LEC 11

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?

Instructor: Ivory & Colin

TAs: Connor
Yang
Dani
Rohan
Wesley

Lecture Outline (1/5)

- Announcements
- toString
- compareTo
- equals
- Interfaces

Announcements

- Twitter Trends (C2) releasing today and due Aug 13th
- Resubmission Cycle 4 (R4) due Aug 11th!
  - C1, P1 eligible for resub
- Quiz 2 on Tuesday, Aug 13th! (Nested Collections, Objects, and Interfaces)
- Finals on Aug 19th and Aug 21st in our normal lecture rooms!

Lecture Outline (2/5)

- Announcements
- **toString**
- equals
- 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 (3/5)

- Announcements
- toString
- equals
- compareTo
- Interfaces

Equals

The `equals()` method returns `true` if the given parameter is considered equal to this object, and `false` otherwise.

- Used by lots of library methods! e.g. `contains`, `remove` for specific elements, etc.

Let's do it!

Each class has one provided by Java, but it checks for **reference equality**. (Thanks?)

If you want equals to check for **value equality**, you need to write this method yourself.

Object

By taking a parameter of type `Object`, the equals method can be passed any type of object.

More to come in CSE 123 on the Java mechanisms that make this work!

We can use the `instanceof` keyword in Java to determine if the parameter is actually a `ShoppingCart`

Almost there…

This is actually **still an imperfect implementation** because we would also need to write a `hashCode()` method for our object to work with `HashSet`, `HashMap`, etc. but more to come on that in CSE 331 and beyond ☺

Lecture Outline (4/5)

- Announcements
- toString
- equals
- compareTo
- 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. Positive

B. Negative

C. Zero

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. Positive

B. Negative

C. Zero

D. Exception

Lecture Outline (5/5)

- Announcements
- toString
- equals
- compareTo
- **Interfaces**

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 <u>one</u> required method:

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