



## CSC110: Tutorial 12

Mr Tony Chung  
a.chung@lancaster.ac.uk  
<http://www.tonychung.net/>

# + Class Changes

- This class is too big.
- If you are registered on course CsC101, then:
  - **same time**
  - **different room:** LT3 in the Engineering Building
  - **same tutor (Tony Chung)**
- If you are registered on another course, then:
  - **same time**
  - **same room:** C32 in Fylde College
  - **different tutor (Alistair Baron)**



# Today's Objectives

- Since everybody has done linked-lists in the lab too...
- Binary Trees
  - also: Trees
  - also: Recursion
- We did Linked Lists last week, code is online and there is time for questions.

# + Trees

- A tree is... (basically a forked linked-list)
- Trees are useful data structures, they appear in:
  - Compilers (to represent the code)
  - Parsers in general (like Java DOM for XML)
  - Databases (to speed up searching)
- There is not enough time to cover parsing and this is not a lecture. So the focus will be on binary trees and data retrieval.

# + Binary Trees

- A binary tree has some rules:
  - Each node (except leaves) has a maximum of two children.
  - All descendants on the left of a node have a LOWER value.
  - All descendants on the right of a node have a HIGHER value.
  
- For example...
  
- So binary trees require the notion of a value for any object that is stored.

# + Java Comparable Interface

- Comparable requires a class to implement the **compareTo()** method:  
**public int compareTo( Object o ){}**
- By using an Interface, rather than a Class, it is possible to store any Object that implements that Interface in our binary tree.
  - Otherwise any Object could be stored.
  - Such objects may not have a compareTo() method!
- Implementing an Interface requires:
  - Providing the actual methods required.
  - Adding “**implements InterfaceName**” after the class name.





# Hash Values

- Many classes don't have a notion of a ranked value.
- But they still benefit from efficient storage.
- Hash values solve this problem by providing a value that can be used for ranking (and other purposes).
- Out of scope for today's class.
- See Java hashCode class.

# + Today's Class Exercise

- Construct a binary tree containing the following numbers:
  - 1,4,7,2,5,7
- Use a BinaryTree class to store each item.
  - **Java automatically converts an int to an Integer for you!**
  - Keep a reference to the 'root' node in the driver.
  - Provide the following methods that work:
    - void insert( Object o )
    - boolean contains( Object o )
- Normally you would provide a get() method that finds an Object using a hashcode and then returns that Object.
- Code to start you off on the next slide...

## + Skeleton Code

```
public class BinaryTree{
    private Comparable item;
    private BinaryTree left;
    private BinaryTree right;
    public BinaryTree( Comparable o ){ ... }
    public void insert( Comparable o ){ ... }
    public boolean contains( Comparable o ){ ... }

    // Recommended methods:
    private int goLeftorRight( Comparable o ){ ... }
}
```

# + Questions

- These slides and the solution will appear on the website.
  - <http://www.tonychung.net/>
- Please email topic suggestions for next week.