ANNOUNCEMENTS
welcome back i hope you had a nice mountain day :)

WARMUP
what is this robot for? →

TODAY
stacks and queues

record LEC-02

where are we?

recap of first half

we built a foundation

- Java!
  - **types**, **operators**, & **compiler errors**
    - when should we compile?
      - early and often!
- avoid repetition!
  - **functions**!
    - for when we have the same chunk of logic repeated in multiple places
  - **classes**!
    - for when we repeatedly pass around the same chunk of data

we started thinking about code more deeply

- the **array**!
  - how *fast* is an array?
  - what are pros and cons of the array?
  - how might we address the cons of an array?
    - the **array list**!
      - how *fast* is an array list? best case? worst case? ...
      - what are pros and cons of the array list?
      - how might we address the cons of an array list?
        - ...

preview of second half

we will look at how to structure data

- we will meet a whole menagerie of data structures
  - stacks
  - queues
  - hash tables
  - linked lists
  - trees
  - graphs
  - heaps
- we will analyze speed and space

we will learn to choose the right data structure for the job

- fun homeworks (at least in my opinion) 🙂 👍
  - Flip Book
    - array list of array lists of array lists of Vector2's
  - PostScript Interpreter
    - stack and hash table
  - Text Generator
    - hash table of hash tables
  - ...

data structures

what is a data structure?
why is a data structure?

data structures

- In computer science, a **data structure** is a data organization, management, and storage format that is usually chosen for efficient access to data.[1][2][3] More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data,[4] i.e., it is an algebraic structure about data. --Wikipedia

data structures

- In computer science, a **data structure** is the organization and implementation of values and information. In simple words, it is the way of organizing information in a computer so that it can be more easily understood and worked with. --Simple Wikipedia

data structures

- A **data structure** is...a system for organizing and using information... It...make[s] information easier to understand and work with..
  --Simple Wikipedia, further simplified by ChatGPT

data structures

- **data** means numbers (and letters)
- a **data structure** organizes your data
  - for a particular task...
    - ...a *good* data structure is...
      - easy to work with (programmer time)
      - runs fast (runtime / user's time)

were array lists a good choice for implementing a Flip Book?

💭 (how) could you have done it with just arrays?

we didn't know how many...

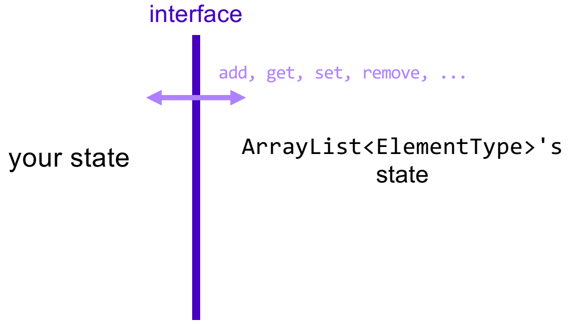
- frames in each animation
- strokes in each frame
- points (Vector2's) in each stroke

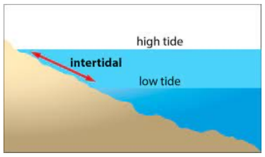
"Alternative" 1: just use arrays, but resize them yourself when needed (essentially, implement the functionality of an array list without actually having a class)

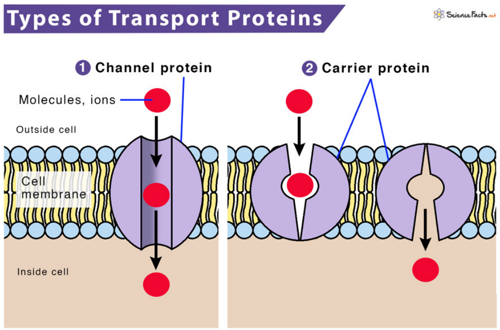
"Alternative" 2:

```
// ~30,000,000 elements
Vector2[][][] animation = new Vector2[64][512][1024];
// need all these counter variables
// can't just call .size() like before!
int numFrames = 0;
int[] numStrokes = ...;
int[][] numPoints = ...;
```

interface







interface

- a data structure's **interface** (**API**) or **abstract data type** is a set of functions a data structure must have
  - a **list** has...
    - **get**
    - **set**
    - **add**
    - **remove**
    - etc.
- a **data structure** is a specific **implementation** (code that does the thing) of that interface
  - an **array list** implements the list interface using an array
  - a **linked list** implements a list using nodes that refer to nodes

you can get (very) formal about this

- The abstract list type L with elements of some type E (a monomorphic list) is defined by the following functions:
  - nil: () → L
  - cons: $E \times L \rightarrow L$
  - first: $L \rightarrow E$
  - rest: $L \rightarrow L$
- with the axioms
  - first (cons (e, l)) = e
  - rest (cons (e, l)) = l
- for any element e and any list l. It is implicit that
  - cons (e, l) ≠ l
  - cons (e, l) ≠ e
  - cons (e_1, l_1) = cons (e_2, l_2) if $e_1 = e_2$ and $l_1 = l_2$
- Note that first (nil ()) and rest (nil ()) are not defined.
- These axioms are equivalent to those of the abstract stack data type.

i typically won't.

just know there is a difference between
interface ("list") and
implementation (ArrayList<Element>)

Tungsten break

did you know the density of Tungsten is 19.25 g/cm^3?
Gold is 19.32 g/cm^3

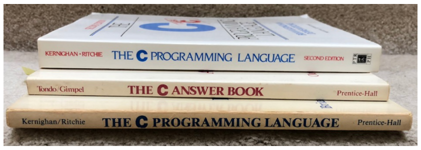
did you know Tungsten is really hard?
Tungsten alloys can be as hard as Sapphire! that's really hard!

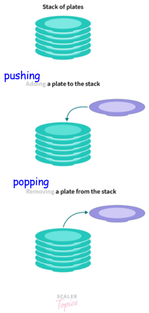
Tungsten is expensive (that cube is $60) but not like...that expensive

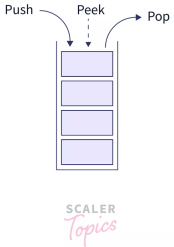
what could YOU do with Tungsten?

stacks

stack


KERNIGHAN • RITCHIE THE C PROGRAMMING LANGUAGE SECOND EDITION
Tondo/Gimpel THE C ANSWER BOOK Prentice-Hall
Kernighan/Ritchie THE C PROGRAMMING LANGUAGE Prentice-Hall





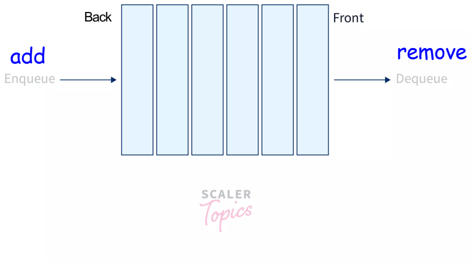
stack interface

```
– // Push (add) a new element to the top of the stack.
  void push(ElementType element);

– // Pop (remove) the top element of the stack.
  // and returns it.
  ElementType pop();

– // Peek (look) at the top element of the stack
  // (without removing it) and return it.
  ElementType peek();

– // Returns the number of elements currently in the stack.
  int size();
```

queues

queue

Back
Front
Last to enter
Last to leave
First to enter
First to leave
SCALER Topics



queue interface

```
– // Add (enqueue) a new element to the back of the queue.
  void add(ElementType element);

– // Remove (dequeue) the front element of the queue
  // and return it.
  ElementType remove();

– // Peek (look) at the front element of the queue
  // (without removing it) and return it.
  ElementType peek();

– // Returns the number of elements currently in the queue.
  int size();
```

why stacks and queues in the same lecture?

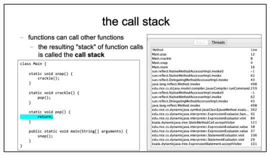
stack interface

```
– // Push (add)    a new element to the top  of the stack.
  void push(ElementType element);

– // Remove (pop)     the top   element of the stack.
  // and returns it.
  ElementType pop();

– // Peek (look) at the top   element of the stack
  // (without removing it) and return it.
  ElementType peek();

– // Returns the number of elements currently in the stack.
  int size();
```

queue interface

```
– // Add (enqueue) a new element to the back of the queue.
  void add(ElementType element);

– // Remove (dequeue) the front element of the queue
  // and return it.
  ElementType remove();

– // Peek (look) at the front element of the queue
  // (without removing it) and return it.
  ElementType peek();

– // Returns the number of elements currently in the queue.
  int size();
```

what are some example uses of stacks and queues in computer science?

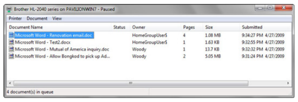


the call stack

- functions can call other functions
  - the resulting "stack" of function calls is called the **call stack**

```
class Main {

    static void snap() {
        crackle();
    }

    static void crackle() {
        pop();
    }

    static void pop() {
        return;
    }

    public static void main(String[] arguments) {
        snap();
    }
}
```

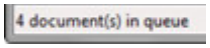
Method	Line
Main.pop	12
Main.crackle	8
Main.snap	4
Main.main	16
sun.reflect.NativeMethodAccessorImpl.invoke0	-1
sun.reflect.NativeMethodAccessorImpl.invoke	62
sun.reflect.DelegatingMethodAccessorImpl.invoke	43
java.lang.reflect.Method.invoke	498
edu.rice.cs.drjava.model.compiler.JavacCompiler.runCommand	259
sun.reflect.NativeMethodAccessorImpl.invoke0	-1
sun.reflect.NativeMethodAccessorImpl.invoke	62
sun.reflect.DelegatingMethodAccessorImpl.invoke	43
java.lang.reflect.Method.invoke	498
edu.rice.cs.dynamicjava.symbol.JavaClass$JavaMethod.evalu...	362
edu.rice.cs.dynamicjava.interpreter.ExpressionEvaluator.han...	92
edu.rice.cs.dynamicjava.interpreter.ExpressionEvaluator.visit	84
koala.dynamicjava.tree.StaticMethodCall.acceptVisitor	121
edu.rice.cs.dynamicjava.interpreter.ExpressionEvaluator.value	38
edu.rice.cs.dynamicjava.interpreter.ExpressionEvaluator.value	37
edu.rice.cs.dynamicjava.interpreter.StatementEvaluator.visit	106
edu.rice.cs.dynamicjava.interpreter.StatementEvaluator.visit	29
koala.dynamicjava.tree.ExpressionStatement.acceptVisitor	101



Brother HL-2040 series on PAVILIONWIN7 - Paused

Document Name	Status	Owner	Pages	Size	Submitted
Microsoft Word - Renovation email.doc		HomeGroupUser$	4	1.08 MB	9:34:27 PM 4/27/2009
Microsoft Word - Test2.docx		HomeGroupUser$	1	1.63 KB	9:32:55 PM 4/27/2009
Microsoft Word - Mutual of America inquiry.doc		Woody	1	13.7 KB	9:32:32 PM 4/27/2009
Microsoft Word - Allow Bongkod to pick up Ad...		Woody	2	5.05 MB	9:31:24 PM 4/27/2009

4 document(s) in queue



4 document(s) in queue

ANNOUNCEMENTS

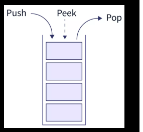
today is Laptop Wednesday!

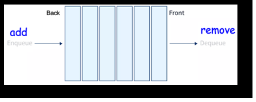
replace interpret(...tokens) with version from **the Writeup**

WARMUP

a couple (slightly silly) interview questions

- implement a stack using a (basic) queue
  - just implement push and pop (and peek?)
  - (don't worry about efficiency)
  - pseudocode is fine
  - what is the runtime (big O)?
- implement a queue using 2 stacks





TODAY stacks and queue tutorial

record LEC-02

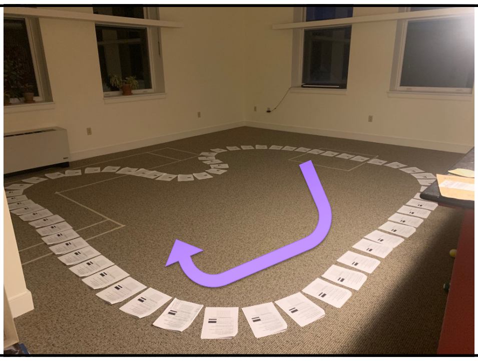
Stack and Queue tutorial
💻

the midterm
😱

i graded the midterm over the weekend

i had so much fun

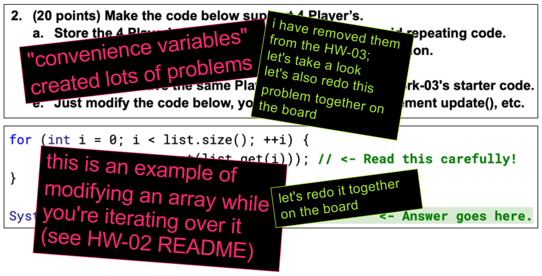
not really



the midterm

- overall, i think it went well!
  - MEAN ~ 87
  - STANDARD DEVIATION ~ 11
  - if you have any questions (how to prep for final, why did you lose points), let's chat in Lab (we can also schedule a chat)
- notes on grading
  - some feedback just for your benefit (no lost points) – variable names, etc.
  - feedback is **very** sparse! (please ask questions if unsure what i mean)

some common pain points



2. (20 points) Make the code below sup… 4 Player's.
   a. Store the 4 Pla… …id repeating code.
   …on.
   …e the same Pla… …rk-03's starter code.
   e. Just modify the code below, yo… …ement update(), etc.

```
for (int i = 0; i < list.size(); ++i) {
                  …t(list.get(i))); // <- Read this carefully!
}

Syst…                                  <- Answer goes here.
```

"convenience variables" created lots of problems

i have removed them from the HW-03; let's take a look let's also redo this problem together on the board

this is an example of modifying an array while you're iterating over it (see HW-02 README)

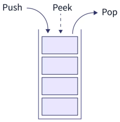
let's redo it together on the board

ANNOUNCEMENTS

today is Friday!

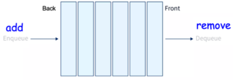
apply to TA next semester! also next next (next?) semester!

WARMUP



implement a queue using two stacks

- just implement add and remove (and peek?)
- (don't worry about efficiency)
- pseudocode is fine
- what is the runtime (big O)?



TODAY odds and ends

record LEC-02

tricky midterm questions

Player -> Player[]

question

```
class HW03 extends App { // BEFORE
    Player player;

    void setup() {
        player = new Player();
        player.color = Vector3.cyan;
        player.radius = 4.0;
        player.position = new Vector2(0.0, 0.0)
    }
}
```

answer

```
class HW03 extends App { // BEFORE
    Player player;

    void setup() {
        player = new Player();
        player.color = Vector3.cyan;
        player.radius = 4.0;
        player.position = new Vector2(0.0, 0.0)
    }
}
```

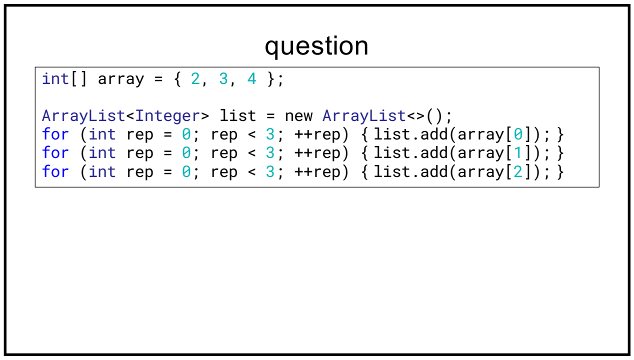
what will this code print? (arrays and array lists)

question

```
int[] array = { 2, 3, 4 };

ArrayList<Integer> list = new ArrayList<>();
for (int i = 0; i < array.length; ++i) {
    for (int rep = 0; rep < 3; ++rep) {
        list.add(array[i]);
    }
}
```

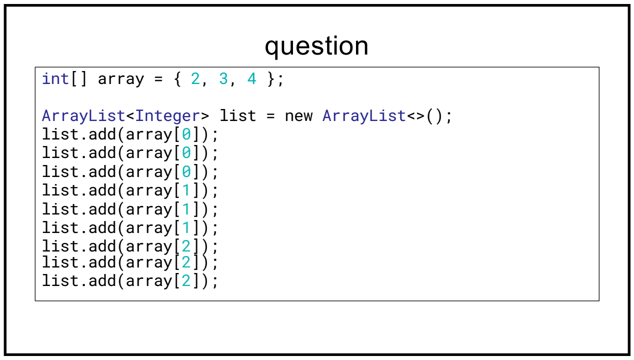
question

```
int[] array = { 2, 3, 4 };

ArrayList<Integer> list = new ArrayList<>();
int i = 0;
for (int rep = 0; rep < 3; ++rep) { list.add(array[i]); }
i = 1;
for (int rep = 0; rep < 3; ++rep) { list.add(array[i]); }
i = 2;
for (int rep = 0; rep < 3; ++rep) { list.add(array[i]); }
```



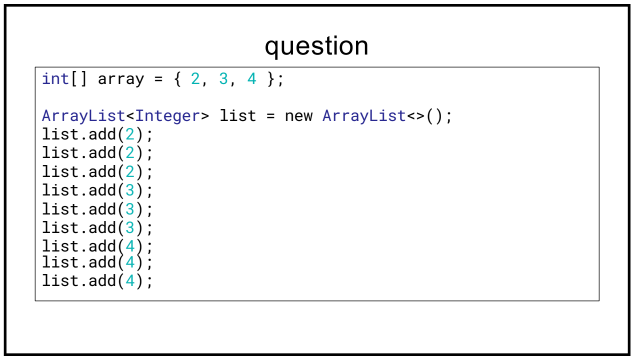
question

```
int[] array = { 2, 3, 4 };

ArrayList<Integer> list = new ArrayList<>();
for (int rep = 0; rep < 3; ++rep) { list.add(array[0]); }
for (int rep = 0; rep < 3; ++rep) { list.add(array[1]); }
for (int rep = 0; rep < 3; ++rep) { list.add(array[2]); }
```



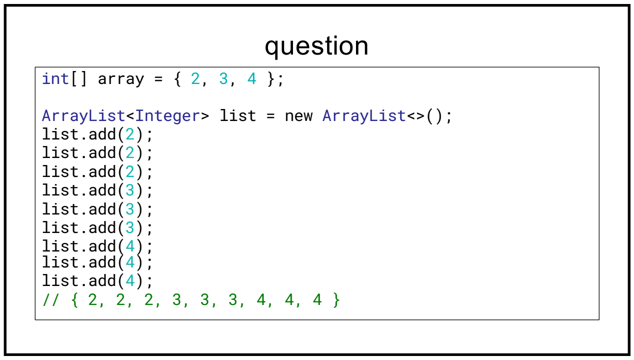
question

```
int[] array = { 2, 3, 4 };

ArrayList<Integer> list = new ArrayList<>();
list.add(array[0]);
list.add(array[0]);
list.add(array[0]);
list.add(array[1]);
list.add(array[1]);
list.add(array[1]);
list.add(array[2]);
list.add(array[2]);
list.add(array[2]);
```



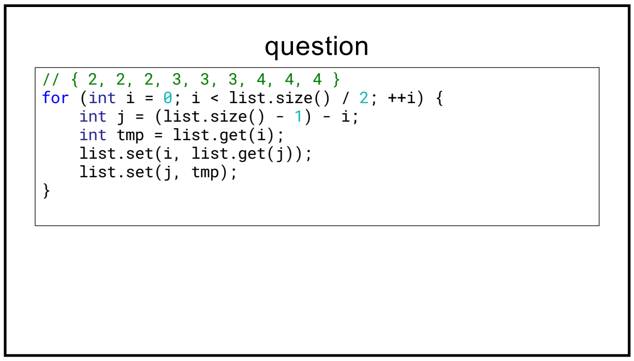
question

```
int[] array = { 2, 3, 4 };

ArrayList<Integer> list = new ArrayList<>();
list.add(2);
list.add(2);
list.add(2);
list.add(3);
list.add(3);
list.add(3);
list.add(4);
list.add(4);
list.add(4);
```



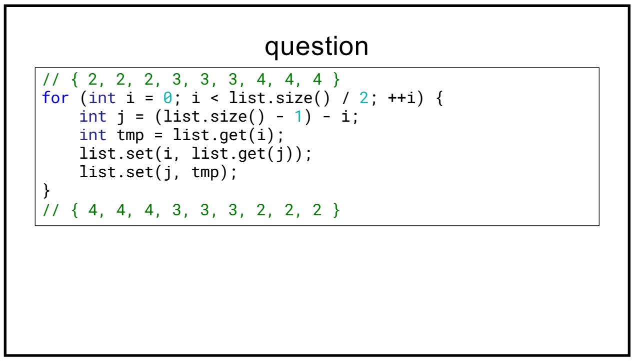
question

```
int[] array = { 2, 3, 4 };

ArrayList<Integer> list = new ArrayList<>();
list.add(2);
list.add(2);
list.add(2);
list.add(3);
list.add(3);
list.add(3);
list.add(4);
list.add(4);
list.add(4);
// { 2, 2, 2, 3, 3, 3, 4, 4, 4 }
```



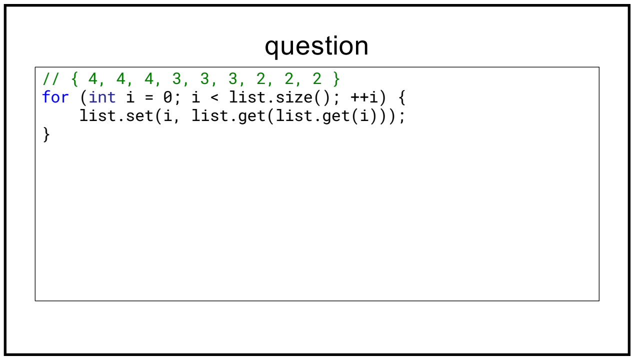
question

```
// { 2, 2, 2, 3, 3, 3, 4, 4, 4 }
for (int i = 0; i < list.size() / 2; ++i) {
    int j = (list.size() - 1) - i;
    int tmp = list.get(i);
    list.set(i, list.get(j));
    list.set(j, tmp);
}
```



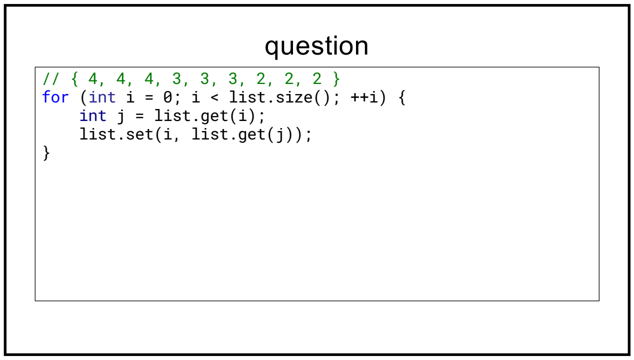
question

```
// { 2, 2, 2, 3, 3, 3, 4, 4, 4 }
for (int i = 0; i < list.size() / 2; ++i) {
    int j = (list.size() - 1) - i;
    int tmp = list.get(i);
    list.set(i, list.get(j));
    list.set(j, tmp);
}
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
```



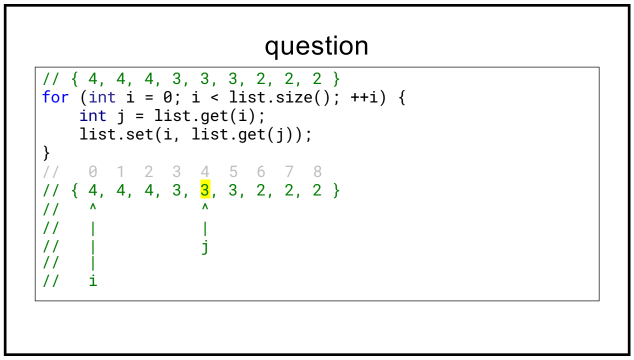
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    list.set(i, list.get(list.get(i)));
}
```



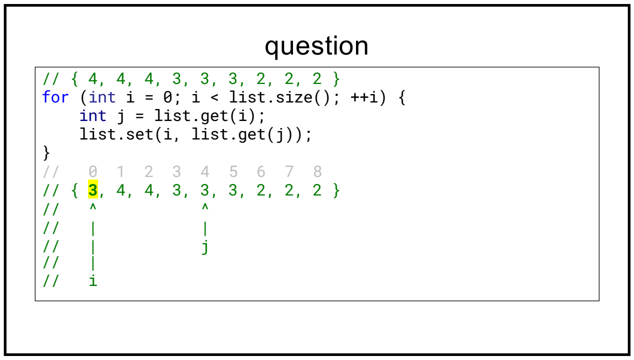
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
```



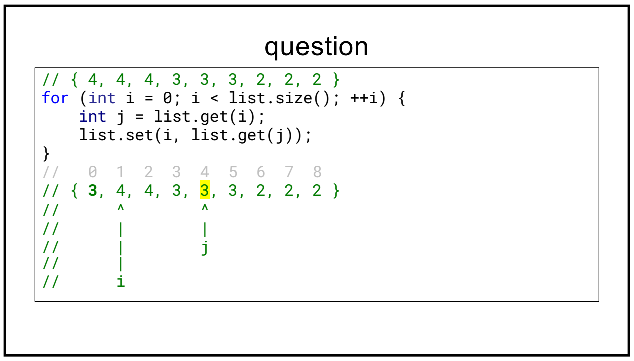
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
//   ^           ^
//   |           |
//   |           j
//   |
//   i
```



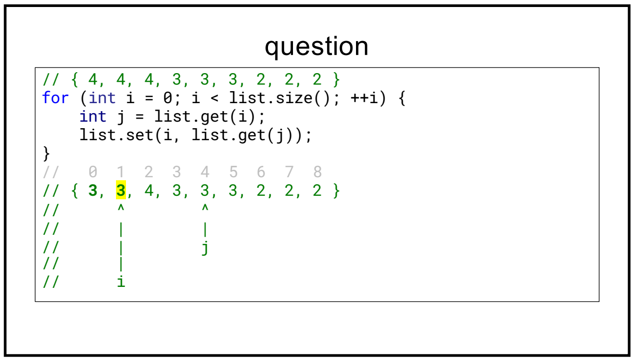
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 4, 4, 3, 3, 3, 2, 2, 2 }
//   ^           ^
//   |           |
//   |           j
//   |
//   i
```



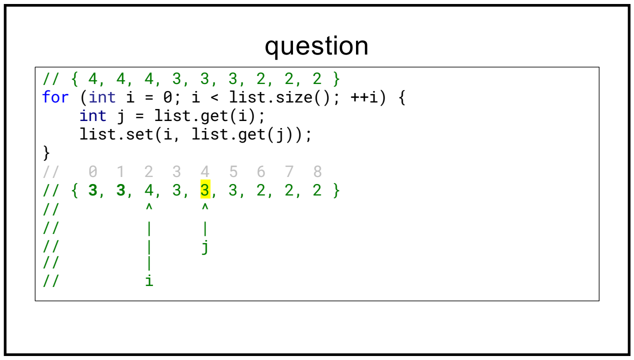
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 4, 4, 3, 3, 3, 2, 2, 2 }
//      ^        ^
//      |        |
//      |        j
//      |
//      i
```



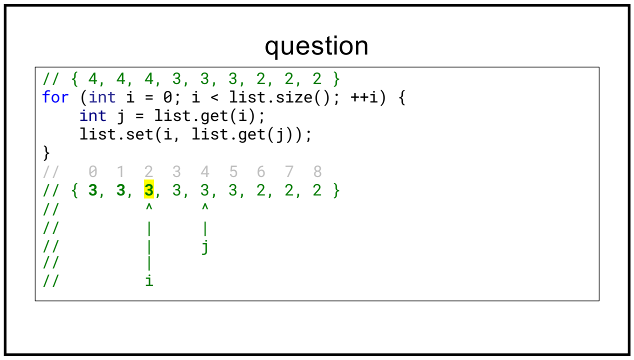
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 4, 3, 3, 3, 2, 2, 2 }
//      ^        ^
//      |        |
//      |        j
//      |
//      i
```



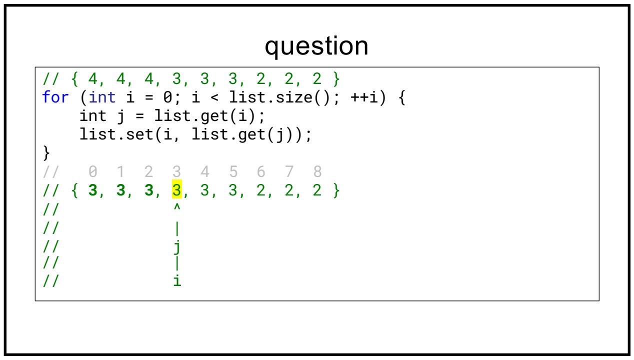
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 4, 3, 3, 3, 2, 2, 2 }
//         ^     ^
//         |     |
//         |     j
//         |
//         i
```



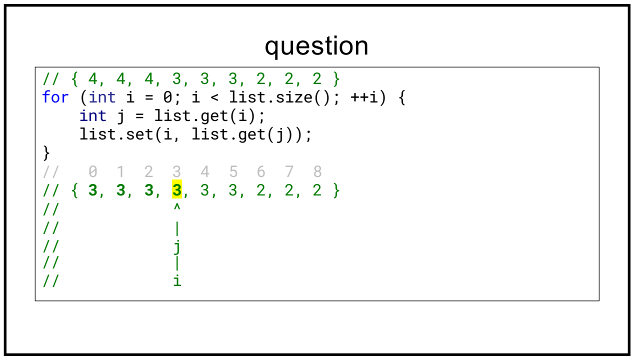
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//         ^     ^
//         |     |
//         |     j
//         |
//         i
```



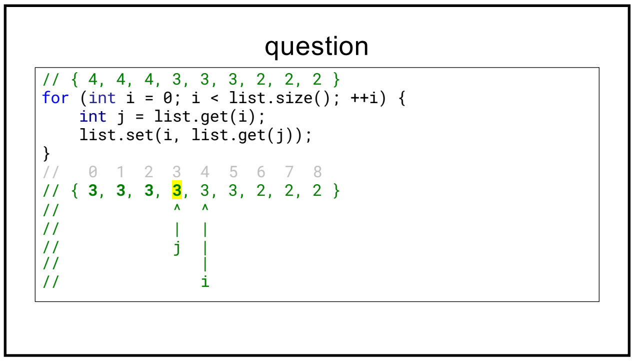
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//            ^
//            |
//            j
//            |
//            i
```



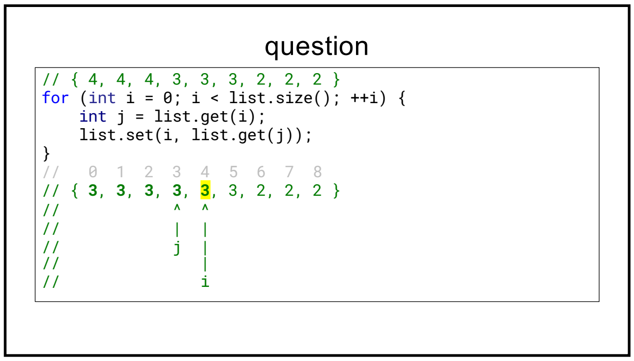
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//            ^
//            |
//            j
//            |
//            i
```



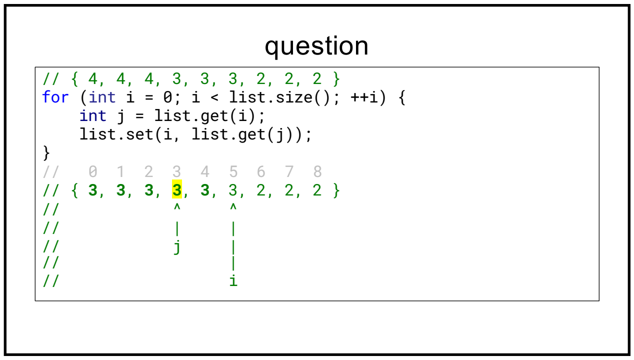
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//            ^  ^
//            |  |
//            j  |
//               |
//               i
```



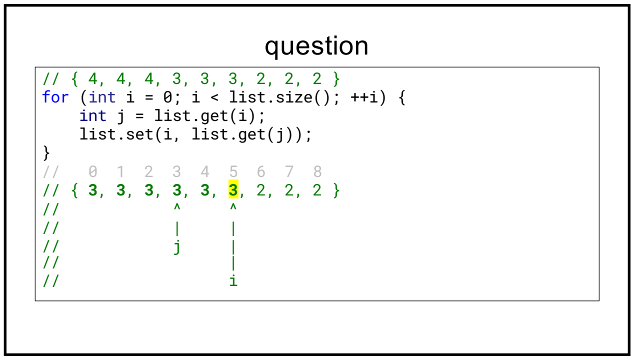
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//            ^  ^
//            |  |
//            j  |
//               |
//               i
```



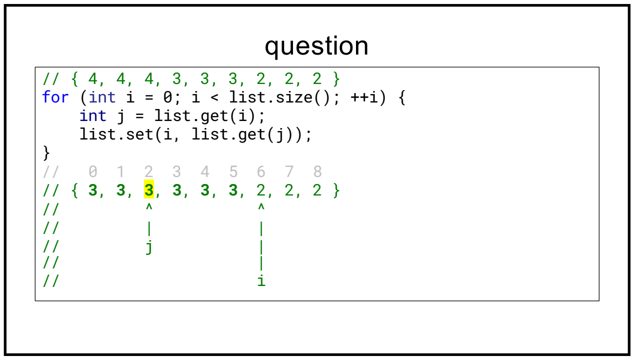
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//            ^     ^
//            |     |
//            j     |
//                  |
//                  i
```



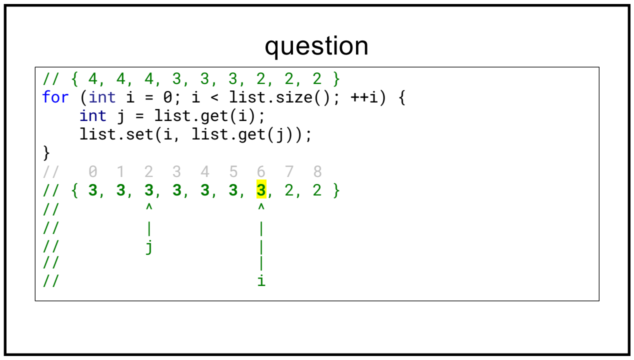
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//            ^     ^
//            |     |
//            j     |
//                  |
//                  i
```



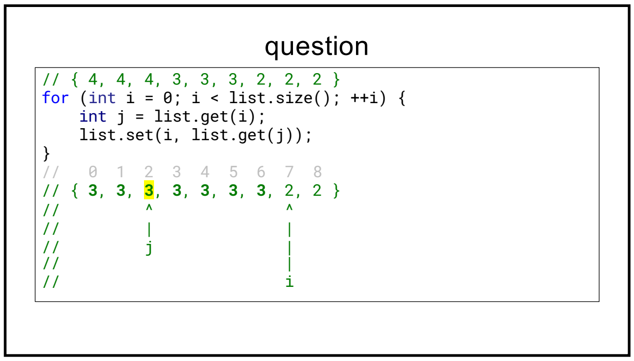
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 2, 2, 2 }
//         ^           ^
//         |           |
//         j           |
//                     |
//                     i
```



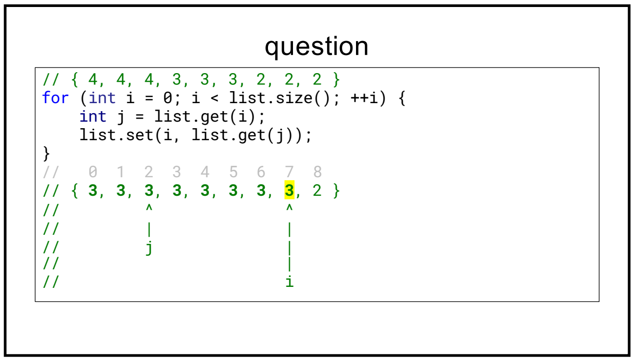
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 3, 2, 2 }
//         ^           ^
//         |           |
//         j           |
//                     |
//                     i
```



question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 3, 2, 2 }
//         ^              ^
//         |              |
//         j              |
//                        |
//                        i
```



question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 3, 3, 2 }
//         ^              ^
//         |              |
//         j              |
//                        |
//                        i
```

question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 3, 3, 2 }
//         ^                 ^
//         |                 |
//         j                 |
//                           |
//                           i
```

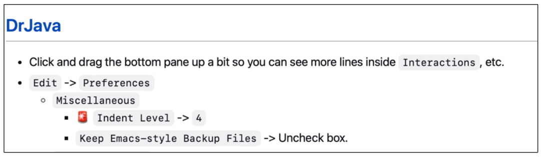
question

```
// { 4, 4, 4, 3, 3, 3, 2, 2, 2 }
for (int i = 0; i < list.size(); ++i) {
    int j = list.get(i);
    list.set(i, list.get(j));
}
//   0  1  2  3  4  5  6  7  8
// { 3, 3, 3, 3, 3, 3, 3, 3, 3 }
//         ^                 ^
//         |                 |
//         j                 |
//                           |
//                           i
```

HW06.java
HW06.class
~HW06.java

filetypes

- HW06.java is **source code** (**input** to the compiler)
- HW06.class is bytecode (**output** from the compiler)
- ~HW06.java is a backup file created by DrJava 😳



DrJava

- Click and drag the bottom pane up a bit so you can see more lines inside `Interactions`, etc.
- `Edit` -> `Preferences`
  - `Miscellaneous`
    - 🚨 `Indent Level` -> `4`
    - `Keep Emacs-style Backup Files` -> Uncheck box.

`_thisUnderscoreMeansDoNotUse`

(unless you really know what you're doing)

where did the Starter Code come from?

(forgive me, I know this should have been on Wednesday) the big decision in the Starter Code is that we're going to chop the program into "tokens"

and each token could be a boolean, double, String, or list

in Python, a variable's type is **dynamic**

```
token = True
print(type(token)) # <class 'bool'>

token = 5.0
print(type(token)) # <class 'float'>

token = "exch"
print(type(token)) # <class 'str'>

token = ["exch", 0.0, "add"]
print(type(token)) # <class 'list'>
```

in Java, a variable's type is **static**

```
class Token {
    int type;
    private boolean _valueIfTypeBoolean;
    private double _valueIfTypeDouble;
    private String _valueIfTypeString;
    private ArrayList<Token> _valueIfTypeList;

    static final int TYPE_BOOLEAN = 0;
    static final int TYPE_DOUBLE = 1;
    static final int TYPE_STRING = 2;
    static final int TYPE_LIST = 3;
}
```

in Java, a variable's type is **static**

```
// Create a String-type Token.
Token token = new Token();
token.type = Token.TYPE_STRING;
token.value = "exch";
```

```
// Get the value of a String-type token.
assert token.type = Token.TYPE_STRING; // check type!
String string = token._valueIfTypeString;
```

this usage code makes me sad 🙁
let's write some ✨**functions**✨!

in Java, a variable's type is **static**

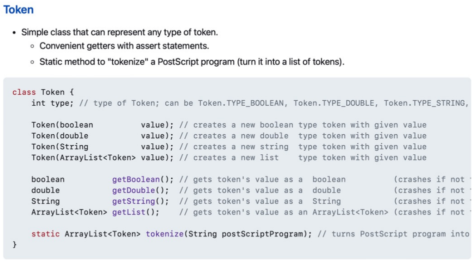
```
// Create a String-type Token.
Token token = new Token("exch");
```

```
// Get the value of a String-type token.

String string = token.getString();
```

all better 🙂

tl;dr

use these functions 🙂 👍

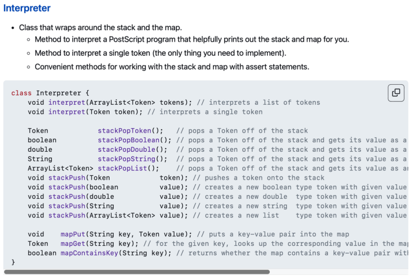


Token

- Simple class that can represent any type of token.
  - Convenient getters with assert statements.
  - Static method to "tokenize" a PostScript program (turn it into a list of tokens).

```
class Token {
    int type; // type of Token; can be Token.TYPE_BOOLEAN, Token.TYPE_DOUBLE, Token.TYPE_STRING,

    Token(boolean          value); // creates a new boolean type token with given value
    Token(double           value); // creates a new double  type token with given value
    Token(String           value); // creates a new string  type token with given value
    Token(ArrayList<Token> value); // creates a new list    type token with given value

    boolean          getBoolean(); // gets token's value as a  boolean          (crashes if not
    double           getDouble();  // gets token's value as a  double           (crashes if not
    String           getString();  // gets token's value as a  String           (crashes if not
    ArrayList<Token> getList();    // gets token's value as an ArrayList<Token> (crashes if not

    static ArrayList<Token> tokenize(String postScriptProgram); // turns PostScript program into
}
```

and these functions!! 🙂 👍



Interpreter

- Class that wraps around the stack and the map.
  - Method to interpret a PostScript program that helpfully prints out the stack and map for you.
  - Method to interpret a single token (the only thing you need to implement).
  - Convenient methods for working with the stack and map with assert statements.

```
class Interpreter {
    void interpret(ArrayList<Token> tokens); // interprets a list of tokens
    void interpret(Token token); // interprets a single token

    Token            stackPopToken();   // pops a Token off of the stack
    boolean          stackPopBoolean(); // pops a Token off of the stack and gets its value as a
    double           stackPopDouble();  // pops a Token off of the stack and gets its value as a
    String           stackPopString();  // pops a Token off of the stack and gets its value as a
    ArrayList<Token> stackPopList();    // pops a Token off of the stack and gets its value as a
    void stackPush(Token            token); // pushes a token onto the stack
    void stackPush(boolean          value); // creates a new boolean type token with given value
    void stackPush(double           value); // creates a new double  type token with given value
    void stackPush(String           value); // creates a new string  type token with given value
    void stackPush(ArrayList<Token> value); // creates a new list    type token with given value

    void    mapPut(String key, Token value); // puts a key-value pair into the map
    Token   mapGet(String key); // for the given key, looks up the corresponding value in the ma
    boolean mapContainsKey(String key); // returns whether the map contains a key-value pair wit
}
```

```
// Pop a double off the stack
assert _stack.size() > 0;
Token token = _stack.pop();
double num = token.getDouble();
```

```
// Push a double onto the stack.
_stack.push(new Token(5.0));
```

🙁

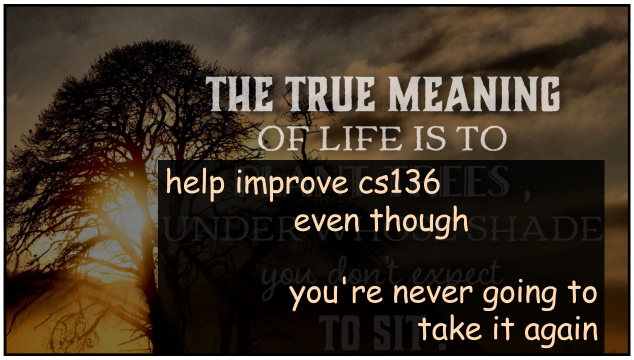
```
// Pop a double off the stack


double num = stack.popDouble();
```

```
// Push a double onto the stack.
stackPush(5.0);
```

🙂

midsemester feedback

do you *want* to give midsemester feedback?



https://tinyurl.com/yf725rdk