

LEC 14

CSE 122

Interfaces

BEFORE WE START

Chat with neighbors:

*If you could be an inanimate object,
which one would you be and why?*

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

Instructor: ~~Elba Garza~~ **Katharine**

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 #cse122a



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

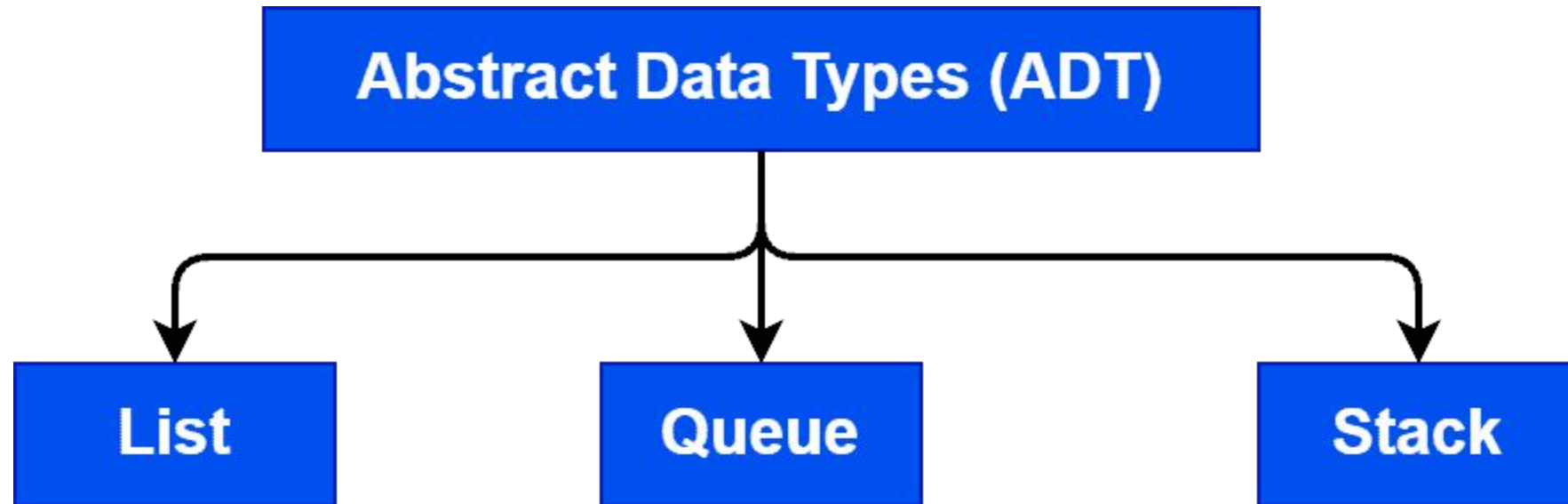
Recall: Abstraction

```
J Scanner.class X
1  /*
2   * Copyright (c) 2003, 2022, Oracle and/or its affiliates. All rights reserved.
3   * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
4   *
5   * This code is free software; you can redistribute it and/or modify it
6   * under the terms of the GNU General Public License version 2 only, as
7   * published by the Free Software Foundation. Oracle designates this
8   * particular file as subject to the "Classpath" exception as provided
9   * by Oracle in the LICENSE file that accompanies this code.
10  *
11  * This code is distributed in the hope that it will be useful, but WITHOUT
12  * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
13  * FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
14  * version 2 for more details (a copy is included in the LICENSE file that
15  * accompanied this code).
16  *
17  * You should have received a copy of the GNU General Public License version
18  * 2 along with this work; if not, write to the Free Software Foundation,
19  * Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
20  *
21  * Please contact Oracle, 500 Oracle Parkway, Redwood Shores, CA 94065 USA
22  * or visit www.oracle.com if you need additional information or have any
23  * questions.
24  */
25
26 package java.util;
27
28 import java.io.*;
29 import java.math.*;
30 import java.nio.*;
31 import java.nio.channels.*;
32 import java.nio.charset.*;
33 import java.nio.file.Path;
34 import java.nio.file.Files;
35 import java.text.*;
36 import java.text.spi.NumberFormatProvider;
37 import java.util.function.Consumer;
38 import java.util.regex.*;
39 import java.util.stream.Stream;
40 import java.util.stream.StreamSupport;
41 import sun.util.locale.provider.LocaleProviderAdapter;
42 import sun.util.locale.provider.ResourceBundleBasedAdapter;
43
44 /**
45  * A simple text scanner which can parse primitive types and strings using
46  * regular expressions.
47  *
48  * -p>A {@code Scanner} breaks its input into tokens using a
49  * -p>delimiter pattern, which by default matches whitespace. The resulting
50  * -p>tokens may then be converted into values of different types using the
51  * -p>various {@code next} methods.
52  *
53  * -p>For example, this code allows a user to read a number from
54  * -p>the console.
55  * -p>{@snippet :
56  * -p>var con = System.console();
```

VS

A Scanner gets
text input and
turns it into data.

Remember ADTs?



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 implement an interface, it must fulfill the contract requirements.

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

building noun

build·ing ('bil-dɪŋ)

[Synonyms of building >](#)

1 : a usually roofed and walled structure **built** for permanent use (as for a dwelling)

Our “ADT”



Our
“Interface”



Our
“Implementation”

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! 🏆

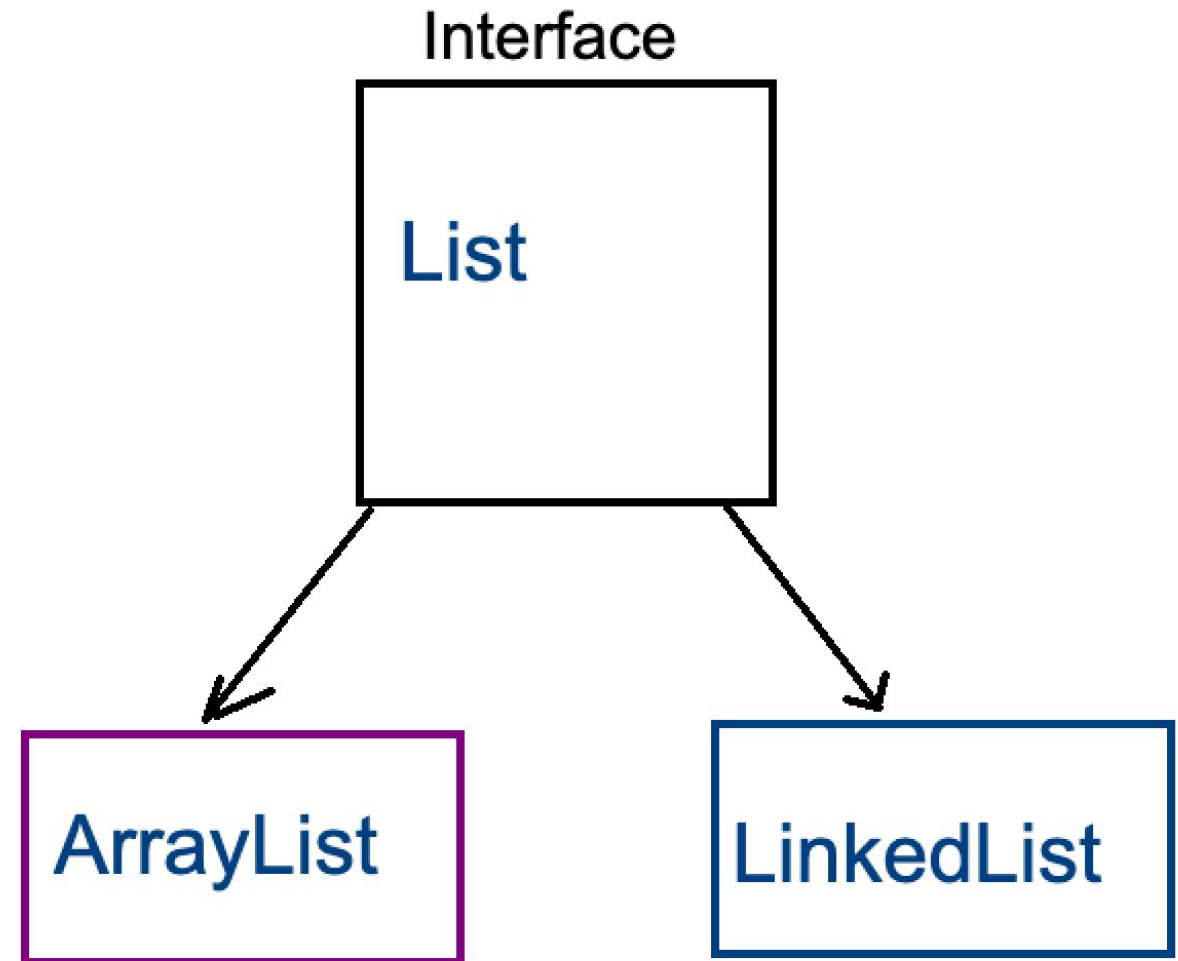
Interfaces vs. Implementation

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
- Brita
- 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!**  [Ed lesson link](#)
- 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 UWMerch {  
    public String getMaterial();  
}
```

```
public class WaterBottle implements Container, UWMerch {  
    ...  
    public String getMaterial() {  
        return this.material;  
    }  
}
```

But WaterBottle would have to implement:

- fill from Container AND getMaterial from UWMerch

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

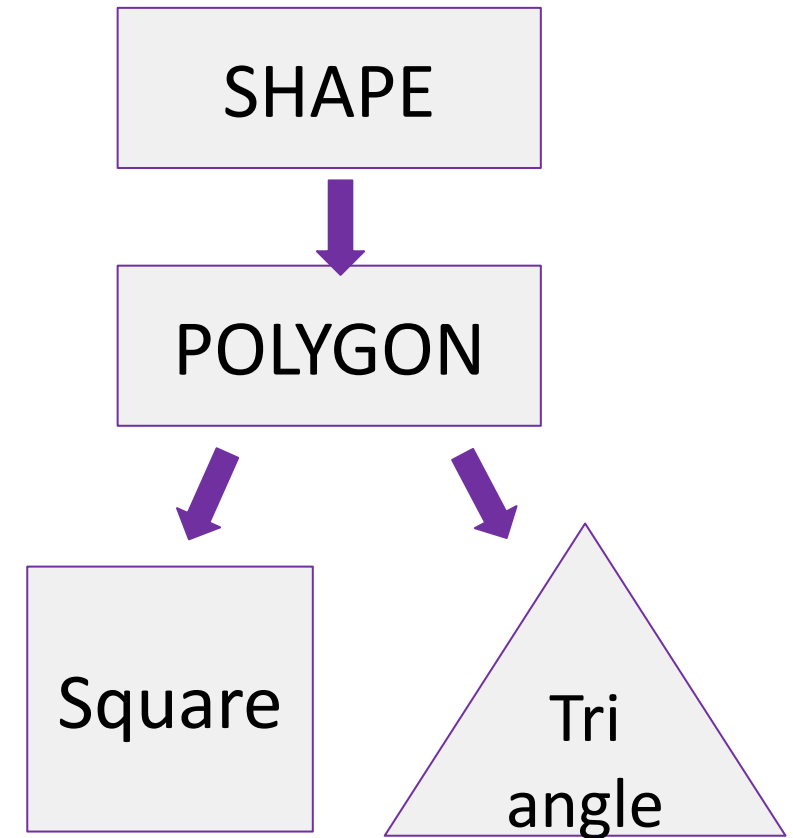
We can write another interface:

Polygon that extends **Shape**

- “**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**)



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