**Design a Task Executor for a Car Workshop.**  
The Car workshop has following Employees on the payroll:

|  |  |
| --- | --- |
| **Employee Name** | **Designation** |
| Joe | Trainee |
| Smith | Expert |
| Walker | Employee |

Following tasks/duties are performed in the workshop:

|  |  |  |
| --- | --- | --- |
| **Task Name** | **Service Fee($)** | **Time Taken(Hours)** |
| Car-Wash | 100 | 2 |
| Car-Repair | 1000 | 5 |
| Car-Paint | 1100 | 4 |

On any given day, a schedule is created in the morning, in which task/tasks are assigned to each Employee. Example, schedule of 2-Jan-2015:

|  |  |
| --- | --- |
| **Employee Name** | **Task Name** |
| Joe | Car-Wash |
| Car-Repair |
| Car-Paint |
| Smith | Car-Repair |
| Walker | Car-Paint |
| Car-Repair |

**Assignment:**   
1. Design and implement Task, Employee and Schedule classes.  
2. Design and Implement Executor which will schedule and execute tasks of all employees

3. All Employees will start their work in Parallel (multi-threaded).   
4. There can be 2 strategies of Task Prioritization –

* Tasks can be prioritized based on the time taken. More time consuming task should be executed prior to other lesser time-consuming tasks assigned to that employee.

Expected output

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee Name** | **Task Name** | **Time taken** | **Fee** |
| Joe | Car-Repair | 5 | 1000 |
| Car-Paint | 4 | 1100 |
| Car-Wash | 2 | 100 |
| Smith | Car-Repair | 5 | 1000 |
| Walker | Car-Paint | 5 | 1100 |
| Car-Repair | 4 | 100 |

* Tasks can be prioritized based on Service Fee. A Task which charges more service fee should be executed first.  
  Expected output

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee Name** | **Task Name** | **Time taken** | **Fee** |
| Joe | Car-Paint | 4 | 1100 |
| Car-Repair | 5 | 1000 |
| Car-Wash | 2 | 100 |
| Smith | Car-Repair | 5 | 1000 |
| Walker | Car-Paint | 4 | 1100 |
| Car-Repair | 5 | 1000 |

**Expectations:**

1. Programming using Interfaces is desired.
2. Use composition wherever required.
3. The design should be flexible to accommodate more task prioritization strategies in future.
4. Classes should be properly packaged.
5. Class names, method and variable names should follow proper Java naming conventions.
6. Correct access modifiers should be used.
7. Use Java Generics wherever applicable.
8. Usage of Java 5 or later features desirable.