```java
import java.util.*;
import java.lang.*;

class Graph {
    private int V;
    private LinkedList<Integer>[] adj;


    Graph(int v) {
        V = v;
        adj = new LinkedList[v];
        for (int i=0; i<v; ++i)
            adj[i] = new LinkedList<>();
    }


    void addEdge(int v,int w) {
        adj[v].add(w);
        adj[w].add(v);
    }


    void BFS(int s) {
        boolean visited[] = new boolean[V];

        LinkedList<Integer> queue = new LinkedList<>();

        visited[s]=true;
        queue.add(s);

        while (queue.size() != 0) {
            s = queue.poll();
            System.out.print(s+" ");

            Iterator<Integer> i = adj[s].listIterator();
            while (i.hasNext()) {
                int n = i.next();
                if (!visited[n]) {
                    visited[n] = true;
                    queue.add(n);
                }
            }
        }
    }

    void dfs(int s) {
        boolean visited[] = new boolean[V];
        Stack<Integer> st = new Stack<>();
        st.push(s);
        while(!st.empty()) {
            s = st.pop();
            if(!visited[s]) {
                visited[s] = true;
                System.out.print(s + " ");

                Iterator<Integer> i = adj[s].listIterator();
                while(i.hasNext()) {
                    int n = i.next();
                    if(!visited[n]) st.push(n);
                }
            }
        }
    }

    void dfs_h(boolean vis[], int s) {
        vis[s] = true;
        System.out.print(s + " ");
        Iterator<Integer> i = adj[s].listIterator();
        while(i.hasNext()) {
            int n = i.next();
            if(!vis[n]) dfs_h(vis, n);
        }
    }

    void dfs_recursive(int s) {
        boolean vis[] = new boolean[V];
        dfs_h(vis, s);
    }
}


class Main {
    public static void main(String[] args) {
        Graph g = null;
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("Choose an option:");
            System.out.println("1. Create a graph");
            System.out.println("2. BFS traversal");
            System.out.println("3. DFS traversal");
            System.out.println("4. DFS recursive traversal");
            System.out.println("5. Exit");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("Enter the number of vertices:");
                    int v = scanner.nextInt();
                    g = new Graph(v);
                    System.out.println("Enter the number of edges: ");
                    int e = scanner.nextInt();
                    System.out.println("Enter the edges (format: v w):");
                    for(int i = 0; i < e; i++) {
                        int v1 = scanner.nextInt();
                        int w = scanner.nextInt();
                        g.addEdge(v1, w);
                    }
                    break;
                case 2:
                    if (g ≠ null) {
                        System.out.println("Enter the starting vertex for BFS:");
                        int s = scanner.nextInt();
                        g.BFS(s);
                    } else {
                        System.out.println("Graph not created yet.");
                    }
                    break;
                case 3:
                    if (g ≠ null) {
                        System.out.println("Enter the starting vertex for DFS:");
                        int s = scanner.nextInt();
                        g.dfs(s);
                    } else {
                        System.out.println("Graph not created yet.");
                    }
                    break;
                case 4:
                    if (g ≠ null) {
                        System.out.println("Enter the starting vertex for DFS Recursive:");
                        int s = scanner.nextInt();
                        g.dfs_recursive(s);
                    } else {
                        System.out.println("Graph not created yet.");
                    }
                    break;
                case 5:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please choose a valid option.");
            }
        } while (choice ≠ 5);
    }
}


/*
----------------------------------------------------------------------------------------------------
OUTPUT
----------------------------------------------------------------------------------------------------

Note: Main.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Choose an option:
1. Create a graph
2. BFS traversal
3. DFS traversal
4. DFS recursive traversal
5. Exit

1

Enter the number of vertices:
4
Enter the number of edges:
4
Enter the edges (format: v w):
0 1
0 2
1 3
2 3

Choose an option:
1. Create a graph
2. BFS traversal
3. DFS traversal
4. DFS recursive traversal
5. Exit

2

Enter the starting vertex for BFS:
0

0 1 2 3

Choose an option:
1. Create a graph
2. BFS traversal
3. DFS traversal
4. DFS recursive traversal
5. Exit

3

Enter the starting vertex for DFS:
0

0 2 3 1
Choose an option:
1. Create a graph
2. BFS traversal
3. DFS traversal
4. DFS recursive traversal
5. Exit
```

```
4
Enter the starting vertex for DFS Recursive:
1

1 0 2 3

Choose an option:
1. Create a graph
2. BFS traversal
3. DFS traversal
4. DFS recursive traversal
5. Exit

5

Exiting...

*/
```