INF2100

Oppgaver 26. september til 1. oktober 2007

1 Språket

Denne uken skal vi implementere en utvidelse av språket fra forrige ukes oppgave. Syntaksen er vist i figur 1 på neste side og betydning er stadig ganske enkel:

- Interpreten vår skal lese uttrykk fra tastaturet (dvs *standard inn*); etter hvert heltall skal det stå et semikolon.
- Hvis uttrykket er et heltall, skal det skrives ut.
- Hvis uttrykket er «⟨name⟩ = ⟨number⟩», skal interpreten huske at dette navnet er tilordnet nummeret; ingenting skal skrives ut.
- Hvis uttrykket er et navn (uten noe «= ⟨number⟩»), skal det tilordnede nummeret skrives ut.

1.1 Et eksempel

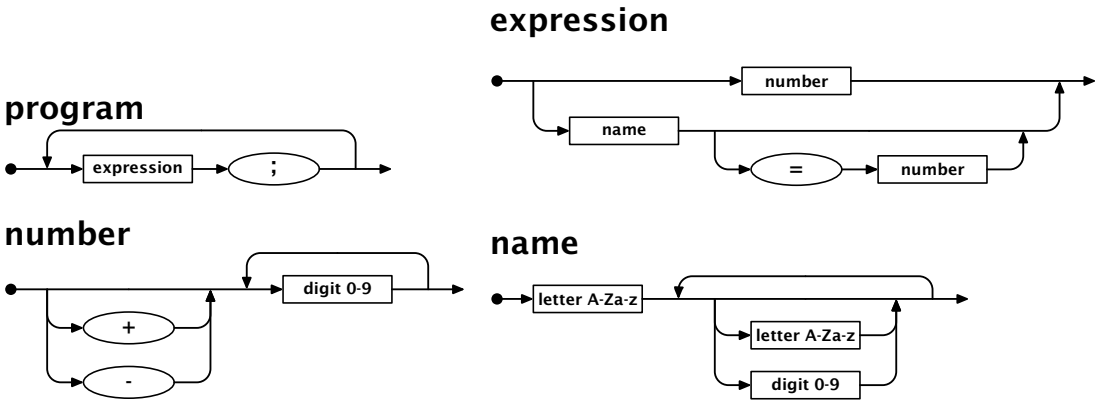
Anta at vi har en «kildefil» med navn `demo.num` og som ser slik ut:

```
pi100 = 314;
r= 6
;
1; -345;  pi = +003;  pi100;  pi;  0; -0999;  r;
r = +;  r;
```

Her er et eksempel på en kjøring med denne filen:

```
> java NumExpr <demo.num
1
-345
314
3
0
-999
6
Error in line 5: Illegal number: '+'!
```

Og her er en testkjøring av skanneren:



Figur 1: Programmeringsspråkets syntaks er definert av disse fire jernbanediagrammene.

```
> java NumExpr -testscanner <demo.num
Scanner: nameToken 'pi100'
Scanner: assignToken
Scanner: numberToken 314
Scanner: semicolonToken
Scanner: nameToken 'r'
Scanner: assignToken
Scanner: numberToken 6
Scanner: semicolonToken
Scanner: numberToken 1
Scanner: semicolonToken
Scanner: numberToken -345
Scanner: semicolonToken
Scanner: nameToken 'pi'
Scanner: assignToken
Scanner: numberToken 3
Scanner: semicolonToken
Scanner: nameToken 'pi100'
Scanner: semicolonToken
Scanner: nameToken 'pi'
Scanner: semicolonToken
Scanner: numberToken 0
Scanner: semicolonToken
Scanner: numberToken -999
Scanner: semicolonToken
Scanner: nameToken 'r'
Scanner: semicolonToken
Scanner: nameToken 'r'
Scanner: assignToken
Error in line 5: Illegal number: `+´!
```

2 Koden

Her er en implementasjon av en interpret for språket; koden ligger også på filen ~inf2100/kode/numbers/NumExpr.java og er tilgjengelig på nettet som http://www.ifi.uio.no/~inf2100/kode/numbers/NumExpr.java.

```
import java.io.*;
import java.util.*;

class NumExpr {
    public static void main(String arg[]) {
        Map<String,Long> nameTab = new HashMap<String,Long>();

        Scanner.readNext();
        if (arg.length>0 && arg[0].equals("-testscanner")) {
            while (true) {
                System.out.print("Scanner: "+Scanner.curToken);
                if (Scanner.curToken == Token.nameToken)
                    System.out.print(" '"+Scanner.curName+"'");
                if (Scanner.curToken == Token.numberToken)
                    System.out.print(" "+Scanner.curNumber);
                System.out.println();

                if (Scanner.curToken == Token.eofToken)
                    break;
                Scanner.readNext();
            }
        } else {
            while (true) {
                if (Scanner.curToken == Token.eofToken)
                    break;

                Scanner.check(Token.nameToken, Token.numberToken);
                if (Scanner.curToken == Token.nameToken) {
                    String name = Scanner.curName;
                    Scanner.readNext();
                    if (Scanner.curToken == Token.assignToken) {
                        // Expression is: <name> = <number>
                        Scanner.readNext();
                        Scanner.check(Token.numberToken);
                        nameTab.put(name, Scanner.curNumber);
                        Scanner.readNext();
                    } else {
                        // Expression is: <name>
                        if (! nameTab.containsKey(name))
                            Error.error("line "+Scanner.curLineNum,
                                        "Unknown name `"+name+"´!");
                        System.out.println(nameTab.get(name));
                    }
                } else {
```

```
                    // Expression is: <number>
                    System.out.println(Scanner.curNumber);
                    Scanner.readNext();
                }
                Scanner.check(Token.semicolonToken);
                Scanner.readNext();
            }
        }
    }
}

enum Token {
    assignToken, eofToken, nameToken, noToken, numberToken, semicolonToken;
}

class Scanner {
    public static Token curToken = Token.noToken;
    public static String curName = "";
    public static long curNumber = 0;
    public static int curLineNum = 0;

    private static LineNumberReader stdIn =
        new LineNumberReader(new InputStreamReader(System.in));
    private static String curLine = "";
    private static int curPos = 0;

    public static void readNext() {
        curToken = Token.noToken;
        while (curToken == Token.noToken) {
            //** Her må det inn kode! **
        }
    }

    private static boolean isLetterAZ(char c) {
        return ('a'<=c && c<='z') || ('A'<=c && c<='Z');
    }

    private static boolean isNameChar(char c) {
        return isLetterAZ(c) || Character.isDigit(c);
    }

    private static String readName() {
        int startPos = curPos-1;
        while (curPos<curLine.length() && isNameChar(curLine.charAt(curPos))) {
            ++curPos;
        }
        return curLine.substring(startPos,curPos);
    }

    private static long readNumber() {
```

```
        String n = ""+curLine.charAt(curPos-1);

        while (Character.isDigit(curLine.charAt(curPos)))
            n += curLine.charAt(curPos++);

        try {
            return Long.parseLong(n.startsWith("+") ? n.substring(1) : n);
        } catch (NumberFormatException e) {
            Error.error("line "+Scanner.curLineNum,
                        "Illegal number: `"+n+"´!");
        }
        return 0;  // Will never be run!
    }

    static void check(Token t1) {
        if (curToken != t1)
            Error.error("line "+curLineNum,
                  "Expected a "+t1+" but found a "+curToken);
    }

    static void check(Token t1, Token t2) {
        if (curToken!=t1 && curToken!=t2)
            Error.error("line "+curLineNum,
                  "Expected a "+t1+" or a "+t2+" but found a "+curToken);
    }
}

class Error {
    static public void error(String where, String message) {
        System.out.println("Error" +
                           (where.equals("") ? "" : " in "+where) +
                           ": " + message);
        System.exit(1);
    }

    static public void error(String message) {
        error("", message);
    }
}
```

3 Oppgaven

Oppgaven er å skrive koden som mangler i Scanner-klassen slik at den oppfører seg som vist i eksemplene.