

LEC 13

CSE 122

Object-Oriented Programming Wrap-Up

Questions during Class?

Raise hand or send here

sli.do #cse122




BEFORE WE START

*Slido vote & chat with neighbors:**What is your favorite kind of bread?*Music: [122 26sp Lecture Jams](#) **Instructor:** Elbas Garza

TAs:	David William Dani Rohan Andrew Ava Shreyank Nicole	Caleb Neha Wesley Isis Colin Naomi Hanna Blake P.	Cole Blake R. Carson Sushma Connor Mahima Nicolae Ivory	Yang Cady Diya Elsa Garza
-------------	--	--	--	------------------------------------

Lecture Outline

- **Announcements** 
- Item Class Review
- toString
- compareTo
- Interfaces

Announcements

- Twitter Trends (C2) released and due May 21st
- Resubmission Cycle 4 (R4) due May 19th!
 - **C1**, P1 eligible for resub
- Quiz 2 on Tuesday, May 26th! (Nested Collections, Objects, and Interfaces)

Lecture Outline

- Announcements
- **Item Class Review** ◀
- toString
- compareTo
- Interfaces

Lecture Outline


- Announcements
- Item Class Review
- **toString** ◀
- compareTo
- Interfaces

toString

```
public String toString() {  
    return "String representation of object";  
}
```

The `toString()` method is automatically called whenever an object is treated like a `String`!

Lecture Outline

- Announcements
- Item Class Review
- toString
- **Interfaces** 

Recall the Item Class Fields

Item has the fields

```
private String name;  
private double price;  
private int quantity;
```

Recall the Item Class Fields

Item has the fields

```
private String name;  
private double price;  
private int quantity;
```

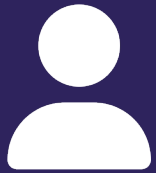
What if we wanted to sort out items? What does it mean for an item to come before another?

compareTo

```
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



Practice : Think

[sli.do](#)[#cse122](#)

Given the following compareTo and ShoppingCarts, what is the expected order?

```
public int compareTo(ShoppingCart other)
    if (capacity == other.capacity) {
        return items.size() - other.items.size();
    }
    return capacity - other.capacity;
}
```

...

```
ShoppingCart c1 = new ShoppingCart(4);
c1.addItem(new Item("Hot Dog", "1.50"));
c1.addItem(new Item("Soda", ".79"));
ShoppingCart c2 = new ShoppingCart(4);
c2.addItem(new Item("Salad", "3.99"));
c2.compareTo(c1);
```

A. c1, c2

B. c2, c1

C. Tied (both are equal)

D. Exception



Practice : Pair

sli.do

#cse122

Given the following compareTo and ShoppingCarts, what is the expected order?

```
public int compareTo(ShoppingCart other)
    if (capacity == other.capacity) {
        return items.size() - other.items.size();
    }
    return capacity - other.capacity;
}
```

...

```
ShoppingCart c1 = new ShoppingCart(4);
c1.addItem(new Item("Hot Dog", "1.50"));
c1.addItem(new Item("Soda", ".79"));
ShoppingCart c2 = new ShoppingCart(4);
c2.addItem(new Item("Salad", "3.99"));
c2.compareTo(c1);
```

A. c1, c2

B. c2, c1

C. Tied (both are equal)

D. Exception

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.

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)
```