```
package com.jinglin;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) throws IOException {
        File familyTreeFile = new File("c:/familyTreeFile.txt");
        Scanner sc = new Scanner(familyTreeFile);

        Tree tree = new Tree();
        tree.readFamilyTree(sc);
        tree.printTree();
        questions(tree, "Jones");
        questions(tree, "Bob");
        questions(tree, "Dan");
        questions(tree, "Brian");
        questions(tree, "Richard");
        questions(tree, "Jake");
        questions(tree, "Michael");
        questions(tree, "Bill");
        questions(tree, "Deville");

        tree.readFamilyTree(sc);
        tree.printTree();
        questions(tree, "Barney");
        questions(tree, "David");
        questions(tree, "Enrique");

        tree.readFamilyTree(sc);
        tree.printTree();
        questions(tree, "Fletcher");
        questions(tree, "Gordon");
        questions(tree, "Ian");

        tree.readFamilyTree(sc);
        tree.printTree();
        questions(tree, "Jon");
        questions(tree, "Kent");
        questions(tree, "King");

        tree.readFamilyTree(sc);
        tree.printTree();
        questions(tree, "Mac");
        questions(tree, "Noe");
        questions(tree, "Paul");
    }

    public static void questions(Tree tree, String name) {
        tree.fatherOfP(name);
        tree.sonsOfP(name);
        tree.brothersOfP(name);
        tree.oldestBrothersOfP(name);
        tree.youngestBrothersOfP(name);
        tree.oldestSonOfP(name);
        tree.youngestSonOfP(name);
        tree.unclesOfP(name);
        tree.grandfatherOfP(name);
        System.out.println();
    }
}
```

```
package com.jinglin;

import java.util.ArrayList;
import java.util.Scanner;

public class Tree {
    private TreeNode root;

    public void readFamilyTree(Scanner sc){
        ArrayList<TreeNode> temp = new ArrayList<TreeNode>();
        ArrayList<TreeNode> temp2 = new ArrayList<TreeNode>();
        String name = sc.next();
        int numOfSon = sc.nextInt();
        int numOfSonOfBottomLevel;
        root = new TreeNode(name, numOfSon, null);
        temp.add(root);
        do{
            numOfSonOfBottomLevel = 0;
            for (int i = 0; i < temp.size(); i++) {
                for (int j = 0; j < temp.get(i).getNumOfSon(); j++) {
                    name = sc.next();
                    numOfSon = sc.nextInt();
                    TreeNode node = new TreeNode(name, numOfSon, temp.get(i));
                    temp.get(i).getSons().add(node);
                    temp2.add(node);
                }
            }
            for(TreeNode t: temp2) numOfSonOfBottomLevel += t.getNumOfSon();
            temp = temp2;
            temp2 = new ArrayList<TreeNode>();
        }while(numOfSonOfBottomLevel != 0);
    }

    public void printTree(){
        ArrayList<TreeNode> temp = new ArrayList<TreeNode>(root.getSons());
        ArrayList<TreeNode> temp2 = new ArrayList<TreeNode>();
        System.out.println("Family tree:");
        System.out.println("====================================");
        System.out.println(root.getName());
        while(!temp.isEmpty()) {
            for (int i = 0; i < temp.size(); i++) {
                System.out.print(temp.get(i).getName() + " ");
                temp2.addAll(temp.get(i).getSons());
            }
            System.out.println();
            temp = temp2;
            temp2 = new ArrayList<TreeNode>();
        }
        System.out.println("====================================");
        System.out.println();
    }

    public TreeNode searchP(String searchName){
        ArrayList<TreeNode> temp = new ArrayList<TreeNode>(root.getSons());
        ArrayList<TreeNode> temp2 = new ArrayList<TreeNode>();
        if(searchName.equals(root.getName())) return root;
        while(!temp.isEmpty()) {
            for (int i = 0; i < temp.size(); i++) {
                if(searchName.equals(temp.get(i).getName())) return temp.get(i);
                temp2.addAll(temp.get(i).getSons());
            }
            temp = temp2;
            temp2 = new ArrayList<TreeNode>();
        }
```

```
    return null;
}

public void fatherOfP(String name){
    TreeNode p = searchP(name);
    if(p == null){
        System.out.println("Person no found");
        return;
    }
    if(p.getParent() == null){
        System.out.println("No record " +
                "about " + p.getName() + "'s father");
    }else {
        System.out.println("The father of " + p.getName() + " is " +
                p.getParent().getName());
    }
}

public void sonsOfP(String name){
    TreeNode p = searchP(name);
    if(p == null){
        System.out.println("Person no found");
        return;
    }
    if(p.getSons().isEmpty()){
        System.out.println(p.getName() + " has no son");
    }else {
        System.out.print("The sons of " + p.getName() + " are");
        for (TreeNode s : p.getSons()) {
            System.out.print(" " + s.getName() + ",");
        }
        System.out.println();
    }
}

public void brothersOfP(String name){
    TreeNode p = searchP(name);
    if(p == null){
        System.out.println("Person no found");
        return;
    }
    if(p.getParent() == null){
        System.out.println("No record " +
                "about " + p.getName() + "'s brothers");
    }else if(p.getParent().getSons().size() < 2){
        System.out.println(p.getName() + " has no brother");
    }else {
        System.out.print("The brothers of " + p.getName() + " are");
        for (TreeNode b : p.getParent().getSons()) {
            if (!(b.getName().equals(p.getName())))
                System.out.print(" " + b.getName() + ",");
        }
        System.out.println();
    }
}

public void oldestBrothersOfP(String name){
    TreeNode p = searchP(name);
    if(p == null){
        System.out.println("Person no found");
        return;
    }
    if(p.getParent() == null){
        System.out.println("No record " +
```

```
                "about " + p.getName() + "'s oldest brother");
    } else {
        String oldestBrother = p.getParent().getSons().get(0).getName();
        if(oldestBrother.equals(p.getName())){
            System.out.println(p.getName() + " has no elder brother");
        }else {
            System.out.println("The oldest brother of " + p.getName() +
                    " is " + oldestBrother);
        }
    }
}

public void youngestBrothersOfP(String name){
    TreeNode p = searchP(name);
    if(p == null){
        System.out.println("Person no found");
        return;
    }
    if(p.getParent() == null){
        System.out.println("No record " +
                "about " + p.getName() + "'s youngest brother");
    } else {
        int end = p.getParent().getSons().size() - 1;
        String youngestBrother = p.getParent().getSons().get(end).getName();
        if(youngestBrother.equals(p.getName())){
            System.out.println(p.getName() + " has no younger brother");
        }else {
            System.out.println("The youngest brother of " + p.getName() +
                    " is " + youngestBrother);
        }
    }
}

public void oldestSonOfP(String name){
    TreeNode p = searchP(name);
    if(p == null){
        System.out.println("Person no found");
        return;
    }
    if(p.getSons().isEmpty()){
        System.out.println(p.getName() + " has no son");
    }else {
        System.out.println("The oldest son of " +
                p.getName() + " is " + p.getSons().get(0).getName());
    }
}

public void youngestSonOfP(String name){
    TreeNode p = searchP(name);
    if(p == null){
        System.out.println("Person no found");
        return;
    }
    if(p.getSons().isEmpty()){
        System.out.println(p.getName() + " has no son");
    }else {
        System.out.println("The youngest son of " +
                p.getName() + " is " + p.getSons().get(p.getSons()
                .size() - 1).getName());
    }
}

public void unclesOfP(String name){
    TreeNode p = searchP(name);
```

```
        if(p == null){
            System.out.println("Person no found");
            return;
        }
        if(p.getParent() == null) {
            System.out.println("No record " +
                    "about " + p.getName() + "'s uncles");
        }else if(p.getParent().getParent() == null){
            System.out.println("No record " +
                    "about " + p.getName() + "'s uncles");
        }else {
            if(p.getParent().getParent().getSons().size() < 2){
                System.out.println(p.getName() + "has no uncle");
            }else {
                System.out.print("The uncles of " + p.getName() + " are");
                for (TreeNode b : p.getParent().getParent().getSons()) {
                    if (!(b.getName().equals(p.getParent().getName())))
                        System.out.print(" " + b.getName() + ",");

                }
                System.out.println();
            }
        }
    }

    public void grandfatherOfP(String name){
        TreeNode p = searchP(name);
        if(p == null){
            System.out.println("Person no found");
            return;
        }
        if(p.getParent() == null) {
            System.out.println("No record " +
                    "about " + p.getName() + "'s grandfather");
        }else if(p.getParent().getParent() == null) {
            System.out.println("No record " +
                    "about " + p.getName() + "'s grandfather");
        }else {
            System.out.println("The grandfather of " +
                    p.getName() + " is " +
                    p.getParent().getParent().getName());
        }
    }
}
```

```java
package com.jinglin;

import java.util.ArrayList;

public class TreeNode {
    private String name;
    private int numOfSon;
    private TreeNode parent;
    private ArrayList<TreeNode> sons;

    public TreeNode(String name, int numOfSon, TreeNode parent) {
        this.name = name;
        this.numOfSon = numOfSon;
        this.parent = parent;
        this.sons = new ArrayList<TreeNode>();
    }

    public String getName() {
        return name;
    }

    public int getNumOfSon() {
        return numOfSon;
    }

    public TreeNode getParent() {
        return parent;
    }

    public ArrayList<TreeNode> getSons() {
        return sons;
    }
}
```