Objectives: recursion on trees; selection sort;

Up next: MP6 - due Monday; Sign up for quiz EARLY this week.

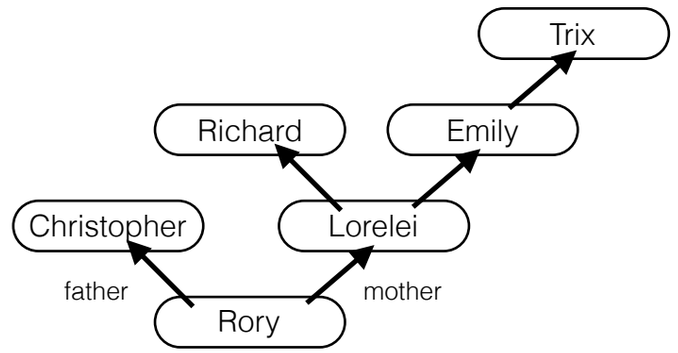
1. **Person class (for a family tree?) - encapsulate data:**

```
class Person {
   private String name;
   private Person mother;
   private Person father;

//  Write setters and getters (read/write) for each instance variable
//  Write convenience methods that get the
//  mother's name and the father's name for the person

//  A constructor that takes a String : newName

}
```



2. Set up the people and the relationships shown above:

```
public static void main(String[] args) {

}
```

3. Assume the head of the family tree is female. Write a *forward* recursive method *getFL1* that returns a string of the entire female lineage of person '*p*'. Insert commas between each person's name and a period at the end.

"*name,mother,grand-mother,great-grand-mother,….*"

```
public String getFL1() {

}
```

5. findMin() finds the **smallest value** of array elements 'lo' to 'hi'…
Example use: `int value = findMin(data, 0, data.length-1);`

```
public static int findMin(double[] array, int lo, int hi) {

   if( ____________ ) return ___________________; // Base Case

// Solve subproblem

   int result = findMin(array, _________, hi);

// Decide what to return to the caller.

   if( _____________ < ___________________ ) return result;


   return ____________________; // my value wins!

}
```

Is findMin method above forward or tail recursive?

6. **Use findMin() above and swap() to implement a recursive selection sort:**

```
public static void sort(double[] data) {




}

public static int findMin(double[] data, int lo, int hi) {
   // returns index
    modify code above
}

public static void swap(double[] data, int posA, int posB)
    {
   double temp = data[      ];
   data[     ] = data[      ];
   data[     ] = temp;
}
```

4. Family Tree: Write a recursive method to return an integer - the oldest known generation (count the starting person as generation 1):

7. Write a recursive method to return the total number of people in this family tree:

8. Write a tail-recursive method getFL2 to return the female lineage in reverse: "great-grand-mother, grand-mother, mother, p." (Your method will be called with an empty string as the parameter)

```
   public String getFL2(String result) {




  }
```

4. Write a recursive method to return the total number of people in this family tree:

5. Write a recursive method to return the oldest known generation (count the starting person as generation 1):

6. Write a tail-recursive method getFL2 to return the female lineage in reverse: "great-grand-mother, grand-mother, mother, p." (Your method will be called with an empty string as the parameter)

```
public String getFL2(String result) {



}
```

7. Write a tail recursive with a string accumulator method and a wrapper method to return the father with the longest name. Only consider the male lineage.

8. **Create an activation diagram for** `f3(31373)`:

```
public static int f3(int x) {
   if (x == 3) return 1;
   if (x < 10) return 0;

   return f3(x/10) + f3(x%10);
}
```