# Snobol
# and
# Snowflake

COMP360

*“The original name was **S**ymbolic **EX**pression **I**nterpreter (SEXI)*

*It was clear that we needed another name!! We sat and talked and drank coffee and shot rubber bands and after much too much time someone said "We don't have a Snowball chance in hell of finding a name". All of us yelled at once, "WE GOT IT -- SNOBOL" in the spirit of all the BOL languages. We then stretched our mind to find what it stood for.”*

David Farber

# Schedule

- The lexical scanner for the Snowflake language is due by noon on Friday, February 10, 2017
- The first exam is Friday, February 17, 2017

# Recursive Descent Token Management

```
java.util.ArrayList<Token>    tokenList;
int         curTok = 0;                    // current token index

Token getNextToken( ) {
      if ( curTok >= tokenList.size( ) ) {
         return null;                      // no more tokens
      }
      return tokenList.get( curTok++ );
   }
```

# Looking Ahead

```
Token peek( ) {
        if ( curTok >= tokenList.size() ) {   // any more tokens
            return nullToken;                 // if not, return null
        }
        return tokenList.get( curTok );
 }
```

# What type is next?

- This method returns true if the next token is the specified type

```
boolean peekIs(String maybe) {
    if (curTok >= tokenList.size()) {        // if no more tokens
        return false;
    }
    return maybe.equals(tokenList.get(curTok).getType());
}
```

# Using the methods

```
boolean production() {
        int savePos = curTok;       // save token index
        Token thing = getNextToken();
        if (peekIs("variable") ) {
                curTok = savePos;   // restore token index
                return false;
        }
        return true;
}
```

# Have to Use All Tokens

- Consider a language that is parsing a list of statements

```
dog = cat + 5;
bad + cow ! good;
```

- The recursive descent method will recognize the first line, but return false when the parsing the second
- The parser has to recognize the difference between the end of the program and ending with bad syntax

# Main Procedure

```
main() {
        expr();                 // start symbol
        if (curTok != tokenList.size()) {
                print "Syntax Error";
        }
}
```

# Different Language Style

- Snobol is significantly different from most languages
- It is an older procedural language
- Snobol is designed for string manipulation
- Pattern matching is a basic part of Snobol

- Snowflake, the toy language we will be compiling, is similar to Snobol, but much simplier

# Snobol

- SNOBOL (**S**tri**N**g **O**riented and sym**BO**lic **L**anguage) is a series of computer programming languages developed between 1962 and 1967 at AT&T Bell Laboratories by David J. Farber, Ralph E. Griswold and Ivan P. Polonsky
- Snobol only has one statement type
- Snobol is line oriented, probably because it was created in the days of punch cards

# Simple Stuff

- Snobol does not require variables to be declared
- Strings and numbers can be used together

```
cat = 'dog'
bird = '17' + 3.21
```

- *Note: Snowflake does not have numbers*

# Concatenation

- The default Snobol operation is concatenation
- If two variables or constants are adjacent, they are concatenated

```
eagle  = 'big'
canary = eagle 'bird'
```

- canary has the value `'bigbird'`

# Input and Output

- Snobol has two special variables
- **output** – when you assign a value to the variable output, the value is written to the screen
- **input** – when you use the variable input, it reads a line from the keyboard

# Example using input and output

```
OUTPUT = 'What is your name?'
Username = INPUT
OUTPUT = 'Thank you, ' Username
END
```

# Pattern Matching

- Snobol allows complex pattern matching
- A pattern can have concatenation and alternation

```
canary = 'big bird sings'
robin  = 'big dog growls`

canary ('small' | 'big') ' ' ('bird' | 'dog')
robin  ('small' | 'big') ' ' ('bird' | 'dog')
```

- Both strings would match

# Keywords

- Java has a few keywords to control pattern matching
- If the variable `&ANCHOR` has the value 1, then patterns much match from the beginning of the string
- For other values of `&ANCHOR`, Snobol will scan the string to see if the pattern matches anywhere

- Snowflake always searches the full string

# Replacement

- If a pattern succeeds in finding a match, then that portion of the string can be replaced with another value

```
canary = 'big bird sings'
canary ('bird' | 'dog') = 'cat'
```

- canary will be `'big cat sings'`

# Success or Failure

- A Snobol statement can succeed or fail
  - succeed – pattern matches
  - fail – pattern does not match
  - fail – end of file on input

# Flow of Control

- Snobol is an old fashion “GOTO” style language
- All Snobol statements can have a label at the beginning
- The label END is required at the end
- At the end of a Snobol statement there can be a colon then a label specifying the next line to be executed
- The next line can differ depending on the success or failure of the line

# Jumping

- An unconditional jump is

```
dog = 'cat'                    :(rabbit)
```

- To jump only if there is a success

```
dog 'cat'                      :s(rabbit)
```

- To jump only if there is a failure

```
dog 'cat'                      :f(rabbit)
```

- To jump either way

```
dog 'cat'                 :s(rabbit)f(hare)
```

# Which Snobol Statement replaces the first occurrence of mouse in cat with rat?

A. rat = cat mouse

B. cat mouse = rat

C. cat = cat (rat | mouse)

D. cat (rat | mouse)

# Using Patterns

```
        OUTPUT = 'What is your name?'
        Username = INPUT
        Username 'J'                                :S(LOVE)
        Username 'K'                                :S(HATE)
MEH     OUTPUT = 'Hi, ' Username                    :(END)
LOVE    OUTPUT = 'Nice to meet you, ' Username      :(END)
HATE    OUTPUT = "Oh. It's you, " Username
END
```

# Functions

Snobol has several function that can be used in patterns or expressions

- break( 'xyz' ) – match until you encounter any of these characters
- span( 'xyz' ) – match a sequence of any of these characters
- trim( '  xyz  ' ) – remove leading and trailing spaces
- gt( 'a', 'b' ) – succeeds if a is greater than b

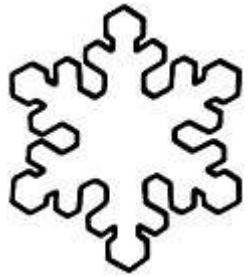
# Saving Pattern Matchings

- The value that matches in a pattern can be saved in a variable by putting a $ and the variable name after the pattern

```
canary = 'big bird sings'
canary ('bird' | 'dog') $ cow = 'cat'
```

- canary will be `'big cat sings'` and cow will have the value `'bird'`

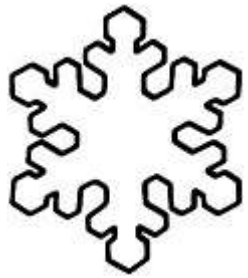
# Snowflake



- This simple Snobol like language creates a Java method that does string manipulation
- Code generation creates a Java method instead of machine language

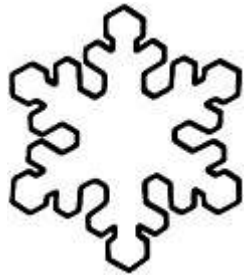
- The idea is to have a language to easily write a Java method to do string manipulation

# Variables



- All Snowflake variables are Strings
- Variables do not have to be declared in Snobol
- Variable names and keywords are case INSENSITIVE
- dog is the same as Dog or DOG or doG

# Snowflake Start and End



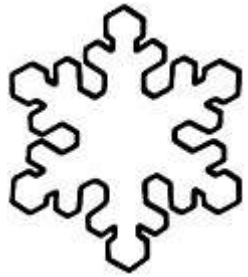
- The first Snowflake statement must be parm to define the variables passed to the method as parameters

```
parm dog    cat;
```

- The last Snowflake statement must be **return** to return a value from the method

```
return variable;
```

# Assignment



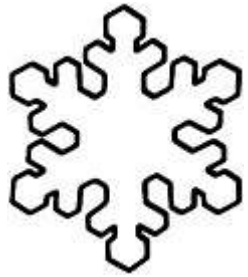
- Simple assignment is like most languages

```
dog = cat; or dog = "quote string";
```

- If two or more strings are on the right side of the equals sign, they are concatenated

```
dog = cat "bird" cow;
```

# Patterns

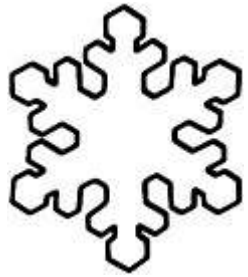


- A pattern is given to the right of a variable on the left of an equals sign

```
dog "Constant pattern" = cow;
```

- If the pattern exists in the variable dog, it is replaced with the contents of cow
- If the pattern does not exist in the variable dog, nothing happens
- Patterns can be variables or string constants

# Multiple Patterns

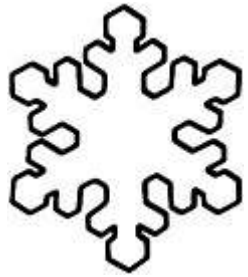


- Patterns can have multiple values to match

```
dog  cat | "mouse" = rat;
```

- If the value of cat appears in dog, then that portion of the dog string is replaced by the value of rat
- If cat does not appear in dog, then dog is searched to see if “mouse” appears and, if so, it will be replaced by the contents of rat

# while Loops



- While loops repeat if a pattern matches

```
while variable pattern {   … }
```

- Patterns work the same way as assignments, but they do not change the value of the variable
- If the pattern exists in the variable, the loop is executed

# Example Snowflake Program

```
parm  dog  cat;
rat = "gerbil" cat;
dog cat | "mouse" = rat;
return dog;
```

# Java Output for Example

```
public static String example(String dog, String cat) {
        String rat;
        int $TEMP1$, $TEMP2$;
        rat = "gerbil" + cat;
        $TEMP1$ = dog.indexOf( cat );
        $TEMP2$ = cat.length();
        if ($TEMP1$ == -1) {
            $TEMP1$ = dog.indexOf( "mouse" );
            $TEMP2$ = "mouse".length();
        }
        if ($TEMP1$ != -1) {
            dog = dog.substring(0, $TEMP1$) + rat +
                                dog.substring($TEMP1$+$TEMP2$);
        }
        return dog;
    }
```

# Write a BNF for

```
parm var1  var2  var3 ;
variable = var1 var2 ;
return var;
```

# Possible Solution

program → parmstmt assign rtn

parmstmt → parm varList **;**

rtn → **return** *variable* **;**

varList → *variable* | *variable* varList

assign → *variable* **=** varList **;**

# Write a Recursive Descent Parser Method

- Write a method to recognize

varList → *variable* | *variable* varList

# Possible Solution

```
boolean varList() {
        int savePos = curTok;                   // save token index
        Token symbol = getNextToken();
        if ( !symbol.getType().equals("variable") ) {
                curTok = savePos;               // restore token index
                return false;
        }
        while (peekIs("variable") ) {
                getNextToken();
        }
        return true;
}
```

# Schedule

- The lexical scanner for the Snowflake language is due by noon on Friday, February 10, 2017
- The first exam is Friday, February 17, 2017