Notes

Next programming assignment has been posted. Due FRIDAY, Jan 30. It's due on Friday instead of Sunday.
GO TO THE LAB – The TAs are there to help.
If you really think you need more time, then
  • Come to see me Thursday. You have to be able to show me that you have made progress on the assignment by then, and have a plan to complete the program.

My sense is that a lot of students put off working on the program until the last minute – you won't be able to do that in this class.

Hint for success on next program: START EARLY!

Partners – if you need or want a new partner, please come see me after class.

Reading Assignment

Read the programming assignment!

Ch 6: The sections "The Abstract Data Type Stack", "Simple Applications of the ADT Stack", and "Implementations of the ADT Stack". – pages 250 – 268 (148-266 old version)

Homework Assignment

Chapter 4 – self-test exercises 2 and 3.

Linked Lists

Last time we began looking at Linked Lists. Each element in the List is a Node (see Node.java). Each Node has two fields: Object item and Node next;

class Node {
  private Object item;
  private Node next;

  // constructors, accessors, and mutators
}
Displaying the elements in a Linked List

```java
for (Node curr = head; curr != null; curr = curr.getNext() ) {
    System.out.println( curr.getItem() );
}
```

Inserting an item into a linked list

Draw a Linked List.

Discuss Node insertion, with two references: prev and curr.

```java
Node newNode = new Node( item );
newNode.setNext( curr );
prev.setNext( newNode );
```

Special case: insertion at beginning of list

```java
newNode.setNext( head );
head = newNode;
```

Does it matter if list is empty?

What if want to 'insert' item at end of list? What is value of curr? (null). Does above still work?

Removing an item from a linked list

```java
prev.setNext( curr.getNext() );
```
What if we want to remove the first element of the list?

head = head.getNext();

Does this work if the list has only one Node?
Does this work if we want to remove the last element?

What do we need to do to get rid of the Node once we have removed it from the list?

curr.setNext( null );
curr = null;

Clears references, so the memory allocated to the Node can be reused.

**How do we get the references prev and curr?**

Traverse the list until you find the proper spot to insert your new item.

For example, say you have a sorted list, and you want to insert a new item. Then you need to traverse the list to find the correct position.

prev = null;
curr = head;

while ( ((Integer) curr.getItem()).compareTo( newValue ) < 0 ) {
   prev = curr;
   curr = curr.getNext();
}

Problem: What if newValue greater than all values in list? What if list is empty? curr gets assigned value of null, and then curr.getItem() throws NullPointerException

Need to check if curr is null before calling getItem()

prev = null;
curr = head;

while ( curr != null && ((Integer) curr.getItem()).compareTo( newValue ) < 0 ) {
   prev = curr;
   curr = curr.getNext();
}

Java doesn't evaluate all of the operands of && if the first one is false – it stops without evaluating second expression.
Does this find the correct place? What if new value is smaller than all values in the list? What if list is empty?

**Look at Linked List implementation of ADT List.**

See ListReferenceBased.java

**Doubly Linked Lists**

class Node {
    Object item;
    Node next;
    Node precede;
}

class Node {
    Object item;
    Node next;
    Node precede;
}

class Node {
    Object item;
    Node next;
    Node precede;
}  

public Node getPrecede() { 
    return precede;
}

**Delete a node**

curr.getPrecede().setNext( curr.getNext() );
curr.getNext().setPrecede( curr.getPrecede() );

**Insert a node**

newNode.setNext( curr );
newNode.setPrecede( curr.getPrecede() );
curr.setPrecede( newNode );
newNode.getPrecede().setNext( newNode );

Doesn't handle special cases at beginning and end of list. See book about using a dummy head node.

STACKS (Chapter 6)

Envision a stack of plates at a cafeteria, where the plates are set on a spring-loaded platform in the counter. What can you tell about the stack? What operations can you perform?

Property of stack – FIFO (first in, first out). Not great in real-life, but very useful with computers.

ADT stack operations
- create an empty stack
- check to see if stack is empty
- add an item to the stack
• remove the most recently added item from the stack
• look at the most recently added item
• remove all items from the stack

createStack() // creates an empty stack

isEmpty() // Determines whether a stack is empty

push( newItem ) throws StackException
// Adds newItem to the top of the stack. Throws StackException if the insertion is not successful

pop() throws StackException
// Retrieves and then removes the item from the top of the stack (the most recently added item). Throws StackException if the deletion is not successful

popAll() // Removes all items from the stack

peek() throws StackException
// Retrieves the top of the stack. Retrieval does not change the stack. Throws StackException if the retrieval is not successful

**Checking for balanced braces**

abc { def {ijk} {} } qr

Braces are balanced if
1. When you encounter a }, it matches an already encountered {
2. When you reach the end of the string, you have matched all the {

stack.createStack()
while ( not at end of string ) {
    if (the next char is a '{' ) {
        stack.push( '{' )
    } else if (the character is a '}' )
        openBrace = stack.pop()
}

This solution keeps track of braces, but does not check for 1 and 2 above.

stack.createStack()
balancedSoFar = true
i = 0
while ( balancedSoFar and i < stringLength ) {
    ch = character at position i in string
    i++
    if (ch is '{' ) {
        stack.push('{')
    }
    else if (ch is '}' ) {
        if (!stack.isEmpty() ) {
            openBrace = stack.pop()
        }
        else {
            balancedSoFar = false    // found '}' without '{'
        }
    }
}

if ( balancedSoFar and stack.isEmpty() )
    // string has balanced braces
else
    // string braces are not balanced