Nama : Andyan Yogawardhana

NIM : 21/482180/PA/21030

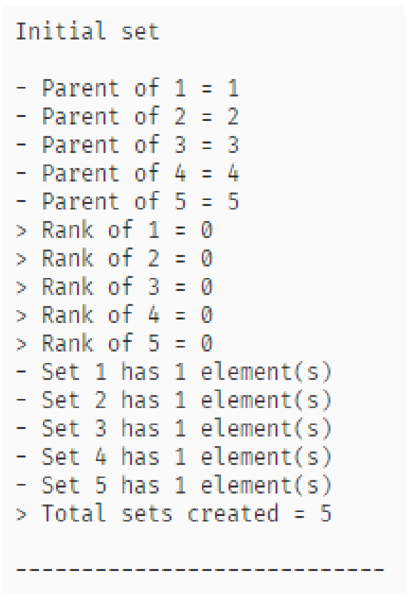
Kelas : KOMB1

Tugas 9 – Disjoint Set

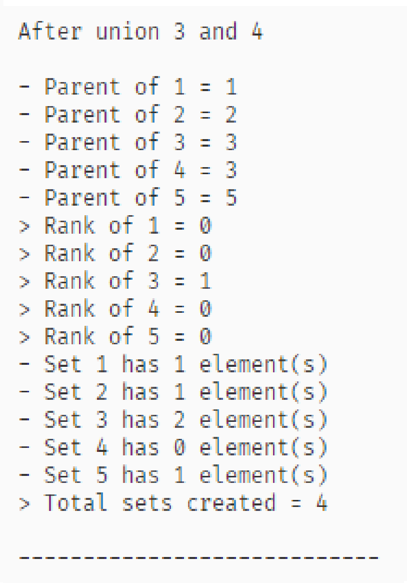
Source Code

```java
public class Disjoint {
    public static void main(String[] args) {
        DisjointSet disjointSet = new DisjointSet(5);
        System.out.println("\nInitial set\n");
        disjointSet.printResult();

        disjointSet.union(3, 4);
        System.out.println("\nAfter union 3 and 4\n");
        disjointSet.printResult();

        disjointSet.union(1, 2);
        disjointSet.union(1, 3);
        System.out.println("\nFinal result\n");
        disjointSet.printResult();
    }
}

class Set {
    private int parent, rank;

    public Set(int data) {
        this.parent = data;
        this.rank = 0;
    }

    public int getParent() {
        return this.parent;
    }

    public void setParent(int data) {
        this.parent = data;
    }

    public int getRank() {
        return this.rank;
    }

    public void setRank(int data) {
        this.rank = data;
    }
}

class DisjointSet {
    private Set[] sets;
    private int[] elements;     // menyimpan jumlah anggota elemen masing-masing set
    private int size, setCount; // setCount: menyimpan jumlah set yang terbentuk

    public DisjointSet(int numItem) {
        this.size = numItem;
        this.setCount = numItem;    // jumlah set awal yang terbentuk sama dengan jumlah elemen awal
        this.sets = new Set[size + 1];
        this.elements = new int[size + 1];

        for(int i = 1; i <= this.size; i++) {
            this.sets[i] = new Set(i);
            this.elements[i] = 1;   // setiap set baru yang terbentuk memiliki satu buah elemen
        }
    }

    public int find(int item) {
        int parent = this.sets[item].getParent();

        if(item == parent) {
            return item;
        }

        else {
            parent = find(parent);
            this.sets[item].setParent(parent); // path compression
            return parent;
        }
    }

    public boolean isSameSet(int firstItem, int secondItem) {
        return find(firstItem) == find(secondItem);
    }

    public void union(int firstItem, int secondItem) {
        int firstItemParent = find(firstItem);
        int secondItemParent = find(secondItem);

        if(firstItemParent != secondItemParent) {
            int firstRank = this.sets[firstItemParent].getRank();
            int secondRank = this.sets[secondItemParent].getRank();

            if(firstRank < secondRank) {
                this.sets[firstItemParent].setParent(secondItemParent);
                this.elements[firstItemParent] += this.elements[secondItemParent];  // menambahkan jumlah elemen pada set yang menjadi root dengan jumlah elemen dari set yang bergabung
            }
            else if (firstRank > secondRank) {
                this.sets[secondItemParent].setParent(firstItemParent);
                this.elements[secondItemParent] += this.elements[firstItemParent];  // menambahkan jumlah elemen pada set yang menjadi root dengan jumlah elemen dari set yang bergabung
            }
            else {
                this.sets[secondItemParent].setParent(firstItemParent);
                this.sets[firstItemParent].setRank(firstRank + 1);
                this.elements[firstItemParent] += this.elements[secondItemParent];  // menambahkan jumlah elemen pada set yang menjadi root dengan jumlah elemen dari set yang bergabung
                this.elements[secondItemParent] = 0;     // set yang bergabung ke set lain tidak memiliki elemen lagi setelah dilakukan union
            }

                setCount--;     // jumlah tree yang terbentuk berkurang setelah dua buah set bergabung menjadi satu
            }
        }

        public void printResult() {
            print();
            printRank();
            countElement();
            countSet();
            System.out.println("----------------------------");
        }

        public void print() {
            for(int i = 1; i <= this.size; i++) {
                System.out.println("- Parent of " + i + " = " + find(i));
            }
        }

        public void printRank() {
            for(int i = 1; i <= this.size; i++) {
                System.out.println("> Rank of " + i + " = " + this.sets[i].getRank());
            }
        }

        public void countElement() {
            for(int i = 1; i <= size; i++) {
                System.out.println("- Set " + i + " has " + elements[i] + " element(s)");
            }
        }

        public void countSet() {
            System.out.println("> Total sets created = " + setCount + "\n");
        }
    }
```

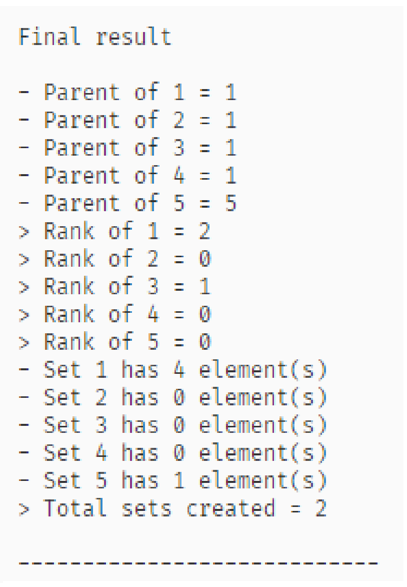
Output Terminal



```
Initial set

- Parent of 1 = 1
- Parent of 2 = 2
- Parent of 3 = 3
- Parent of 4 = 4
- Parent of 5 = 5
> Rank of 1 = 0
> Rank of 2 = 0
> Rank of 3 = 0
> Rank of 4 = 0
> Rank of 5 = 0
- Set 1 has 1 element(s)
- Set 2 has 1 element(s)
- Set 3 has 1 element(s)
- Set 4 has 1 element(s)
- Set 5 has 1 element(s)
> Total sets created = 5

----------------------------
```



```
After union 3 and 4

- Parent of 1 = 1
- Parent of 2 = 2
- Parent of 3 = 3
- Parent of 4 = 3
- Parent of 5 = 5
> Rank of 1 = 0
> Rank of 2 = 0
> Rank of 3 = 1
> Rank of 4 = 0
> Rank of 5 = 0
- Set 1 has 1 element(s)
- Set 2 has 1 element(s)
- Set 3 has 2 element(s)
- Set 4 has 0 element(s)
- Set 5 has 1 element(s)
> Total sets created = 4

----------------------------
```



```
Final result

- Parent of 1 = 1
- Parent of 2 = 1
- Parent of 3 = 1
- Parent of 4 = 1
- Parent of 5 = 5
> Rank of 1 = 2
> Rank of 2 = 0
> Rank of 3 = 1
> Rank of 4 = 0
> Rank of 5 = 0
- Set 1 has 4 element(s)
- Set 2 has 0 element(s)
- Set 3 has 0 element(s)
- Set 4 has 0 element(s)
- Set 5 has 1 element(s)
> Total sets created = 2

----------------------------
```