Talk to your neighbors:

What is your most-used emoji?
Lecture Outline

• Announcements

• Interfaces Review

• More Shapes!

• Comparable
Announcements

• P3 will be released today
  • Due Sunday, August 13 at 11:59pm
  • JUnit will be covered on Wednesday
• Quiz 2 on Monday, August 7
  • Interfaces section will be released early for extra practice
• Reminder that the final exam is scheduled for Wednesday + Friday (Aug 16 + Aug 18) at 10:50 – 11:50 am
Lecture Outline

• Announcements

• Interfaces Review

• More Shapes!

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

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

*note: interfaces say nothing about a class’ state*
List Interface

List is an interface – defines the behaviors which make something a List, inc:

- add
- clear
- contains
- get
- isEmpty
- size

Any class with these behaviors can implement List

List documentation enumerates the full list of methods required to be a List: https://docs.oracle.com/javase/8/docs/api/java/util/List.html
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
...
(PCM) Why interfaces?

Abstraction

Interfaces support *abstraction*

(the separation of ideas from details)
**Why interfaces?**

Car interface
- Press gas pedal
- Press brake pedal
- Turn steering wheel

\[\text{implements} \quad \text{implements} \]
Why interfaces?

Flexibility

```java
public static void driveToWork(Car c) {...}
```

This method does not need to change if we update our implementation of Car.
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 \textit{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

```java
public interface Company {
    public String getName();
    public String getMissionStatement();
}

public class Square implements Shape, Company {
    ...
}

But Square would have to implement:
- `getPerimeter, getArea` from Shape
  *AND*
- `getName, getMissionStatement` from Company
An interface can extend another

You can have one interface *extend* another

So if `public interface A extends B`, then to implement A a class needs to have all methods from A and B.

In the above example, A is *more specific* than B.
An interface can extend another

We can write another interface

Polygon that extends Shape

- Square is a Polygon (and Shape)
- Triangle is a Polygon (and Shape)
- Circle is a Shape (but not a Polygon)
Select all of the following statements that would cause an error.

A) B foo = new Two();
   foo.b();

B) D bar = new Two();
   bar.a();

C) D baz = new Three();
   baz.a();

D) A waldo = new Three();
   waldo.b();
Select all of the following statements that would cause an error.

```java
public interface A {
    public void a();
}

public interface B extends A {
    public void b();
}

public interface C {
    public void c();
}

public interface D extends A {
    public void d();
    public void e();
}

public class One implements A {
    ...
}

public class Two implements B, D {
    ...
}

public class Three implements B, C {
    ...
}

A) B foo = new Two();
   foo.b();

B) D bar = new Two();
   bar.a();

C) D baz = new Three();
   baz.a();

D) A waldo = new Three();
   waldo.b();
```
Select all of the following statements that would cause an error.

```java
public interface A {
    public void a();
}

public interface B extends A {
    public void b();
}

public interface C {
    public void c();
}

public interface D extends A {
    public void d();
    public void e();
}
```
Select all of the following statements that would cause an error.

public class One implements A {
    ...  
}
public class Two implements B, D {
    ... 
}
public class Three implements B, C {
    ... 
}
Select all of the following statements that would cause an error.

A) B foo = new Two();
   foo.b();

B) D bar = new Two();
   bar.a();

C) D baz = new Three();
   baz.a();

D) A waldo = new Three();
   waldo.b();
What will each of the following lines of code output? (or DNC)

```java
public interface A {
    public void a();
}

public interface B {
    public void b();
}

public class One implements A {
    public void a() {
        System.out.println("hi");
    }
}

public class Two implements B {
    public void a() {
        System.out.println("howdy");
    }
    public void b() {
        System.out.println("bye");
    }
}

public class Three implements A, B {
    public void a() {
        System.out.println("hey");
    }
    public void b() {
        System.out.println("cya");
    }
}

A a = new One();
B b = new Two();
A a2 = new Three();
B b2 = new Three();
a.a();
b.b();
a2.a();
a2.b();
b2.a();
b2.b();
```
Lecture Outline

• Announcements
• Interfaces Review
• More Shapes!
  • Comparable
Recall the Song / Artist Example

Course stored a field

```java
private List<Song> songs;
```

We also had a suggestion to use a Set to store the songs...

Seems like a great idea (no duplicates, not worried about keeping a specific order or indexing into it) but ... 
• HashSet won’t work because of the hashCode() business I mentioned on Wed
• TreeSet won’t work because what does it mean to “sort” Song
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

Comparable documentation:

https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html