COMPUTER SCIENCE 12B (SPRING TERM, 2017)
ADVANCED PROGRAMMING TECHNIQUES

PROGRAMMING ASSIGNMENT 7

Due: Sunday April 23rd at 11:55pm

Part 1

You are to write a program that uses recursion to find all family ancestors and descendants of a given person.

The program asks for the name of the family data file and then prompts for a specific person's name. If the person is in the family database, the program shows all the ancestors and all the descendants for that person. For ancestors it must show all parents, all grandparents, all great grandparents, etc. For descendants it must show all children, all grandchildren, all great grandchildren, etc. The program also must use indentation to make it clear who is a parent of whom and who is a child of whom. As a test data file, you will be using information about the Tudor kings and queens of England. The name of the file is "tudor.txt".
Format of `tudor.txt`: The first section of the file (until the line with "END" has all the unique family members. The section of the `tudor.txx` after the "END" is divided into groups of three - first the family member, then his/her mother, then his/her father. This is how you can tell who is related to whom.

i.e.:

`Arthur` –> Family Member
`Elizabeth of York` -> Mother
`Henry VII` -> Father
`Henry VIII` –> Family Member
`Elizabeth of York` -> Mother
`Henry VII` -> Father

etc.

Special cases: The data file contains no information prior to Henry VII, so when you ask for information about him, you get a lot of descendants but no ancestor information. Below is the log of execution you should obtain for him.

```
What is the input file? tudor.txt
Person's name ('quit' to end)? Henry VII

Ancestors:
    Henry VII
Descendants:
    Henry VII
        Arthur
        Henry VIII
            Mary I
            Elizabeth I
            Edward VI
        Margaret
            James V
                Mary, Queen of Scots
                    James VI & I
            Margaret Stuart
                Henry, Lord Darnley
                    James VI & I
        Mary
            Frances
                Lady Jane Grey
```

At the other extreme is James VI of Scotland who became James I of England. The data file has no information about his descendants, but a great deal of information about his ancestors. Below is the log you should obtain for him.

```
What is the input file? tudor.txt
Person's name ('quit' to end)? James VI & I
Ancestors:
    James VI & I
        Mary, Queen of Scots
            Mary of Guise
            James V
                Margaret
                    Elizabeth of York
                    Henry VII
                James IV
        Henry, Lord Darnley
            Margaret Stuart
                Margaret
                    Elizabeth of York
                    Henry VII
Descendants:
    James VI & I
```

Your task: You are required to demonstrate that your program works for these two individuals and for another in the middle of this family tree. In particular, you should test it for Margaret. Below is the log you should obtain for her.

```
What is the input file? tudor.txt
Person's name ('quit' to end)? Margaret
Ancestors:
    Margaret
        Elizabeth of York
        Henry VII
Descendants:
    Margaret
        James V
            Mary, Queen of Scots
                James VI & I
        Margaret Stuart
```

```
                    Henry, Lord Darnley
                        James VI & I
```

Your client program should be named "**`Relatives.java`**" and it should include only static methods. It should prompt the use for the test file and then for the name of the person. If the name does not exist it should print a message and quit. If the name exists it should print the ancestors and descendants as shown in the above examples.

Design Hints:

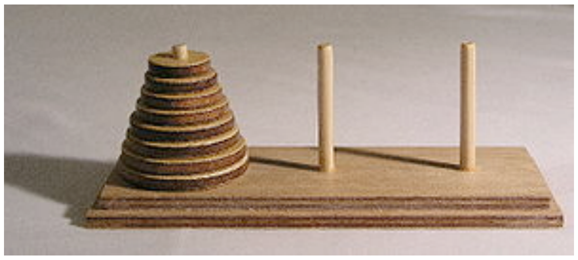
It might be useful to create one class that stores information about a person such as name, mother, father and any kids (if any). The class is responsible for setting and getting information for a specific person.
You might also find it useful to a have class that holds basic family relationship information about a particular family. The class could store a list of persons in the family and include a number of methods for setting and getting information about a family (read from a file and add a new relative and its info such as father, mother, kids, etc). This class could compliment the person class, i.e., it could read from a family test file and store the info of a specific person.

If you choose this design, then your client program simply read the family file that the user provides, stores the info of the family and once a Person's name is given, it parses the list of persons in the family and uses recursive methods that print out the ancestors and descendants.

You DON'T need to follow this design. You are free to design your own program as you wish. However, you need to make sure that you're writing object-oriented code and that the ancestors and descendants are produced using recursive methods. You should also explain these methods in your README.txt file.

Part 2

Write a recursive program to solve the Tower of Hanoi puzzle. The puzzle involves manipulating disks that you can move between three different towers (A, B, C). You are given a certain number of disks on one of the three towers. The disks have decreasing diameter, with the smallest disk on top, as shown in the next figure.



The object of the puzzle is to get all of the disks from one tower to another (say from the A to the B). The third tower (C) is provided as a temporary storage space as you move disks around. You are allowed to move only one disk at a time, and you are not allowed to place a disk on top of a smaller one (i.e., one with smaller diameter).
Examine the rather simple solutions for one, two and three disks, and see if you can discern a pattern. Then write a program that will solve the Towers of Hanoi puzzle for any number of disks (Hint: Moving four disks is a lot like moving three disks, except that one additional disk is on top.)
Your program should be named "Tower.java" and first prompt the user for the number of disks. Once the user types a number greater or equal to 1 then it solves the puzzle for that number of disks and prints the movements, i.e.,

```
Move disk from A to B
Move disk from A to B
Move disk from C to B
```

Your program should include only static methods.

Submission:

Your Java source code should be submitted via Latte the day it is due.

Grading:

You will be graded on

- **External Correctness:** The output of your program should match exactly what is expected. Programs that do not compile will not receive points for external correctness.
- **Internal Correctness:** Your source code should follow the stylistic guidelines shown in class. Also, remember to include the comment header at the beginning of your program.
- **One-on-one interactive grading:** By the end of the day that the assignment is due, please make an appointment with your TA for an interactive 10-15 minute grading session. You will receive an email notifying you of the name of the TA who has been assigned to you for this assignment with further instructions on setting up the appointment. (You will be meeting with a different TA for each assignment). One-on-one interactive grading will help you improve your programming skills and avoid repeating mistakes from one assignment to the next.