EECS2011 Fundamentals of Data Structures (Winter 2022)

Q&A - Lectures 7

Wednesday, March 9

Announcements

will be covered in WT2

more challenging than WT1

- Lecture W8 released
  - General Trees vs. Binary Trees
  - Implementing a Generic, General Tree in Java

    Part B1-B4.
  - Terminology and Mathematical Properties
- Assignment 2 released

  task 2: O(n)
- Assignment 1 results released

  1. office hours
  2. submit regrading request
- Written Test 2 coming soon

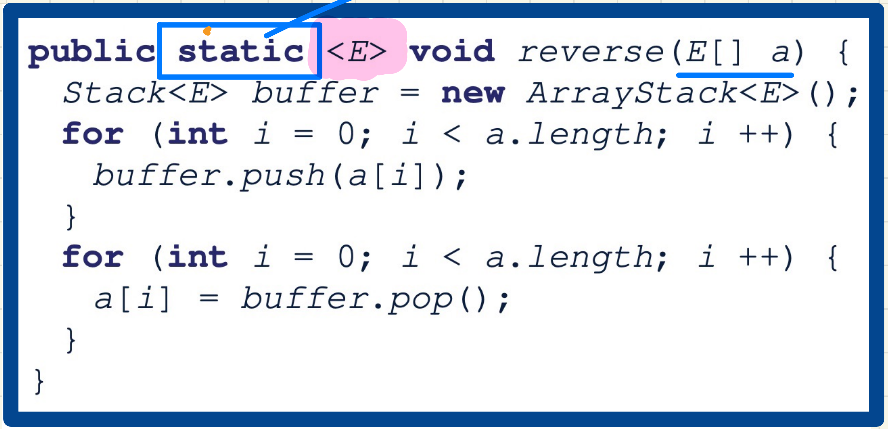
If we made a github repository of a past course, like eecs1022, is it ok if we share some projects that we did with employers or is it a breach for academic integrity.

EECS2011

↳ private before official end of the course (end of april / early may)

On week 6 slides (ADTs-stack....),

why do we declare our methods as static on pages 28 and 30 ?



```
public static <E> void reverse(E[] a) {
  Stack<E> buffer = new ArrayStack<E>();
  for (int i = 0; i < a.length; i ++) {
    buffer.push(a[i]);
  }
  for (int i = 0; i < a.length; i ++) {
    a[i] = buffer.pop();
  }
}
```

1. method does not need to use any attributes in StackUtilities class

(this method only operates on the parameter).

2. StackUtilities. reverse(...);

StackUtilities su = new ...;

su. reverse(...);

↳ warning!

× ArrayList

ListOrdereps ×

what names can we use as generic parameters.

public MyClass < ___ , ___ , ... >

any name that does not clash with:

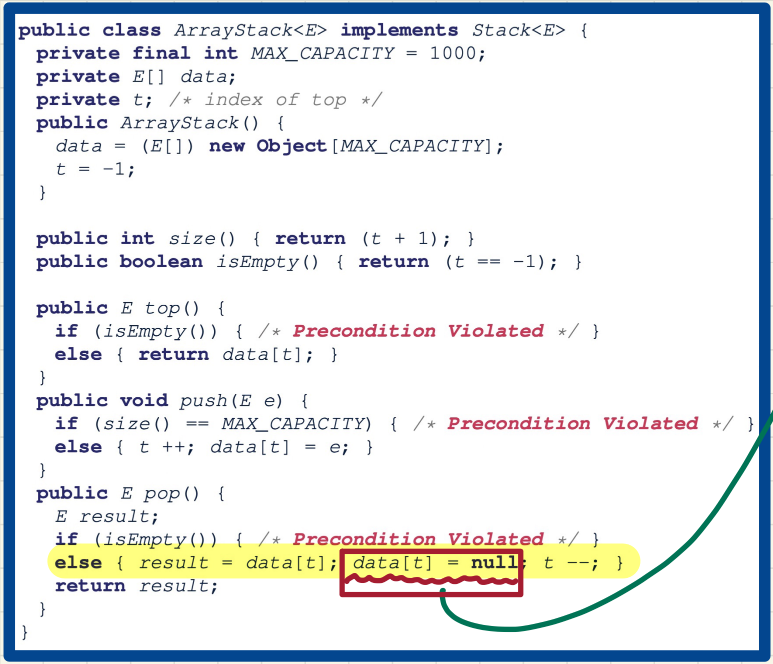
(1) library class names

(2) classes created under the same project

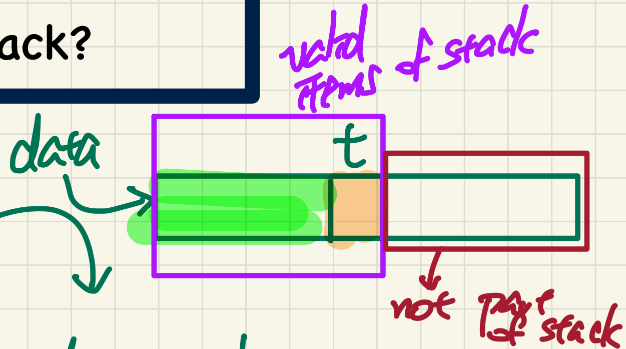
E , I , K , V (single character in upper case)

G , H , I , J , . . .

Why do we need to assign null to the returned array element when we pop from an array stack?
Can't we just decrement the head of the stack?



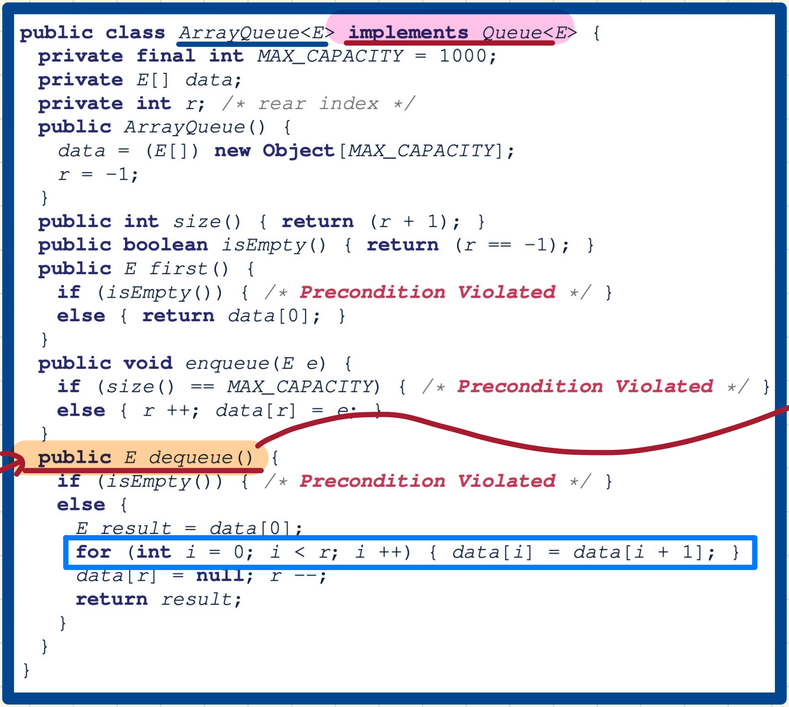
```
public class ArrayStack<E> implements Stack<E> {
  private final int MAX_CAPACITY = 1000;
  private E[] data;
  private t; /* index of top */
  public ArrayStack() {
    data = (E[]) new Object[MAX_CAPACITY];
    t = -1;
  }

  public int size() { return (t + 1); }
  public boolean isEmpty() { return (t == -1); }

  public E top() {
    if (isEmpty()) { /* Precondition Violated */ }
    else { return data[t]; }
  }
  public void push(E e) {
    if (size() == MAX_CAPACITY) { /* Precondition Violated */ }
    else { t ++; data[t] = e; }
  }
  public E pop() {
    E result;
    if (isEmpty()) { /* Precondition Violated */ }
    else { result = data[t]; data[t] = null; t --; }
    return result;
  }
}
```



1. t indicates where the top is

2. items stored at indices > t are not valid stack items
↳ they should be detached ASAP whenever possible

my question is about the dequeue method in circular array. The dequeue method in in circular array has to be void rather than returning a new array like arrayqueue?



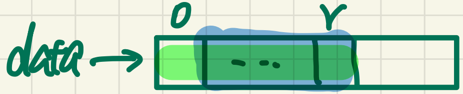
```
public class ArrayQueue<E> implements Queue<E> {
  private final int MAX_CAPACITY = 1000;
  private E[] data;
  private int r; /* rear index */
  public ArrayQueue() {
    data = (E[]) new Object[MAX_CAPACITY];
    r = -1;
  }
  public int size() { return (r + 1); }
  public boolean isEmpty() { return (r == -1); }
  public E first() {
    if (isEmpty()) { /* Precondition Violated */ }
    else { return data[0]; }
  }
  public void enqueue(E e) {
    if (size() == MAX_CAPACITY) { /* Precondition Violated */ }
    else { r ++; data[r] = e; }
  }
  public E dequeue() {
    if (isEmpty()) { /* Precondition Violated */ }
    else {
      E result = data[0];
      for (int i = 0; i < r; i ++) { data[i] = data[i + 1]; }
      data[r] = null; r --;
      return result;
    }
  }
}
```

Q1. Why do we need C.A.?

Q2. Why do we need D.A?

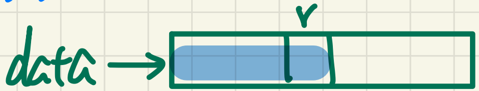
this header was declared in the Queue interface

pre-state

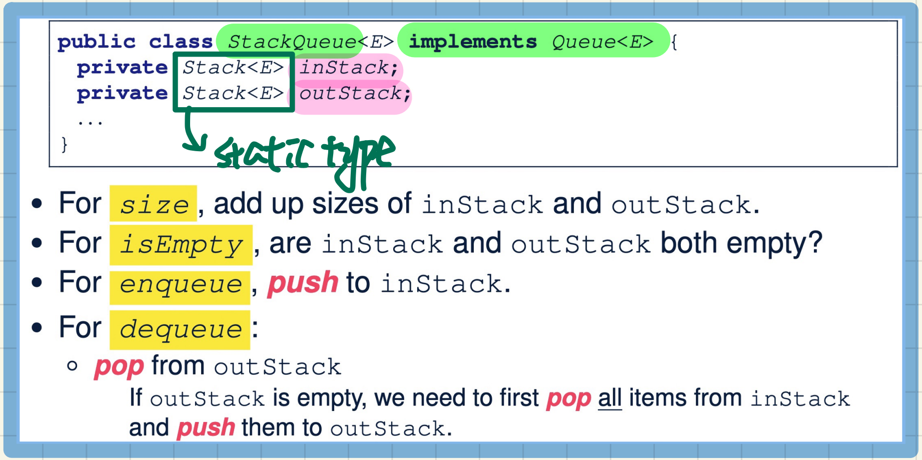


q. deque();

post-state



Hello professor. Can you please elaborate a bit on how to implement a queue using two stacks? Thanks.



```
public class StackQueue<E> implements Queue<E> {
  private Stack<E> inStack;
  private Stack<E> outStack;
  ...
}
```

static type

- For *size*, add up sizes of inStack and outStack.
- For *isEmpty*, are inStack and outStack both empty?
- For *enqueue*, ***push*** to inStack.
- For *dequeue*:
  - ***pop*** from outStack

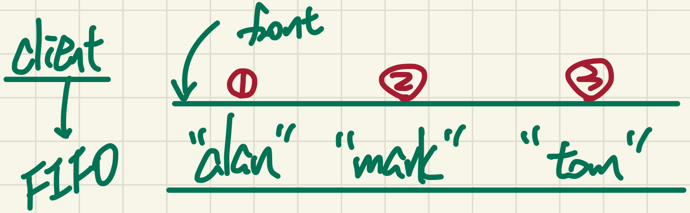
    If outStack is empty, we need to first ***pop*** all items from inStack and ***push*** them to outStack.

✓ push to inStack

```
Queue<String> q = new StackQueue<>();
q.enqueue("alan");
q.enqueue("mark");
q.enqueue("tom");
String first = q.dequeue();
String second = q.dequeue();
String queue = q.dequeue();
```

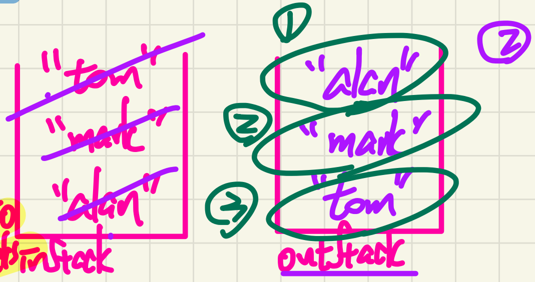
① first pop all items in stack then push to outStack

② pop off outStack



Supporter

Simulate FIFO using two LIFO stacks



One pattern:

$\underbrace{\text{enqueue} \cdots\cdots \text{enqueue}}_{n \text{ operation}}\ \underbrace{\overset{O(n)}{\boxed{\text{dequeue}}}\ \overset{O((n-1)\cdot 1)=O(n)}{\boxed{-\,-\,-\ \text{dequeue}}}}_{n \text{ operations}}$

Amortized RT: $O\left(\frac{n+(n-1)}{n}\right) = O(1)$

Amortized RT of Stack Queue	Amortized (doubling) RT of dynamic arrays (for queue)
dequeue worst case: $O(n)$ ∵ pushing to outstack items popped from instack However, such transfer of items from in- to out-stack happens rarely. ⇒ ART: $O(1)$	enqueue worst case: $O(n)$ ∵ copying items to the resized array Hower, such resizing of array is not needed often ⇒ A.RT: $O(1)$

It seems that we can implement this without recursion as well. We can simply keep updating the “from” and “to” in a loop. Why do we not do that?

```
boolean binarySearch(int[] sorted, int key) {
  return binarySearchH(sorted, 0, sorted.length - 1, key);
}
boolean binarySearchH(int[] sorted, int from, int to, int key) {
  if (from > to) { /* base case 1: empty range */
    return false; }
  else if(from == to) { /* base case 2: range of one element */
    return sorted[from] == key; }
  else {
    int middle = (from + to) / 2;
    int middleValue = sorted[middle];
    if(key < middleValue) {
      return binarySearchH(sorted, from, middle - 1, key);
    }
    else if (key > middleValue) {
      return binarySearchH(sorted, middle + 1, to, key);
    }
    else  { return true; }
  }
}
```

Exercise: implement binary search using a loop

binary search tree