ID2207 -Modern Methods in Software Engineering

Homework 4

Group 8

Joakim Ljung
Xiaoyng Sun

September 2020

Question:

Based on the SEP business description that you have in Canvas, develop a system design and object design document which includes:

- **Subsystem decomposition structure (using class diagrams)**
  Expected output in the document:
  1. A class diagram showing the decomposition.
- **Mapping subsystems to processors and components (hardware/software mapping) using UML deployment diagrams.**
  Expected output in the document:
  1. UML deployment diagram
  2. Brief description about the selected technologies and software components.
- **Persistent storage solution for the problem.**
  Expected output in the document:
  1. List of persistent objects
  2. Brief description of the selected storage managment strategy
- **Access control, global control flow, and boundary conditions.**
  Expected output in the document:
  (Access Control):
  1. Access matrix
  2. Brief description about security, authentication/authorization strategy, confidentiality of data, and network/infrastructure security.

  (Global control flow):
  1. Brief description of the selected control flow for the system

  (Boundary conditions):
  1. Boundary use cases
- **Applying design patterns to designing object model for the problem**
  Expected output in the document:

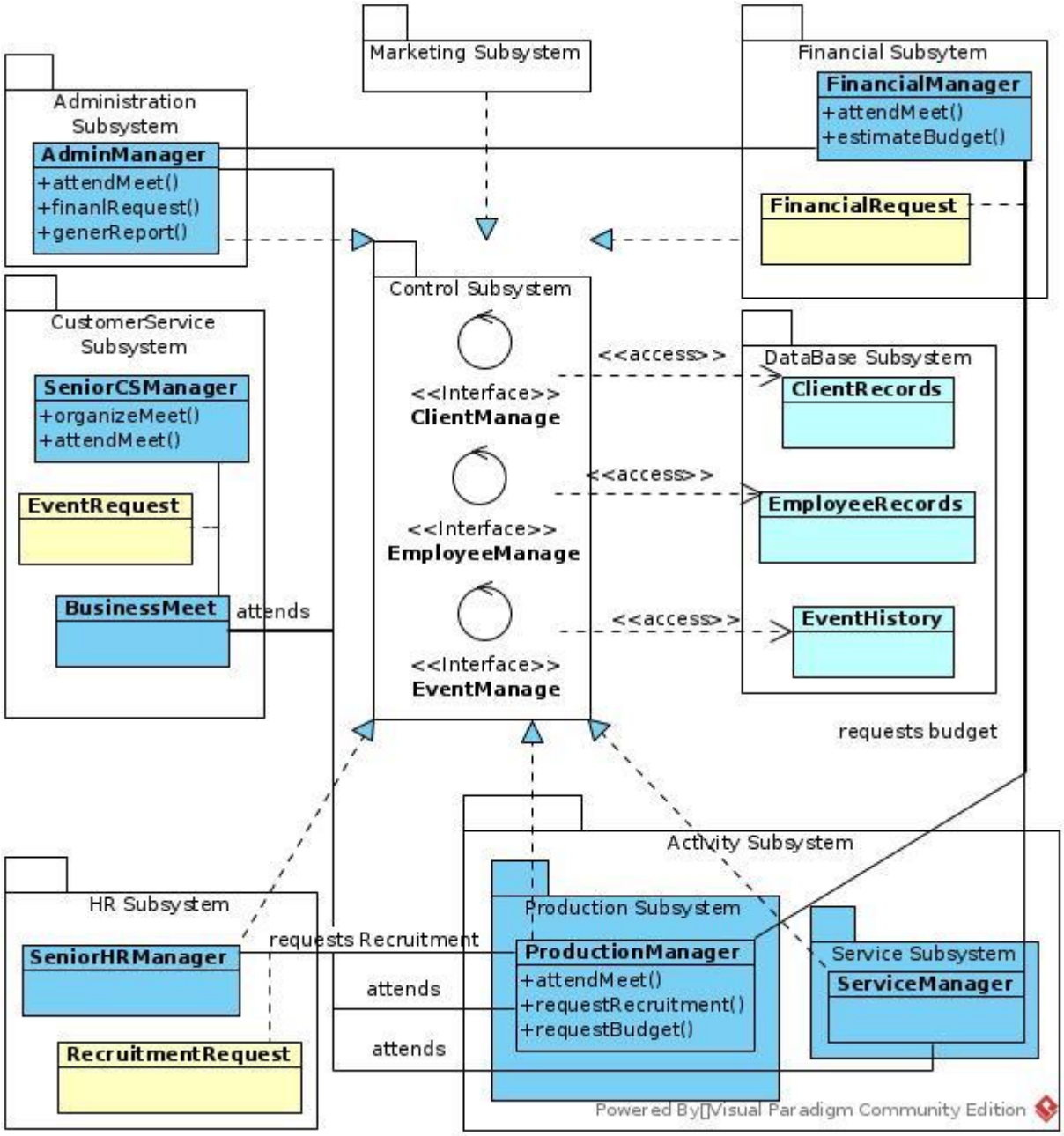
1. Brief description about the selected design patterns and why/how you are going to use them

- **Writing contracts for noteworthy classes**

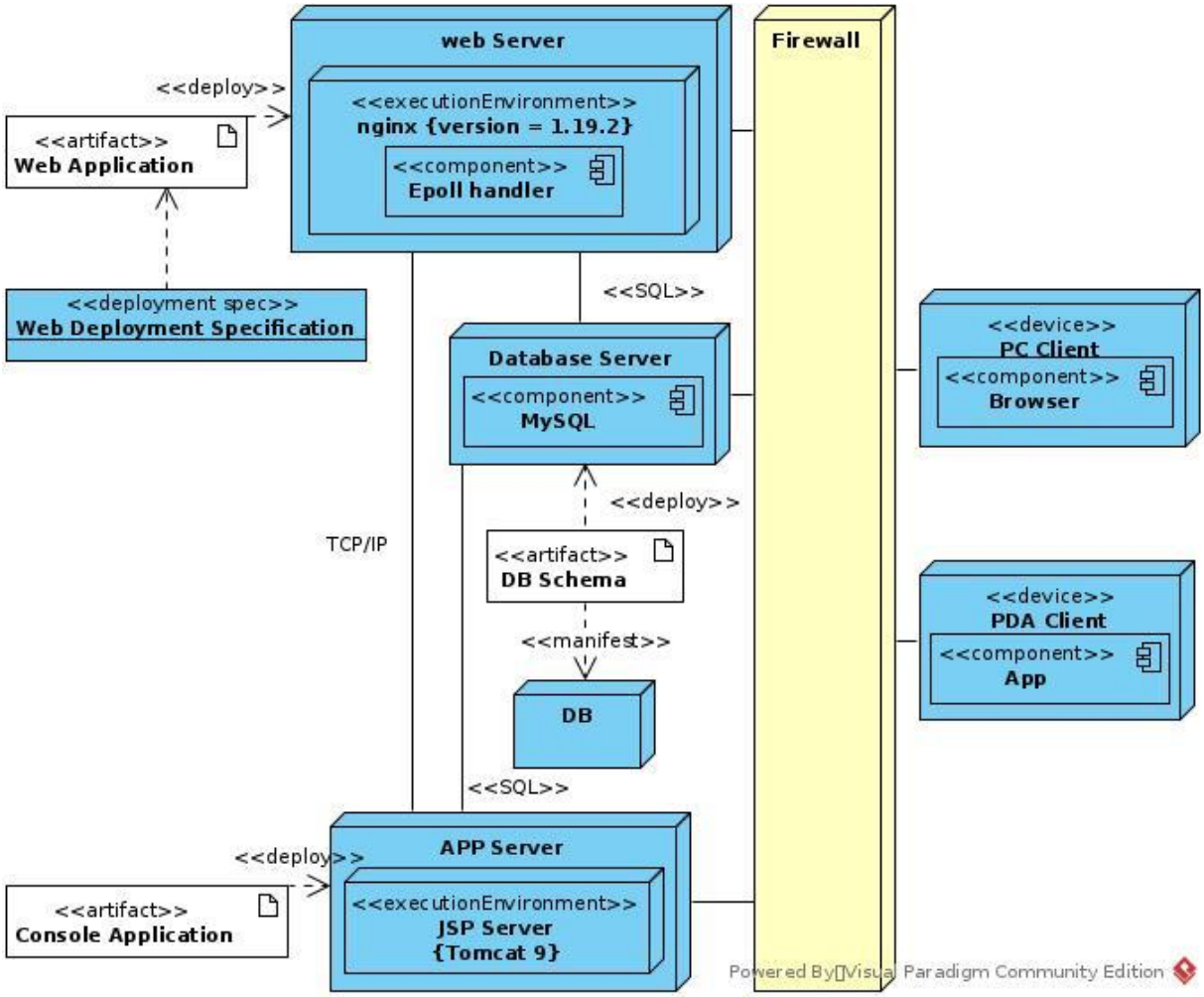
  <u>Expected output in the document:</u>

  1. contracts described in OCL

1. Subsystem Class Diagram



2. UML deployment Diagram



PC Client	The user's personal PC is pre-installed with a browser.
PDA Client	Personal Digital Assistant, like mobile phone and iPad, is pre-installed with SEP's APP.
Firewall	All operations on the server are filtered through the firewall.
Web Server	Nginx is a lightweight HTTP server, and supports forward and reverse proxy, Virtual host.
Web Deployment Specification	Web deployment specification lists most required hardware and software technologies.
Database Server	MySQL is small size, fast speed, and supports all mainstream operating systems.
APP Server	Tomcat processes dynamic requests and is a container for compiling JSP\Servlet. Nginx has a dynamic separation mechanism. Static requests can be directly processed through Nginx, and dynamic requests are forwarded to the background and handed over to Tomcat for processing.

3. Persistent Storage Solution

Persistent Objects

Client records
Employee records
Event history
Staff schedule
Scheduling information
Event requests

Storage Management Strategy

Database subsystem The database subsystem stores all of the client records, employee records, staff schedules, event requests and event history. It is important for all of this data to be accessed simultaneously and to be safe when a potential system crash occurs. This is why it is a database and not flat files. The database is a relational database, since the database will potentially handle large data sets and it will query over attributes.

4.1 Access Control

Object actors	ClientRecords	EmployeeRecords	EventHistory	EventRequests
Activity-Manager				initiateTask() receiveRequest() updateRequest()
Administration-Team	generateClientReport()	searchRecord() viewRecord() generateEmployeeReport()	generateEarningsReport()	finalizeRequest() receiveEventRequest() updateEventRequest()
Marketing-Team	searchRecord() viewRecord() generateClientReport()		generateEarningsReport()	
Financial-Manager	searchRecord() viewRecord() makeDiscount()	searchRecord() viewRecord()		receiveRequest() updateRequest() estimateBudget()
SeniorHR-Manager		searchRecord() viewRecord()		
HRTeam		searchRecord() viewRecord()		
Customer-Service				initiateNewEvent() redirectRequest()
SeniorCS-Manager	createRecord() searchRecord() viewRecord()			viewEventRequest() searchEventRequest() receiveRequest() updateRequest() redirectRequest()

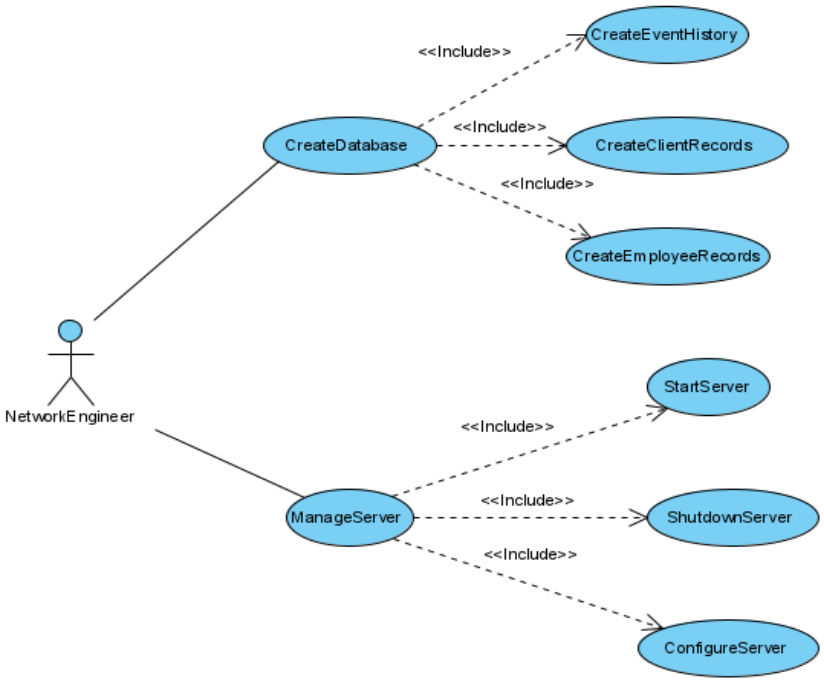
4.1 Access Control
Access Matrix

In order for staff to access the different objects, they must first log in to the system. This is done by the staff inputting their login credentials in the login portal. After this is done their credentials are verified and they are logged in. They can only access the objects according to the access matrix. For example, the financial manager can view records of clients and employees, as well as receiving event requests. They cannot access the EventHistory object.

4.2 Global Control Flow

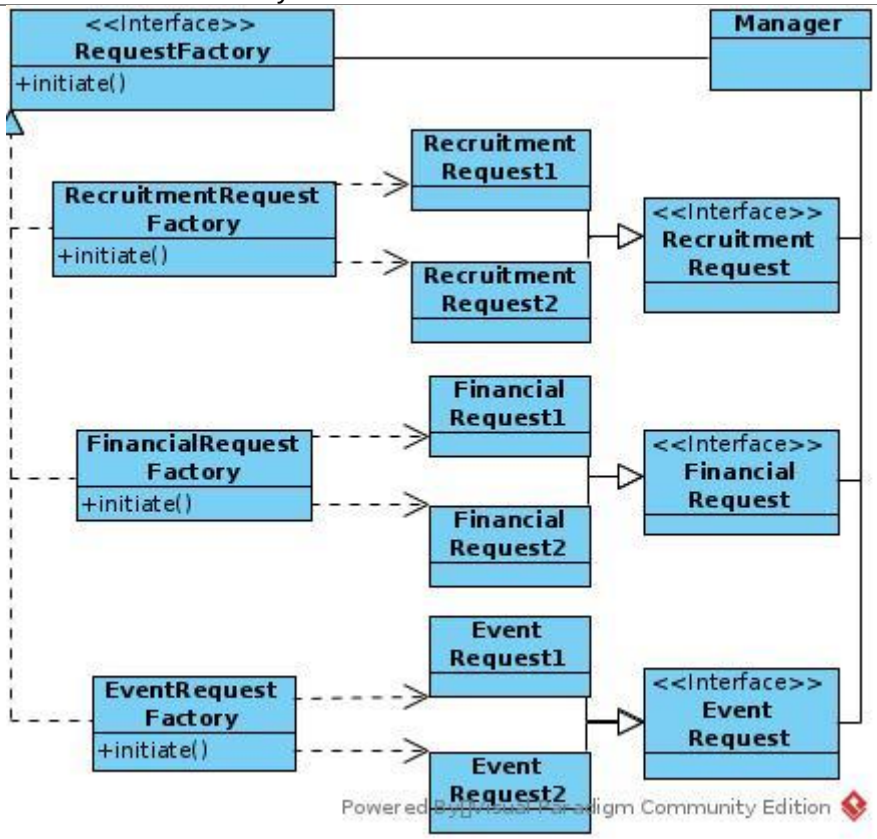
Event-driven control
We have selected event-driven control for the system. We felt that procedure-driven control would not be ideal, since we will be implementing the system in java, which is an object-oriented language. Procedure-driven control has difficulties with object-oriented languages. Threads seemed like a good choice, but because of problems arising when debugging and testing threads we decided to use event-driven control instead. When debugging tools become better suited for threads then it would most probably be a better choice for this system.

4.3 Boundary Conditions



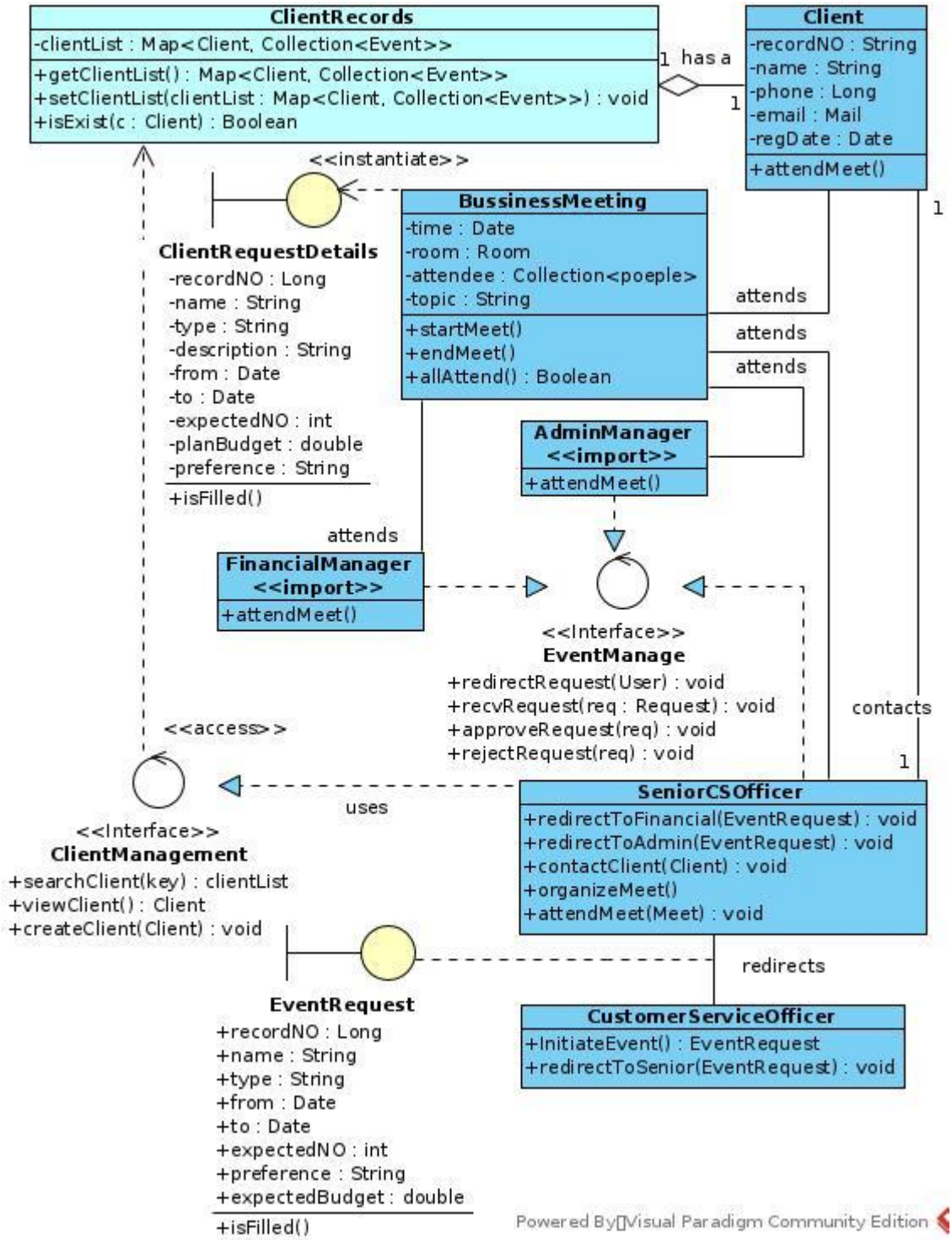
Name: NetworkManagement
Participating actor: NetworkEngineer
Entry conditions: 1. The network engineers logs in to the system. 2. The network engineer decides to create a database or manage the server.
Exit condition: There's no new database that needs to be created and the server doesn't need to be managed
Event flow: 1. The network creates a new database for the system. 2. The different records of event history, clients and employees are added to the database. 3. The network engineer decides to manage the server. 4. If the server had been shut down, the network engineer can try to start it up again. 5. If the server is experiencing issues and needs to be shutdown, the network engineer can do so. 6. If the server needs to be configured, the network engineer can make changes to the configuration.

5.1 Abstract Factory Pattern

Name	Abstract Factory Pattern
Problem description	When materials are insufficient or there are new request, the responsible department needs to submit special request form to the relevant department
Solution	We create RecruitmentRequest, FinancialRequest and EventRequest interfaces and entity classes that implements these interfaces. The next step is to create the abstract factory class RequestFactory. Then define the factory classes RecruitmentRequestFactory, FinancialRequestFactory and EventRequestFactory , all of which extend abstract factory.
Class diagram	

6. OCL

Customer Service Department



context CustomServiceOfficer :: redirectToSenior (EventRequest) **pre:**
EventRequest -> isFilled()

context SeniorCSOfficer :: organizeMeet () **pre:**
self -> contactClient()

context SeniorCSOfficer :: redirectToFinancial () **post:**
FinancialManager -> recvRequest()

context SeniorCSOfficer :: redirectToAdmin () **post:**
AdminManager -> recvRequest()

context ClientManage :: viewClient() **pre:**
ClientRecords -> isExist()

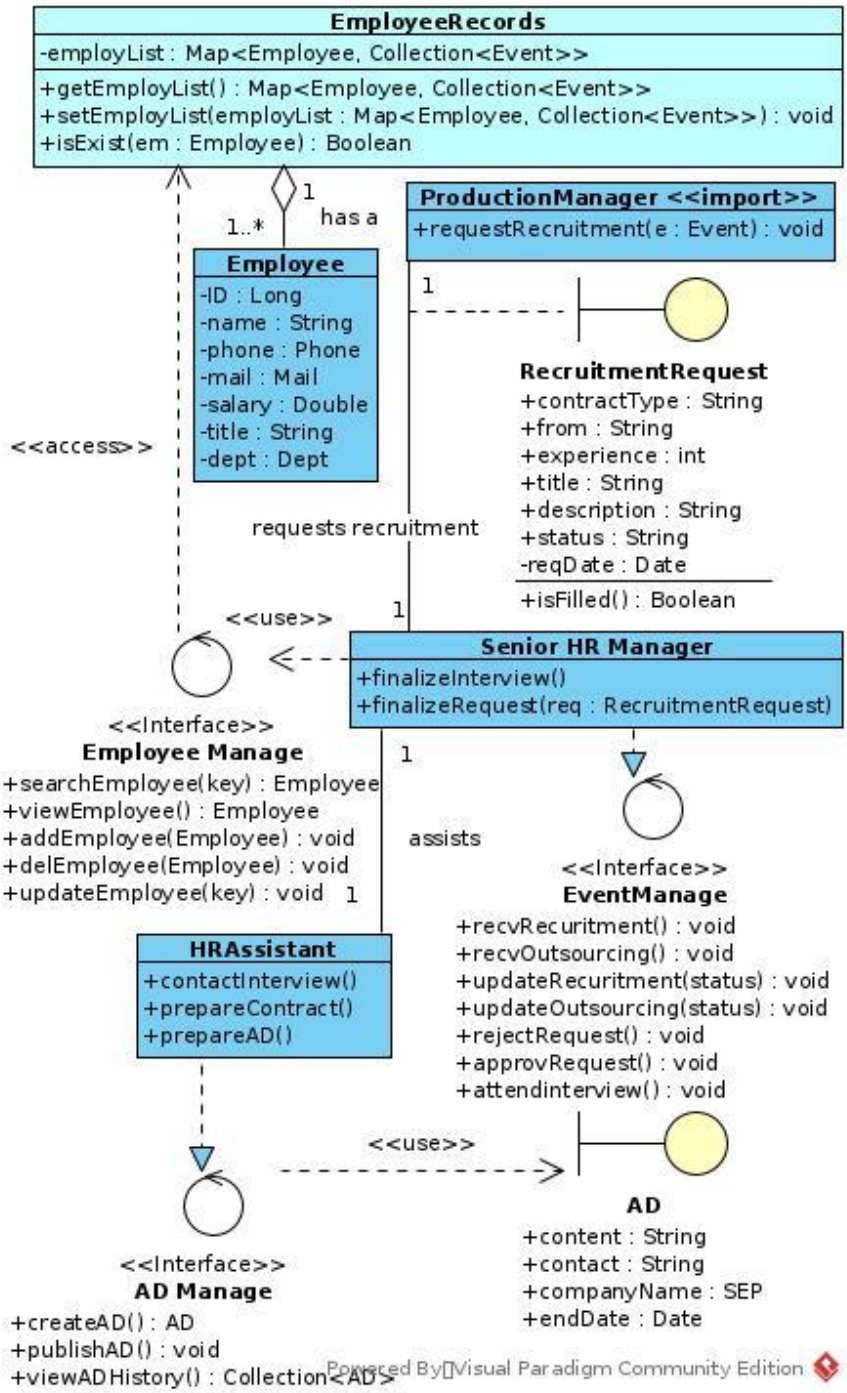
context BusinessMeeting :: startMeet() **pre:**
self -> allAttend()

context BusinessMeeting :: endMeet() **pre:**
ClientRequestDetails -> isFilled()

context EventRequest **inv:**
self.from < self.to
expectedNO > 0
expectedBudget > 0.0

context ClientRequestDetails **inv:**
self.from < self.to
expectedNO > 0
planBudget > 0.0

HR Department



context ProductionManager :: requestRecruitment(e:Event) **pre:**
RecruitmentRequest -> isFilled()

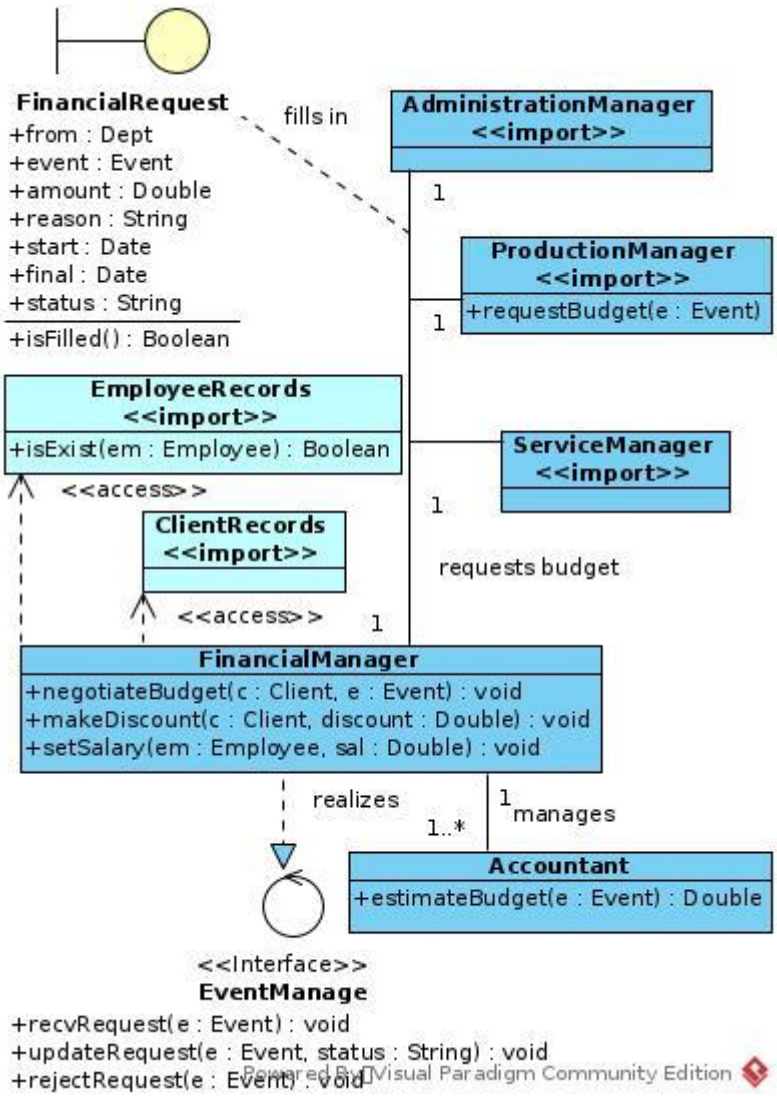
context ProductionManager :: requestRecruitment(e:Event) **post:**
SeniorHRManger -> recvRequest()

context EmployeeManage :: delEmployee(em:Empolyee) **pre:**
EmployeeRecords -> isExist()

context EmployeeManage :: addEmployee(em:Empolyee) **pre:**
! EmployeeRecords -> isExist()

context ADManage :: publishAD() **pre:**
self -> createAD()

Financial Department



context FinancialManager ::makeDiscount(c:Client, discount:Double) **pre:**
c.totalOrder > 1
discount >= 0.6

context FinancialManager ::setSalary(em:Employee, sal:Double) **pre:**
EmployeeMange -> isExist(em:Employee)
sal > 0.0

context ProductionManager ::requestBudget(e:Event) **pre:**
FinancialRequest -> isFilled()

context FinancialRequest **inv:**
self.final – self.start <= 5
self.amount > 0.0