Introduction

CSE 373
Data Structures & Algorithms
Linda Shapiro
Spring 2013

Today's Outline

- Introductions
- Administrative Info
- What is this course about?

Staff

- Professor
  - Linda Shapiro, (shapiro@cs.washington.edu)
- TA's
  - Richard Kedziorski, (kedzior@cs.washington.edu)
  - Jacob Gile, (jjgile at@cs.washington.edu)
  - Daphna Khen, (khend@cs.washington.edu)
  - Gene Kim, (genelkim@cs.washington.edu)
  - Adam Nelson, (ace91@cs.washington.edu)
  - Sam Wilson, (samw11@cs.washington.edu)
  - Hang Yin, (yinh@cs.washington.edu)

Me (Linda Shapiro)

- Taught Computer Science and Electrical Engineering at the University of Washington for 27 years
- Taught Computer Science at Virginia Tech and Kansas State Universities before that.
- Research: Computer Vision, Pattern Recognition, Image Databases, Biomedical Imaging & Informatics
- Recently Taught: computer vision, artificial intelligence, medical imaging
- Taught since 1974: Data Structures
- Taught once: ENGR 100

Today's Outline

- Introductions
- Administrative Info
- What is this course about?

Web Page

- All info is on the web page for CSE 373
  - http://www.cs.washington.edu/373
  - also known as
    - http://www.cs.washington.edu/education/courses/373/13sp
- Look there for schedules, contact information, assignments, links to discussion boards and mailing lists, etc.

Office Hours

- Linda Shapiro– 634 CSE (Allen Center)
  - › Monday & Wednesday 3:30-4:30pm, or by appointment
- TAs office hours will be posted.
- 390 Sections will be posted and available for one credit.

CSE 373 E-mail List

- If you are registered for the course, you will be automatically subscribed.
- The E-mail list is used for posting announcements by instructor and TAs.
- You are responsible for anything sent here

CSE 373 Discussion Board

- The course will have a Catalyst Go-Post message board
- Use for:
  - › General discussion of class contents
  - › Hints and ideas about assignments (but **not** detailed code or solutions)
  - › Other topics related to the course.

Computer Lab

- College of Arts & Sciences Instructional Computing Lab
  - › http://depts.washington.edu/aslab/
- We'll be using Java for the programming assignments.
- Eclipse is the recommended (not required) programming environment.

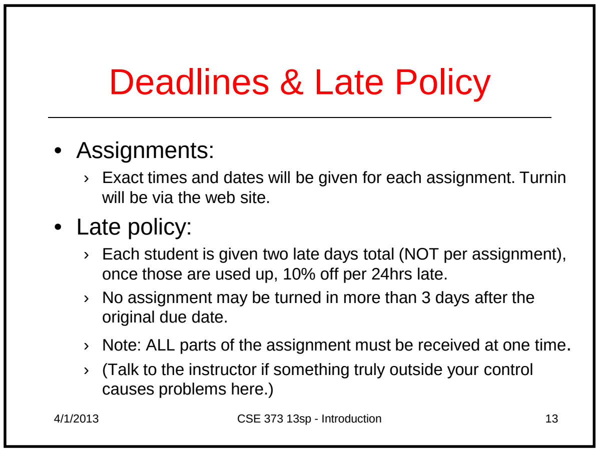
Textbook

- *Data Structures and Algorithm Analysis in Java*, by Mark Allen Weiss, 3rd edition, Addison-Wesley, 2012.

- We will also try to support the 2nd edition (2007).

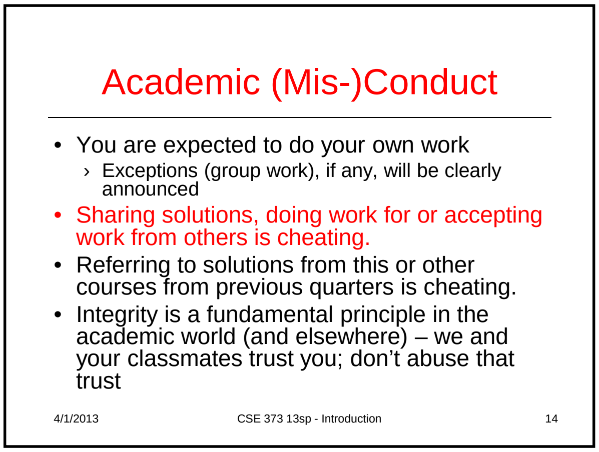
Grading - Estimated Breakdown:

- Assignments 50%
  - › Weights may differ to account for relative difficulty of assignments
  - › Assignments will be a mix of shorter written exercises and longer programming projects
- Midterm 20% (probably May 3)
- Final Exam 30%
  - › 2:30-4:20pm Tuesday, June 11, 2013.



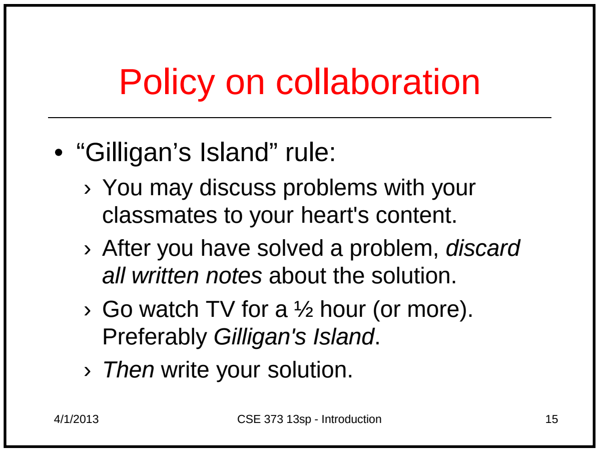
Deadlines & Late Policy

- Assignments:
  - › Exact times and dates will be given for each assignment. Turnin will be via the web site.
- Late policy:
  - › Each student is given two late days total (NOT per assignment), once those are used up, 10% off per 24hrs late.
  - › No assignment may be turned in more than 3 days after the original due date.
  - › Note: ALL parts of the assignment must be received at one time.
  - › (Talk to the instructor if something truly outside your control causes problems here.)



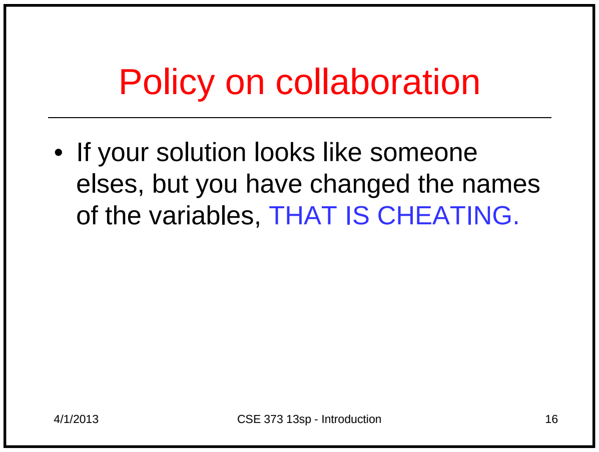
Academic (Mis-)Conduct

- You are expected to do your own work
  - › Exceptions (group work), if any, will be clearly announced
- Sharing solutions, doing work for or accepting work from others is cheating.
- Referring to solutions from this or other courses from previous quarters is cheating.
- Integrity is a fundamental principle in the academic world (and elsewhere) – we and your classmates trust you; don’t abuse that trust



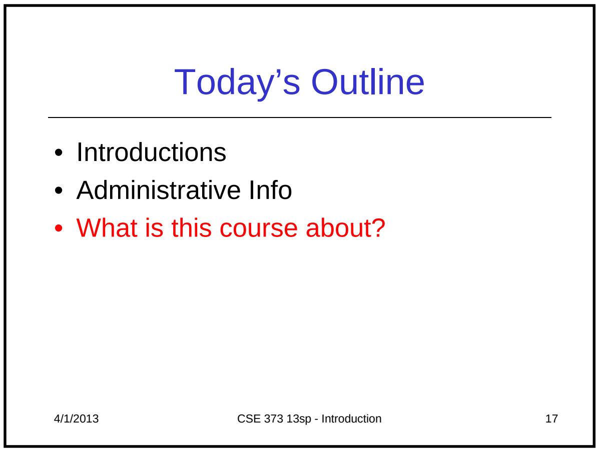
Policy on collaboration

- “Gilligan’s Island” rule:
  - › You may discuss problems with your classmates to your heart's content.
  - › After you have solved a problem, *discard all written notes* about the solution.
  - › Go watch TV for a ½ hour (or more). Preferably *Gilligan's Island*.
  - › *Then* write your solution.



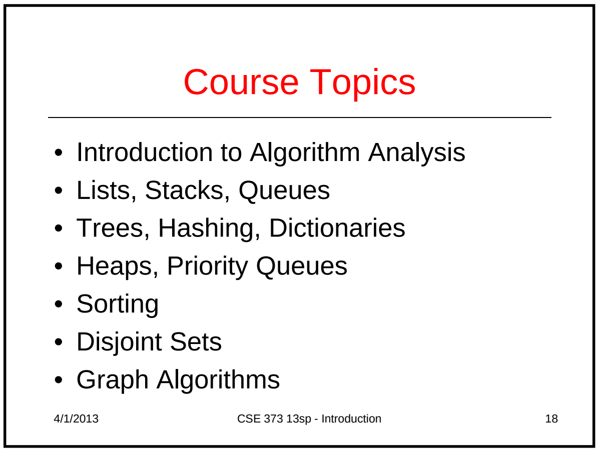
Policy on collaboration

- If your solution looks like someone elses, but you have changed the names of the variables, THAT IS CHEATING.



Today’s Outline

- Introductions
- Administrative Info
- What is this course about?



Course Topics

- Introduction to Algorithm Analysis
- Lists, Stacks, Queues
- Trees, Hashing, Dictionaries
- Heaps, Priority Queues
- Sorting
- Disjoint Sets
- Graph Algorithms

Background

- Prerequisite is CSE 143 and you need Java
- Topics you should have a basic understanding of:
  - › variables, conditionals, loops, methods
  - › fundamentals of defining classes and inheritance
  - › arrays, singly linked lists, simple binary trees
  - › recursion
  - › some sorting and searching algorithms
  - › basic algorithm analysis (e.g., O(n) vs O(n^2))

What is 373 about?

- Introduction to many of the basic data structures and algorithms used in computer software:
  - › Understand the data structures and the **trade-offs** they make
  - › Rigorously **analyze** the algorithms that use them (math!)
  - › Learn how to **pick "the right data structure for the job"**
  - › More thorough and rigorous take on topics introduced in CSE 143 (plus more new topics)
  - › Applications
- Practice design and analysis of data structures/algorithms
- Practice implementing and using these data structures by writing programs

Goals

- You will understand:
  - › what the tools are for storing and processing common data types
  - › which tools are appropriate for which need
- So that you will be able to:
  - › make good design choices as a developer, project manager, or system customer
  - › justify and communicate your design decisions

What is a data structure?

- A way to *organize information* in order to enable *efficient* computation over that information.
- What data structures have ***you*** used?

Data structures!

A data structure supports certain *operations*, each with a:

- › **Meaning**: what does the operation do/return?
- › **Performance**: how efficient is the operation?

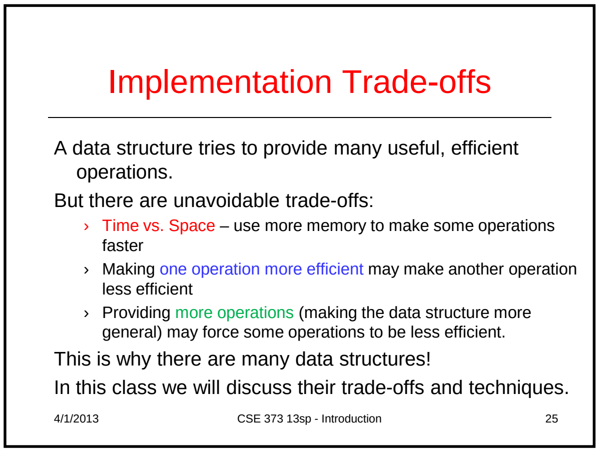
Examples:

- › ***List*** with operations **`insert`** and **`delete`**
- › ***Stack*** with operations **`push`** and **`pop`**

Picking the best data structure

Things we care about:

- Does this data structure support the operations I need?
  - › e.g. find an item quickly, insert in any location, print in sorted order, delete?
- Does it support them in an ***efficient*** manner?
  - › Time (Speed)
  - › Space (Memory)
- How easy will it be to implement, debug, and test it?



Implementation Trade-offs

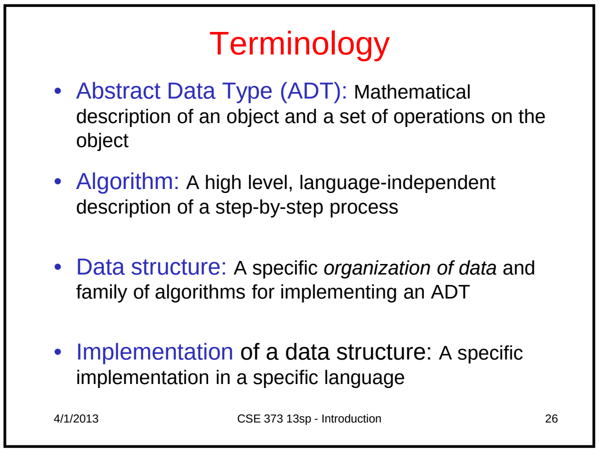
A data structure tries to provide many useful, efficient operations.

But there are unavoidable trade-offs:

- Time vs. Space – use more memory to make some operations faster
- Making one operation more efficient may make another operation less efficient
- Providing more operations (making the data structure more general) may force some operations to be less efficient.

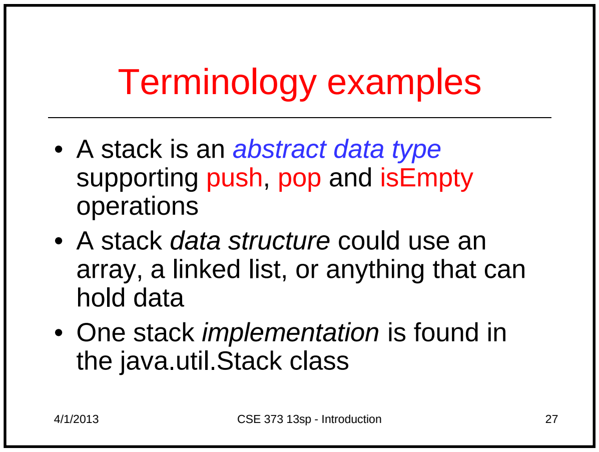
This is why there are many data structures!

In this class we will discuss their trade-offs and techniques.



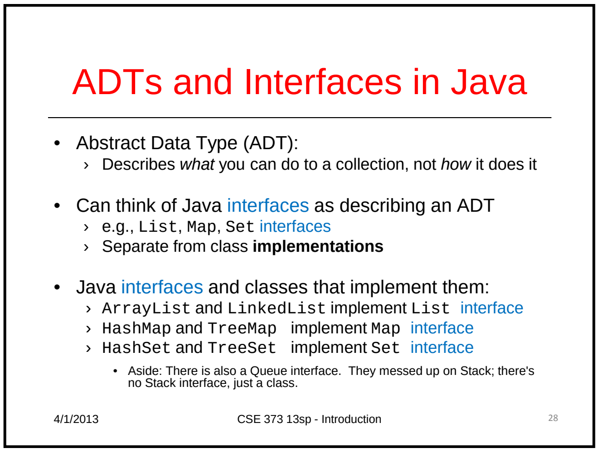
Terminology

- Abstract Data Type (ADT): Mathematical description of an object and a set of operations on the object
- Algorithm: A high level, language-independent description of a step-by-step process
- Data structure: A specific *organization of data* and family of algorithms for implementing an ADT
- Implementation of a data structure: A specific implementation in a specific language



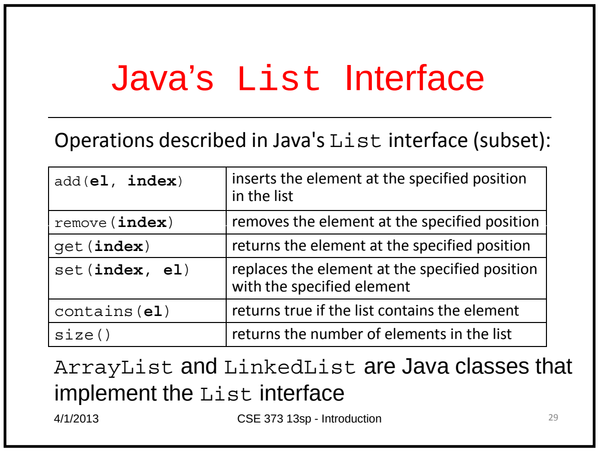
Terminology examples

- A stack is an *abstract data type* supporting push, pop and isEmpty operations
- A stack *data structure* could use an array, a linked list, or anything that can hold data
- One stack *implementation* is found in the java.util.Stack class



ADTs and Interfaces in Java

- Abstract Data Type (ADT):
  - Describes *what* you can do to a collection, not *how* it does it
- Can think of Java interfaces as describing an ADT
  - e.g., `List`, `Map`, `Set` interfaces
  - Separate from class **implementations**
- Java interfaces and classes that implement them:
  - `ArrayList` and `LinkedList` implement `List` interface
  - `HashMap` and `TreeMap` implement `Map` interface
  - `HashSet` and `TreeSet` implement `Set` interface
    - Aside: There is also a Queue interface. They messed up on Stack; there's no Stack interface, just a class.

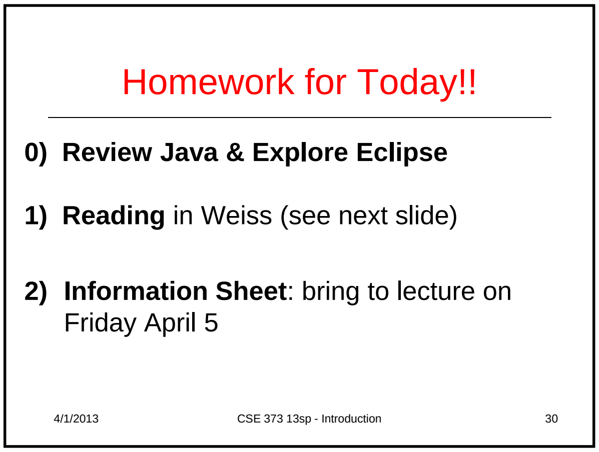


Java's `List` Interface

Operations described in Java's `List` interface (subset):

`add(`**`el, index`**`)`	inserts the element at the specified position in the list
`remove(`**`index`**`)`	removes the element at the specified position
`get(`**`index`**`)`	returns the element at the specified position
`set(`**`index, el`**`)`	replaces the element at the specified position with the specified element
`contains(`**`el`**`)`	returns true if the list contains the element
`size()`	returns the number of elements in the list

`ArrayList` and `LinkedList` are Java classes that implement the `List` interface



Homework for Today!!

0) **Review Java & Explore Eclipse**

1) **Reading** in Weiss (see next slide)

2) **Information Sheet**: bring to lecture on Friday April 5

Reading

- Reading in *Data Structures and Algorithm Analysis in Java*, by Weiss (2nd & 3rd Eds.)
- For this week:
  - › (Wed) Weiss 3.1-3.7 –Lists, Stacks, & Queues (Topic for Assignment #1)
  - › (Fri) Weiss 1.1-1.6 –Mathematics Review and Java
  - › Weiss 2.1-2.4 –Algorithm Analysis (Topic for next week)

Bring to Class on Friday:

- Name
- Email address
- Year (1,2,3,4)
- Major
- Hometown
- Interesting Fact or what I did over break.