LEC 11

CSE 122

Instance Methods

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

What class are you most excited to take next quarter?

Respond on sli.do!

Instructor: Brett Wortzman

TAs:

Archit	Daniel	Naomi	Sreshta
Blake P.	David	Neha	Suyash
Blake R.	Hanna	Nicole	Trisha
Cady	Isis	Rio	Wesley
Carson	Jun	Rohan	Yash
Cole	Landon	Shreyank	
Connor	Michelle	Somya	

Music: 🎷 CSE 122 26au Lecture Tunes 🎶

Lecture Outline

- Announcements
- `toString`
- `equals`
- Bigger Example: Courses and Students

Announcements

- P2 out tonight, due Thursday 11/12
- PQ0 grades out this weekend
- PQ1 Tuesday 11/10
  - Practice quiz out already; solutions tomorrow
- No class Wednesday 11/11
  - Also no IPL, message board responses delayed

Shopping Cart Practice

- Add two new methods to the `ShoppingCart` class:
  - `spaceRemaining` returns the number of additional items the cart can hold
  - `removeByName` takes a name as a parameter and removes all items from the cart with that name

Lecture Outline

- Announcements
- `toString`
- `equals`
- Bigger Example: Courses and Students

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
- `toString`
- `equals`
- Bigger Example: Courses and Students

`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 a default implementation that checks for **reference equality**.

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`

ShoppingCart'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

👍 1

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
- `toString`
- `equals`
- **Bigger Example: Courses and Students**

Student class

Write a `Student` class that you can construct by saying:

```
new Student(1234567, "Cornbear")
```

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

A cautionary tale:



What if we added a field to the Student class:

```
private boolean isLocalStudent;
```

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

- `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