CSC 212: Programming with Data Structures

Midterm: Spring 2016

Thursday, March 10, 1-2:50pm

- This is an in-class exam on the material through Lab 6.
- It is closed notes, closed Internet, and closed technology, but you may use a "cheat sheet".
- Your cheat sheet must be hand-written, created by you, 8.5 × 11 inches, and double-sided (at most).
- I will come in a few times to answer questions.
- If there is a clarification to be made, I will write it on the board.
- Do not discuss the exam with other students and respect the honor code of doing your own work.
- If you are unable to make progress on any part of the exam, tell me what you tried; describe your thought process.
- Even if you are not finished, your exam must be turned in at the end of lab to maintain fairness.
- For ease of writing, I'll encourage using *uppercase, single-letter* variable names, but just for the exam!

Name	Solution Sketches

Part 1	/15
Part 2	/15
Part 3	/20
Part 4	/20
Part 5	/15
Part 6	/15
Total	/100

Part 1: Warmup and Vocab

1 (a) What does FIFO stand for? What classical data structure has this property?

First In First Out, a queue

1 (b) What does FILO stand for? What classical data structure has this property?

First I Last Out, a stack

(c) For each line of the code below, use the following phrases to describe what is happening: "declare", "initialize", "assign", "memory allocated", "get", "set"

5

```
int[] X;
```
declare X

```
X = new int[3];
```
initialize X, memory allocated (3 ints worth)

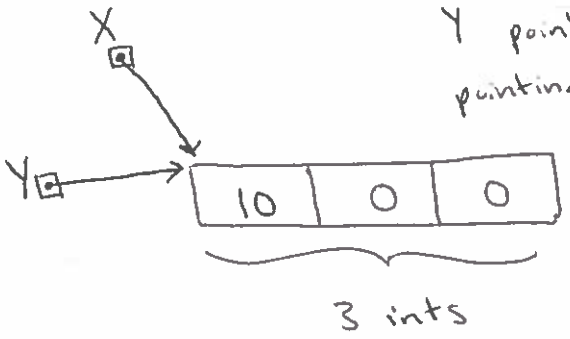
```
int[] Y;
```
declare Y

```
Y = X;
```
initialize Y, assign it the value of X

```
Y[0] = 10;
```
"set" the first element of Y to 10 (assignment)
(this also changes X)

```
int first = X[0];
```
"get" the first element of X

3 Draw a diagram of the end result, showing X and Y clearly. What is first?



- assigning X to Y makes Y point to what X is pointing to. So first is equal to 10.

(d) Student X is writing a `ListTester` class, but they are confused about how to use the following `LinkedList` class to add an element to the *end* of a list. Student X writes:

```
public class ListTester {

    public static void main(String[] args) {
        List myList = new List();
        Node firstClass = new Node("212");
        Node secondClass = new Node("103");
        myList.tail = firstClass;
        myList.tail = secondClass;
    }
}
```

```
public class LinkedList {
    public Node head;
    public Node tail;
    ...
    public
    private void add(String element) {
        ...
    }
    ...
}
```

What is wrong with this code? What changes could be made to `LinkedList` (and/or `Node`) to prevent Student X from doing this?

"tail" is being overridden, not updated properly. This is partly because "head" & "tail" are public and "add" is private. This is an example of the bad things that can happen when fields are public!

(e) Justify your answers to the following questions.

i. Is `JComponent` an interface or a class?

Node could also be a private subclass of LinkedList

a class, because we can "extend" it

ii. What is an example of a class we have studied that is built upon `JComponent`?

JQueue (it extends JComponent)

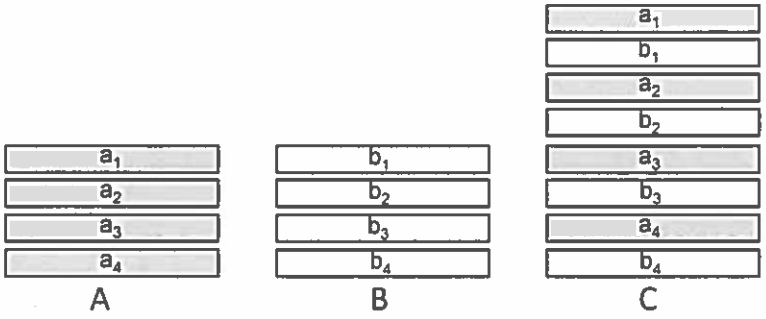
others: JCircle, MapViewer, JKeyProcess.

iii. In the model-view-controller paradigm, what piece does your example fall under?

the "view", it is visualizing the data, which in the case of JQueue is a queue of characters.

Part 2: Stacks

10 Write a method that will take in two stacks (A and B) of the same size (n) and interleave them, creating one stack (C). Whatever was at the top of stack A should now be at the top of stack C, as shown in the picture below for $n = 4$:



It is okay to destroy the original stacks in the process. The only Stack methods you may use are peek, push, pop, and isEmpty.

```
public static Stack interleave(Stack A, Stack B) {
    Stack C = new Stack();
    Stack D = new Stack();
    while (! A.isEmpty()) {
        D.push ( A.pop());
        D.push ( B.pop());
    }
    while (!D.isEmpty()){
        C.push(D.pop());
    }
    return C;
}
```

temp Stack so order of C is correct. ✗

2 • Based on the description and partial signature of the method, what Java keyword(s) should also be included in the method signature? Write on the dotted line.

(static) since it doesn't make sense for each stack to have an instance of this method.

3 • What is the runtime of your method, in terms of the size of each stack, n? Justify your answer.

O(n), each while loop takes linear time, & they are separate (not nested).

Part 3: Arrays and Iterators

Complete the code below to write an iterator for an array of Strings. In particular, think about whether or not you want to add any fields or methods.

```
import java.util.Iterator;

public class ArrayIterator implements Iterator<String> {

    private String[] array;
    private int curr;    // keep track of our "page"

    public ArrayIterator(String[] array) {
        this.array = array;
        this.curr = 0;
    }

    public boolean hasNext() {
        if (this.curr < this.array.length) {
            return true;
        }
        else
            return false;
    }

    public String next() {
        String data = this.array[this.curr];
        this.curr ++;
        return data;
    }
}
```

- Is it necessary to have an iterator for an array? Are there any advantages? Be specific.

  Iterators improve the speed of indexing within loop constructs. Arrays don't need faster 'indexing'. But, having an iterator allows us to have for each loops.

- What Java construct is `java.util.Iterator`? What do `hasNext()` and `next()` look like inside `java.util.Iterator`?

  an interface.

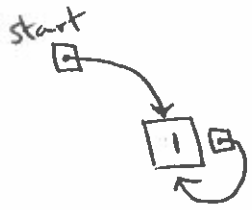
  hasNext() & next() do not contain any code in Iterator

  ↳ only have signatures

Part 4: Loops and Linked Lists

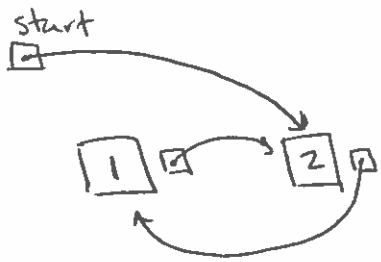
6 In this question we'll be implementing a new *generic* data structure called a `Loop<E>`, using a **singly linked list of nodes** as the underlying data structure. Each node should have a pointer to the next node so that they form a continuous loop (like beads on a necklace). However, your loop data structure should have one field, a pointer to the "start" Node.

(a) Draw a diagram showing a single element being added to an empty loop. Then write code that corresponds to this operation (does not have to be encapsulated as a method).




```
Node new1 = new Node(data1);   // of type E
new1.setNext(new1);
this.start = new1;
```

6 (b) Draw a new diagram showing another element being added to the loop at the "start" (so now there are exactly two elements, starting with the newest one). Then write code to execute this operation (does not have to be a method).



end picture

```
Node new2 = new Node(data2);   // type E
this.start.setNext(new2);
new2.setNext(this.start);
this.start = new2;
```

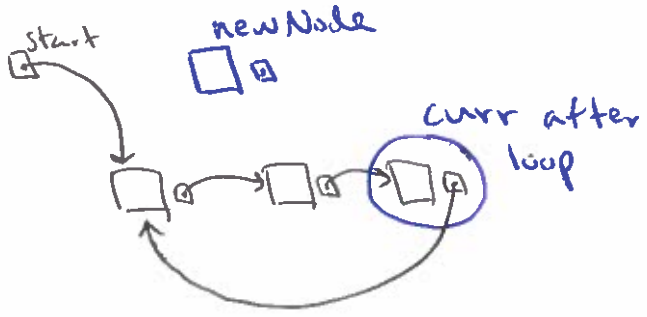
8 (c) Finally, draw a new diagram that shows an element being added to the start of the loop, but with an arbitrary number of elements already in the loop. Then write code for a **method** that will accomplish this general task. Explain the issues involved, including efficiency (although it's not a requirement that your method be efficient). You can choose to augment the fields of your `Node<E>` data structure or your `Loop<E>` data structure, or leave them as they are, as long as your method accomplishes the goal.

```
public void addToStart(E data) {   // assumption: loop not empty
    Node newNode = new Node(data)
    Node curr = this.start;

    while (curr.getNext() != this.start) {
        curr = curr.getNext();
    }
    // after while loop, curr should be the "last" node.

① newNode.setNext(this.start);
② curr.setNext(newNode);
③ this.start = newNode;
}
```

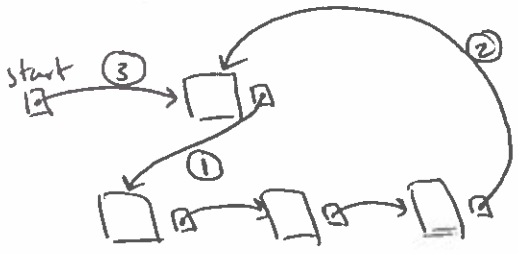
check if we're back at the beginning.

this is O(n) and inefficient because we're walking all the way through the loop.

Start picture



end picture



→ Note: you could have also added a prev field to Node

Part 5: Sorting

10 Inspired by 212, after graduation you decide to work for *Sorters, Inc.* that provides sorting algorithms for every possible situation. One client explains that often their arrays are already mostly sorted (smallest to largest), so the version of insertion sort that you've provided (the one we covered in class) is doing too much extra work. Describe (using code or pseudocode), a modified insertion sort that will be more efficient with somewhat sorted arrays. An example is below:

```
int[] A = {3, 5, 1, 8, 17, 15, 29}
```

Requirements:

- Your algorithm should not modify the original array
- You should return a new sorted `LinkedList`

```
public LinkedList myInsertSort(int[] A){
    LinkedList sorted = new LinkedList();   n = A.length;
    for (int i = n-2; i >= 0; i--){   // start from the back
        j=0
        while ( j < sorted.size() and sorted.get(j).compareTo(array[i
                                                          <0) {
            j++;
        }
        sorted.add(j, array[i]);
    }
    return sorted;
}
```

only difference from class

5 If you ran your algorithm on the example list above, *exactly* how many comparisons are performed?

comparison	list after comparison
15 - 29	15, 29
17 - 15	15, 17, 29
17 - 29	15, 17, 29
8 - 15	8, 15, 17, 29
1 - 8	1, 8, 15, 17, 29
5 - 1	1, 5, 8, 15, 17, 29
5 - 8	1, 5, 8, 15, 17, 29
3 - 1	1, 3, 5, 8, 15, 17, 29
3 - 5	

9 comparisons

Part 6: Runtime analysis

9 (a) What is the runtime of the shunting yard algorithm, in terms of the number of tokens n? Justify your answer.

→ $O(n)$, each token is pushed / popped on / off the stack at most once, then is added to the linked list at most once. (Note this is not obvious when we think about the loops (including a while loop for precedence)).

6 (b) What is the runtime of the code below, in terms of n?

```
for (int i=0; i < n; i++) {
    for (int j=0; j < (int)(n/Math.pow(2,i)); j++) {
        array[i][j] = 1;
    }
}
```

Justify your answer. For fun: prove your answer.

$i=0$, j max $= \frac{n}{2^0} = n$

$i=1$, j max $= \frac{n}{2^1} = \frac{n}{2}$

$i=2$, j max $= \frac{n}{2^2} = \frac{n}{4}$

$i=3$, j max $= \frac{n}{2^3} = \frac{n}{8}$

$i=n$, ... 1

add together

$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots 1$

$= n(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots *)$

Let $S = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots$

Claim: $S \leq 2$

Proof:

$2S = 2 + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots$

$(-) \quad S = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots$

$S = 2$

$\Rightarrow$ Runtime is $2n$ $\Rightarrow$ $O(n)$