Lecture 14: Stacks & Queues

CS 62

Fall 2017

Kim Bruce & Alexandra Papoutsaki

Stack

- `Interface Stack<E> {`
  - `void push(E value)`
  - `E pop()`
  - `E peek()`
- Example: Trays in cafeteria
- Last In - First Out (LIFO)
  No changes to middle of list ever!

Stack Applications

- Run-time stack:
  - See sum program
- Backtracking
  - Solving Maze
- Evaluating expression in postfix form:
  - (52 - ((5 + 7) * 4) $\Rightarrow$ 52 5 7 + 4 * - $\Rightarrow$ 4
- Tools to parse programs
- Undo

Evaluation of postfix expressions

1. Create a stack to store operands (or values).
2. Scan the given expression and do following for every scanned element.
   1. If the element is a number, push it into the stack
      `push(operand)`
   2. If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack
      ```
      result1 = pop()
      result2 = pop()
      result = result2 operator result1
      push(result)
      ```
3. When the expression is ended, the number in the stack is the final answer
   `peek()`

Stack Implementations

- ArrayList:
  - Which end should be head?
  - How complex for push, pop, peek?
- SinglyLinkedList:
  - Which end should be head?
  - How complex for push, pop, peek?
- Space differences?
  - What if there are several stacks?
- java.util.Stack based on Vector - don't use!
  - ArrayDeque is better choice (*more details later)*

Queue

- FIFO: Waiting in line
- Operations:
  - enqueue (at end) – or add
  - dequeue (from beginning) – or remove
- Examples:
  - Simulations
  - Event queue
  - Keeping track when searching

Queue Implementations

- SinglyLinkedList:
  - Which end should be front, rear?
  - How complex for enqueue, dequeue?
- ArrayList:
  - Which end should be front, rear?
  - How complex for enqueue, dequeue?
- Space differences?

Dequeue

- Steque:
  - Add and remove from one end. Only add from other.

- java.util.Deque: Double-Ended Queue
  - Can add or remove from either end.
  - Resizable array implementation
  - Faster than Java Stack class when used as stack, faster than LinkedList (doubly-linked) when used as queue.