# KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

# Information and Computer Science Department

# ICS-202 Data Structures

## Lab Project

Guidelines:

**Group Formation:**

1. You must work in groups of a maximum of two students each. One-person groups (Individual work) are also allowed.

2. Groups may be formed across sections (i.e. Students of Sections 51 and 52 may work in a team)

3. Due date for the project is the week of 25th Dec 2012. You may submit the project during lab-time.

3. There will be a project demonstration followed by Question/Answer Session from the instructor.

4. You also need to include a report which should have the following

* title page with (a) project name, (b) the names, IDs and Section# of the group members.
* listing of all source code of the project in readable format (Courier font size 10 is preferred)
* output screen shots of the project.

**PROJECT QUESTON**

**Write a class GeneralTree in java that models a GeneralTree. A GeneralTree has the following characteristics:**

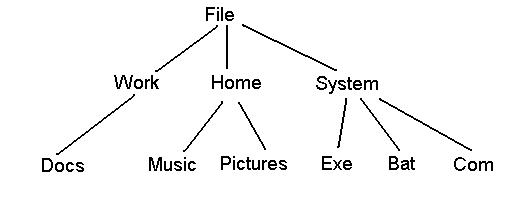
1. Design and implement a class **GeneralTreeNode**. Each **GeneralTreeNode** contains a **Comparable** object as its key.
2. A **GeneralTree** consists of **GeneralTreeNodes**. The maximum number of children for any node of a general tree is given by a constant MAXIMUM (fixed for the entire tree). The value of this MAXIMUM is set to be 10, but can be changed. Each **GeneralTreeNode** may have MAXIMUM or fewer children.
3. A **GeneralTree** supports the following operations:
   1. constructor(**GeneralTreeNode** rootNode)
   2. void insert(**GeneralTreeNode** parentNode, **GeneralTreeNode** newNode),
   3. **String** search(**GeneralTreeNode** newNode)

//The **String** prints the path of the newNode starting with the rootNode

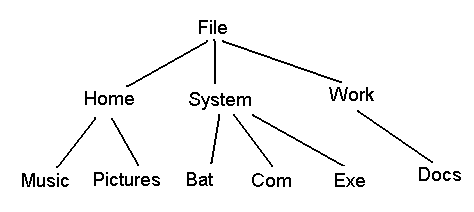
* 1. void delete(**GeneralTreeNode** existingNode).

1. A method **sortByLevel()**, which sorts each level of the **GeneralTree** according to **Comparable**.

For example, if a **GeneralTree** of Strings is formed as follows:



Then after **sortByLevel()**, it becomes the following:



Include methods for traversing the **GeneralTree**, i.e. **bfs**, **preorder**, **inorder** and **postorder**.

**Application**:

Now apply the class **GeneralTree** to a folder structure, i.e. given a folder in your system as the base folder, your program will create a **GeneralTree** out of the entire folder structure. You may add a variable to your **GeneralTreeNode** class to indicate whether a name is a folder name or a file name.

Implement a GUI-based interface that supports the following operations:

1. Given a root folder name, constructs the GeneralTree for the entire set of files and folders contained within it.
2. Searches for and prints the path to a given filename.
3. Prints the tree in bfs, preorder, inorder or postorder.
4. Sorts the GeneralTree and prints it as mentioned above.
5. Prints the number of files and number of folders in the tree.

Assume that all filenames and folder names are unique and that each folder contains no more than MAXIMUM number of files.