**LEC 01: Review; Comparable**

### CSE 123

**Review; Comparable!**

---

**BEFORE WE START**

*Talk to your neighbors:*

*Introduce yourself to your neighbor!*

*What is your name? Major? What have you been up to the past week?*

**Music:** [123 24su Lecture Tunes](#)

---

**Instructor:** Joe Spaniac

**TAs:** Andras, Eric, Sahej, Trien, Zach

---

Questions during Class?

Raise hand or send here

sli.do  #cse123
Lecture Outline

• Introductions

• About this Course
  - Course Components & Tools
  - Making the Most of this Class

• OOP / Junit Review

• Comparable
Course Staff

- Instructor: Joe Spaniac

- Teaching Assistants: 7 Fantastic TAs!
  - Available in section, office hours, and discussion board
  - Invaluable source of information & help in this course

- We’re excited to get to know you!
  - Our goal is to help you succeed 😊
What is this Class?

CSE 121 – Computer Programming I
- Data types (int, String, boolean)
- Methods / Functions
  - Parameters, Returns
- Control structures
  - Loops, Conditionals
- Arrays & 2D arrays
- Computational Thinking
  (language agnostic)

CSE 122 – Computer Programming II
- Functional Decomposition
- File I/O
- Using data structures
  - List, Stacks / Queues, Sets, Maps
- Object Oriented Programming
  - Interfaces

CSE 123 – Computer Programming III
- Advanced Object Oriented Programming
  - Comparable, Inheritance/Polymorphism, Abstract Classes
- Implementing data structures
  - ArrayLists, LinkedLists, Trees
- Recursion
- Critical analysis of design
Why 123?

1. To solve more complex problems by leveraging more complex programming structures / patterns

2. To better rationalize specific design decisions
   - How to “best” structure classes to reduce redundancy
   - Which ADT implementations are “most” appropriate to use

3. To understand and critically analyze intersections between Computer Science and society
   - Search engines, algorithmic art, machine learning, etc.
   - Developing informed opinions on current issues
Lecture Outline

• Introductions

• About this Course
  - Course Components & Tools
  - Making the Most of this Class

• OOP / Junit Review

• Comparable
Course Website

Contains most course info – check frequently!
Announcements, Calendar, Lecture Slides, Office Hours schedule, Staff Bios, Important Links

Get to know the staff

Course Website

cs.uw.edu/123
Other Course Tools

Ed
- Community & Information
  - Discussion Board (please ask & answer!; anonymous option)
  - Chat
  - Announcements
- Pre-Class Materials / Section Handouts
- Assignments
  - Online IDE
  - Submit assignments
  - View Feedback

My Digital Hand
- Queueing in office hours

VSCode
- Develop offline
- Visual debugger

Canvas
- Lecture recordings

Sli.do
- In-class activities (ungraded)
- No account needed
Lecture Outline

• Introductions

• **About this Course**
  - Course Components & Tools
  - *Making the Most of this Class*

• OOP / Junit Review

• Comparable
How Learning Works

• Learning requires **active participation** in the process. It’s not as simple as sitting and listening to someone talk at you.
  - Requires **deliberate practice** in **learning by doing**
  - Benefits from **collaborative learning**

• Hybrid classroom model
  - Asks you to do some preparation before class in the form of readings and practice problems.
    - Should take ~30 minutes a day
  - Class will start with brief recap, then pick up where the reading and practice problems leave off.
  - Attendance isn’t graded, but showing up and trying is the first step in succeeding in the class!

• Pre-class materials are ungraded, but...
  - It’s okay if you find them challenging! That means you are learning!
Getting Help

• Discussion Board
  - Feel free to make a public or private post on Ed
  - We encourage you to answer other peoples’ questions! A great way to learn

• Introductory Programming Lab (Office Hours)
  - TAs can help you face to face in office hours, and look at your code
  - You can go to the IPL with any course questions, not just assignments

• Section
  - Work through related problems, get to know your TA who is here to support you

• Your Peers
  - We encourage you to form study groups! Discord or Ed are great places to do that

• Email
  - We prefer that all content and logistic questions go on the Ed discussion board (even if you make them private). 80 of you >>> 8 of us!
  - For serious personal circumstances, you can email Joe directly. It never hurts to email us, but if it’s a common logistic question, we may politely ask you to post on the discussion board instead.
Lecture Outline

• Introductions
• About this Course
  - Course Components & Tools
  - Making the Most of this Class
• OOP / Junit Review
• Comparable
Lecture Outline

• Introductions
• About this Course
  - Course Components & Tools
  - Making the Most of this Class
• OOP / JUnit Review
• Comparable
Comparable

- Comparable\(<E>\) is an interface that allows implementers to define an ordering between two objects
  - Used by TreeSet, TreeMap, Collections.sort, etc.

- One required method:
  
  ```
  public int compareTo (E other);
  ```

- Returned integer falls into 1 of 3 categories
  
  ```
  a.compareTo(b);
  ```
  this     other

  - < 0: this is “less than” other
  - = 0: this is “equal to” other
  - < 0: this is “greater than” other
Subtraction Trick

- `compareTo` implementation when comparing two integers (a) ascending:

  ```java
  if (this.a < other.a)  // negative number
    -> negative number
  else if (this.a > other.a)  // positive number
    -> positive number
  else
    -> 0
  ```

- This is just subtraction!

  ```java
  this.a - other.a
  ```

- What if we wanted to sort descending?

  ```java
  other.a - this.a
  ```

- **Warning**: this only works for integers! Doubles have issues with truncation.