Design And Analysis Algorithms

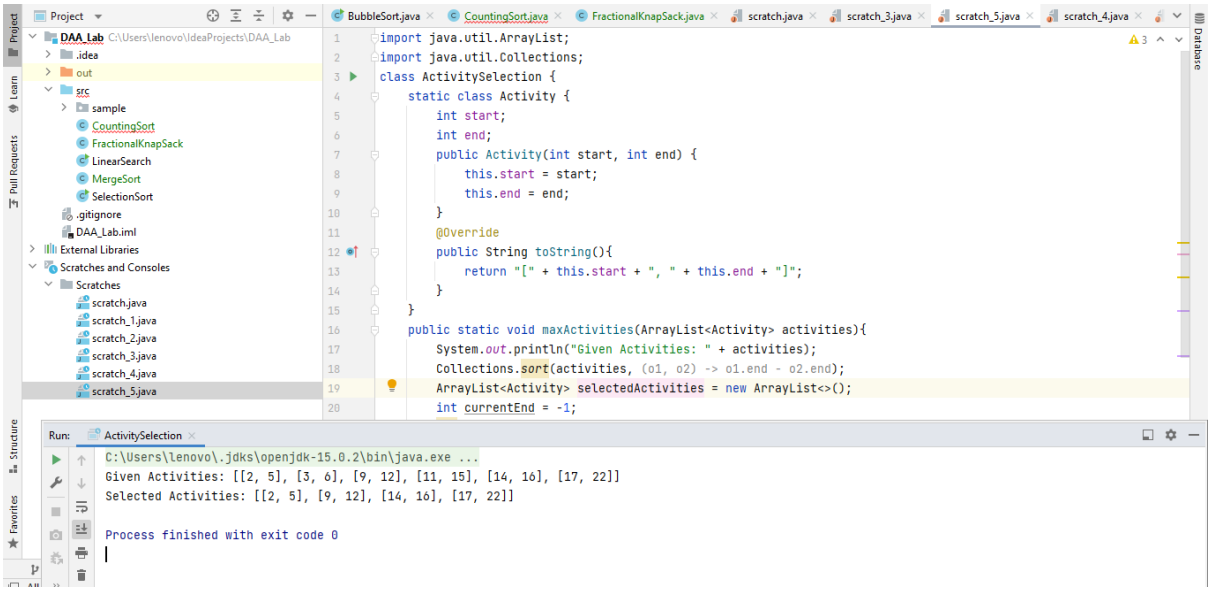
Practical

Objective:- Implement and analyze the complexity of Selection Sort.

Code:-

```
import java.util.ArrayList;
import java.util.Collections;
class ActivitySelection {
   static class Activity {
       int start;
       int end;
       public Activity(int start, int end) {
           this.start = start;
           this.end = end;
       }
       @Override
       public String toString(){
           return "[" + this.start + ", " + this.end + "]";
       }
   }
   public static void maxActivities(ArrayList<Activity> activities){
       System.out.println("Given Activities: " + activities);
       Collections.sort(activities, (o1, o2) -> o1.end - o2.end);
       ArrayList<Activity> selectedActivities = new ArrayList<>();
       int currentEnd = -1;
       for (int i = 0; i <activities.size() ; i++) {
           Activity currentActivity = activities.get(i);
           if(currentActivity.start >currentEnd){
               selectedActivities.add(currentActivity);
               currentEnd = currentActivity.end;
           }
       }
       System.out.println("Selected Activities: " + selectedActivities);
   }
   public static void main(String[] args) {
       ArrayList<Activity> activities = new ArrayList<>();
       activities.add(new Activity(2, 5));
       activities.add(new Activity(3,6));
       activities.add(new Activity(9, 12));
       activities.add(new Activity(11, 15));
       activities.add(new Activity(14, 16));
       activities.add(new Activity(17, 22));
       maxActivities(activities);
   }
}
```

Output:-



Analyze:-

The complexity of this problem is **O(n log n)** when the list is not sorted. When the sorted list is provided the complexity will be O(n).