# 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: _____ Email: _____**

**Office Hours: _____**

**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
- Test-Driven Development (TDD)
- Recall General Tree classes (Treenode and Tree)

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

# Learning Outcomes

## Grades

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

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)

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

## 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
   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?

What is a terminal?

## 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: _____
    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

---

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

Where are you in the file system?

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

What is in your home directory?

Who has permission to read your home directory?

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

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

If you use dots, with cp, it means place file in current directory you are in

```
/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)

In VIM you need:
I - Insert mode
Esc - leave insert mode
: - write and quit (wq)

*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?**

- Lack of Clarity
- Verbal arrangements fall apart as the program evolves

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

Design a theoretical model Becomes an ADT.

- data items are:
  Generics:
  - Key = must be unique, comparable
  - Value = what you are storing
- data structure has:
  these behaviors.
  - Insert some key and its value
  - remove just with key
  - Get value using key
  - Contains with key
  - Size ()
- enforce this with:
  - In java, make an interface that specifies the ADT
  - The compiler can then enforce this "contract".

# Abstract Data Type (ADT)

Abstract Datatype: A data type described by predefined user operations without indicating how they are implemented

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

An ADT is not language specific

Different ADT:
- Stack
- List
- Queue
  - Priority Queue
- Binary Search Trees (border line)

Data Structures:
- Heap
- Linked List
- Oversized Array
-

# Our p1 DataStructureADT

```
// 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();
}
```

# Testing

*Good Tests have:*

*Comprehensive Testing*

*Black-Box Testing*

*White Box Testing*

*Unit Testing*

*System (or End-to-End) Tests*

# Test-Driven Development

*Let's try it!*

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

| Test Name | What it tests? How? (pseudo-code is fine) |
|---|---|
| test00_ | is newly constructed data structure empty size=0 |
| test01_ | does inserting a single key,value pair increase size to 1 |
| test02_ | doe it handle a null key correctly (Hint: what should happen?) |

# Test Program Structure

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

*Test Multiple Implementations of a single ADT*

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
- Add an abstract method that creates and returns an instance of the ADT type
- 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
- 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:

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

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.***