

LEC 12

CSE 122**Advanced OOP****BEFORE WE START***Talk to your neighbors:**If you could eliminate one thing from your daily routine, what would it be & why?***Instructors Tristan Huber & Hunter Schafer****TAs**

Ambika
Andrew
Audrey
Autumn
Ayush
Ben
Colton
Di
Eesha
Elizabeth

Evelyn
Jacob
Jaylyn
Jin
Joe
Kevin
Leon
Megana
Melissa
Mia

Poojitha
Rishi
Rucha
Shivani
Shreya
Steven
Suhani
Yijia
Ziao


Questions during Class?

Raise hand or send here

sli.do #cse122



Lecture Outline

- **Announcements** 
- Equals
- Bigger Example
- Constructors (cont.)
- For next quarter...

Announcements

- P2 due tomorrow (Thursday May 11)
- C2 will be released on Friday (Friday May 12)
- Quiz 2 next Tuesday
 - Objects, Maps, Nested Collections
- Friday Lecture on *fancy datastructures* is optional

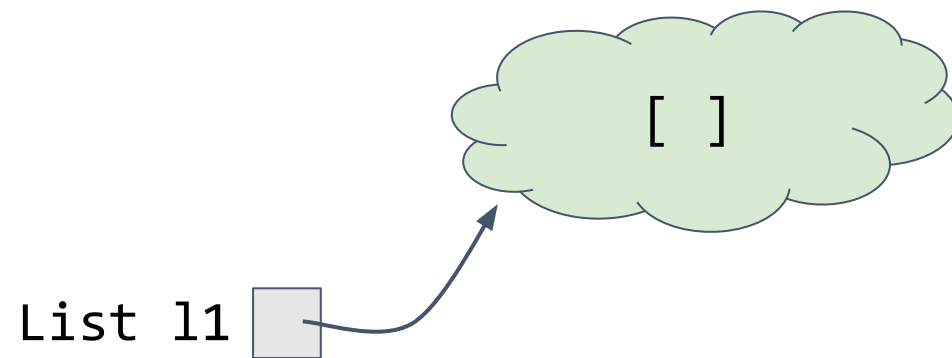
Announcements


- New templates for Ed Questions

Null/Empty Review

```
List<Integer> l1 = new ArrayList<>();
```

```
List<Integer> l2 = null;
```



List l1 



Practice : Think



sli.do

#cse122

Null/Empty Review

```
List<Integer> l1 = new ArrayList<>();
```


```
List<Integer> l2 = null;
```

```
l1.add(5);
```

```
l2.add(4);
```

What is the result of running this code?

Lecture Outline

- Announcements
- **Equals** 
- Bigger Example
- Constructors (cont.)
- For next quarter...

(PCM) Equals

Like toString java gives us equals() “for free” on all our classes.
but....

default .equals() does *reference equality*.

(same as ==)

Usually we want *value equality*

so.. write our own .equals() method

(PCM) 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! (`contains`, `remove` for specific elements, etc.)

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 `Point`

(PCM) Equals


```
public boolean equals(Object other) {  
    if (other == this) {  
        return true;  
    } if (other instanceof Point) {  
        Point otherPoint = (Point) other;  
        return otherPoint.x == this.x && otherPoint.y == this.y;  
    } else {  
        return false;  
    }  
}
```

Most of .equals is boiler plate, change just the highlighted parts to re-implement for other types.

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

- Announcements
- Equals
- **Bigger Example** 
- Constructors (cont.)
- For next quarter...

Student class

Write a `Student` class that you can construct by saying `new Student(1234567, "Miya")` where the first parameter is their student number and the second parameter is their name. Your `Student` class should also implement the following methods:

- `getName()` returns the student's name
- `getStudentNumber()` returns the student's number
- `setName(String newName)` sets the student's name to the given newname
- `toString()` returns a `String` representation of the student formatted as `"name (studentNumber)"`
- `equals(Object other)` that returns `true` if the given parameter is considered equal to this object

Student class

What if we added a field to the Student class:

```
private boolean isMale;
```

You are the *designer* now. Think carefully about what assumptions you are making!

Why shouldn't we include a `setStudentNumber` method?

Course class

Write a Course class that represents a course at UW. Implement the following methods and constructors:

Constructors

- Write a constructor so that you can construct a Course by saying `new Course(23213, "CSE 122", 4)` where the first parameter is the course's SLN, the second parameter is the code for the course, and the third parameter is the number of credits.
- Write another constructor so that you can construct a Course by saying `new Course(23239, "CSE 122", 4, enrollment)` where the first parameter is the course's SLN, the second parameter is the code for the course, the third parameter is the number of credits, and the fourth parameter is a `Student[]` containing a Student for each student enrolled in the course.

Course class

Instance Methods

- `updateRoster(Student[] students)` replaces the current roster with the content of the given students
- `addStudent(Student s)` adds the given student to the roster if they are not already on it
- `dropStudent(Student s)` removes the given student from the roster if they are on it
- `checkStudentEnrolled(Student s)` returns true if the given student is on the current roster, and false otherwise
- `getSLN()` returns the course's SLN
- `getCourseCode()` returns the course's code
- `getCredits()` returns the number of credits for the course
- `getRoster()` returns a reference to the course's roster

Course class

Instance Methods

- `updateRoster(Student[] students)` replaces the current roster with the content of the given students
- `addStudent(Student s)` adds the given student to the roster if they are not already on it
- `dropStudent(Student s)` removes the given student from the roster if they are on it
- `checkStudentEnrolled(Student s)` returns true if the given student is on the current roster, and false otherwise
- ...

Lecture Outline

- Announcements
- **Constructors (cont.)** ◀
- Equals
- Bigger Example
- For next quarter...

OOP Principles Recap

Class - the java code which defines a type of object

Object - an instance of a class. Exists only when code is running.

OOP - Objects should *encapsulate state* and *expose behavior*

Encapsulation - state should be *private*. Access controlled through the objects behavior

(PCM) Constructors

```
public Point(int initialX, int initialY) {  
    x = initialX;  
    y = initialY;  
}
```

Principle: All fields should be initialized in the constructor(s)!

If we write *any* constructors, Java no longer provides one for us.

(PCM) this keyword

The `this` keyword refers to the current object in a method or constructor.

You can use it to refer to an object's fields

```
this.x, this.y
```


You can use it to refer to an object's instance methods

```
this.setX(newX)
```

You can use it to call one constructor from another

```
this(0, 0)
```

Lecture Outline

- Announcements
- Constructors (cont.)
- Equals
- Bigger Example
- **For next quarter...** 

If you want to...

- Learn more about programming techniques
 - Recursion!
- Learn about even more fundamental data structures!
 - And implement *your own* data structures
- Gain a stronger understanding of efficiency
- Pursue a software-intensive major and/or career

Consider taking...

CSE 123

- CSE 123 offered 23sp (Wortzman), 22su (Boinpally)

If you want to...

- Do something with data science
 - The world is run on decisions made from data. Data science requires processing large amounts of data collected to help people make decisions.
- Learn the programming concepts, libraries, and tools that make up the modern data science ecosystem.
 - Data programming = The programming that supports data science

Consider taking...

CSE 163 and other courses in the Data Science Minor & Option

- CSE 163 offered 23sp (Lin), 23su

If you want to...

- Build a website or web app
 - Either the frontend (what visitors see in their browser) or the backend (what runs on the server to compute data)
- Learn the fundamentals of a number of web technologies that make it easier for you to learn more on your own

Consider taking...

CSE 154, INFO 343, or INFO 344

- CSE 154 offered 23sp (Wolman), 23su