```
import java.util.*;


class BiconnectedComponents {
  private int V, time;
  private List<Integer>[] adj;


  public BiconnectedComponents(int v) {
     V = v;
     adj = new ArrayList[V];
     for (int i = 0; i < V; i++) adj[i] = new ArrayList<>();
  }


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


  private void BCCUtil(int u, int[] disc, int[] low, Stack<Edge> stack, int parent) {
     disc[u] = low[u] = ++time;
     int children = 0;


     for (int v : adj[u]) {
        if (disc[v] == -1) {
           children++;
           stack.push(new Edge(u, v));
           BCCUtil(v, disc, low, stack, u);
           low[u] = Math.min(low[u], low[v]);
```

```
            if ((parent == -1 && children > 1) || (parent != -1 && low[v] >= disc[u])) {
                System.out.print("Biconnected Component: ");
                while (true) {
                    Edge e = stack.pop();
                    System.out.print("(" + e.u + ", " + e.v + ") ");
                    if (e.u == u && e.v == v) break;
                }
                System.out.println();
            }
        } else if (v != parent && disc[v] < disc[u]) {
            low[u] = Math.min(low[u], disc[v]);
            stack.push(new Edge(u, v));
        }
    }
}


public void findBCCs() {
    int[] disc = new int[V], low = new int[V];
    Arrays.fill(disc, -1);
    Stack<Edge> stack = new Stack<>();
    time = 0;


    for (int i = 0; i < V; i++) {
        if (disc[i] == -1) {
            BCCUtil(i, disc, low, stack, -1);
            if (!stack.isEmpty()) {
                System.out.print("Biconnected Component: ");
                while (!stack.isEmpty()) {
                    Edge e = stack.pop();
                    System.out.print("(" + e.u + ", " + e.v + ") ");
```

```
                }
                System.out.println();
            }
        }
    }
  }


  private static class Edge {
    int u, v;
    Edge(int u, int v) { this.u = u; this.v = v; }
  }


  public static void main(String[] args) {
    BiconnectedComponents g = new BiconnectedComponents(7);
    g.addEdge(0, 1); g.addEdge(1, 2); g.addEdge(2, 0);
    g.addEdge(1, 3); g.addEdge(3, 4); g.addEdge(4, 5);
    g.addEdge(5, 3); g.addEdge(5, 6);
    System.out.println("Biconnected Components in the graph:");
    g.findBCCs();
  }
}
```