

LEC 13

**CSE 122****Advanced OOP**

BEFORE WE START

***Talk to your neighbors:****What are your plans over winter break?*Music: [122 24au Lecture Tunes](#) **Instructors:** Miya Natsuhara and Elba Garza

<b>TAs:</b>	Ayush	Heon	Harshitha	Aishah
	Andrew	Izak	Marcus	Ben
	Logan	Colin	Carson	Ivory
	Kyle	Jessica	Jack	Cady
	Maggie	Shivani	Connor	Diya
	Nicole H	Ken	Cora	Katharine
	Caleb	Mia	Hannah	
	Nicole W	Ashley	Leo	
	Jacob	Chaafen	Anya	


Questions during Class?

Raise hand or send here

sli.do #cse122




# Lecture Outline

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

# Announcements

- Creative Project 2 (C2) releasing later tonight
  - Focused on OOP!
- Resubmission Cycle 4 (R4) out; due Tuesday, Nov 12 by 11:59 PM
  - Eligible Assignments: **C1**, P1
- Thanksgiving Week Plans
  - Tuesday, Nov 26 – TA's Choice quiz section!
    - Not all sections will be held, but you are welcome to visit a quiz section different from your registered section
  - Wednesday, Nov 27 – Optional Lecture (non-122 material)

# Lecture Outline

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

# Constructor Syntax

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

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

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

# 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...

# 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.

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 `Point`

# Point's equals()



**Hunter Schafer** 1 minute ago



I also think it would be good to highlight the fact that every Java programmer and their mother just copies (or has memorized) that equals method template and the “real work” is filling in the middle case




# 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
- Constructors (cont.)
- Equals
- **Bigger Example** 
- 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;
```

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

Also...

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 copy of 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...** 

# 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 25wi (Natsuhara, Wortzman), 25sp (Brunelle)

# 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 25wi (Lin), 25sp (Lin)

# 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 likely offered 25su (?)