2. Depth First Search (DFS)

Program :

```
// Java program to print DFS
// mtraversal from a given given
// graph
import java.io.*;
import java.util.*;

// This class represents a
// directed graph using adjacency
// list representation
public class DGraph
{
   private int V; // No. of vertices

   // Array  of lists for
   // Adjacency List Representation
   private LinkedList<Integer> adj[];

   // Constructor
   @SuppressWarnings("unchecked") DGraph(int v)
   {
      V = v;
      adj = new LinkedList[v];
      for (int i = 0; i < v; ++i)
         adj[i] = new LinkedList();
   }

   // Function to add an edge into the graph
   void addEdge(int v, int w)
   {
      adj[v].add(w); // Add w to v's list.
   }

   // A function used by DFS
   void DFSUtil(int v, boolean visited[])
   {
      // Mark the current node as visited and print it
```

```
    visited[v] = true;
    System.out.print(v + " ");

    // Recur for all the vertices adjacent to this
    // vertex
    Iterator<Integer> i = adj[v].listIterator();
    while (i.hasNext()) {
        int n = i.next();
        if (!visited[n])
            DFSUtil(n, visited);
    }
}

// The function to do DFS traversal.
// It uses recursive
// DFSUtil()
void DFS(int v)
{
    // Mark all the vertices as
    // not visited(set as
    // false by default in java)
    boolean visited[] = new boolean[V];

    // Call the recursive helper
    // function to print DFS
    // traversal
    DFSUtil(v, visited);
}

// Driver Code
public static void main(String args[])
{
    DGraph g = new DGraph(4);

    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);
```

```
        System.out.println(
            "Following is Depth First Traversal "
            + "(starting from vertex 2) : ");

        g.DFS(2);
    }
}
```

Output: