6.178: Introduction to Software Engineering in Java

Lecture 3: Data Structures

Last lecture...

Good Coding Practice

#1: Don’t Repeat Yourself! (DRY)

#2: Comment where needed!

#3: Use good names for variables, classes, etc...

#4: Use indentation and spaces

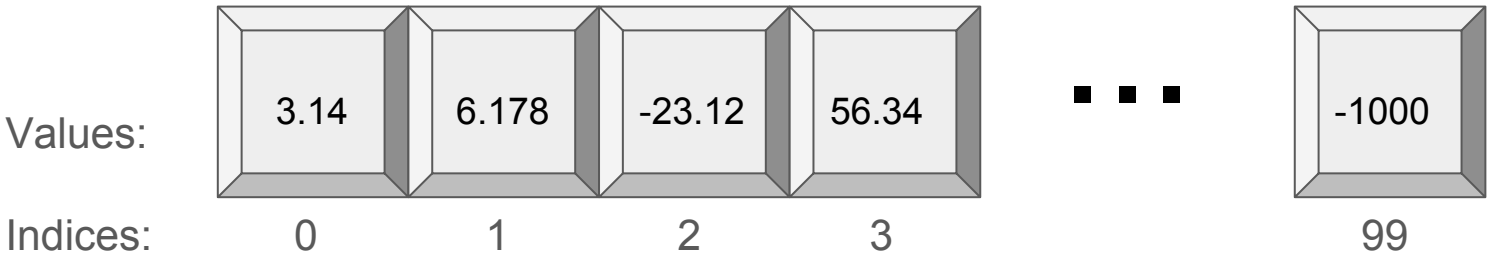
Debugging

- System.out.println() is your best friend!
- Think about what variables change after every executed line.
- The computer isn’t stupid. It just does what you tell it.
- Eclipse has a good debug tool too.

Arrays

- Type[] arrayName = new Type[INT_SIZE];
- Arrays are of a fixed size. (Coming soon: ArrayList)
- Zero-indexing

```
double[] myNumbers = new double[100];
```



Dynamic Arrays - aka Lists

Example:

```
String myName = “Andreea”;
String yourName = “Kat”;
List names = new ArrayList();
names.add(myName);
System.out.println(names.size()); // Prints 1
names.add(yourName);
System.out.println(names.size()); // Prints 2
String myTempName = (String) names.get(0); // myTempName = “Andreea”
String yourTempName = (String) names.get(1); // yourTempName = “Kat”
```

Today!

Let’s continue on the Data Structures roadmap!

But first, quick word on Collections

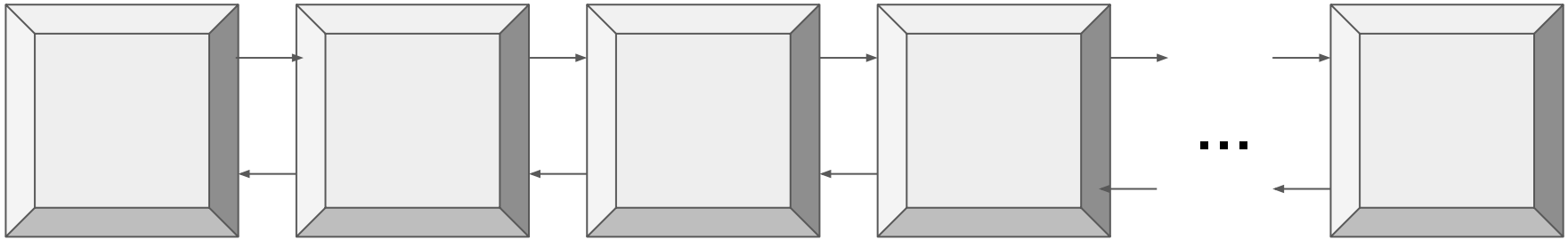
A *collection* — sometimes called a container — is simply an object that groups multiple elements into a single unit. Collections are used to store, retrieve, manipulate, and communicate aggregate data. The Java Collections Framework has several useful classes which have tons of useful functions which makes a programmer task super easy.

Data Structures

- Arrays
- Vector
- **Linked List**
- Set (HashSet)
- Map (HashMap)
- Stack
- Queue

Linked List

- Objects are arranged in a linear order.
- Unlike an array, however, in which the linear order is determined by the array indices, the order in a linked list is determined by a pointer in each object.
- Singly linked, doubly linked



Data Structures

- Arrays
- Vector
- Linked List
- **Set (HashSet)**
- Map (HashMap)
- Stack
- Queue

Set

- A Set is a collection of zero or more unique objects.
- Like a mathematical set or a Python set – and unlike a List – an object cannot appear in a set multiple times. Either it's in or it's out.
- A set: 1, 3, 4, 7, 8; Not a set: 1, 3, 4, 4, 4, 5, 7
- Depending on the implementation, sets usually don't have an ordering
- Set is the interface. HashSet is one implementation
- Important methods: add(Item), isEmpty(), contains(Item), remove (Item), size(), iterator()

HashSet

```
Set mySet = new HashSet();
String element = "element 1";
mySet.add(element);
System.out.println( mySet.contains(element) ); // Prints true
mySet.remove(element);
System.out.println( mySet.contains(element) ); // Prints false
```

Data Structures

- Arrays
- Vector
- Linked List
- Set (HashSet)
- **Map (HashMap)**
- Stack
- Queue

HashMap

- Map<K, V> is the interface, HashMap is one implementation.
- Maps a Key to a Value.
- Good for storing pairs of things because it’s fast at retrieving values for a given key (constant time).
- Important methods: get(Key), put(Key, Val), containsKey(Key), containsValue(Val), keySet(), values()

HashMap

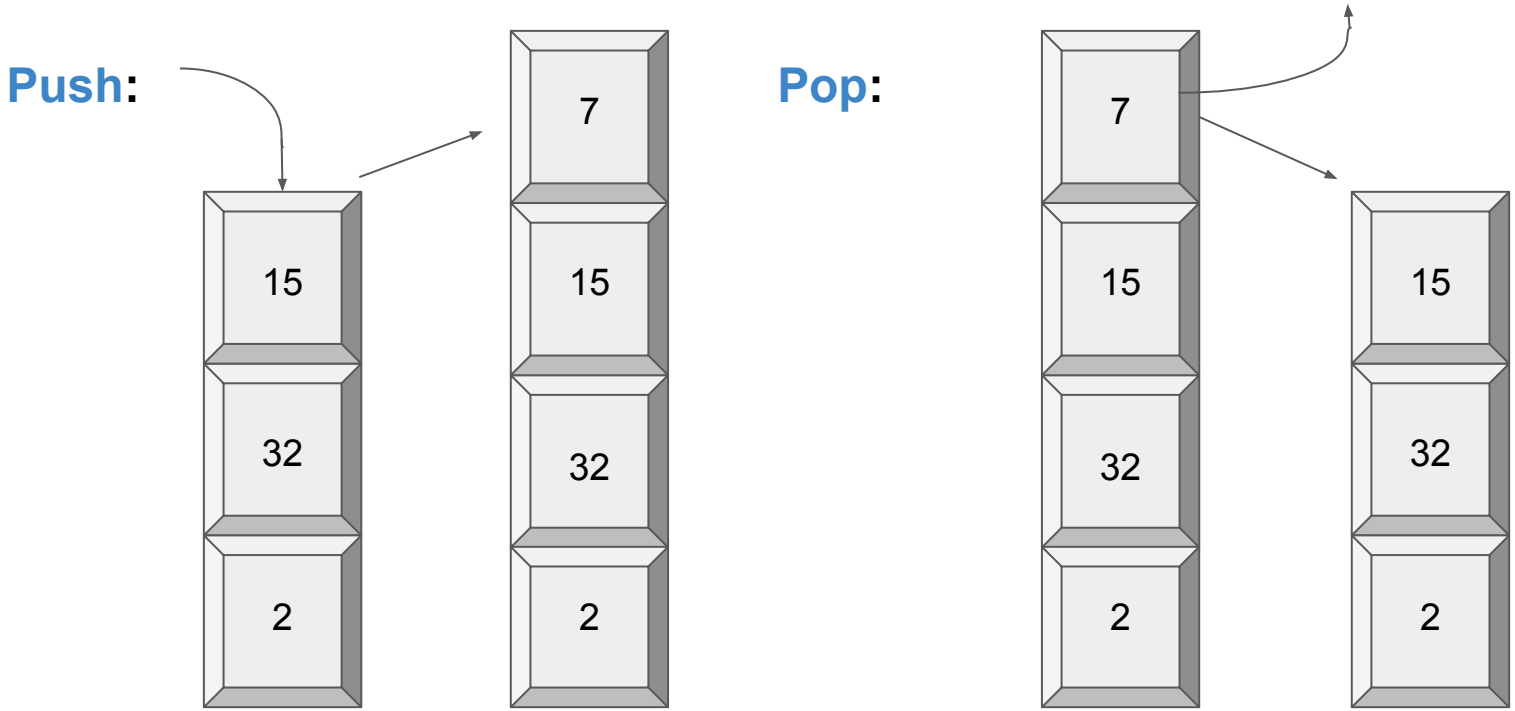
```
class Contact{ … }
HashMap<String, Contact> allContacts = new HashMap<String,
Contact>();
map.put(“617-456-6178”, new Contact(...));
map.put(“1-800-MIT-JAVA”, new Contact(...));
Contact person = map.get(“617-456-6178”);

Set<String> set = map.keySet(); // Has all phone #s
Collection<Contact> contacts = map.values();//Contacts
```

Data Structures

- Arrays
- Vector
- Linked List
- Set (HashSet)
- Map (HashMap)
- **Stack**
- Queue

Stack

- Subclass of Vector that implements a standard last-in, first-out stack.



Stack

- Dinner plates example
- Methods: pop(), push(Item), peek(), empty(), search(Item)
- We’ll play with them in the coding exercises

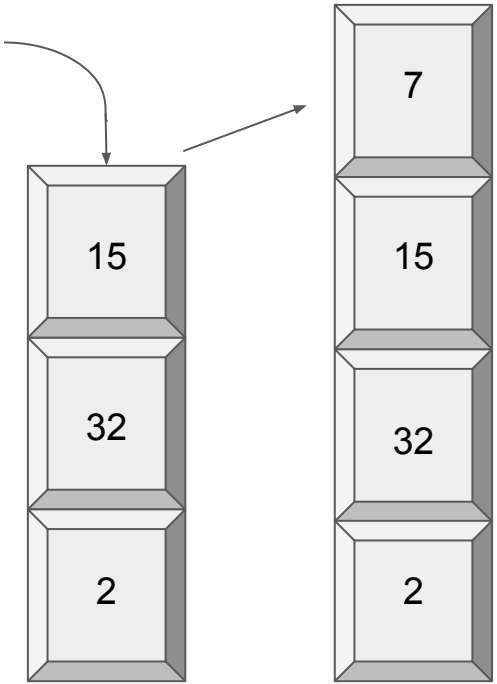
Data Structures

- Arrays
- Vector
- Linked List
- Set (HashSet)
- Map (HashMap)
- Stack
- **Queue**

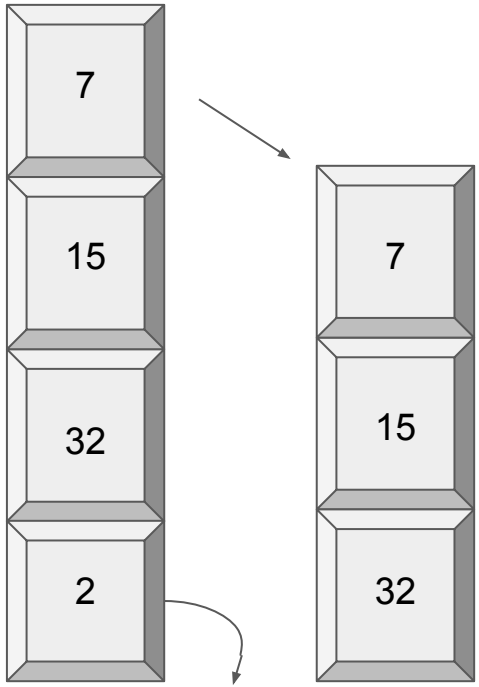
Queue

- Like Stack, but First-In-First-Out.

Push:



Pop:



Queue

- Queues in real life
- Methods: add(Item), element(), peek(), poll(), remove()
- Computing applications: serving requests of a single shared resource (printer, disk, CPU), transferring data asynchronously (data not necessarily received at same rate as sent) between two processes (pipes, file IO, sockets), Spotify playlist.

Coding exercises

Let’s play with these structures!

Demo Exercises:

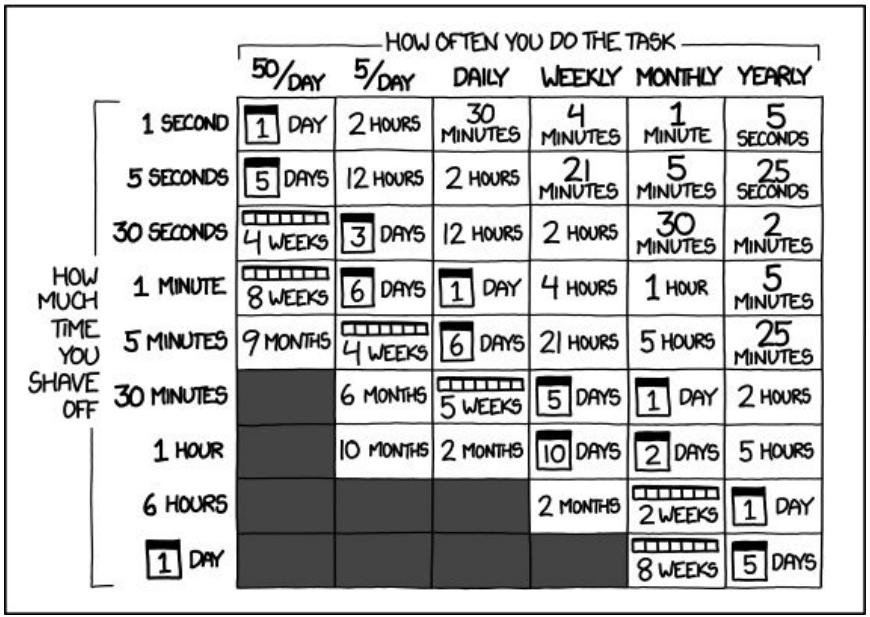
http://shoutkey.com/ate

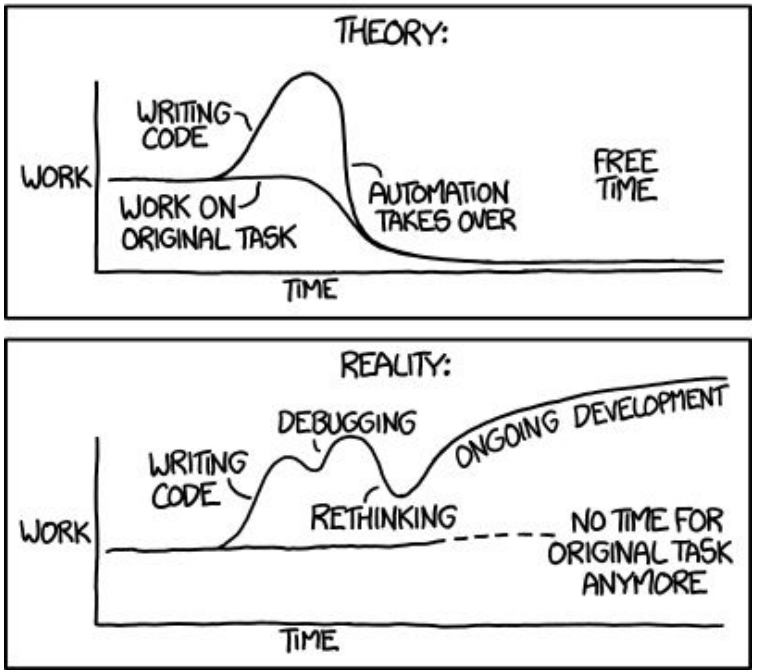
You do not need to turn in the exercises in this repo only, but you should definitely follow along to see how awesome Eclipse is.

git clone https://github.mit.edu/6178-iap16/lec3_eclipse_demo.git

When you come up with a better variable name after you’ ve already used it a billion times… (pictured: me)

Obligatory XKCDs





Neat Eclipse Tricks

Generally, there IS a better way.

Better Ways

- Autocomplete (Ctrl + Space)
- Fix Indentation (Cmd + I) or (Ctrl + I)
- Find in Project (Shift + Cmd + T) or (Shift + Ctrl + T)
- Refactor/Rename (Alt + Cmd + R) or (Shift + Alt + R)
- Refactor/Extract Method (Alt + Cmd + M) or (Alt + Shift M)
- Organize Imports (Shift + Cmd + O) or (Shift + Ctrl + O)
- Save Actions
- Boilerplate Code Automatic Generation
- Eclipse’s Git Status
- Documentation Hover
- Line Numbers

To Eclipse!

You can follow along in BetterWays.

We'll start in Main.java.

Javadocs

- All of the built-in Java stuff is documented by Oracle at https://docs.oracle.com/javase/8/docs/api/
- Third party APIs also generally have something basically identical to this.
- Even the Card and Deck things you used in the last PSet have one!
  - http://web.mit.edu/abartow/www/cardAPIdocs/
- They're generally written in a rather specific, technical manner.
  - You'll learn how to write them if you take 6.005!

- How do you read these things?

Here be OOP Dragons

- You can learn a LOT by reading Javadocs.
- They contain a lot of ideas that won't make much sense until next week
- An important skill to learn is filtering out what you don't need to understand immediately
    - even experienced programmers do this!

I want to learn more about Lists!

See if you can find the Javadocs for it!

Javadoc for List

Interface List<E>

Type Parameters:

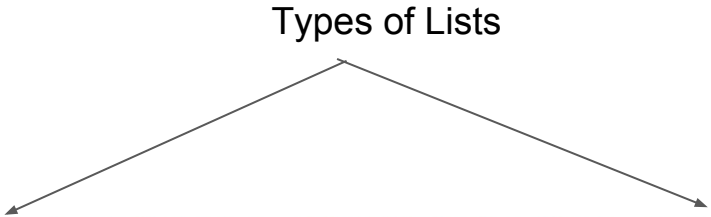
`E - the type of elements in this list`

All Superinterfaces:

`Collection<E>, Iterable<E>`

All Known Implementing Classes:

`AbstractList, AbstractSequentialList, ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector`



```
public interface List<E>
extends Collection<E>
```

An ordered collection (also known as a *sequence*). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements `e1` and `e2` such that `e1.equals(e2)`, and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

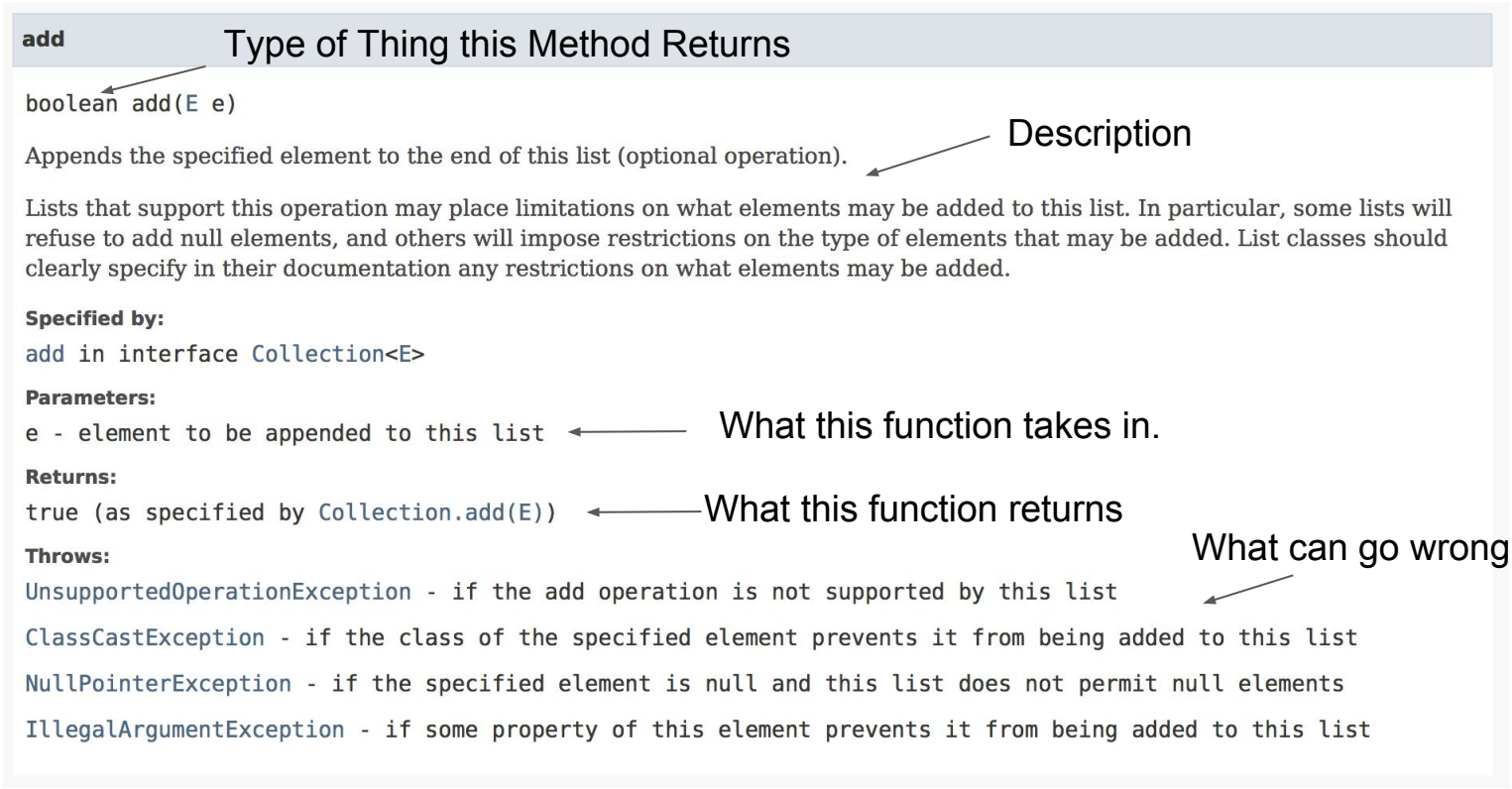
What can I do with Lists?

Method Summary

All Methods | **Instance Methods** | **Abstract Methods** | **Default Methods**

Modifier and Type	Method and Description
boolean	**add**(**E** e) Appends the specified element to the end of this list (optional operation).
void	**add**(int index, **E** element) Inserts the specified element at the specified position in this list (optional operation).
boolean	**addAll**(**Collection**<? extends **E**> c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator (optional operation).
boolean	**addAll**(int index, **Collection**<? extends **E**> c) Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Click on a method!



Implement drawPokerHand() without helper methods

It might be difficult/confusing

You might not understand most of it

This is ok! It will make a lot more sense after next week

Do what you can and you will get full credit for making an attempt at it

Ask if you need help!