LEC 14

CSE 122

Collections

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

coffee or tea?

Music: Hunter/Miya's Playlist

Instructor Hunter Schafer / **Miya Natsuhara**

TAs

Ajay	Gaurav	Melissa
Andrew	Hilal	Noa
Anson	Hitesh	Parker
Anthony	Jake	Poojitha
Audrey	Jin	Samuel
Chloe	Joe	Sara
Colton	Joe	Simon
Connor	Karen	Sravani
Elizabeth	Kyler	Tan
Evelyn	Leon	Vivek

Lecture Outline

- Announcements
- Optional
- Recap of Collections
- Dumb Data Structures
- Collections

Announcements

- Next week weirdness (11/21-11/25)
  - See announcements from yesterday
- P3 will be released later today (but will not be due until 12/1)
- Canvas grades will *actually* be posted this weekend
  - Hunter continues to apologize in advance for notification spam

Lecture Outline

- Announcements
- Optional
- Recap of Collections
- Dumb Data Structures
- Collections

(PCM) Optional

`Optional` is a Java class that is used to handle situations where a value is *sometimes* there.

You give `Optional` a type to hold (or potentially not hold) when you are referring to its type.

e.g., `Optional<String>`, `Optional<Integer>`, `Optional<Point>`

(PCM) Optional Methods

Method	Description
`Optional.empty()`	Creates an empty `Optional` object
`Optional.of(…)`	Creates an `Optional` object holding the object it's given
`isEmpty()`	Returns `true` if there *is no* value stored, and `false` otherwise
`isPresent()`	Returns `true` if there *is* a value stored, and `false` otherwise
`get()`	Returns the stored object from the `Optional` (if one is stored; otherwise throws a `NoSuchElementException`)

The Optional class has more than just these methods, but these are what you'll need to focus on for this class!

(PCM) Optional Methods

`isEmpty()`, `isPresent()`, and `get()` are called like normal instance methods (on an actual instance of `Optional`).

`Optional.of(…)` and `Optional.empty()` are called differently

(Like the `Math` class methods)

(PCM) Why Optional?

Using `Optional` can help programmers avoid `NullPointerExceptions` by making it explicit when a variable may or may not contain a value.

There are other `Optional` methods (that you should explore in your own time if you're interested) that can be really useful to cleanly work with data that may or may not be present.

Student / Course Example one more time…

Let's add two more methods to `Course.java`:

```
public void setCourseEvalLink(String url)

public Optional<String> getCourseEvalLink()
```

The link to the evaluations for a course doesn't usually exist until the last few weeks of the quarter. What if a client calls `getCourseEvalLink` before one is set up?

`Optional` to the rescue!

Lecture Outline

- Announcements
- Optional
- **Recap of Collections**
- Dumb Data Structures
- Collections

Goal for Today

Review some of the data structures we've talked about this quarter

Understand how Java organizes them with *interfaces*

Collections: What *classes* have we seen so far?

Collections: What *interfaces* have we seen so far?

Lecture Outline

- Announcements
- Optional
- Recap of Collections
- **Dumb Data Structures**
- Collections

Dumb Data Structures

We're going to create our own versions of these classes so we can dig into how they all relate to each other!

BUT they're going to be real dumb.

If you want to get a sense of how they're *actually* implemented, go take CSE 123!

DumbArrayIntList

```
DumbArrayIntList()
add(int value)
add(int index, int value)
contains(int value)
isEmpty()
get(int index)
set(int index, int value)
remove(int index)
size()
indexOf(int value)
toString()
```

Lecture Outline

- Announcements
- Optional
- Recap of Collections
- Dumb Data Structures
- **Collections** 

DumbIntCollection Relationships