CS 340 Spring 2020
Sorted List Example
Sorted List

• Implement the class shown on the following slides. The class implements a sorted linked list of Comparable objects. The list is sorted in **descending order**.
import java.io.*;
import java.util.*;

public class SortedList<T extends Comparable<? super T>> {
    //Implements a generic singly linked list of Comparable objects
    //Sorted in descending order

    private class Node {
        private T data;
        private Node next;
        private Node(T d, Node n) {
            data = d;
            next = n;
        }
    }

    private Node head; //Reference to the first node in the list
    private int size; //The number of elements in the list
Sorted List

// When comparing elements in the list use equals or compareTo
// do not use ==

public SortedList() {
    // Constructor for an empty list
    head = null; // no sentinel node
    size = 0;
}
public void insert(T item) {
    // If an element in the list matches item do nothing (i.e. the list must not contain duplicates)
    // otherwise insert item into the list so the list remains sorted
}

public void remove(T item) {
    // If an element in the list matches item then remove the element from the list
    // otherwise do nothing
}
public boolean find(T item) {
//Return true if the list contains an element that matches item
//otherwise return false
//Your implementation must be **recursive** n
//You can do most of the work in an auxiliary private recursive method
}

public int size() {
//Return the number of items in the list
}

public String toString() {
//Return a string representation of the list
//The string representation of the list is a [ followed by the items in the list
//separated by commas followed by a ]
//For example a list of integers could look like [107,50,10,7,3,2]
}
public SortedList(SortedList<T> s1, SortedList<T> s2) {
   //PRE: s1.size() > 0 && s2.size() > 0
   // Constructor for the list created from two SortedLists
   // A new SortedList is created containing all the data from s1 and s2
   // The implementation must take advantage of the fact that s1 and s2
   // are sorted. The implementation cannot use the insert method

}