// File: DFSApp.java

class Stack

{

private final int SIZE = 20;

private int[] stackArray;

private int top;

public Stack()

{

stackArray = new int[SIZE];

top = -1;

}

public void insert(int j)

{

top++;

stackArray[top] = j;

}

public int remove()

{

int j = stackArray[top];

top--;

return j;

}

public int getTop()

{

return stackArray[top];

}

public boolean isEmpty()

{

return (top == -1);

}

}

//-------------------------------------------------------------

class Vertex

{

public char label; // label (e.g. 'A')

public boolean wasVisited;

public Vertex(char lab) // constructor

{

label = lab;

wasVisited = false;

}

} // end class Vertex

//-------------------------------------------------------------

class Graph

{

private final int MAX\_VERTS = 20;

private Vertex vertexList[]; // list of vertices

private int adjMat[][]; // adjacency matrix

private int nVerts; // current number of vertices

private Stack theStack;

public Graph() // constructor

{

vertexList = new Vertex[MAX\_VERTS];

// adjacency matrix

adjMat = new int[MAX\_VERTS][MAX\_VERTS];

nVerts = 0;

for(int j = 0; j < MAX\_VERTS; j++) // set adjacency

for(int k = 0; k < MAX\_VERTS; k++) // matrix to 0

adjMat[j][k] = 0;

theStack = new Stack();

} // end constructor

public void addVertex(char lab)

{

vertexList[nVerts++] = new Vertex(lab);

}

public void addEdge(int start, int end)

{

adjMat[start][end] = 1;

adjMat[end][start] = 1;

}

public void displayVertex(int v)

{

System.out.print(vertexList[v].label + " ");

}

// depth first search

public void dfs()

{

vertexList[0].wasVisited = true;

displayVertex(0);

theStack.insert(0);

while(!theStack.isEmpty())

{

int v = getAdjUnvisitedVertex(theStack.getTop());

if(v == -1)

theStack.remove();

else

{

vertexList[v].wasVisited = true;

displayVertex(v);

theStack.insert(v);

}

}

for(int j = 0; j < nVerts; j++)

vertexList[j].wasVisited = false;

} // end depth first search

public int getAdjUnvisitedVertex(int v)

{

for(int j = 0; j < nVerts; j++)

if(adjMat[v][j] == 1 && vertexList[j].wasVisited == false)

return j;

return -1;

}

} // end class Graph

//-------------------------------------------------------------

public class DFSApp

{

public static void main(String[] args)

{

Graph theGraph = new Graph();

theGraph.addVertex('A'); // 0 (start for dfs)

theGraph.addVertex('B'); // 1

theGraph.addVertex('C'); // 2

theGraph.addVertex('D'); // 3

theGraph.addVertex('E'); // 4

theGraph.addVertex('F'); // 5

theGraph.addVertex('G'); // 6

theGraph.addVertex('H'); // 7

theGraph.addVertex('I'); // 8

theGraph.addEdge(0, 1); // AB

theGraph.addEdge(0, 3); // AD

theGraph.addEdge(1, 4); // BE

theGraph.addEdge(2, 5); // CF

theGraph.addEdge(3, 2); // DC

theGraph.addEdge(4, 6); // EG

theGraph.addEdge(5, 7); // FH

theGraph.addEdge(5, 8); // FI

theGraph.addEdge(6, 7); // GH

theGraph.addEdge(7, 3); // HD

theGraph.addEdge(8, 1); // IA

theGraph.addEdge(8, 2); // IB

System.out.print("Visits: ");

theGraph.dfs(); // depth-first search

System.out.println();

}

}// File: DFSApp.java

class Stack

{

private final int SIZE = 20;

private int[] stackArray;

private int top;

public Stack()

{

stackArray = new int[SIZE];

top = -1;

}

public void insert(int j)

{

top++;

stackArray[top] = j;

}

public int remove()

{

int j = stackArray[top];

top--;

return j;

}

public int getTop()

{

return stackArray[top];

}

public boolean isEmpty()

{

return (top == -1);

}

}

//-------------------------------------------------------------

class Vertex

{

public char label; // label (e.g. 'A')

public boolean wasVisited;

public Vertex(char lab) // constructor

{

label = lab;

wasVisited = false;

}

} // end class Vertex

//-------------------------------------------------------------

class Graph

{

private final int MAX\_VERTS = 20;

private Vertex vertexList[]; // list of vertices

private int adjMat[][]; // adjacency matrix

private int nVerts; // current number of vertices

private Stack theStack;

public Graph() // constructor

{

vertexList = new Vertex[MAX\_VERTS];

// adjacency matrix

adjMat = new int[MAX\_VERTS][MAX\_VERTS];

nVerts = 0;

for(int j = 0; j < MAX\_VERTS; j++) // set adjacency

for(int k = 0; k < MAX\_VERTS; k++) // matrix to 0

adjMat[j][k] = 0;

theStack = new Stack();

} // end constructor

public void addVertex(char lab)

{

vertexList[nVerts++] = new Vertex(lab);

}

public void addEdge(int start, int end)

{

adjMat[start][end] = 1;

adjMat[end][start] = 1;

}

public void displayVertex(int v)

{

System.out.print(vertexList[v].label + " ");

}

// depth first search

public void dfs()

{

vertexList[0].wasVisited = true;

displayVertex(0);

theStack.insert(0);

while(!theStack.isEmpty())

{

int v = getAdjUnvisitedVertex(theStack.getTop());

if(v == -1)

theStack.remove();

else

{

vertexList[v].wasVisited = true;

displayVertex(v);

theStack.insert(v);

}

}

for(int j = 0; j < nVerts; j++)

vertexList[j].wasVisited = false;

} // end depth first search

public int getAdjUnvisitedVertex(int v)

{

for(int j = 0; j < nVerts; j++)

if(adjMat[v][j] == 1 && vertexList[j].wasVisited == false)

return j;

return -1;

}

} // end class Graph

//-------------------------------------------------------------

public class DFSApp

{

public static void main(String[] args)

{

Graph theGraph = new Graph();

theGraph.addVertex('A'); // 0 (start for dfs)

theGraph.addVertex('B'); // 1

theGraph.addVertex('C'); // 2

theGraph.addVertex('D'); // 3

theGraph.addVertex('E'); // 4

theGraph.addVertex('F'); // 5

theGraph.addVertex('G'); // 6

theGraph.addVertex('H'); // 7

theGraph.addVertex('I'); // 8

theGraph.addEdge(0, 1); // AB

theGraph.addEdge(0, 3); // AD

theGraph.addEdge(1, 4); // BE

theGraph.addEdge(2, 5); // CF

theGraph.addEdge(3, 2); // DC

theGraph.addEdge(4, 6); // EG

theGraph.addEdge(5, 7); // FH

theGraph.addEdge(5, 8); // FI

theGraph.addEdge(6, 7); // GH

theGraph.addEdge(7, 3); // HD

theGraph.addEdge(8, 1); // IA

theGraph.addEdge(8, 2); // IB

System.out.print("Visits: ");

theGraph.dfs(); // depth-first search

System.out.println();

}

}