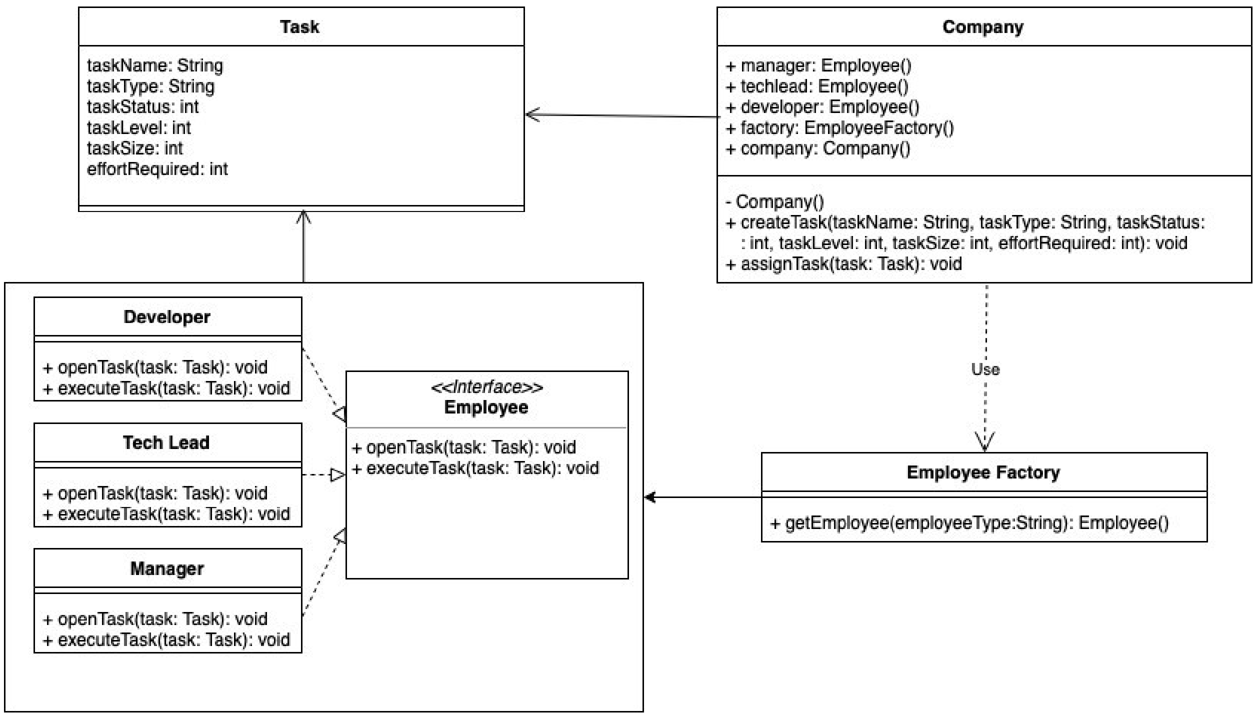
Object Oriented Analysis and Design using Java (UE19CS353)

Assignment-2: Design Patterns

Name: Royston E Tauro
SRN: PES1UG19CS396
Section: F

UML:


Task
taskName: String
taskType: String
taskStatus: int
taskLevel: int
taskSize: int
effortRequired: int
Company
+ manager: Employee()
+ techlead: Employee()
+ developer: Employee()
+ factory: EmployeeFactory()
+ company: Company()
- Company()
+ createTask(taskName: String, taskType: String, taskStatus: : int, taskLevel: int, taskSize: int, effortRequired: int): void
+ assignTask(task: Task): void
Developer
+ openTask(task: Task): void
+ executeTask(task: Task): void
Tech Lead
+ openTask(task: Task): void
+ executeTask(task: Task): void
Manager
+ openTask(task: Task): void
+ executeTask(task: Task): void
<<Interface>>
Employee
+ openTask(task: Task): void
+ executeTask(task: Task): void
Use
Employee Factory
+ getEmployee(employeeType:String): Employee()

Code:
App.java:

```
public class App { public static void main(String[] args) {
    Company company = Company.getCompany();
}
```

Company.java:

```
class Company

{
    private static Company company = null;
    private Employee manager; private Employee techLead; private Employee
developer; private EmployeeFactory factory;
    Company()
    {
        ef = new EmployeeFactory();
        manager = ef.getEmployee("Manager");
        techLead = ef.getEmployee("TechLead");
        developer = ef.getEmployee("Developer");
    }
    public static Company getCompany()
    {
        if (company == null)
            company = new Company();
        return company;
    }
    public void createTask(String taskName, String taskType, int taskStatus, int
taskLevel, int taskSize, int effortRequired)
    {
        Task task = new Task(taskName, taskType, taskStatus, taskLevel,
taskSize, effortRequired);
        assignTask(task);
    }
    public void assignTask(Task task)
    {
        if(task.getTaskLevel() == 3)
        {
            manager.openTask(task); manager.executeTask(task);
        }
        else if(task.getTaskLevel() == 2){
            manager.openTask(task); techLead.executeTask(task);
        }
        else if(task.getTaskLevel() == 1)
        {
            techLead.openTask(task); developer.executeTask(task);
        }
    }
}
```

Employee.java:

```java
public interface Employee {
    void openTask(Task task); void executeTask(Task task);
    }
```

Manager.java:

```java
public class Manager implements Employee {
    public void openTask(Task task) {
        System.out.println("Manager opens the task");
        System.out.println("Task level is " + task.getTaskLevel());
    }
    public void executeTask(Task task) {
        System.out.println("Manager executes the task");
        System.out.println("Task level is " + task.getTaskLevel());
    }
}
```

TechLead.java:

```java
public class TechLead implements Employee {
    public void openTask(Task task) {
        System.out.println("TechLead opens the task");
        System.out.println("Task level is " + task.getTaskLevel());
    }
    public void executeTask(Task task) {
        System.out.println("TechLead executes the task");
        System.out.println("Task level is " + task.getTaskLevel());
    }
}
```

Developer.java:

```java
public class Developer implements Employee {
    public void openTask(Task task) {
        System.out.println("Developer opens the task");
        System.out.println("Task level is " + task.getTaskLevel());
    }
    public void executeTask(Task task) {
        System.out.println("Developer executes the task");
        System.out.println("Task level is " + task.getTaskLevel());
    }
}
```

Task.java:

```java
public class Task {
    private String taskName;
    private String taskType;
    private int taskStatus;
    private int taskLevel;
    private int taskSize;
    private int effortRequired;
    Task(String taskName, String taskType, int taskStatus, int taskLevel, int 
taskSize, int effortRequired)
    {
        this.taskName = taskName;
        this.taskType = taskType;
        this.taskStatus = taskStatus;
        this.taskLevel = taskLevel;
        this.taskSize = taskSize;
        this.effortRequired = effortRequired;
    }
    public String getTaskName() {
        return taskName;
    }

    public void setTaskName(String taskName) {
        this.taskName = taskName;
    }

    public String getTaskType() {
        return taskType;
    }

    public void setTaskType(String taskType) {
        this.taskType = taskType;
    }

    public int getTaskStatus() {
        return taskStatus;
    }

    public void setTaskStatus(int taskStatus) {
        this.taskStatus = taskStatus;
    }

    public int getTaskLevel() {
        return taskLevel;
    }

    public void setTaskLevel(int taskLevel) {
        this.taskLevel = taskLevel;
    }

    public int getTaskSize() {
        return taskSize;
    }

    public void setTaskSize(int taskSize) {
        this.taskSize = taskSize;
    }

    public int getEffortRequired() {
        return effortRequired;
    }

    public void setEffortRequired(int effortRequired) {
        this.effortRequired = effortRequired;
    }
}
```

EmployeeFactory.java:

```java
public class EmployeeFactory {
    public Employee getEmployee(String employeeType){
        if(employeeType == null){
            return null;
        }
        if(employeeType.equalsIgnoreCase("Manager")){
            return new Manager();
        }
        else if(employeeType.equalsIgnoreCase("TechLead"))
        {
            return new TechLead();
        }
        else if(employeeType.equalsIgnoreCase("Developer"))
        {
            return new Developer();
        }
        return null;
    }
}
```