Week 1

MAC OSX Users: find your "terminal" app (make a shortcut to it)
Windows Users: install Git for Windows (comes with git-bash) and one of these
https://mobaxterm.mobatek.net/ https://www.putty.org/

Instructor: _____ **Office:** 5376 **Email:** _____

Office Hours: Wed 9:30 - 11:30 Thursdays 11:00 - 12:30

Lec 001: TR 9:30am - 10:45am, 113 Psychology Building
Lec 002: TR 2:30pm - 3:45pm, 113 Psychology Building
Lec 004: MWF 1:20pm - 2:10pm, 132 Noland Hall

Announcements and Course Policies: https://pages.cs.wisc.edu/~deppeler/cs400/
Readings and Assignments: https://canvas.wisc.edu/143052

There is no textbook.
Assigned readings are found in Weekly Modules on Canvas.
Print lecture outlines (pdfs) before lecture each week. Complete problems during lecture.

THIS WEEK
x0 available and due by end of week (See Canvas assignment x0)
h0 available and due before 10pm on Fridays
p1 available later and require you to implement DataStructureADT

Read: Modules: Review, Week 1 & get started on Week 2 topics

THIS WEEK

- Course Overview
- Linux
- p1 Preview
- ADT
- DataStructureADTTest p1
- Test-Driven Development (TDD)
- Recall General Tree classes (Treenode and Tree)

Binary Search tree

NEXT WEEK

- p1 released
- Height of a General Tree
- Binary Search Tree Review
- Balanced Search Trees
- AVL Rotations

Learning Outcomes

- non-linear data structure
- edit - compile - run from command line
- implement interfaces
- become a better team mate (XO)
- analyse and compare complexity and performance
- design an object-oriented program with Backend D.S. + GUI
- become skill crafts programmers

Grades

- (20 %) **Final Exam: Friday May 10th, 5:05pm-7:05pm**
- (20 %) **Midterm Exam: Thursday March 7th, 5:00pm-7:00pm**
- (20 %) **Team Project:** You will need to form a 4 person team by Week 8 for this assignment } ind + team
- (10 %) **X-Team:** We will assign you to an x-team based on your lecture and availability
- (20 %) **Programs:** completed individually (or some may allow pairs)
- (10 %) **Homeworks:** completed individually (assigned and due weekly)

1/week 10/semester

Note: If a student's final percent is close to a letter grade border, exam percentage used to decide.

Work from the Computer Sciences Department computers

Computer Science Instructional Labs: 1358CS, 1366CS, 1368CS
Operating System: Ubuntu (Linux)
IDE: Eclipse
Java: Java 8 (Java 11 is available, but is not default -- Java 8 is fine for this semester)

CS login name

To print from computer in the CS Labs:
https://csl.cs.wisc.edu/services/printing-instructional-labs

cs.wisc.edu
Eclipse
Chrome

final new project → 产生新文件 → 在别的电脑上打开

Work from your personal computer

Install:

1. Install Updated and Secure Web Browser (Chrome works for me)
2. Install Java Development Kit (Java 8 is fine for cs400, jdk-8u201 is preferred)
   Note: Java 11.0.2(LTS) is available but may not work on older systems.
3. Install Eclipse IDE for Java Developers (Eclipse Proton in labs)
4. Install a terminal application (Win users: MobXTerm and Putty are popular and free)
5. Install Git   Github use wisc.edu → student package
   Note: Git for Windows includes GitBash (this is a good thing)

To connect from your personal computer to a CS Computer (best-linux.cs.wisc.edu):
https://csl.cs.wisc.edu/services/remote-access-csl-computers

Why learn Linux?

- free OS
- open source
- efficient
- stable operating system based on Unix
- knowledge is expected of cs graduates

What is a terminal?

a work station on a network - log in to the network to use resources

← command-line

How can I learn to use Linux?

We can use it today (if you have a terminal app like: Putty, or MobXTerm)

Remote connect to a CS Workstation running Linux (Ubuntu)

1. Open your terminal application (terminal, Putty, MobXTerm, _____)
2. Find connect dialog and enter these details
   a. computer: best-linux-cs.wisc.edu
   b. username: your CS account's username: _____
   c. password: your CS account's password
   d. click connect (or open)
3. Or, from a different Unix/Linux terminal's command-line prompt

ssh username@best-linux.cs.wisc.edu

What's next? Let's use Linux commands to answer these questions

Where are you in the file system? pwd

What does the **`tree`** command do? tree

What is in your home directory? private public

Who has permission to read your home directory? you & sys admin & everybody

How can I make a new directory? change to a different directory?

mkdir
rmdir

DEMO: In-class demonstration of the following from a command-line terminal window:

1. create a course directory (folder)
2. create a project sub-directory
3. copy file(s) from another location on CS file system

```
/p/course/cs400-deppeler/public/html-s/code/Hello.java
```

4. edit Hello.java (vi or emacs)
5. compile Hello.java
6. run Hello using Java Virtual Machine (jvm)

p1 Preview:

We wish to divide up the work on a project and we decide to start with the backend-database structure for our application. We won't use the data structure until we know that it works correctly, so we must test it. We decide to have some sub-team pairs implement different types of data structures for storing our data items and other sub-team pairs write code to test all of the implementations.

What are some problems that this approach may face?

- what is stored
- how is stored and retrieved

inconsistent code that does not compile or work as expected

How can we ensure that all data structures provide same functionality and can be tested by each teams test codes?

- data items are:
  - generic
  - have unique Comparable key (key, value) pair
- data structure has:
  - insert
  - remove
  - get
  - contains (boolean)
  - size
- enforce this with:
  - interface for Abstract Data Type
  - compiler will report errors

Abstract Data Type (ADT)

Abstract not defined, state what it does, but not how

Data: able to store related information

Type: a name for a new kind of object - class, interface

What ADTs did you learn to implement in CS300?
Note: These are the ADTs that we expect that students can implement in Java.

Queue

Stack

List

Heap, data structure or ADT

Priority Queue

Tree d.s. or ADT

Search Tree - ADT

Our p1 DataStructureADT

throw IllegalArgumentException

```
// Required operations for storing multiple instances of (key,value) pairs
// From: /p/course/cs400-deppeler/public/html-s/assignments/p1/files/DataStructureADT.java
public interface DataStructureADT<K extends Comparable<K>, V> {
    void insert(K key, V value);
    boolean remove(K key);
    boolean contains(K key);
    V get(K k);
    int size();
}
```

public Abstract

key comparable can't be null, value can be.

key can't be null, if k found, remove

if key is null / not found, return false

number of items in D.S.

DS_my -> my class implement the D.S.

Main.java -> try the main method.

Testing

How do we write tests so they can be run as 'batch'?

Good Tests have: test one thing (if possible)
- descriptive name
- known, pre-determined input sequence
- expected output (out come)
- need to determine actual output (out come)
- produce a fail or pass

Comprehensive Testing
- thoroughly test all operations: correct input, incorrect input
- boundary condition

Black-Box Testing
- we know operation, but not implementation
- testing the functionality
- can not ensure all code tested

White Box Testing
- we know the implementation, have access to internal variables
- we can write tests specific to internal structure
- we can ensure all code tested.

Unit Testing
- test smallest part as individual unit (method class, module, package).
- make sure unit test pass before using in larger code (structures).

System (or End-to-End) Tests
- test interaction between units
- correct output (out come) for entire run of application
- difficult to thoroughly test all system functionly

Test-Driven Development

design tests before writing solution
great way to ensure all team members understand design
help document design choices.
TDD can be done by individual, team, poject, companies, industry

Let's try it!

Given: `DataStructureADT` (as defined earlier in notes)
Problem: Design some tests for `DataStructureADTTest.java`

void

Test Name	What it tests? How? (pseudo-code is fine)

test00_
ds_empty()
	is newly constructed data structure empty size=0
	DataDS — test = new DS_my( );
	if (test.size( ) != 0) fail ("size should be zero!");
	assertEquals (0, size( ));

test01_	does inserting a single key,value pair increase size to 1 test.insert (new long (1), "CS 400"); if (test.size( ) != 0) fail
test02_	doe it handle a null key correctly (Hint: what should happen?) try { test.insert (null, "value"); fail (-----) } catch (IllegalArgumentException e) { } } catch (Exception e) { fail (---); }

Test Program Structure

What is a good structure for a Java Program that will be used to test a single other class?

- one test class for each structure being tested
- P1 black-box unit tests (implementation)
- have a main method to run all tests - provided by J-unit
- tests be independent of each other

Test Multiple Implementations of a single ADT (P1)

Can we use inheritance to reduce the amount of code needed to test multiple types that implement the same ADT?

- Define a test super class DataStructureADTTests
- Add an abstract method that creates and returns an instance of the ADT type createInstance
- Place all tests in the super test class
- Define a sub-class of the test super class for each data type you wish to test ~12 TestDS_name
- Define the abstract create method in each sub-type

JUnit Testing

JUnit is a testing framework for Java classes that does some of the tedious work of setting up, configuring, and running a set of unit tests.

You will learn the form for writing JUnit tests and some additional configuration options.

A JUnit Test "super" class will be provided for p1. You will add tests to that class and then sub-class it for each data structure implementation to be tested.

Data Structure implementations will be provided via classes/*.class files. You will not have the source for the structures you are required to test.

You will also be required to define your own DataStructure that implements the DataStructureADT provided. No need to wait, you can start on that today.

General Trees

Given:

T a placeholder for type of data

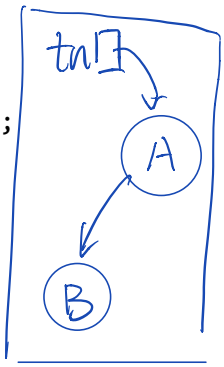
```
class Treenode<T> {
    private T data;
    private ListADT<Treenode<T>> children;
    ...
}
```

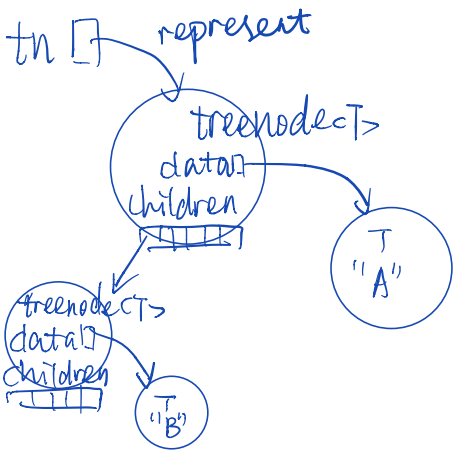
// Package level access

better.

And:

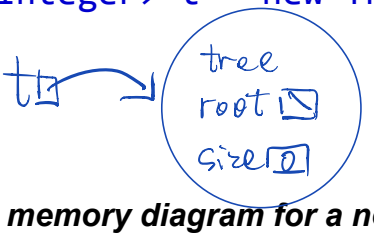
```
public class Tree<T> {
    private Treenode<T> root;
    private int size;
    public Tree() {
        root = null;
        size = 0;
    }
    ...
}
```





Draw a diagram of the memory allocated for an empty general tree.

```
Tree<Integer> t = new Tree();
```



Draw a memory diagram for a non-empty general tree that contains a root node with 3 children.