# Java Week 10

**public** **class** DepthFirstTraversal<T> **extends** AdjacencyGraph<T> **implements** Traversal<T>

{

List<T> ourGraph = **new** ArrayList<T>();

List<T> visited = **new** ArrayList<T>();

@Override

**public** List<T> traverse() **throws** GraphError

{

**for**(**int** i = 0; i<getNodes().size(); i++)

{

//If we have not visited all the nodes

**if**(visited.size() <getNodes().size())

{

//store our current node

@SuppressWarnings("unchecked")

T startNode = (T) getNodes().toArray()[i];

//If our visited array does not contain this node

**if**(!visited.contains(startNode))

{

recursiveDepthFirstTraversal(startNode);

}

}

**else** **break**;

}

**return** ourGraph;

}

**public** **void** recursiveDepthFirstTraversal(T node) **throws** GraphError

{

//Add our node to the visited list and our graph

visited.add(node);

ourGraph.add(node);

//set what the neighbours are

Set<T> neighboursSet = getNeighbours(node);

//put the neighbours into an object array

@SuppressWarnings("unchecked")

T[] neighbouringNodes = (T[]) neighboursSet.toArray();

//check through all the neighbouring nodes

**for** (**int** i = 0; i < neighbouringNodes.length; i++)

{

//this node is the current neighbour

T n = neighbouringNodes[i];

// if it is not visited and exists

**if** (n != **null** && !visited.contains(n))

{

//start again from here

recursiveDepthFirstTraversal(n);

}

}

}

**public** **static** **void** main(String[] args) **throws** GraphError

{

DepthFirstTraversal<Integer> graph = **new** DepthFirstTraversal<>();

Integer node0 = **new** Integer(0);

Integer node1 = **new** Integer(1);

Integer node2 = **new** Integer(2);

Integer node3 = **new** Integer(3);

Integer node4 = **new** Integer(4);

graph.add(node0);

graph.add(node1);

graph.add(node2);

graph.add(node3);

graph.add(node4);

graph.add(0, 3);

graph.add(0, 2);

graph.add(1, 0);

graph.add(2, 1);

graph.add(3, 4);

graph.add(4,0);

graph.traverse();

System.***out***.println("Recursive Depth First Traversal: "+Arrays.*toString*((graph.ourGraph.toArray())));

}

}

Results:

Recursive Depth First Traversal: [0, 2, 1, 3, 4]