```java
/**
 * @author Amar Bessedik
 * This class designs a disjoint set data structure using an array of ints.
 * The index represents the label of a set. The cell content hold vertices.
 */
public class DisjointSet
{
   private int[] set;//Holds graph's vertices represented by their labels.
   private int N;//# of subsets where labels start from 1 up to N included.

   /**
    * Constructor
    * @param n  # of vertices
    *
    */
   public DisjointSet(int n)
   {
      this.N = n;
      initializeSubsets(N);
   }//end Constructor

   private void initializeSubsets(int n)
   {
      this.set = new int[n + 1]; // There is no vertex called 0, therefore:
      // Start from 1 up to n included as vertices called 1, 2, . . ., n
      for (int x = 1; x <= n; x++)
         this.set[x] = x;
   }//end initializeSubsets()

   /**
    * Finds the label of a vertex and does path compression along the way.
    * @param x is a vertex
    * @return the label of x.
    */
   public int find2(int x)
   {
      int r = x;
      while (r != set[r]) //If x is not its own representative (label)
         r = set[r];

      int i = x;
      int j;

      while (i != r) //Proceed to path compression only if x != set[x]
      {
         j = set[i];
         set[i] = r;
         i = j;
      }//All visited nodes would have been updated to point to same representative.
      return r;
   }//end find2()


```

```
54     /**
55      * merges two vertices in different disjoint sets into one disjoint set.
56      *
57      * @param a
58      * @param b
59      */
60     public void merge(int a, int b)
61     {
62        if (a < b)
63           set[b] = a;
64        else
65           set[a] = b;
66     }//end merge()
67
68     //Helps in debugging and to display the effect of path compression.
69     private void showPathCompression(int[] d_set)
70     {
71        for (int u : d_set)
72           System.out.printf("%3d", d_set[u]);
73        System.out.println();
74     }//end showPathCompression()
75 }//end class
76
```