Linked Data Structures

Linked lists; Stacks; preview of Assignment 8

But first, a bit more on Assn 7

Some problems that several people seem to have ...

Bubbling values up from recursion

I’ve seen ...

    boolean isPalindrome( ... ) {
        if ( ... ) {
            isPalindrome( ... );
        }
        return false;
    }

What’s missing?

Search ... trying all paths

The code

    ok = ok || something( ... );
    ok = ok || something( ... );
    ...
    return ok;

has been puzzling people. What’s it about?
Short circuit evaluation

```java
if ( 1 < 5 || 17 > 27 ) { ...
    Never even tries 17 > 27 ... because
    true || x is true

    ok = ok || something( ... )
    executes something( ... ) if ok is false
```

Searching, searching ...

```java
boolean ok = false;
ok = ok || aAa( ... );
ok = ok || bBb( ... );
ok = ok || cCc( ... );
ok = ok || dDd( ... );
...
return ok;
Suppose only cCc( ... ) returns true
```

The search tree

![Search Tree Diagram]

Buffering

`hasNext()` can be called multiple times
without consuming a result

Try testing with `Spelling.java` from the
assignment page (added today)
Spelling.java

```java
static void test1() {
    String badWord = "fooze";
    Scanner good = new Scanner("fooze\n");
    Suggester suggest = new Suggester(badWord, good, 1);
    expect(suggest.hasNext(), "Exact match: Available");
    expect(suggest.hasNext(),
            "Buffering: ok to call hasNext() twice");
    String s = suggest.next();
    expect( badWord.equals( s ),
            "Exact match, fooze==fooze");
    expect(! suggest.hasNext(),
            "Should have no more suggestions.");
}
```

Linked Data Structures

References

We’ve seen references from the stack to the heap, and from the heap to the heap ...

References can “link” objects

```java
class Cell {
    String name;
    Cell next;
    public Cell( String name ) {
        this.name = name;
        this.next = null;
    }
    public void linkTo(Cell neighbor) {
        this.next = neighbor;
    }
    ...
```
We can build “linked lists” of objects

```
public void printAll() {
    System.out.print( name + "-\->");
    if (next != null) {
        next.printAll();
    } else {
        System.out.println("[null]");
    }
}
```

Traversing a List (with recursion)

Adding to the front ...

```
Cell c = new Cell("caboose");
for (int i=0; i < 3; ++i) {
    Cell head = new Cell( Integer.toString(i) );
    head.linkTo(c);
    c = head;
}
c.printAll();
What does it build? What does it print?
```

Cell c = new Cell("caboose");
for (int i=0; i < 3; ++i) {
    Cell head = new Cell( Integer.toString(i) );
    head.linkTo(c);
    c = head;
}

c.printAll();

for (int i=0; i < 3; ++i) {
    Cell head = new Cell( Integer.toString(i) );
    head.linkTo(c);
    c = head;
}

public void printAll() {
    System.out.print( name + "->");
    if (next != null) {
        next.printAll();
    } else {
        System.out.println("[null]");
    }
}

this
public void printAll() {
    System.out.print( name + "->");
    if (next != null) {
        next.printAll();
    } else {
        System.out.println("[null]");
    }
}

this

Cell 2
     next

Cell 1
     next

Cell 0
     next

Cell caboose
     next

public void printAll() {
    System.out.print( name + "->");
    if (next != null) {
        next.printAll();
    } else {
        System.out.println("[null]");
    }
}

this

Cell 2
     next

Cell 1
     next

Cell 0
     next

Cell caboose
     next

this

Cell 2
     next

Cell 1
     next

Cell 0
     next

Cell caboose
     next