THE UNIVERSITY OF SYDNEY

INFO1113 Week 9 Tutorial

Recursion

Recursion

Recursive functions are broken into one or more base cases and recursive cases. The base case is a condition under which the recursion stops and some value is returned, while the recursive case is where the function calls itself.

```
f(x):
    if x is 0:
        return 1
    else
        f(x - 1)
```

The java virtual machine is responsible for the stack allocation prior to your java application's execution. Typically this stack size is set to 1MB. The size of the method call will impact how many method calls you are able to make.

Question 1: Fibonacci

Write a java program that outputs and pints the n^{th} Fibonacci number with recursion.

Use the following method declaration:

```
public int fibonacci(int n)
```

The fibonacci recurrence relation is defined as:

$$F_n = F_{n-1} + F_{n-2}$$

The fibonacci sequence is as follows: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...$

You will need to consider the first two initial values of this sequence for your base case.

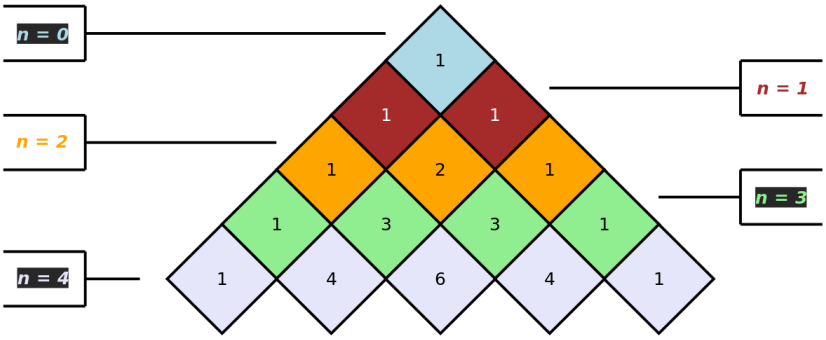
Question 2: Pascal's Triangle

Each row in pascal's triangle will include two 1's (at the beginning and end) except for the first row. The mid section of the row can be calculated using the previous row. A value in the midsection of the new row is calculated by adding the previous row's i-1 and i value (or sum of the two numbers directly above it).

If n is 0, you will print out at least 1 row of pascal's triangle (the first).

Consider using a collection type to store the previous rows of Pascal's Triangle.

The following image may help you visualise the problem.




```
public class PascalsTriangle {
        //Your utility methods and structures

        public void outputTriangle(int n) {
                //Your code here
        }
}
```

Example output:

```
java Pascal 4

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

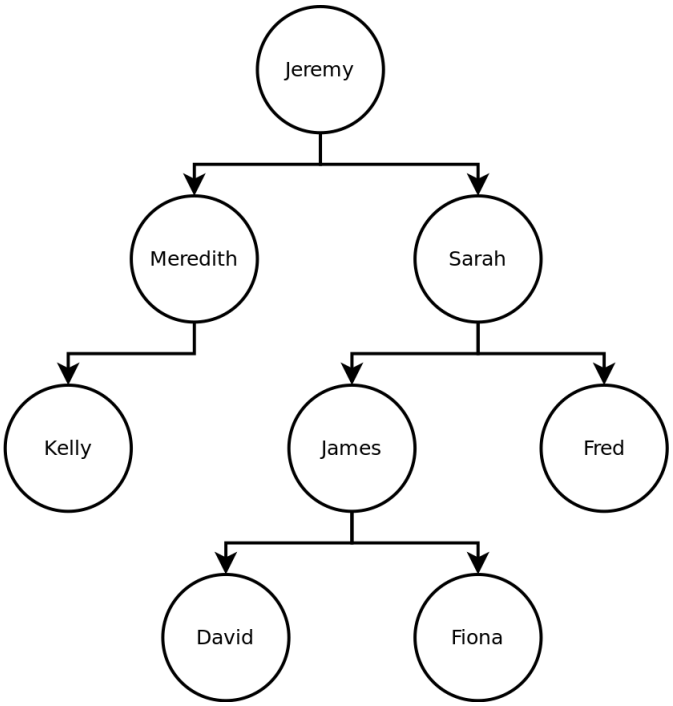
Question 3: Family Tree

You will need to create two classes, **FamilyTree** and **FamilyMember**. The objective of the **FamilyMember** class is to represent a person within a family tree.

Use the following scaffold as part of your solution.

```
import java.util.List;
import java.util.ArrayList;

public class FamilyMember {
        private String name;
        private int age;
        private List<FamilyMember> children;

        //Rest of your code here
}

import java.util.List;

public class FamilyTree {
        private FamilyMember headOfFamily;

        public FamilyTree(FamilyMember headOfFamily) {
                this.headOfFamily = headOfFamily;
        }

        public List<FamilyMember> getAll() {
                //Your code here
        }

        public List<FamilyMember> getParents() {
                //Your code here
        }

        public List<FamilyMember> getChildless() {
                //Your code here
        }

        public List<FamilyMember> getUnclesAndAunties() {
                //Your code here
        }

        public static void main(String[] args) {
                //Test your code here
        }
}
```

Consider the following family tree:



After implementing the `FamilyTree` and `FamilyMember` classes, implement the following queries

- Write a method `public List<FamilyMember> getAll()` which will retrieve all family members that are part of the family tree.
- Write a method `public List<FamilyMember> getAllParents()` which will retrieve all family members that have children, after writing getAllParents, write the inverse that will find all FamilyMember's that do not have a child.
- **Difficult:** Write a method `public List<FamilyMember> getUnclesAndAunties()` which will find all uncles and aunties.

Question 4: Assessed Task - Online Task 5

Remember you are required to complete a Online Task within the due date. Go to EdStem for this unit and click on Assessment to find out the task and the due date. This is a marked task.

Extension

Question 5: Selection Sort

Selection sort is a simple and common sorting algorithm that picks the smallest element and swaps it with the current unsorted index. The current unsorted index starts at zero and once an element is swapped with it, it the index will increment by 1.

```
public static void selectionSort(List<Integer> numbers) {
        //Your code here
}
```