CSci 1933 Assignment 5

This assignment is due on **Friday, December 18th, 2015** at **11:55 PM**

General:

1. **You are supposed to work in group of 2 (unless exempted by the instructor), otherwise you will lose 20% of your grade.**
2. **When an assignment description tells you to give something a particular name, you must give it exactly that name. Failure to do so may result in a very low or zero score due to inability to properly run your code.**
3. **Your program must output its results in exactly the format we describe, with no additional text being written. If you insert additional output statements to aid in debugging, make sure they are removed before you submit.**

   You have the options 1) to work with the previous teammate or 2) to find a new teammate. If you have difficulties finding a teammate, please post to Student Forum on Moodle site **no later than Dec. 12th**.

1. Overview

In this assignment, you will implement Wikipedia categories using trees and your family trees using graphs and print them.

2. Part 1: Model Wikipedia categories using Tree

------Step 1------

2.1. In this step, you will create a "`Node`" and "`Tree`" interface in a new project in IntelliJ. You will then create implementations of this interfaces (e.g. "`MyNode.java`", "`MyTree.java`").

2.2. The "`Node`" interface is simple and must have the following methods:

- ➔ `public void addChild(Node n)`
  - ◆ adds n to the list of children for this node
- ➔ `public String getName()`
  - ◆ returns the name of the string (e.g. the name of the Wikipedia category)
- ➔ `public List<Node> getChildren()`

- returns a list that contains all children of this node

2.3. The "`Tree`" interface must have the following methods:

➔ `public void traverseBfs()`
- Prints out the nodes in your tree in breadth-first search order, with one node on each line.
- Hint: you'll want to use the `Queue<T>` interface in Java and the `LinkedList` implementation of `Queue<T>`. This means you'll likely have the following line of code in your method:
  - `Queue<Node> queue = new LinkedList<Node>();`

➔ `public Double getBranchingFactor()`
- Returns the average branching factor of your tree (do NOT include the leaves in this calculation).

➔ `public Boolean isBinaryTree()`
- Returns true if the tree is a formal binary tree and false otherwise

➔ `public void preorderDfsTraverseRecursive()`
- Prints out the nodes in your tree in pre-order depth-first search order, with one node on each line using a recursive approach. Unlike `traverseBfs()`, nodes at each level should be distinguished by the number of tabs before the name of the node is printed. For example

```
NodeAtLevel0
   NodeAtLevel1
   NodeAtLevel1
        NodeAtLevel2
   NodeAtLevel1
```

Hints:

- I recommend that your Tree class (e.g. `MyTree.java`) have a constructor that looks like this:
  - `public MyTree(Node root){…}`

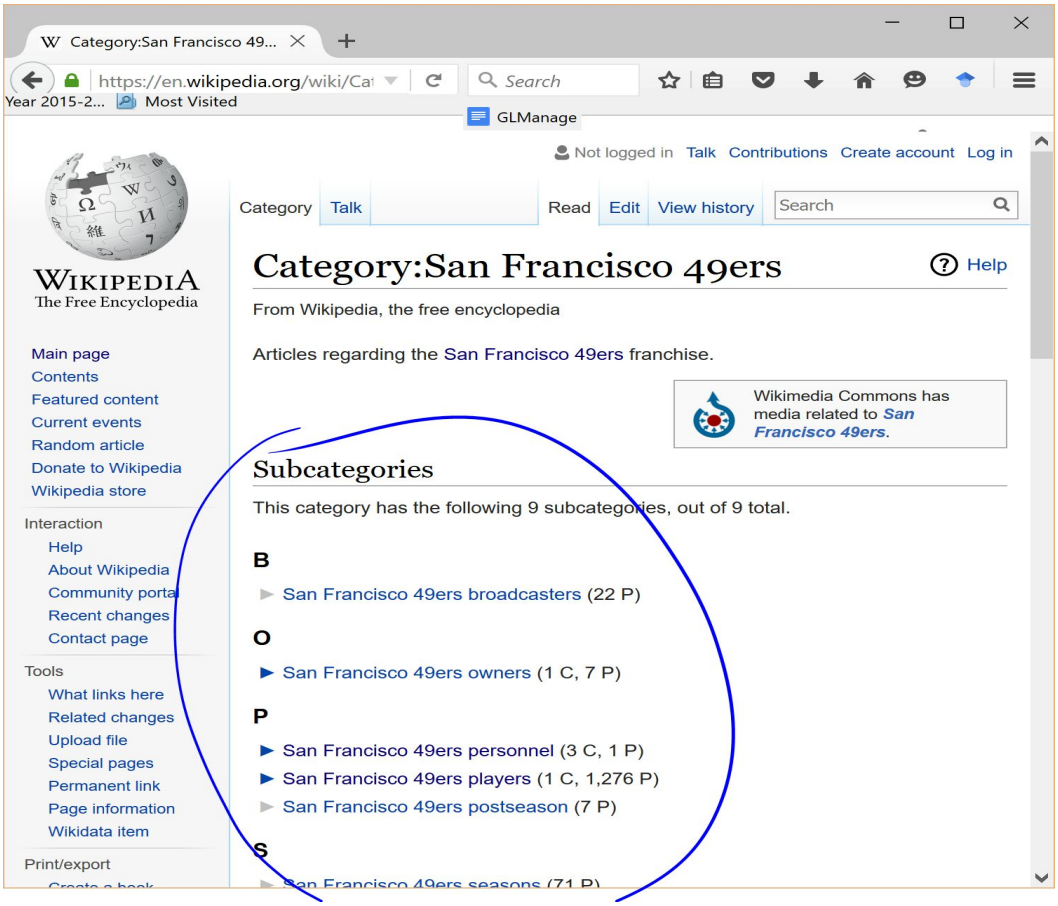
- You can save yourself a lot of time (and learn a valuable new programming paradigm) if you take the following approach for the breadth-first search portion of the assignment:
    - Make a new interface called `TraverseOperation` (or something like it)
    - Instead of "hard-coding" `System.out.println()` and other operations that need to occur during traversals, implement a `TraverseOperation` for each type of operation (e.g. I have a "`PrintTraverseOperation.java`" class).

------Step 2------

2.4. In this step, you will use your interfaces and classes to model a small section of Wikipedia's category structure.

- Go to a page of interest in Wikipedia (e.g. "San Francisco 49ers").
- Scroll to the bottom of the page and choose a category of which the page is a member. For example, in the image below, I chose the "San Francisco 49ers" category. I could have chosen any of the other as well (e.g. "National Football League teams"). NOTE: Try to choose a relatively general category (e.g. not "Sports clubs established in 1946"). You'll thank me later.

- Click on your chosen category. You should see a screen that looks like that below. Pay attention to the "Subcategories" section. These are effectively the next level in the Wikipedia category "tree", with your chosen category being the root.



- Using your interfaces and classes, model this part of the category structure of Wikipedia.
  - Your modeled tree must contain at least 3 levels and at least 10 nodes, although it can have more
  - You do not need to include all the nodes at every level if there are many, many nodes at each level.

2.5. Create the deliverables for this section of the assignment.

- (10 points)Draw a picture of the section of category structure you are modeling. You can do this by hand, in Powerpoint, etc. Just make sure it's legible and that you include it as "categorytree.png" (if you do it by hand, take a picture of the piece of paper).
- In a class called Main, build your tree and print out the results of the `Tree.java` methods in the following order:

- (20 points): preorderDfsTraverseRecursive()
- (20 points): traverseBfs()
- (20 points): getBranchingFactor()
- (20 points): isBinaryTree()

IMPORTANT: *Include a line between the output of each method as a buffer for the grader.*

3. Part 2: Model your family tree using Graph

Although family trees are called "trees", they are best modeled by computer scientists using graphs (why?). In this part of the assignment, you'll model and print out **BOTH** of your family trees.

As this is the final part of the final assignment, we're not going to provide as much structure as we have in the past: we're simply going to describe what we'd like you to do, and it's going to be your job to figure out how to structure your approach and implementation.

3.1. 10 points (per tree): For BOTH partners, a drawing of both your family trees up through your maternal and paternal grandparents
- If you have an enormous family (nodes > 20), you can truncate your tree as you desire.
- If for some reason you are uncomfortable using real names, go ahead and use fake ones (e.g. "Marge", "Homer") ☺

3.2. 10 points: A `Graph.java` class that models family trees and that contains a `getChildren()` and `getParent()` method for each node.

3.3. In a class called `Main.java`, model **BOTH** of your family trees up through your maternal and paternal grandparents using your `Graph.java` class.

3.4. 40 points: In a class called `Main.java`, print out both trees using the following format:
- For each node in the tree, print
  - "*<NODE NAME>* is the child of *<PARENT1>* and *<PARENT2>* and has the following children: *<CHILD1>, <CHILD2>, <CHILD3>*, … \n"

        - E.g. “Bart is the child of Marge and Homer and has the following children:” (Bart has no children).
    - Please include a full empty line between each tree.

4. Submission

Before you submit your solution, **you must create a text file called `group.txt` in your `src/` directory**. In this file, put the names and x.500 ID's of the members of your group.
Now compress your project into tar.gz file for submission. Please make sure the `src/` files are in the created tar.gz file.

Submit the tar.gz to the **Lecture Moodle Site.**