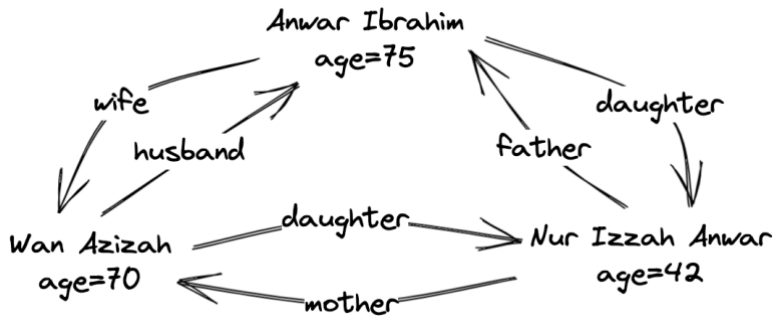
UM Data Structure 2022/2023 Generics Lab Test (Wednesday)

In Computing Mathematics I, you have learned that graphs and trees are very flexible and can handle complex data relationships. Thus, they are widely used in applications with complex relationships, such as recommendation engines, fraud detection, and social networks.

In case you have forgotten what are graphs and trees. These are the simple descriptions.

- A graph consists of a set V called a vertex set and a set E called an edge set. Each member of V is called a vertex and each member of E is called an edge.
- A tree is a connected acyclic undirected graph *G*. There is a unique path between every pair of vertices in *G*. A tree with *N* number of vertices contains *(N−1)* number of edges. The vertex which is of 0 degree is called root of the tree. The vertex which is of one degree is called leaf node of the tree, and the degree of an internal node is at least two.

This lab test has four pages with **three (3) tasks to complete**. For the first and second tasks, you will be required to design a data structure as follows:



Then, in the final task (Task 3), you will run Test.java to identify the killer of a crime scene.

Task 1

Your first task is to create a generic **Node class** that can store edges to other nodes. Please note that the Node class should be a generic class, and can take in any custom type (for example, a Java `String` type or a user-defined `Person` class).

Task 2

Next, within the generic **Node<ValueType> class**,

⇒ Create the following instance variables:

a. **private ValueType value;**
   i. Stores the value of the node
b. **private String edgeName1;**
   i. Stores the name of the first edge of the node
c. **private String edgeName2;**
   i. Stores the name of second edge of the node
d. **private Node<ValueType> edgeNode1;**
   i. Points to the first target node with edgeName1
e. **private Node<ValueType> edgeNode2;**
   i. Points to the second target node with edgeName2

⇒ Create the following public functions:

a. **public void setEdge1(String edgeName, Node<ValueType> node) {}**
   i. This will add set the first edge from the current node (`this`) to the target node (`node`).
   ii. This will also set the first edge name from the current node to the target node.
b. **public void setEdge2(String edgeName, Node<ValueType> node) {}**
   i. Like (a), but this will set the second edge name & the second node the current node will be pointing to.
c. **public NodeType getValue() {}**
   i. This simply returns the value the node is storing
d. **public void printEdgesAndNodes() {}**
   i. This will print the current node's value, and also the current node's edge names, and the value of the target nodes. (See SAMPLE OUTPUT below).

Assumption:

- You may assume that there will only be a maximum of two outgoing edges for each node.
- You may also assume that each edge will only point to ***one and only one node*** (*e.g.* In the image above, Anwar Ibrahim has only two edges (wife & daughter), and each of the edges points to only one node).

Notes:

- You must copy these two files (Main.java & Person.java) to develop and test your Node class.
- Your output should display the following when Main.java runs. (Slight difference in string formatting is acceptable, but it must look like an indented tree, with clear nodes & edge relationships:)

SAMPLE OUTPUT

```
-------------------------------------
Anwar Ibrahim (age=75)
  :wife->
    Wan Azizah (age=70)
  :daughter->
    Nur Izzah Anwar (age=42)
-------------------------------------

-------------------------------------
Wan Azizah (age=70)
  :husband->
    Anwar Ibrahim (age=75)
  :daughter->
    Nur Izzah Anwar (age=42)
-------------------------------------

-------------------------------------
Nur Izzah Anwar (age=42)
  :father->
    Anwar Ibrahim (age=75)
  :mother->
    Wan Azizah (age=70)
-------------------------------------
```

UM Data Structure 2022/2023 Generics Lab Test (Wednesday)

Main.java

```
class Main {
  public static void main(String[] args) {
    Node<Person> anwarIbrahim = new Node<>(new Person("Anwar Ibrahim", 75));
    Node<Person> wanAzizah = new Node<>(new Person("Wan Azizah", 70));
    Node<Person> nurIzzah = new Node<>(new Person("Nur Izzah Anwar", 42));

    anwarIbrahim.setEdge1("wife", wanAzizah);
    wanAzizah.setEdge1("husband", anwarIbrahim);

    anwarIbrahim.setEdge2("daughter", nurIzzah);
    nurIzzah.setEdge1("father", anwarIbrahim);

    wanAzizah.setEdge2("daughter", nurIzzah);
    nurIzzah.setEdge2("mother", wanAzizah);

    anwarIbrahim.printEdgesAndNodes();
    wanAzizah.printEdgesAndNodes();
    nurIzzah.printEdgesAndNodes();
  }
}
```

Person.java

```
public class Person {
  private String name;
  private int age;

  public Person(String name, int age) {
    this.name = name;
    this.age = age;
  }

  public String toString() {
    return String.format("%s (age=%s)", this.name, this.age);
  }
}
```

Task 3

It was a crime scene. Dave was dead lying on the floor and there were five suspects. After investigations, police found that:

- the killer was Dave's subordinate's husband's son's wife's friend.
- You and Alice were friends
- Alice's husband was Bob
- Bob's father was Frank
- Frank and Elsa were married
- Elsa's boss was Dave

In this task, create a public function in Node class:

a. **public Node<ValueType> traverse(String edgeName) {}**
   i. This should return the target node that matches the `edgeName`. For example, `anwarIbrahim` node has two edges (`wife` & `daughter`). When we run `anwarIbrahim.traverse("wife")`, it should return `wanAzizah` node).

b. Run the Test.java and identify who is the killer?

Test.java

```
class Test {
  public static void main(String[] args) {
    Node<String> alice = new Node<>("Alice");
    Node<String> bob = new Node<>("Bob");
    Node<String> charlie = new Node<>("You");
    Node<String> dave = new Node<>("Dave");
    Node<String> elsa = new Node<>("Elsa");
    Node<String> frank = new Node<>("Frank");

    alice.setEdge1("friend", you);
    you.setEdge1("friend", alice);

    alice.setEdge2("husband", bob);
    bob.setEdge1("wife", alice);

    bob.setEdge2("father", frank);
    frank.setEdge1("son", bob);

    frank.setEdge2("wife", elsa);
    elsa.setEdge1("husband", frank);

    elsa.setEdge2("boss", dave);
    dave.setEdge1("subordinate", elsa);

    // find the killer
    System.out.println(
        String.format("Killer is %s!",
            dave.traverse("subordinate")
                .traverse("husband")
                .traverse("son")
                .traverse("wife")
                .traverse("friend")
                .getValue()));

  }
}
```

// **END OF PAGE //**