Utilization Of Algorithms, Dynamic Programming, Optimal Memory Utilization

```
public class Candidate {
    private String name;
    private String contactInformation;
    private List<InternalMark> internalMarks;

    public Candidate(String name, String contactInformation, List<InternalMark>
internalMarks) {
        this.name = name;
        this.contactInformation = contactInformation;
        this.internalMarks = internalMarks;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
    public String getContactInformation() {
        return contactInformation;
    }

    public void setContactInformation(String contactInformation) {
        this.contactInformation = contactInformation;
    }

    public List<InternalMark> getInternalMarks() {
        return internalMarks;
    }

    public void setInternalMarks(List<InternalMark> internalMarks) {
        this.internalMarks = internalMarks;
    }
}

public class InternalMark {
    private String type;
    private Integer score;

    public InternalMark(String type, Integer score) {
```

```
        this.type = type;
        this.score = score;
    }
    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public Integer getScore() {
        return score;
    }

    public void setScore(Integer score) {
        this.score = score;
    }
}
```