の言語なりの長所を我々が生かしてやる必要があると考えている。A3.
このレポートを提出したのはタイトルにもある通り 2 月なので、第 1 回講義にどのような内容をやったかはテキストを参照して思い出している形になるが、2 月になった今でも共通で、java を触るのはおおよそ 4 年ぶりであり、更に直前まで研究のために Python を触っていたことから、セミコロンを打ち忘れるエラーが多発した。プログラムを書く楽しみがまた増えたので良かったなと感じている。