BEFORE WE START

Chat with neighbors:

What is your favorite fast food chain?

Music: [122 25au Lecture Tunes](#)

LEC 14

CSE 122

Interfaces

Instructor: Elba Garza 1 & Elba Garza2

TAs:

Sreshta	Merav	Shivani	Yang
William	Nicole	Naomi	Cady
Arjun	Vrinda	Hanna	Diya
Dani	Shreya	David	Katharine
Rohan	Wesley	Sushma	
Andrew	Isis	Rio	
Saachi	Colin	Nicolae	
Ava	Medha	Ivory	

Questions during Class?

Raise hand or send here

sli.do #cse122b

Lecture Outline

- **Announcements**

- Interface Review

- More Shapes!

- Comparable

Announcements

- Creative Project 2 (C2) due Thursday, November 13th

- Resubmission Cycle 5 (R5) out Thursday, November 13th
  - Eligible: **P1**, P2

- Programming Assignment 3 (P3) out Friday!
  - Due November 20th by 11:59 PM

- Quiz 2 Thursday, November 20th

  - Practice Quiz coming out before then!

- Reminder on Final Exam: **Monday, December 8th 12:30 PM - 2:20 PM**

Lecture Outline

- Announcements

- **Interfaces Review**

- More Shapes!

- Comparable

Recall: Abstraction

VS

A Scanner gets text input and turns it into data.

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.

```
List<String> myList = new ArrayList<>();
```

Interfaces

Interfaces serve as a sort of "contract"– in order for a class to <u>implement</u> 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 <u>must</u> include <u>all</u> these

methods (and any others the `List` interface specifies)

Interfaces vs. Implementation

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

`List` is an interface

    `ArrayList` is a <u>class</u> that <u>implements</u> the `List` interface

    `LinkedList` is a <u>class</u> that <u>implements</u> 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
- Brita
- Tub or
- <u>Any</u> 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
- <u>Any</u> other class that implements Set and whose element type is String!

Why interfaces?

Abstraction

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

UNIVERSITY *of* WASHINGTON

Lecture Outline

- Announcements

- Interfaces Review

- **More Shapes!**

- Comparable

Classes can Implement <u>Multiple</u> Interfaces

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

If a class implements an interface A <u>and</u> 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 <u>Multiple</u> Interfaces

```java
public interface UWMerch {
    public String getTypeOfMerch();
}



public class WaterBottle implements Container, UWMerch {
    ...
    public String getTypeOfMerch() {
        return "bottle";
    }
}
```

But `WaterBottle` would have to implement:

 - `fill` from `Container` <u>*AND*</u> `getTypeOfMerch` from `UWMerch`

An interface can extend another

You can have one interface <u>extend</u> another

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

An interface can extend another

We can write another interface:
    Polygon that extends Shape
    - "Polygon is a 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)

UNIVERSITY *of* WASHINGTON

Lecture Outline

- Announcements

- Interfaces Review

- More Shapes!

- **Comparable**

Recall the **Student** / **Course** Example from Wed

Course stored a field

```java
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 <u>one</u> required method:
    `public int compareTo(E other)`

Its return value is:

    `< 0` if `this` is "less than" other

      `0` if `this` is <u>equal</u> to other

    `> 0` if `this` is "greater than" other