Data structures project

 WMS

Worflow Management System

By

Omar Shehab |202403769|

Lama Razzouk |202403296|

Wissam Moh. Haydar |202403603|

**Objective:**

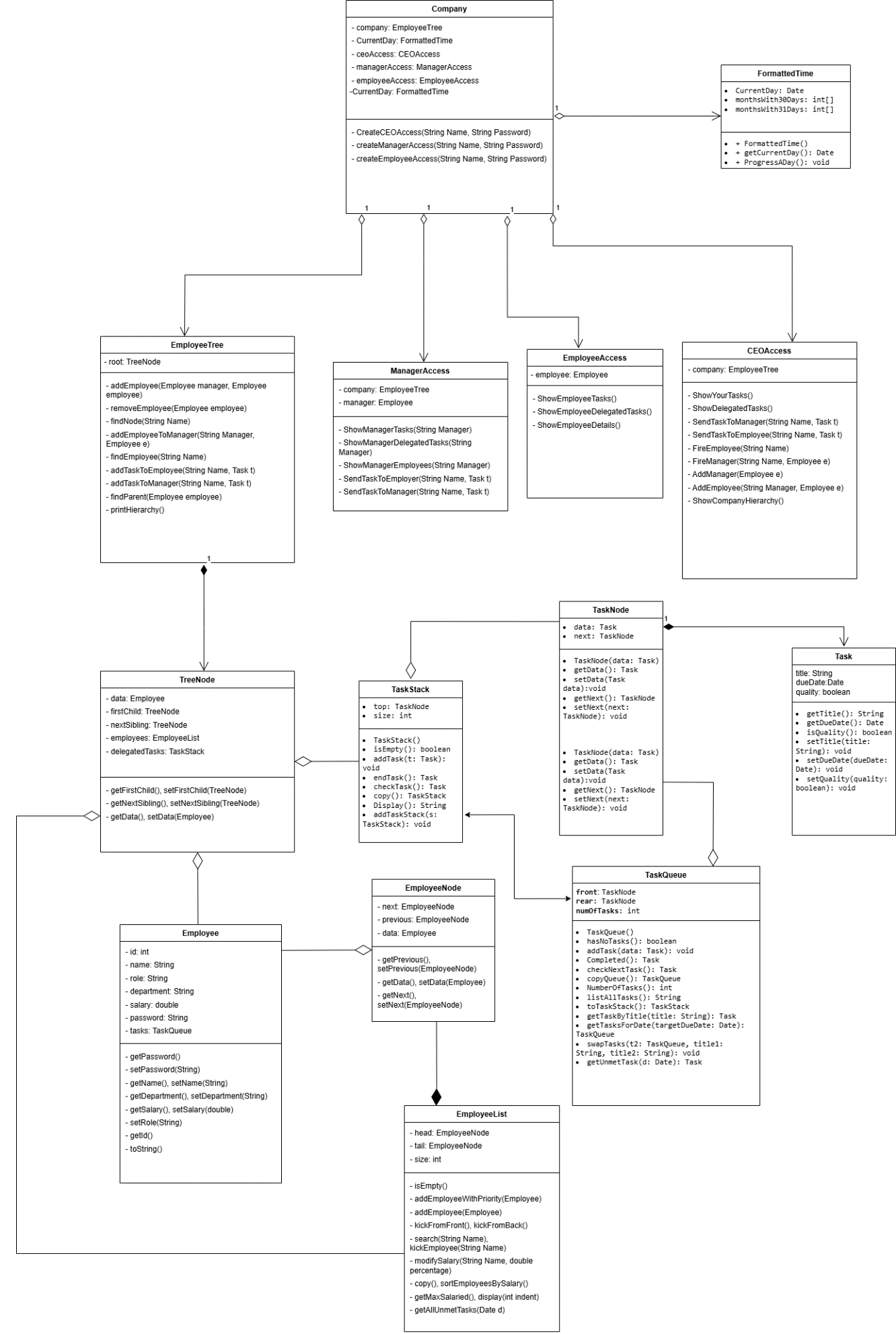
The problem at hand is that when companies grow big , handling all the data of employees and managers becomes difficult and inefficient in traditional methods , and as such this project aims to fix this by arranging company hierarchy and tasks withing a java project that handles this data appropriately , from showing it , to manipulating it based on your position in the company .

**Tasks of Each Member:**

**Wissam Haydar :** UML , EmployeeTree Class , TreeNode Class.

**Omar Shehab :** EmployeeList , EmployeeNode , FormattedTime ,Employee , TaskNode , TaskStack , Task , Company , CEOAccess, ManagerAccess , EmployeeAccess Classes

**Lama Razzouk :** TaskQueue, [WelcomePage, Employeeframe, Manager, CEO (Jframe design+source) ]



**Classes and Functions:**

EmployeeList:

* **Constructor:** initiates an empty list.
* **addEmployeeWithPriority(Employee e) :** inserts an employee at the very front of the list.
* **addEmployee(Employee e) :** inserts an employee at the end of the list
* **kickFromFront() :** kicks the employee at the very front of the list if empty do nothing
* **kickFromBack():** kicks the employee at the very back of the list if empty do nothing
* **search(String Name):** returns an employee after searching for it by name , if not found then return null
* **kickEmployee(String Name):** kicks an employee by name , if not found , do nothing
* **modifySalary(String Name,*double* percentage) :** unused but still present , it’s supposed to give a raise to an employee by name by certain percentage (if he was 10,000 and the percentage was 10%the salary becomes 11,000 )
* **copy :** returns an exact replica of the EmployeeList
* **sortEmployeesBySalary():** returns an exact replica of the EmployeeList but sorted by salary in ascending order
* **getMaxSalaried():** returns the employee with the highest Salary
* **display(*int* indent):** returns a string with all the employees with an indent value for spaces before each employee
* **getAllUnmetTasks(Date d):** returns a taskStack with all the current (at the front) unmet Tasks from each employee in the list
* **isEmpty() :** returns true if the list is empty , false other wise

TaskStack :

* **Constructor :** initiates an empty stack
* **isEmpty()** : returns true if and only if the stack is empty
* **addTask(Task t) :** adds a task at the top of the stack
* **endTask():** returns the current task and removes it from the stack if there is no tasks return a null
* **checkTask():** returns the current task without removing it from the stack if there is no tasks return a null
* **copy() :** returns a replica of the stack at hand
* **Display() :** returns a string with each task in order
* **addTaskStack(TaskStack s) :** adds a TaskStack to the current TaskStack

TaskNode:

* **Constructor(Task data)** : initiates a Task Node with no next (null) and a Task data that will be assigned to the data Attribute
* **Getters and setters** for all

Task :

* **Constructor (String title , Date dueDate) :** initiates a Task with a quality attribute set to true by default , assigns the title to the this.title attribute , and sets the year of the due date to itself -1900 since the equivalent of 2024 in the Date java class is 124 and then assigns this date to the this.dueDate attribute
* **Getters and setters** for all
* **toString() :** returns a string representation of the Task object , while accounting for the properties of the Date class

TaskQueue:

- **Constructor :** initiates an empty qucoeue.

- **hasNoTasks() :** return true if the queue is empty (employee has no ongoing tasks), or false if the queue is not empty (employee has ongoing tasks)

- **addTask(Task data) :** adds a task at the rear of the queue.

-**Completed()** : removes the front of the queue if the queue is not empty and returns it, else returns null.

-**checkNextTask() :** returns the next Task fromt the front if the queue is not empty, else returns null.

-**copyQueue() :** copies the queue and returns the copy.

-**NumberOfTasks():** returns the number of tasks in the queue.

-**toTaskStack() :** pushes the tasks in the queue to a stack and returns the new stack.

-**getTaskByTitle(String title) :** searches for a task byt its title and returns it if the Queue is not empty, else returns null.

-**getTasksForDate(Date targetDueDate) :** searches for all the Tasks due to certain date, puts them in a new queue and returns it.

-**swapTasks(TaskQueue t2, String title1, String title 2) :** searches for 2 tasks for 2 different employees by their title and swaps them. If at least one of the tasks is not found, it does nothing.

-**getUnmetTask(Date d):** returns unmet tasks (quality unmet or past dueDate) if not completed yet. If task is completed before the dueDate and the quality is met then it returns null.

EmployeeNode:

* **Constructor (Employee e) :** initiates a new employee node and assigns e as the object data and sets the next attribute to null
* **Getters and setters** for all

Employee :

* **Constructor(String name , String role , String department ,double salary , String pass) :** sets the name , role ,department ,salary and password attributes to their respective values in the parameters , and assigns the employee a new unique id (since currid attribute is static and auto increments with each object instance), and initiates a new TaskQueue() for the employee’s tasks
* **Getters and setters** for all but TaskQueue and ID (ID is permanent)
* **Getter** for ID
* **toString() :** returns a string representing the employee , not including his password

FormattedTime:

* **Constructor():** sets the CurrentDay attribute as 1-1-2024 equivalent in java Date class
* **Getter** for current Day
* **ProgressADay() :** simulates the passing of a day by manipulating the currentDay object in accordance with rules on how calendars function

EmployeeAccess:

* **Constructor(Employee employee) :** sets the employee attribute as the employee paremeter
* **showYourTasks() :** returns a string showing all the tasks of the employee
* **showOwn() :** returns a string showing the current employee details
* **completeTask() :** completes the current task for the current employee and returns a string containing the current completed task with a message confirming its completion
* **returnName() :** returns a string containing the current employee’s Name

ManagerAccess :

* **Constructor(TreeNode manager) :** sets the manager attribute as the parameter
* **showOwn() :** returns a string showing the current manager details
* **completeTask() :** completes the current task for the current manager and returns a string containing the current completed task with a message confirming its completion
* **showDelegatedTasks() :** returns a string containing all the delegated tasks from the employees to the manager stack
* **sendTaskToEmployee(String employeeName, Task task) :** sends the task to said employee if found under the current manager , otherwise do nothing
* **addEmployee(Employee employee)** : adds an employee to the current manager
* **removeEmployee(String employeeName) :** searches for said employee if he is found delegate all his tasks to the current manager and then kick him otherwise do nothing
* **showEmployeeTasks(String employeeName):** returns a string containing the employee’s tasks If found , otherwise throws an IllegalArgumentException
* **showEmployee(String Name):** returns the string representation of the employee’s details
* **showEmployees() :** returns a string representation of all the employees details under the current manager.

WelcomePage:

**-Login button:** when action performed , it takes the text of the name text field and the password textfield and compares them with the company’s data using createAccess methods. Returns the position access object and returns the other positions access objects as null. And when one of the access objects is not null anymore. It opens its new screen ( CEO, Manager, EmployeeJframe) and it disposes.

CEO:

-**Hierarchy button:** when action performed, it sets the InfoTextArea text as the returned value from ShowCompanyHierarchy() (displays the details about employees from ceo to employees)

- **Details button:** when action performed, it sets the InfoTextArea text as the returned value from ShowOwn() (details of the ceo)

-**Tasks button :** when action performed, it sets the InfoTextArea text as the returned value from ShowYourTasks() (Tasks of the ceo)

-**Dlg button :** when action performed, it sets the InfoTextArea text as the returned value from showDelegatedTasks() ( ceo’s delegated tasks)

- **Manager Dlg. Tasks button :** when action performed, shows the desired manager’s delegated tasks (takes the MnameTextField value to search for manager)

-**Employees button:** when action performed, shows details about the desired manager’s employees (takes the MnameTextField value to search for manager)

-**get Dlg. Tasks button :** when action performed, takes the desired manager’s delegated tasks and give them to the ceo and then displays the ceo’s new delegated tasks (takes the MnameTextField value to search for manager)

-**Manager Tasks button:** when action performed, displays the desired manager’s tasks(takes the MnameTextField value to search for manager)

-**Manager Details button:** when action performed, displays the desired manager’s details(takes the MnameTextField value to search for manager)

- **Kick prev. button:** when action performed, takes the MnameTextField value to search for manager and takes the new manager’s details, kick the first and adds the second as manager and then displays his informations.

- **Add button :** when action performed, takes the new manager’s details and adds them as manager and then displays their informations.

- **Send button :** when action performed, takes the MnameTextField value to search for manager and the task’s information(title , dueDate) and then sends the task to the desired manager and displays his new Task queue.

- **pass a day button :** when action performed, advances the date by one day.

-**Complete current task button**: when action performed, complete the ceo’s ongoing task and displays that it was completed.

-**Log out button:** when action performed, logs out of the ceo access page (while keeping the data) and then goes back to the login page.

Manager:

- **Details button:** when action performed, it sets the InfoTextArea text as the returned value from ShowOwn() (details of the manager)

-**Tasks button :** when action performed, it sets the InfoTextArea text as the returned value from ShowYourTasks() (Tasks of the manager)

-**Dlg button :** when action performed, it sets the InfoTextArea text as the returned value from showDelegatedTasks() ( manager’s delegated tasks)

**- Add button :** when action performed, takes the new employee’s details and adds them and then displays their informations.

**- Employees button:** when action performed, shows details about all the employees under manager’s department.

**- Kick button:** when action performed, takes the enameTextField value to search for employeeand takes the new employee’s details, kick them.

-**Employee Tasks button:** when action performed, displays the desired employee’s tasks(takes the enameTextField value to search for employee)

-**Employee Details button:** when action performed, displays the desired employee’s details (takes the enameTextField value to search for employee)

- **Send button :** when action performed, takes the enameTextField value to search for employee and the task’s information(title , dueDate) and then sends the task to the desired employee and displays his new Task queue.

- **pass a day button :** when action performed, advances the date by one day.

-**Complete current task button**: when action performed, complete the manager’s ongoing task and displays that it was completed.

-**Log out button:** when action performed, logs out of the manager access page (while keeping the data) and then goes back to the login page.

EmployeeJframe:

-**Show informations button :** when action performed, it sets the InfoTextArea text as the returned value from ShowOwn() (details of the employee)

-**Show Tasks button :** when action performed, it sets the InfoTextArea text as the returned value from ShowYourTasks() (Tasks of the employee)

- **pass a day button :** when action performed, advances the date by one day.

-**Complete current task button**: when action performed, complete the employee manager’s ongoing task and displays that it was completed.

-**Log out button:** when action performed, logs out of the employee access page (while keeping the data) and then goes back to the login page.