Stacks

Feb 18
Stacks

• Insertion and deletions are First In Last Out
  • FILO
  • or LIFO
• Physical stacks are everywhere!
• Function names (in the following slides) follow java.util.Stack rather than the book.
null is returned from peek() and pop() when stack is empty

• throw exception?

```
public interface StackInft<E> {
  public boolean empty();
  public E push(E e);
  public E peek();
  public E pop();
  public int size();
}
```
### Example

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Value</th>
<th>Stack Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>push(5)</td>
<td>5</td>
<td>{5}</td>
</tr>
<tr>
<td>push(3)</td>
<td>3</td>
<td>{5, 3}</td>
</tr>
<tr>
<td>size()</td>
<td>2</td>
<td>{5, 3}</td>
</tr>
<tr>
<td>pop()</td>
<td>3</td>
<td>{5}</td>
</tr>
<tr>
<td>empty()</td>
<td>FALSE</td>
<td>{5}</td>
</tr>
<tr>
<td>pop()</td>
<td>5</td>
<td>{}</td>
</tr>
<tr>
<td>empty()</td>
<td>TRUE</td>
<td>{}</td>
</tr>
<tr>
<td>pop()</td>
<td>null</td>
<td>{}</td>
</tr>
<tr>
<td>push(7)</td>
<td>7</td>
<td>{7}</td>
</tr>
<tr>
<td>push(9)</td>
<td>9</td>
<td>{7, 9}</td>
</tr>
<tr>
<td>peek()</td>
<td>9</td>
<td>{7,9}</td>
</tr>
</tbody>
</table>
Array-based Stack

- Implement the stack with an array
- Add elements onto the end of the array
- Use an int `size` to keep track of the top

```
S
0 1 2 ...
```

```
Performance and Limitations

• Performance
  ▫ let $n$ be the number of objects in the stack
  ▫ The space used is $O(n)$
  ▫ Each operation runs in time $O(1)$

• Limitations
  ▫ Max size is limited and can not be changed
  ▫ Pushing onto a full stack can fail
Push

- Array has set size and may become full
- A push will fail if the array becomes full
  - Limitation of the array-based implementation
  - Alternatives?
    - Make the array grow (use ArrayList)?
    - Linked Lists?
  - What do to on fail?
    - return null
    - throw exception
public class ArrayStack<K> implements StackIntf<K> {
    private static final int DEFAULT_CAPACITY = 40;
    private int size;
    private K[] underlyingArray;

    public ArrayStack() {
        this(DEFAULT_CAPACITY);
    }

    public ArrayStack(int capacity) {
        size=0;
        underlyingArray = (K[]) new Object[capacity];
    }
}
Code
empty, peek and pop
public K pop() {
    if (size>0) {
        size--;
        K tmp = underlyingArray[size];
        underlyingArray[size]=null;
        return tmp;
    }
    return null;
}
Method Stack in the JVM

- The JVM keeps track of the chain of active methods with a stack
  - printStackTrace() — only within catch block of exception
  - Thread.dumpStack() — anywhere
- On a method call, the JVM pushes onto the stack a frame containing:
  - parameters
  - local variables
  - return address
- When a method ends, control passes onto the method on top of the stack
- Using VSC to view the stack — MethodStack.java
Stack Applications

• Reversing
• Prefix/postfix algebraic interpreter
• Palindromes
  • Madam Im adam
  • A man a plan a canal panama!
  • Dennis, Nell, Edna, Leon, Nedra, Anita, Rolf, Nora, Alice, Carol, Leo, Jane, Reed, Dena, Dale, Basil, Rae, Penny, Lana, Dave, Denny, Lena, Ida, Bernadette, Ben, Ray, Lila, Nina, Jo, Ira, Mara, Sara, Mario, Jan, Ina, Lily, Arne, Bette, Dan, Reba, Diane, Lynn, Ed, Eva, Dana, Lynne, Pearl, Isabel, Ada, Ned, Dee, Rena, Joel, Lora, Cecil, Aaron, Flora, Tina, Arden, Noel, and Ellen sinned.

• Recursion
• OS Tasks
## Using Stacks

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Value</th>
<th>Stack Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>push(5)</td>
<td>5</td>
<td>{5}</td>
</tr>
<tr>
<td>push(3)</td>
<td>3</td>
<td>{5, 3}</td>
</tr>
<tr>
<td>size()</td>
<td>2</td>
<td>{5, 3}</td>
</tr>
<tr>
<td>pop()</td>
<td>3</td>
<td>{5}</td>
</tr>
<tr>
<td>empty()</td>
<td>FALSE</td>
<td>{5}</td>
</tr>
<tr>
<td>pop()</td>
<td>5</td>
<td>{}</td>
</tr>
<tr>
<td>empty()</td>
<td>TRUE</td>
<td>{}</td>
</tr>
<tr>
<td>pop()</td>
<td>null</td>
<td>{}</td>
</tr>
<tr>
<td>push(7)</td>
<td>7</td>
<td>{7}</td>
</tr>
<tr>
<td>push(9)</td>
<td>9</td>
<td>{7, 9}</td>
</tr>
<tr>
<td>peek()</td>
<td>9</td>
<td>{7, 9}</td>
</tr>
</tbody>
</table>