

LEC 13

CSE 122**Interfaces**

BEFORE WE START

*Talk to your neighbors:**Classes you're most excited for during Spring quarter. Nervous For?*Music: [122 25wi Lecture Tunes](#) **Instructor:** Elba Garza

TAs:	Anya	Daniel Ryan	Ken	Nicole
	Ashley	Diya	Kuhu	Nicole
	Cady	Elizabeth	Kyle	Niyati
	Caleb	Hannah	Leo	Sai
	Carson	Harshitha	Logan	Steven
	Chaafen	Ivory	Maggie	Yang
	Colin	Izak	Mahima	Zach
	Connor	Jack	Marcus	
	Dalton	Jacob	Minh	


Questions during Class?

Raise hand or send here

sli.do #cse122

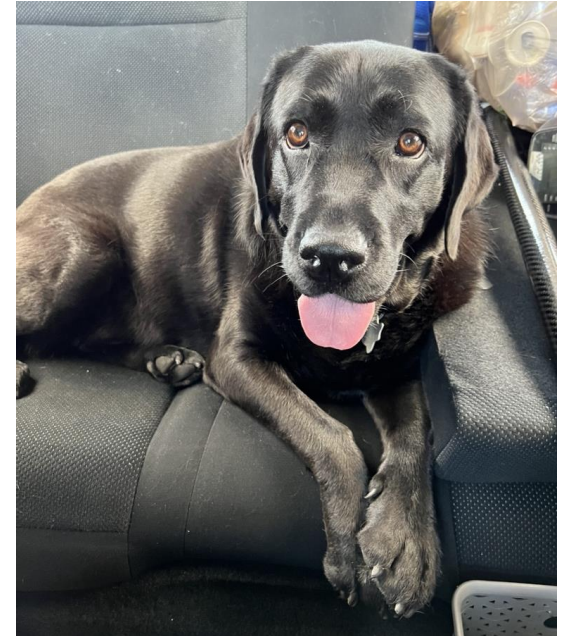


Lecture Outline

- **Announcements** 
- Interface Review
- More Shapes!
- Comparable

Announcements

- Creative Project 2 (C2) due Thursday, February 27th
- Resubmission Cycle 5 (R5) out Thursday, February 27th
 - Eligible: **P1**, P2
- Programming Assignment 3 (P3) out Friday!
 - Due March 6th by 11:59 PM
- Quiz 2 Tuesday, March 4th
 - Practice Quiz coming out Friday, February 28th!
- Reminder on Final Exam:
Wednesday, March 19th 12:30-2:20 PM
- Gigi visits during finals week!



Lecture Outline

- Announcements
- **Interfaces Review** ◀
- More Shapes!
- Comparable

Recall from L6: Wait, ADT? Interfaces?

- **Abstract Data Type (ADT):** A *description of the idea* of a data structure including what operations are available on it and how those operations should behave. For example, the English explanation of what a list should be.
- **Interface:** Java construct that lets programmers *specify what methods a class should have*. For example the `List` interface in java.
- **Implementation:** *Concrete code* that meets the specified interface. For example, the `ArrayList` and `LinkedList` classes that implement the `List` interface.

Interfaces

Interfaces serve as a sort of “contract” – in order for a class to implement an interface, it must fulfill the contract requirements.

The contract requirements are certain methods that the class must implement.

Lists

One ADT we've talked a lot about in this course is a list.

Within Java, there exists a `List` interface – its contract includes methods like:

`add`, `clear`, `contains`, `get`, `isEmpty`, `size`

There's also an `ArrayList` class (implementation)

To get the certificate, it must include all these methods (and any others the `List` interface specifies)

Interfaces vs. Implementation

Interfaces require certain methods, but they do not say anything about how those methods should be implemented – that's up to the class! 🏆

List is an interface

ArrayList is a class that implements the List interface

LinkedList is a class that implements the List interface

...

Why interfaces?

```
public static void fill(WaterBottle w) {...}
```

This method only accepts a WaterBottle!

Why interfaces?

Flexibility



```
public static void fill(Container c) {...}
```

This method can accept either a:

- WaterBottle
- Mug
- Tub or
- Any other class that implements Container!

Why interfaces?

Flexibility



```
public static void method(Set<String> s) {...}
```

This method can accept either a:

- `HashSet<String>` or
- `TreeSet<String>` or
- Any other class that implements `Set` and whose element type is `String`!

Why interfaces?

Abstraction

Interfaces also support *abstraction*
(the separation of ideas from details)



Lecture Outline

- Announcements
- Interfaces Review
- **More Shapes!** 
- Comparable

Classes can Implement Multiple Interfaces

A class can implement multiple interfaces – it's like one person signing multiple contracts!

If a class implements an interface A and an interface B, it'll just have to include all of A's required methods along with all of B's required methods

Classes can Implement Multiple Interfaces

```
public interface Parallel {  
    public int numParallelPairs();  
}
```

```
public class Square implements Shape, Parallel {  
    ...  
    public int numParallelPairs() {  
        return 2;  
    }  
}
```

But Square would have to implement:

- getPerimeter, getArea from Shape

AND

- numParallelPairs from Parallel

An interface can extend another

You can have one interface extend another

So if **public interface A extends B**, then any class that implements A must include all the methods in A's interface and all the methods in B's interface

An interface can extend another

We can write another interface:

Polygon that extends **Shape**

Make modifications such that:

- Square is a **Polygon** (and **Shape**)
- Triangle is a **Polygon** (and **Shape**)
- Circle is a **Shape** (but not a **Polygon**)

Lecture Outline

- Announcements
- Interfaces Review
- More Shapes!
- **Comparable** ◀

Recall the Student / Course Example from Wed

Course stored a field

```
private List<Student> roster;
```

Why not use a Set to store the students?...

Seems like a great idea (no duplicates, not worried about keeping a specific order or indexing into it) but ... Java reasons:

- HashSet won't work because of lack of hashCode() implementation
- TreeSet won't work because... what does it mean to "sort" Students?

Comparable

TreeSet uses an **interface** called Comparable<E> to know how to sort its elements!

Only has one required method:

```
public int compareTo(E other)
```

Its return value is:

- < 0 if this is “less than” other
- 0 if this is equal to other
- > 0 if this is “greater than” other