Stack limitations

- You cannot loop over a stack in the usual way.

```
Stack s = new Stack();
…
for (int i = 0; i < s.size(); i++) {
    do something with s.get(i);
}
```

- Instead, you pull elements out of the stack one at a time.
  - common : Pop each element until the stack is empty.

```
// process (and destroy) an entire stack
while (!s.isEmpty()) {
    do something with s.pop();
}
```

Stack implementation: H.W

You have one week to do the following

- Suppose we're asked to write a method **max** that accepts a Stack of integers and returns the largest integer in the stack:

```
// Precondition: !s.isEmpty()

public static void max(Stack  s) {

    int maxValue = s.pop();
    while (!s.isEmpty()) {
        int next = s.pop();
        maxValue = Math.max(maxValue, next);
    }
    return maxValue;
}
```

The algorithm is correct, but what is wrong with the code?

Stack implementation: H.W

You have one week to do the following

- **Implement a Stack function called printList.**

  public void printList(); // to print all elements inside the Stack

Case Study: Infix To Postfix

infix notation

- Conventional notation is called infix notation. The arithmetic operators appears between two operands.

- Parentheses are required to specify the order of the operations. For example: a + (b * c).

Post fix notation

- The operator is placed directly after the two operands it needs to apply. For example: a b c * +

- Eliminates the need for parentheses

- some popular hand-held calculators use postfix notation to avoid the complications of multiple sets of parentheses

Algorithm for Infix to Postfix

4 * (5 + 3) * 1

A * (B + C) * D ---> ABC+*D*

1) Examine the next element in the input.
2) If it is operand, output it.
3) If it is opening parenthesis, push it on stack.
4) If it is an operator, then
   i) If stack is empty, push operator on stack.
   ii) If the top of stack is opening parenthesis, push operator on stack
   iii) If it has higher priority than the top of stack, push operator on stack.
   iv) Else pop the operator from the stack and output it, repeat step 4
5) If it is a closing parenthesis, pop operators from stack and output them until an opening parenthesis is encountered. pop and discard the opening parenthesis.
6) If there is more input go to step 1
7) If there is no more input, pop the remaining operators to output.

Case Study: Infix To Postfix

Example : Convert **A * (B + C) * D** to postfix form.

Move	Current Token	Stack	Output

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

	Move	Current Token	Stack	Output
Before				

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
…….	…….	Empty	Empty
1	2	Empty	2

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
…….	…….	Empty	Empty
1	2	Empty	2
2	*	*	2

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
…….	…….	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
…….	…….	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
…….	…….	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
.....	.....	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
.....	.....	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
.....	.....	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2
8	1	/(-	23*21

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
…….	…….	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2
8	1	/(-	23*21
9	)	/	23*21-

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
.....	.....	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2
8	1	/(-	23*21
9	)	/	23*21-
10	+	+	23*21-/

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
.....	.....	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2
8	1	/(-	23*21
9	)	/	23*21-
10	+	+	23*21-/
11	5	+	23*21-/5

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
.....	.....	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2
8	1	/(-	23*21
9	)	/	23*21-
10	+	+	23*21-/
11	5	+	23*21-/5
12	*	+*	23*21-/5

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
.....	.....	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2
8	1	/(-	23*21
9	)	/	23*21-
10	+	+	23*21-/
11	5	+	23*21-/5
12	*	+*	23*21-/5
13	3	+*	23*21-/53

Case Study: Infix To Postfix

Example : convert 2*3/(2-1)+5*3 into Postfix form.

Before

Move	Current Token	Stack	Output
…….	…….	Empty	Empty
1	2	Empty	2
2	*	*	2
3	3	*	23
4	/	/	23*
5	(	/(	23*
6	2	/(	23*2
7	-	/(-	23*2
8	1	/(-	23*21
9	)	/	23*21-
10	+	+	23*21-/
11	5	+	23*21-/5
12	*	+*	23*21-/5
13	3	+*	23*21-/53
14	…….	Empty	23*21-/53*+

Evaluating a postfix expression

- Use a stack to evaluate an expression in postfix notation.
- The postfix expression to be evaluated is scanned from left to right.

- Initialise an empty stack
- While token remain in the input stream
  - Read next token
  - If token is a number, push it into the stack
  - Else, if token is an operator, pop top two tokens off the stack,apply the operator, and push the answer back into the stack
- Pop the answer off the stack.

Case Study: Infix To Postfix

Example:

Evaluate the expression **2 3 4 + * 5 *** which was created by the previous algorithm for infix to postfix

Move	Current Token	Stack

Case Study: Infix To Postfix

Example: Evaluate the expression **23*21-/53*+** which was created by the previous algorithm for infix to postfix.

Move	Current Token	Stack

Extra Exercises

- Convert **(((A + B) * (C - E)) / (F + G))** to postfix notation.

- Evaluate the following expression in postfix :
  - **623+-382/+*2^3+**
  - **45+72-***
  - **752*+434+*4*-**

Extra Exercises

- **Using the following algorithm. Write a java code to Evaluate a Postfix Expression**

```
opndstk = the empty stack
/* scan the input string reading one element */
/* at a time into symb */
while (not end of input) {

     symb = next input character;

     if (symb is an operand)
          push(opndstk, symb)

     else {
          /* symb is an operator */
           op2   = pop(opndstk);
           op1   = pop(opndstk);
 value = result of applying symb to op1 & op2
           push(opndstk, value);
     } /* end else */

} /* end while */
return (pop(opndstk));
```

Example:

Postfix Expression: 6 2 3 + - 3 8 2 / + * 2 $ 3 +

symb	opnd1	opnd2	value	opndstk
6				6
2				6,2
3				6,2,3
+	2	3	5	6,5
-	6	5	1	1
3	6	5	1	1,3
8	6	5	1	1,3,8
2	6	5	1	1,3,8,2
/	8	2	4	1,3,4
+	3	4	7	1,7
*	1	7	7	7
2	1	7	7	7,2
$	7	2	49	49
3	7	2	49	49,3
+	49	3	52	52

Extra Exercises

- Fill the table below (Efficiency of the Stack Implementations)

	Array Stack	Linked List Stack
push ()	O (1)	O (1)
pop()		
peek()		
Space efficiency		

- Consider the following sequence of stack operations:

push(d), push(h), pop(), push(f), push(s), pop(), pop(), push(m).

Assume the stack is initially empty, what is the sequence of popped values, and what is the final state of the stack? (Identify which end is the top of the stack.)

Extra Exercises

Suppose you have three stacks s1, s2, s3 with starting configuration shown on the left, and finishing condition shown on the right. Give a sequence of push and pop operations that take you from start to finish. For example, to pop the top element of s1 and push it onto s3, you would write `s3.push( s1.pop())`.

Question?

“Success is the sum of small efforts, repeated day in and day out.”
Robert Collier

Reference:

1. Azhar Maqsood Lecture Notes
2. Qamar Rehman Lecture Notes
3. Java Au Naturel by William C. Jones
4. http://condor.depaul.edu/ichu/csc415/notes/notes9/Infix.htm