CS2123 Data Structures
Stack

Dr. Weining Zhang

Department of Computer Science
University of Texas at San Antonio

August 23, 2009

Tasks of the Week

- Work with postfix expressions.
- Case Study: Develop a PostfixExpression to represent a postfix expression.
- Case Study: Implement the StackADT.

Outline

1. Stack
   - Concepts

2. Application of Stack
   - Postfix Notation

Stack

- A linear structure accessible at one end only
- Provide First In Last Out (FILO) access
public interface StackADT<T> {
    public void push(T element);
    public T pop();
    public T peek();
    public boolean isEmpty();
    public int size();
    public String toString();
}

There are several other possibilities:
- Implement using a Java Stack object
- Directly implement it using a linked list
- Implement it as a subclass of a list (will be discussed in the future)

Postfix Notation

- Arithmetic expressions can be written in several forms

**Example**

Infix form: \((5 + 8) \times 10 - 7\)
Prefix form: \(- * + 5 8 10 7\)
Postfix form: \(5 8 + 10 * 7 -\)

- Postfix is widely used in computer arithmetic because
  - no need for parentheses
  - easily evaluated using a stack
The basic algorithm for evaluating postfix expression is the following.

1. Read postfix expression one token at a time from left to right
2. If token is an operand, push it into the stack
3. If token is an operator, pop the top operand(s), perform the operation, and push the result back into the stack
4. If the end of the expression is reached, pop the stack and return the final result

Design A PostfixEvaluator

- Represent an operand or an operator as an object of a Token class.
  - Each Token object has a name and a value
  - Token objects have an evaluate() method that takes an operand stack as the parameter
- Specific operators and operand in the expression are defined as subclasses of Token
  - Each subclass implements the evaluate() according to its specific operation, using the stack to hold the operand(s) and the result
- The PostfixEvaluator class will read the expression and perform the overall control
- Use a class to create different types of Token from an input String that represents a postfix expression