HR SCHEDULING SYSTEM
- group 044 -

Specification

The HR scheduling system manages scheduling for an organization: shifts and meetings for employees. It maintains a roster of employees at the organization (which may be modified as employees join or leave the organization), along with scheduling constraints. The manager of the organization can assign employees work according to specific rules:

- Minimum and maximum number of hours an employee may work per week.
- Minimum and maximum number of employees for a given time, as too few employees can't get the work done, while an excessive number of employees would cause redundancy.
- Leave tracking, to avoid scheduling an employee on holiday
- Scheduling contiguously where possible.

These scheduling requirements translate to requirements for the general employee management: the number of working hours and leave of absence time relates to the salary of the employee (part time vs full time), while the workplace requirements are a property of the organization.

All persistent state (employee data, schedules, and so on) must be serialized to an on-disk database as it is updated, and loaded from the database on start-up, to guarantee data integrity even if the system fails.

Entity classes

Organization: Map of Employees (key ID, value Employee)
Employee: name, employee ID, Events(list of events), salary(hourly wage), maxHoursPerWeek(max hours can work in a day), schedulable(Boolean) (false if the Employee is going to leave).

Calendar: Map with date as key and List of Events as value.
Event (abstract class): start, name, location, duration, capacity(maximum number of Employee for an event)

> Shift: concrete instantiation of Event
> Meeting: a subclass of Event contains attributes of meeting host (Employee), participants (set of Employees) for this meeting

Remark: need to have an interface + dependency inversion so setting/getting properties is backed by a database class on the outside.

Use case classes

Scheduler(Abstract Class): Scheduler class aims to Create & Delete an event to the Calendar. This Abstract class have several abstract methods:

> constructor:
> createEvent(date): create an Event and add to the Calendar by its date.

deleteEvent(date): delete an existing Event from the Calendar
assign(Employee employee, Event event): This method will assign tasks to the employee. By adding the Event to the Employee's attribute: Events.

ShiftScheduler(Subclass of Scheduler): override scheduler and create & delete shift.
createEvent: create a Shift then add it to the Calendar.
deleteEvent: Delete an existing Shift from the Calendar.

MeetingScheduler(Subclass of Scheduler): override scheduler and create & delete meetings.
createEvent(Employee holder, ArrayList<Employee> participants): create a Meeting event according to the holder and participants. Then, add it to the Calendar
deleteEvent: Use the Superclass delete event

EmployeeModifier: A Manager has the power to modify employees in this organization.

addEmployee(): Add an employee to the organization. Assign a job through Shift Scheduler until his/her workload is full. Used when hiring.

removeEmployee(): Delete an employee from the organization after some time. For a certain time, marks the employee unschedulable, cancels events after that time, and prevents scheduling during that time. Used when firing.

Controllers:

EmployeeManager: enable users to access EmployeeManager in use case classes.

EventManager: enable user to access Scheduler in use case classes.

External Interfaces:

Command line that is used to access EmployeeManager, EventManager
Database class(es) implementing the interface for serialization/deserialization. SQLite backend.

If we add a GUI or a web frontend in phase 2, that would use the controller interfaces as-is.

Entity Classes

Event(abstract class)

- Event() - Constructor method of Event contains of following attributes:
    - start - the start time of the event, using java.time Instant variable.
    - duration - the amount of time that from event start to end.
    - name - the name of the event
    - location - the name of the specific address of the event.
    - capacity - the maximum number of employees for the event. In the consideration of efficiency, too many employees would cause redundancy.
- Getter for some attributes above.

1. Parent Class of:
    a. Shift
    b. Meeting

Meeting

- Meeting() - Constructor of Meeting:
    - super()
    - Holder - the Employee to host the meeting
    - Participants - list of Employees who attend the meeting

1. Child class of “Event”
2. Employee

Shift

- Constructor of Shift:
    - super()

1. Child class of “Event”

Employee	
● Employee() - Constructor of Employee contains of following attributes: ○ name - the name of an employee. Different employees might have the same name. ○ ID - the unique ID of the employee. ○ events - a list of events involved by the employee ○ maxHoursPerWeek - The max hours employee can work in a day ○ salary - hourly wage of the employee. ○ schedulable - a boolean variable indicating whether the employee can be assigned an event (due to leave) ● Getter for some attributes above.	1. Event 2. Organization

Organization	
● Organization() - constructor method of organization. All of the Employees in this Organization are stored in a dictionary with Int of Employee's ID as key and the corresponding Employee as value. ● getEmployee(int id) - based on id, return the corresponding employee. ● addEmployee(int id, Employee) - add an employee ● removeEmployee(Employee) - removes an employee	1. Employee

Calendar	
● Calendar() - constructor method of Calendar. ○ events - arraylist of all events ● addEvent(Event) -- adds an event to the event's list	1. Events

Use Case Classes

Scheduler	
• Scheduler() - An abstract class which includes several abstract methods: • createEvent() - create an Event and add to the Calendar by its date • deleteEvent() -delete an existing Event from the Calendar • assign() - Assigning Event to Employee	1. Event 2. Calendar 3. Employee

ShiftScheduler	
• ShiftScheduler() - An abstract class which includes several abstract methods: • Inherit create, delete and assign method from the superclass	1. Employee 2. Event 3. Calendar 4. Subclass of Scheduler

MeetingScheduler	
• createEvent() - Override create method from parent class Scheduler. Input more information as Holder(Employee), Participants(List of Employee). • inherit delete & assign method	1. Employee 2. Event 3. Clendar 4. Subclass of Scheduler

EmployeeModifier	
• EmployeeModifier() - constructor method of EmployeeModifier contains of attribute: ○ organization - an Organization object • hireEmployee(): Add an employee to the organization. • fireEmployee(): Delete an employee from the organization after a certain amount of time. Within that time, mark the employee unschedulable, thus stop assigning a new Event to him. After that, delete him from the organization and Events he participated in. • SalaryEvaluation(): Calculate the Salary for the Employee based on hourly salary and workTime.	• Employee • Organization • Event

Controller

EventManager	
controller that enables HR access to create Event, delete Event and assign Event to employees	Scheduler • ShiftScheduler • MeetingScheduler

EmployeeManager	

controller that enables HR access Employee Modifier to hire and fire employees	EmployeeModifier

External Interfaces:

Command / UI	
1. Implement a command line interface for exposing access to the employee roster and the calendars via their respective controllers.	EventManager EmployeeManager

Scenario Walk-Through

When a programmer Jack wants to enter a new company, the HR manager will firstly accept his application by EmployeeManager, which will call the use case classes's hireEmployee method, instantiating an Employee object with Jack's name, salary, and ID. After instantiating the Employee, it will instantiate a Calendar for Jack, and add the Employee to the Organization's map. Then, the manager will assign events to Jack by the SchedularManager. This controller will call several use case classes, which allow the manager not only to create a new event by time, location, duration, and event's name for Jack to do, but also assign other events that have already existed to Jack. When the weekly meeting is held, another special event called "Meeting" is created by using additional names of attenders and the meeting holder and then was assigned to Jack. Later, the manager decides to fire Jack, using the fireEmployee method of the EmployeeManager, Jack's schedulable is set to false, and he was announced to hand over his work in a few days. All the events sent to him after a few days are cancelled. After that, he is finally fired.

Progress Report

The HR scheduling system adds events to a calendar among certain rules based on the target employee's max or min of work hours and number of employees, enabling employees to join and exit the event, also evaluating their salaries.

By clean architecture, there are four layers in HR systems, **external interface -> controller -> use case classes -> entity,** we implement systems by first implementing our command line(**external interface)** to access EmployeeManager, SchedulerManager and SalaryManager(**controller)**,then **controller** will call schedule class(**use case classes)** to further access to our **entity**, where our class follows SOLID principle

We have finished a brief description of Classes(**Entity classes, user classes, controller)** with all constructor and methods covered. All relationships between each class and their function are clearly indicated in CRC cards and each CRC card only gets called from the upper layer and provides a path to access the lower layer.

From our Scenario Walk-Through, Human resources managers can use three different types of management systems to fire or hire employees, placing different events to employees, and calculating each employees salary based on their workload. Through the story of Jack, Jack gets hired through hireEmployee called by EmployeeManager. His work term is scheduled and assessed by SchedulerManager and Salary Manager and eventually he is fired by FireEmployee called by EmployeeManager. Through this story, we demonstrate the whole process of how an employee enters, works and leaves in one company.

Group works:

Yuxiang Lin: Finish Scenario Walk-Through and help fix some parts in CRC cards. Next will focus on designing parts of Entity Classes.

Kunhan Duan: go through and check all parts of my teammates task and make a summary in progress report. coming up with an open question in our group. Also adding and editing some parts on CRC cards and Scenario Walk-Through. Further planning to work on the Skeleton Program to implement our HR system.

Jiawei Chen: Scheduling in-person meetings. Define classes and clarify which layer those classes belong to based on CLEAN Architecture.

Haitao Zeng: Working on CRC model, discussing specification part with teammates, and entity part of skeleton program. I may work on entity class or use case class, in the future.

Alyssa Rosenzweig: Skeleton program, most of the specification.

Xinglin Chen: most part of CRC model, refining the specification.