Stacks

Fundamentals

- A **stack** is an ordered list in which the last element added is the first element retrieved or removed (Last-In, First-Out).
- *Example:* A stack of Korean dishes: Kimchi, Bibimbap, Bulgogi, Japchae

Japchae
Bulgogi
Bibimbap
Kimchi

Kimchi, the first item added, is placed on the bottom of the stack; japchae, the last item added, is on the top of the stack.

- During program execution, the stack can be used to store information about the parameters and return points for all the methods that are currently executing.
- Stacks are used by compilers to store information while evaluating expressions.
- The methods of the **`Stack`** class from the **`java.util`** package are used to implement stacks in Java.
- The list methods are used to implement stacks in Python.
- The two (2) main stack operations are the following:
  - **Push** – adds an item to the top of the stack
    - Method in Java: `push()`
      ```
      Stack stack = new Stack();
      stack.push("Mairo");
      ```
    - Method in Python: `append()`
      ```
      stack = []
      stack.append("Berlin");
      ```
  - **Pop** – removes an item from the top of the stack
    - Method in Java and Python: `pop()`
- Other stack operations:
  - **Peek** – looks at the item at the top of the stack without removing it from the stack
    - Method in Java: `peek()`
    - Syntax in Python: `stack_name[-1]`
  - **Test whether stack is empty**
    - For Java, use the **`isEmpty()`** method.
    - For Python, use the **`if not`** condition, followed by the stack name and a colon.
      Example:
      ```
      stack = []
      if not stack:
          print("Stack is empty.")
      ```

Stack Applications

- Finding palindromes: A **palindrome** is a string that reads the same in either direction. *Ex. level, radar, civic*
- Evaluating a postfix expression: A normal expression is in infix notation. In a postfix expression, the operands (variable or number) precede the operators (symbol).
  Algorithm:
  1. Classify the token (operand or operator).
  2. If token is an operand, push it to the stack.
  3. If token is an operator, perform two (2) pop operations.
  4. Perform the operation on the two (2) popped operands based on the operator. Push its result to the stack.
  5. Repeat until the end of expression.

Example: **6 4 + 9 3 - ***

Token	Operation	Evaluation	Stack
6	Push 6		6
4	Push 4		6, 4
+	Pop 4 then 6 Perform + Push result	6 + 4 = 10	10
9	Push 9		10, 9
3	Push 3		10, 9, 3
-	Pop 3 then 9 Perform - Push result	9 - 3 = 6	10, 6
*	Pop 6 then 10 Perform * Push result	10 * 6	60

6 4 + 9 3 - * = 60

- Converting from infix to postfix
  Algorithm:
  1. Classify the token (operand or operator).
  2. If token is an operand, append it to the postfix expression.
  3. If token is an operator, push it to the stack if stack is empty or has higher precedence than the operator on the top of the stack. If it has lower or equal precedence, pop the operator from the top of the stack, append it to the postfix expression, and push the current operator to the stack.
  4. If a right parenthesis is encountered, pop all the elements until a left parenthesis is encountered. Append all these elements to the postfix expression except the parentheses.
  5. Repeat Steps 1 to 4 until the last token.
  6. If there are no more tokens, pop all the operators and append to the postfix expression.

  Example: (**6 / 3 + 2) * (9 - 3)**

Token	Operation	Stack	Postfix
(	Push (	(	
6	Append 6 to postfix	(	6
/	Push /	(, /	6
3	Append 3 to postfix	(, /	6 3
+	+ < / Pop / and append to postfix Push +	(, +	6 3 /
2	Append 2 to postfix		6 3 / 2
)	Pop all operators until (Append + to postfix		6 3 / 2 +
*	Push *	*	6 3 / 2 +
(	Push (	*, (	6 3 / 2 +
9	Append 9 to postfix	*, (	6 3 / 2 + 9
-	Push -	*, (, -	6 3 / 2 + 9
3	Append 3 to postfix	*, (, -	6 3 / 2 + 9 3
)	Pop all operators until '(' Append - and * to postfix		6 3 / 2 + 9 3 - *

Postfix: **6 3 / 2 + 9 3 - ***

References:

Koffman, E. & Wolfgang, P. (2016). *Data structures: Abstraction and design using Java.* Hoboken: John Wiley & Sons, Inc.

Python Software Foundation (n.d.). *The Python tutorial.* Retrieved from https://docs.python.org/3/tutorial/index.html