Lecture 1: Welcome to CSE 373

Data Structures and Algorithms
Agenda

- Introductions
- Syllabus
- Dust off data structure cob webs
- Meet the ADT
Waitlist/ Overloads

- I have no control over these things :/
- Email cse373@cs.washington.edu for all registration requests/questions
- Many students move around, likely a spot will open
- Keep coming to lecture!
Hello!

I am Kasey Champion
Software Engineer @ Microsoft
High School Teacher @ Franklin High
champk@cs.washington.edu
Office Hours: Fridays 2:30pm-5:30pm

@techie4good
Class Style

Kasey has to go to “real work” after this
- The internets
- Your TAs
- Each other

Please come to lecture
- Collaboration
- Demos
- Ask questions! Point out mistakes!

Sections
- TAs = heroes
- Practice problems
- Sections start this week
Course Administration

Course Page
- All course content/announcements posted here
- Pay attention for updates!

Canvas
- Grades will be posted here

Office Hours
- Will be posted on Course Page
- Will start next week

Piazza
- Great place to discuss questions with other students
- Will be monitored by course staff
- No posting of project code!

Textbook
- Optional!
- Data Structures and Algorithm Analysis in Java by Mark Allen Weiss
Grade Break Down

Homework (65%)
- 4 Projects (50%)
  - Partners encouraged
- 3 Written Assignments (15%)
  - Must be individual

Exams (35%)
- Midterm Exam – Friday April 27\textsuperscript{th} in class (15%)
- Final Exam – Tuesday June 5\textsuperscript{th} 8:30-10:30 here! (20%)
Syllabus
Questions?

Clarification on syllabus, General complaining/moaning
### What is this class about?

**CSE 143 – OBJECT ORIENTED PROGRAMMING**
- Classes and Interfaces
- Methods, variables and conditionals
- Loops and recursion
- Linked lists and binary trees
- Sorting and Searching
- \(O(n)\) analysis
- Generics

**CSE 373 – DATA STRUCTURES AND ALGORITHMS**
- Design decisions
- Design analysis
- Implementations of data structures
- Debugging and testing
- Abstract Data Types
- Code-base development
Data Structures and Algorithms

What are they anyway?
Basic Definitions

Data Structure
- A way of organizing and storing related data points
- Examples from CSE 14X: arrays, linked lists, stacks, queues, trees

Algorithm
- A series of precise instructions used to perform a task
- Examples from CSE 14X: binary search, merge sort, recursive backtracking
Review: Clients vs Objects

CLIENT CLASSES

A class that is executable, in Java this means it contains a Main method

```java
public static void main(String[] args)
```

OBJECT CLASSES

A coded structure that contains data and behavior

Start with the data you want to hold, organize the things you want to enable users to do with that data
Abstract Data Types (ADT)

Abstract Data types
- A definition for expected operations and behavior

Start with the operations you want to do then define how those operations will play out on whatever data is being stored

*Review:* List - a collection storing an ordered sequence of elements

- each element is accessible by a 0-based index
- a list has a size (number of elements that have been added)
- elements can be added to the front, back, or elsewhere
- in Java, a list can be represented as an ArrayList object
**Review: Interfaces**

**interface**: A list of methods that a class promises to implement.

- Interfaces give you an is-a relationship *without* code sharing.
  - A Rectangle object can be treated as a Shape but inherits no code.
- Analogous to non-programming idea of roles or certifications:
  - "I'm certified as a CPA accountant. This assures you I know how to do taxes, audits, and consulting."
  - "I'm 'certified' as a Shape, because I implement the Shape interface. This assures you I know how to compute my area and perimeter."

```java
public interface name {
    public type name(type name, ..., type name);
    public type name(type name, ..., type name);
    ...
    public type name(type name, ..., type name);
}
```

**Example**

```java
// Describes features common to all shapes.
public interface Shape {
    public double area();
    public double perimeter();
}
```

Diagram:

```
<Interface>
  Shape
    area()
    perimeter()
  Circle
    radius
    area(radius)
    perimeter()
  Rectangle
    width, height
    area(width, height)
    perimeter()
  Triangle
    a, b, c
    area(a, b, c)
    perimeter()
```
Review: Java Collections

Java provides some implementations of ADTs for you!

You used:

- **Lists**
  ```java
  List<Integer> a = new ArrayList<Integer>();
  ```

- **Stacks**
  ```java
  Stack<Character> c = new Stack<Character>();
  ```

- **Queues**
  ```java
  Queue<String> b = new LinkedList<String>();
  ```

- **Maps**
  ```java
  Map<String, String> d = new TreeMap<String, String>();
  ```

But some data structures you made from scratch... why?

- **Linked Lists** - LinkedList was a collection of ListNode

- **Binary Search Trees** – SearchTree was a collection of SearchTreeNodeNodes
Abstract Data Type (ADT)
- A definition for expected operations and behavior
- A mathematical description of a collection with a set of supported operations and how they should behave when called upon
- Describes what a collection does, not how it does it
- Can be expressed as an interface
- Examples: List, Map, Set

Data Structure
- A way of organizing and storing related data points
- An object that implements the functionality of a specified ADT
- Describes exactly how the collection will perform the required operations
- Examples: LinkedList, ArrayIntList
List of ADTs

- List
- Set
- Map
- Stack
- Queue
- Priority Queue
- Graph
Case Study: The List ADT

**list:** stores an ordered sequence of information.
- Each item is accessible by an index.
- Lists have a variable size as items can be added and removed

**Supported Operations:**
- `get(index):` returns the item at the given index
- `set(value, index):` sets the item at the given index to the given value
- `append(value):` adds the given item to the end of the list
- `insert(value, index):` insert the given item at the given index maintaining order
- `delete(index):` removes the item at the given index maintaining order
- `size():` returns the number of elements in the list
Case Study: List Implementations
TODO list

Skim through full Syllabus on class web page
Sign up for Piazza
Review 142/143 materials. Materials provided on class webpage