Hausaufgabe 11.3: Divide-and-Conquer

```
public class MinMaxFinder {

	private int [] a;

	public MinMaxFinder(int[] input){
		a = input;
	}
//	find min and max based on naive approach
	public MinMaxPair findMinAndMaxNaive(){
		int min = Integer.MAX_VALUE;
		int max = Integer.MIN_VALUE;
		for(int i = 0; i< a.length; i++){
			if(a[i]< min)
				min = a[i];
			if(a[i]> max)
				max = a[i];
		}
		return new MinMaxPair(min, max);
	}
	//find min and max using devide-and-conquer
	public MinMaxPair findMinAndMaxClever(int l, int r){
		if(r-l==1){
			if(a[l]<a[r])
				return new MinMaxPair(a[l], a[r]);
			else
				return new MinMaxPair(a[r], a[l]);
		}
		else{
			MinMaxPair leftMinMax=findMinAndMaxClever(l, (l+r)/2);
			MinMaxPair rightMinMax=findMinAndMaxClever((l+r+1)/2, r);
			int min = leftMinMax.min;
			int max = leftMinMax.max;
			if(leftMinMax.min > rightMinMax.min)
				min = rightMinMax.min;
			if(leftMinMax.max < rightMinMax.max)
				max = rightMinMax.max;
			return new MinMaxPair(min, max);
		}
	}
```

```
    public static void main(String[] args) {
        int[] test = {3,5,6,8,9,2,4,5};
        MinMaxFinder f = new MinMaxFinder(test);
        System.out.println(f.findMinAndMaxNaive());
        System.out.println(f.findMinAndMaxClever(0, test.length-1));
    }

    public class MinMaxPair{
        public int min, max;
        public MinMaxPair(int min, int max){
            this.min = min;
            this.max = max;
        }
        public String toString(){
            return "Min: "+min+" / Max: "+max;
        }
    }
}
```