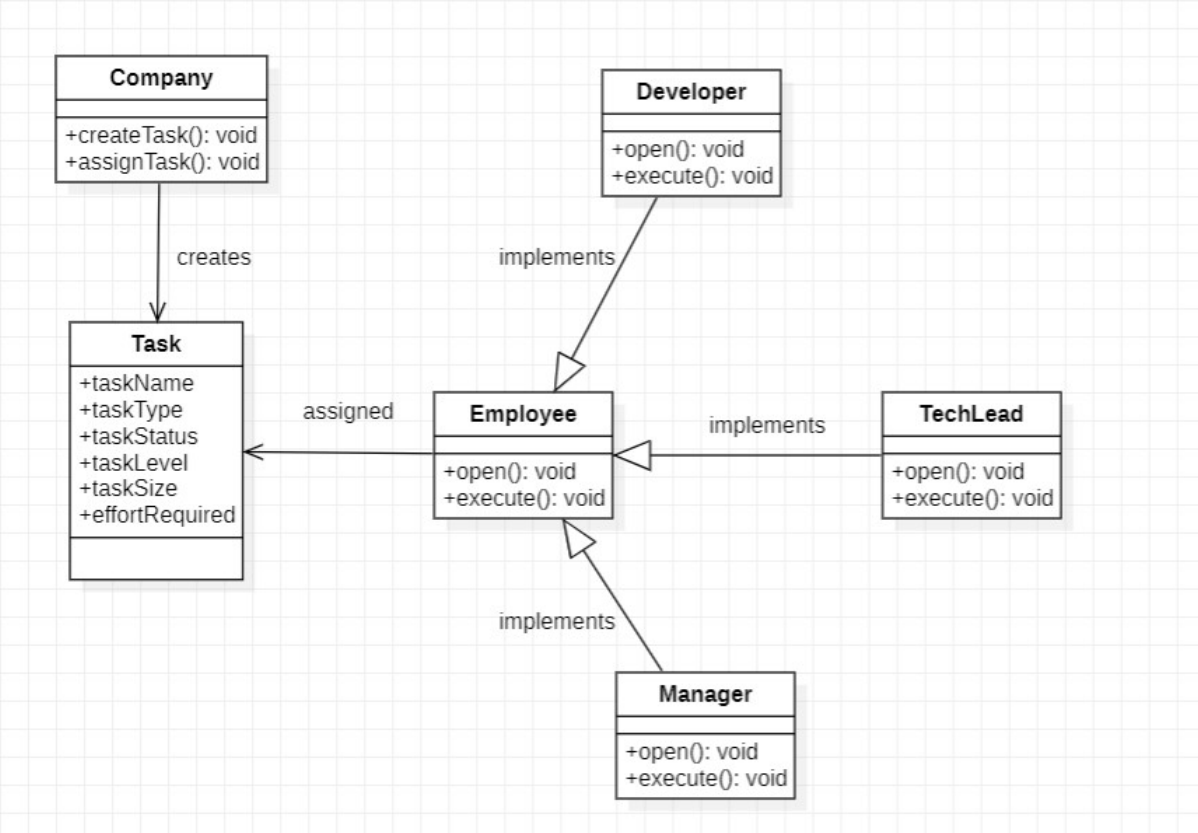
Object Oriented Analysis and Design using Java (UE18CS353)

Assignment-2: Design Patterns

Name – B Pravena **Sec** - B

SRN – PES2UG19CS076

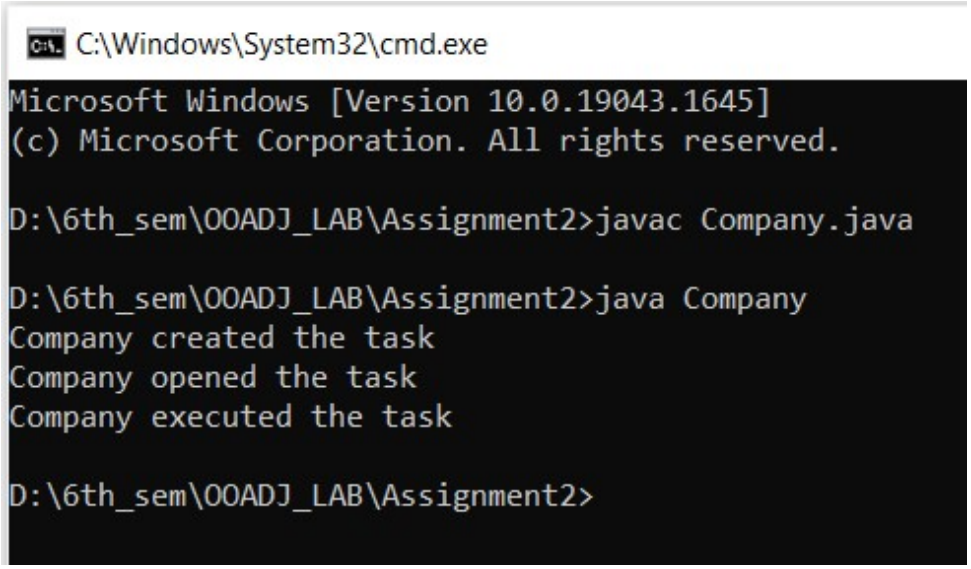
Class Diagram -:



Design Patter Information -:

In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time. The purpose of the singleton class is to control object creation, limiting the number of objects to only one. The singleton allows only one entry point to create the new instance of the class called the global access point.

Output -:



Code -:

Company.java 3

D: > 6th_sem > OOADJ_LAB > Assignment2 > Company.java > Company > main(String[])

```java
interface Employee
{
    public void showEmployeeDetails();
}

class Task
{
    private String taskName;
    private String taskType;
    private String taskStatus;
    private int taskLevel;
    private int taskSize;
    private int effortRequired;

    public Task(String taskName, String taskType, String 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;
    }
```

```
35      public String getTaskType()
36      {
37          return taskType;
38      }
39
40      public void setTaskType(String taskType)
41      {
42          this.taskType = taskType;
43      }
44
45      public String getTaskStatus()
46      {
47          return taskStatus;
48      }
49
50      public void setTaskStatus(String taskStatus)
51      {
52          this.taskStatus = taskStatus;
53      }
54
55      public int getTaskLevel()
56      {
57          return taskLevel;
58      }
59
60      public void setTaskLevel(int taskLevel)
61      {
62          this.taskLevel = taskLevel;
63      }
64
65      public int getTaskSize()
66      {
67          return taskSize;
68      }
```

```
70      public void setTaskSize(int taskSize)
71      {
72          this.taskSize = taskSize;
73      }
74
75      public int getEffortRequired()
76      {
77          return effortRequired;
78      }
79
80      public void setEffortRequired(int effortRequired)
81      {
82          this.effortRequired = effortRequired;
83      }
84
85      public void showTaskDetails()
86      {
87          System.out.println("Task Name: " + taskName);
88          System.out.println("Task Type: " + taskType);
89          System.out.println("Task Status: " + taskStatus);
90          System.out.println("Task Level: " + taskLevel);
91          System.out.println("Task Size: " + taskSize);
92          System.out.println("Effort Required: " + effortRequired);
93      }
94  }
```

```java
class Manager implements Employee
{
    private String name;
    private long empId;
    private String position;

    public Manager(long empId, String name, String position)
    {
        this.empId = empId;
        this.name = name;
        this.position = position;
    }

    @Override
    public void showEmployeeDetails()
    {
        System.out.println(empId + " " + name + " " + position);
    }

    public void createTask(Task task)
    {
        System.out.println("Manager created the task");
    }

    public void openTask(Task task)
    {
        System.out.println("Manager opened the task");
    }

    public void executeTask(Task task)
    {
        System.out.println("Manager executed the task");
    }
}
```

```java
class Developer implements Employee
{
    private String name;
    private long empId;
    private String position;

    public Developer(long empId, String name, String position)
    {
        this.empId = empId;
        this.name = name;
        this.position = position;
    }

    @Override
    public void showEmployeeDetails()
    {
        System.out.println(empId + " " + name + " " + position);
    }

    public void createTask(Task task)
    {
        System.out.println("Developer created the task");
    }

    public void openTask(Task task)
    {
        System.out.println("Developer opened the task");
    }

    public void executeTask(Task task)
    {
        System.out.println("Developer executed the task");
    }
}
```

```java
class TechLead implements Employee
{
    private String name;
    private long empId;
    private String position;

    public TechLead(long empId, String name, String position)
    {
        this.empId = empId;
        this.name = name; this.position = position;
    }

    @Override
    public void showEmployeeDetails()
    {
        System.out.println(empId + " " + name + " " + position);
    }

    public void createTask(Task task)
    {
        System.out.println("TechLead created the task");
    }

    public void openTask(Task task)
    {
        System.out.println("TechLead opened the task");
    }

    public void executeTask(Task task)
    {
        System.out.println("TechLead executed the task");
    }
}
```

```java
class CompanyDirectory
{
    private Manager manager;
    private TechLead techLead;
    private Developer developer;

    public CompanyDirectory(Manager manager, TechLead techLead, Developer developer)
    {
        this.manager = manager;
        this.techLead = techLead;
        this.developer = developer;
    }

    public void createTask(Task task)
    {
        System.out.println("Company created the task");
    }

    public void openTask(Task task)
    {
        System.out.println("Company opened the task");
    }

    public void executeTask(Task task)
    {
        System.out.println("Company executed the task");
    }
}
```

```java
class Company
{
    public static void main(String[] args)
    {
        Manager manager = new Manager(12345, "John", "Manager");
        TechLead techLead = new TechLead(12346, "Peter", "TechLead");
        Developer developer = new Developer(12347, "Paul", "Developer");
        CompanyDirectory companyDirectory = new CompanyDirectory(manager, techLead, developer);
        Task task = new Task("Task1", "TaskType1", "TaskStatus1", 1, 1, 1);
        companyDirectory.createTask(task);
        companyDirectory.openTask(task);
        companyDirectory.executeTask(task);
    }
}
```