Union-Find for Cycle Detection

Write a Union-Find data structure with path compression. Use this data structure to detect a cycle in an undirected graph.

ANS:

```
package com.Day15;
public class CycleDetection {

   private int[] parent;
   private int[] rank;
   // Constructor to initialize the Union-Find data structure
   public CycleDetection(int size) {
      parent = new int[size];
      rank = new int[size];
      for (int i = 0; i < size; i++) {
         parent[i] = i;
         rank[i] = 0;
      }
   }
   // Find operation with path compression
   public int find(int node) {
      if (parent[node] != node) {
         parent[node] = find(parent[node]); // Path compression
      }
      return parent[node];
   }
   // Union operation with union by rank
   public void union(int node1, int node2) {
      int root1 = find(node1);
      int root2 = find(node2);
      if (root1 != root2) {
         if (rank[root1] > rank[root2]) {
            parent[root2] = root1;
         } else if (rank[root1] < rank[root2]) {
            parent[root1] = root2;
         } else {
            parent[root2] = root1;
            rank[root1]++;
```

```
            }
        }
    }
    // Method to detect cycle in an undirected graph
    public boolean hasCycle(int[][] edges) {
        for (int[] edge : edges) {
            int u = edge[0];
            int v = edge[1];
            int rootU = find(u);
            int rootV = find(v);
            if (rootU == rootV) {
                return true; // Cycle detected
            }
            union(u, v);
        }
        return false; // No cycle detected
    }
    public static void main(String[] args) {
        // Example graph: edges list and number of vertices
        int[][] edges = {
            {0, 1},
            {1, 2},
            {2, 3},
            {3, 4},
            {4, 2} // This edge creates a cycle
        };
        int numVertices = 5;
        // Create an instance of the class
        CycleDetection cycleDetection = new CycleDetection(numVertices);
        // Check if the graph has a cycle
        if (cycleDetection.hasCycle(edges)) {
            System.out.println("Cycle detected in the graph");
        } else {
            System.out.println("No cycle detected in the graph");
        }
    }
}
```

OUTPUT:

Cycle detected in the graph