Week 2

ASSIGNMENTS
~~**h0** available and due before 10pm on Monday 1/28~~ - LAST NIGHT
h1 available and due before 10pm on Monday 2/4
p1 available and due before 10pm on Thursday 2/7

In-class Demo
Open Eclipse and
Web Browser to p1 assignment.

→ **TA Lab Consulting** - See schedule (cs400 home pages)
→ **Peer Mentoring available** - Friday 8am-12pm, 12:15-1:30pm in 1289CS

→ **TAs and Peer Mentors will focus on helping students get p1 JUnit tests running in Lab.**
→ **We can not guarantee that we can get your personal computers configured.**

Module: Week 2 (start on week 3 before next week)

THIS WEEK

- → Ready Set Program 1!
  - Read Assignment - there are getting started instructions there
  - Create and configure project for JUnit5, compile and run **TestDS_My**
  - Put tests in DataStructureADTTest (not the sub-classes)
- → Testing: JUnit5
- → Java: inner classes
- Determining Height of a Tree (Recursion Review)
- Binary Search Trees (BST) (Review?)
  - operations
  - implementing
  - complexities
- Classifying Binary Trees
- Balanced Search Trees
- George Adelson-Velsky and Evgenii Landis

NEXT WEEK

- X-team Exercise x1 (in-class exercise with your assigned teams)
- Watch for instructions to find your team number and how to meet

Test-Driven Development

- design tests before writing solution
- great way to ensure all team members understand design
- helps to document design choices
- TDD can be done by ind, team, project, companies, industry

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)

void test00_ ds_empty()	is newly constructed data structure empty size=0
	`DataStructureADT ds = new DS_My();` ← in all tests
	`if (ds.size() != 0) fail("size should be zero");`
	`assertEquals(0, ds.size());`

test01_	does inserting a single key,value pair increase size to 1 "1" ~~nex~~ k()
	`ds.insert(new Long(1), "cs400");`
	`if (ds.size() != 1) fail`

test02_	doe it handle a null key correctly (Hint: what should happen?)
	`try { ds.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
- write black-box unit tests
- have a main method to run all tests – provided by JUnit
- tests must 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 DataStructureADTTest
- 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_ *
- 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.~~

Writing JUnit5 Tests

What is JUnit?

- a testing framework for Java class
- has test runner that runs the tests you write

How do I use it?

1. Add Junit to the build-path
2. Add Junit Test Case (class)
3. Add units test to Test ~~cla~~ class
4. Run the Test Class

What does a @Test method look like?

- code to try things
- code to detect problems

if (bad cond) fail( )

assertEquals(expr1 , expr2)

Details:

1. **Import JUnit classes**

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
```

Eclipse imports these

```
public class TestClass {

    @Test
    void test123_try_someting_check_results() {
        // try something
        // if (cond_expression) fail("descriptive failure message")
        // assertEquals(  expr1, expr2 )
        // fail if something unexpected occurs or if something expected does not occur.
    }

}
```

– annotations

One or other not both

General Trees

Given:

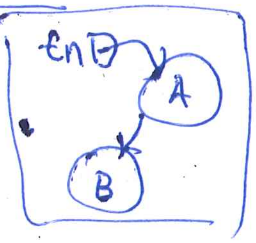
T a placeholder for type of data

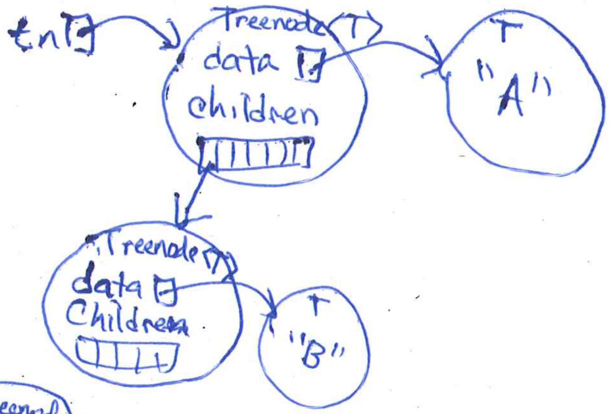
// package level access

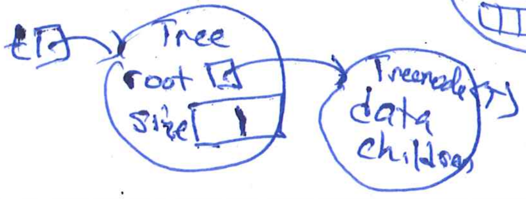
```
class Treenode<T> {
    private T data;
    private ListADT<Treenode<T>> children;
    ...
}
```

And:

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

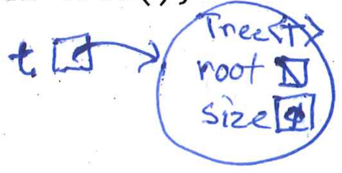




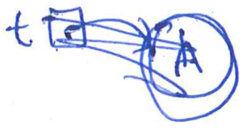


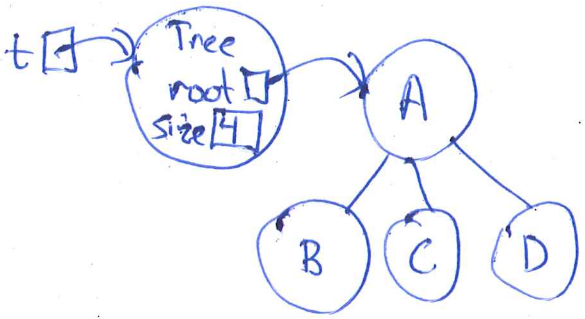
Draw a diagram of the memory allocated for an empty general tree.

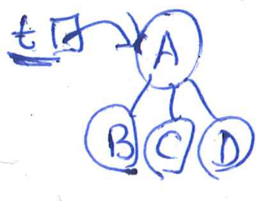
`Tree<Integer> t = new Tree();`



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







Java's Inner Classes

https://docs.oracle.com/javase/tutorial/java/javaOO/innerclasses.html

What is an inner class?

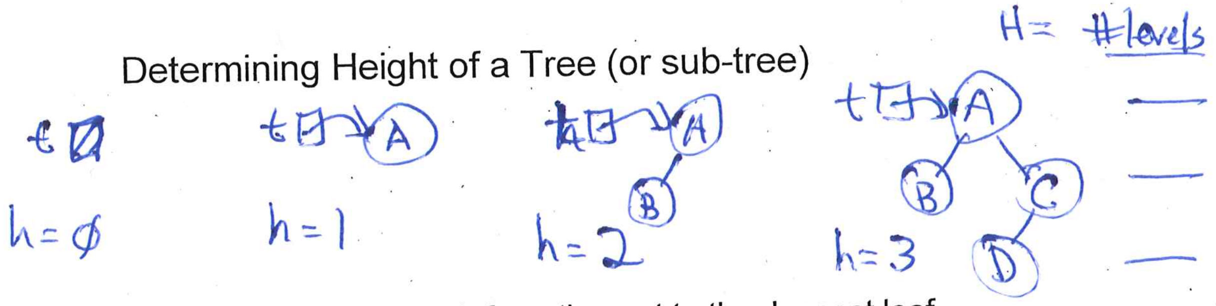
- a class defined within another class
- make inner class private , make members of inner class private

Define a generic search Tree cass using an inner class for the individual node type of the tree.

```
public class Tree<K extends Comparable<K>> {

    private class Treenode <K> {
        private K data;
        private ListADT<Treenode<K>> children;
        //private constructors and methods
            ...

    }
    private Treenode<K> root;
    private int size;
    public Tree(        ) {    ...   }

     add

     remove

       .
       ,
       ,
       ,

}
```

Determining Height of a Tree (or sub-tree)

height of a tree



We define height of a tree as the # nodes on path from the root to the deepest leaf.

Write a recursive definition for the height of a general tree.

height(Tree t)

h= 0, if t is null
h= 1, if ~~root~~ t is a leaf
} base cases

h= 1+ max height of children, if root is not a leaf } recursive

Complete the recursive height method based on the recursive definition.
Assume the method is added to a Tree class having a root instance variable.

```
public int height() { if (root == null) return ∅
                      else return height( root);
                    }

private int height( Treenode <T> node ) {




}
```