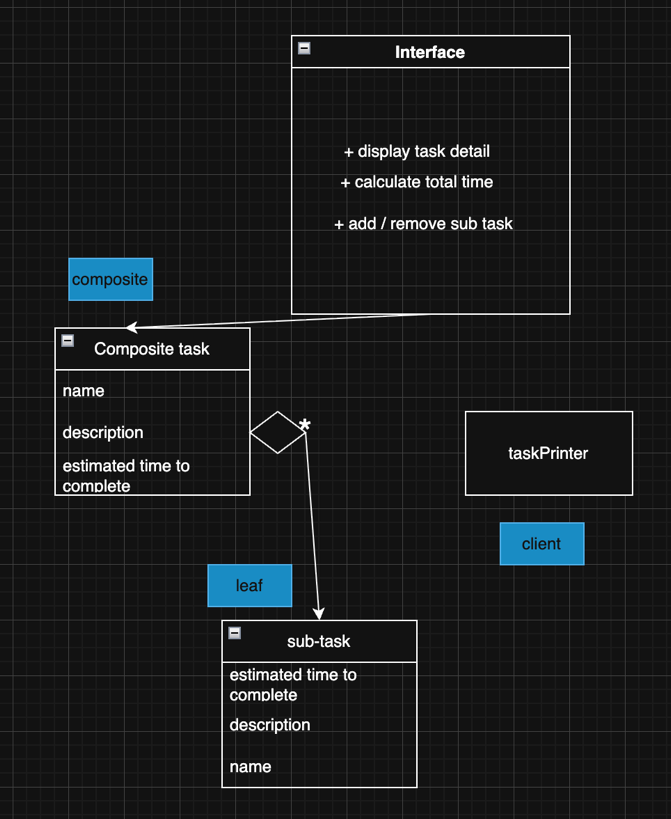
1. Identify the classes needed to represent the individual and composite tasks. What classes or objects play which participant role in the composite pattern?
2. Draw a class diagram to illustrate the relationships between these classes.



1. Give pattern-relevant code (Pseudocode/Java/C++/C#):
   * 1. Write pseudocode or code for the Component interface.

interface Task {

method displayDetails()

method calculateTotalTime()

method addSubtask(Task task)

method removeSubtask(Task task)

}

* + 1. Write pseudocode or code for one of the Leaf class.

class IndividualTask implements Task {

string name

string description

double estimatedTime

constructor(name, description, estimatedTime) {

this.name = name

…

}

method displayDetails() {

print("Task: " + this.name)

print("Description: " + this.description)

print("Estimated Time: " + this.estimatedTime + " hours")

}

method calculateTotalTime() {

return this.estimatedTime

}

}

* + 1. Write pseudocode or code for one class that acts as the  Composite classes  
       class CompositeTask implements Task {  
        string name   
        string description   
        list<Task> subtasks

constructor(name, description) {   
 this.name = name   
 …  
}  
   
method addSubtask(Task task) {   
 this.subtasks.add(task)   
}   
  
method removeSubtask(Task task){   
 this.subtasks.remove(task)   
}   
  
method displayDetails() {   
 print("Project: " + this.name)   
 print("Description: " + this.description)

print("Sub-tasks:") for each task in this.subtasks { task.displayDetails() }  
}   
  
method calculateTotalTime() {   
 totalTime = 0 for each task in this.subtasks { totalTime += task.calculateTotalTime() } return totalTime }

}

1. Client Code: Behavior Simulation  
   Write code/pseudocode or class stubs to simulate the following scenarios:
   * + Creating a composite task with sub-tasks

projectA = new CompositeTask("Project A", "Launch of new product line")

// Add sub-tasks to the composite task

projectA.addSubtask(task1)

projectA.addSubtask(task2)

* + - A TaskPrinter that takes any kind of Task as input and prints its details uniformly.
    - The TaskPrinter calls calculateTotalTime() on both an individual and composite task while displaying the details.

class TaskPrinter {

method printTaskDetails(Task task) {

task.displayDetails()

totalTime = task.calculateTotalTime()

print("Total estimated time: " + totalTime)

}

}

printer = new TaskPrinter()

printer.printTaskDetails(IndividualTask task1)

printer.printTaskDetails(CompositeTask projectA)