```java
import java.io.*;
import java.util.*;

class BTNode{

    private int nodeid;
    private int data;
    private int levelNum;
    private BTNode leftChildPtr;
    private BTNode rightChildPtr;

    public BTNode(){}

    public  void setNodeId(int id){
        nodeid = id;
    }

    public  int getNodeId(){
        return nodeid;
    }

    public  void setData(int d){
        data = d;
    }

    public  int getData(){
        return data;
    }

    public  void setLevelNum(int level){
        levelNum = level;
    }

    public  int getLevelNum(){
        return levelNum;
    }

    public  void setLeftChildPtr(BTNode ptr){
        leftChildPtr = ptr;
    }

    public  void setRightChildPtr(BTNode ptr){
        rightChildPtr = ptr;
    }

    public  BTNode getLeftChildPtr(){
        return leftChildPtr;
    }

    public  BTNode getRightChildPtr(){
        return rightChildPtr;
    }

    public  int getLeftChildID(){
        if (leftChildPtr == null)
            return -1;

        return leftChildPtr.getNodeId();
    }

    public  int getRightChildID(){
        if (rightChildPtr == null)
            return -1;

```

```
65          return rightChildPtr.getNodeId();
66      }
67  }
68
69
70
71  class BinaryTree{
72
73      private int numNodes;
74      private BTNode arrayOfBTNodes[];
75
76      public BinaryTree(int n){
77          numNodes = n;
78          arrayOfBTNodes = new BTNode[numNodes];
79
80          for (int id = 0; id < numNodes; id++){
81              arrayOfBTNodes[id] = new BTNode();
82              arrayOfBTNodes[id].setNodeId(id);
83              arrayOfBTNodes[id].setLevelNum(-1);
84              arrayOfBTNodes[id].setLeftChildPtr(null);
85              arrayOfBTNodes[id].setRightChildPtr(null);
86          }
87      }
88
89      public  void setLeftLink(int upstreamNodeID, int downstreamNodeID){
90          arrayOfBTNodes[upstreamNodeID].setLeftChildPtr(arrayOfBTNodes[downstreamNodeID]);
91      }
92
93      public  void setRightLink(int upstreamNodeID, int downstreamNodeID){
94
            arrayOfBTNodes[upstreamNodeID].setRightChildPtr(arrayOfBTNodes[downstreamNodeID])
            ;
95      }
96
97
98      public void printLeafNodes(){
99
100         for (int id = 0; id < numNodes; id++){
101
102             if (arrayOfBTNodes[id].getLeftChildPtr() == null &&
                arrayOfBTNodes[id].getRightChildPtr() == null)
103                 System.out.print(id + " ");
104             }
105
106         System.out.println();
107     }
108
109
110     public  boolean isLeafNode(int nodeid){
111
112         if (arrayOfBTNodes[nodeid].getLeftChildPtr() == null &&
            arrayOfBTNodes[nodeid].getRightChildPtr() == null)
113             return true;
114
115         return false;
116     }
117
118
119     public  int getNodeHeight(int nodeid){
120
121         if (nodeid == -1)
122             return -1;
123
124         if (isLeafNode(nodeid) )
```

```
125             return 0;
126
127         int leftChildID = arrayOfBTNodes[nodeid].getLeftChildID(); // -1 if not exist
128         int rightChildID = arrayOfBTNodes[nodeid].getRightChildID(); // -1 if not exist
129
130         return Math.max(getNodeHeight(leftChildID), getNodeHeight(rightChildID)) + 1;
131
132     }
133
134
135     public  int getTreeHeight(){
136         return getNodeHeight(0);
137     }
138
139
140     public  void PreOrderTraversal(int nodeid){
141
142         if (nodeid == -1)
143             return;
144
145         System.out.print(nodeid + " ");
146
147         PreOrderTraversal(arrayOfBTNodes[nodeid].getLeftChildID());
148         PreOrderTraversal(arrayOfBTNodes[nodeid].getRightChildID());
149
150     }
151
152
153     public void PrintPreOrderTraversal(){
154
155         PreOrderTraversal(0);
156         System.out.println();
157
158     }
159
160 }
161
162
163 class BinaryTreeTraversal{
164
165     public static void main(String[] args){
166
167     try{
168
169     Scanner input = new Scanner(System.in);
170
171     String filename;
172     System.out.print("Enter a file name: ");
173     filename = input.next();
174
175     int numNodes;
176     System.out.print("Enter number of nodes: ");
177     numNodes = input.nextInt();
178
179     BinaryTree binaryTree = new BinaryTree(numNodes);
180
181     FileReader fr = new FileReader(filename);
182     BufferedReader br = new BufferedReader(fr);
183
184     String line = null;
185
186     while ( (line = br.readLine()) != null){
187
188         StringTokenizer stk = new StringTokenizer(line, ",: ");
```

```
189
190             int upstreamNodeID = Integer.parseInt(stk.nextToken());
191
192             int childIndex = 0;
193
194             while (stk.hasMoreTokens()){
195
196                 int downstreamNodeID = Integer.parseInt(stk.nextToken());
197
198                 if (childIndex == 0 && downstreamNodeID != -1)
199                     binaryTree.setLeftLink(upstreamNodeID, downstreamNodeID);
200
201                 if (childIndex == 1 && downstreamNodeID != -1)
202                     binaryTree.setRightLink(upstreamNodeID, downstreamNodeID);
203
204                 childIndex++;
205
206             }
207
208         }
209
210
211         System.out.print("PreOrderTraversal: ");
212         binaryTree.PrintPreOrderTraversal();
213
214
215         }
216         catch(Exception e){e.printStackTrace();}
217
218         }
219     }
```

```
Enter a file name: binaryTreeFile_1.txt
Enter number of nodes: 10
PreOrderTraversal: 0 1 3 6 2 4 7 9 8 5
```