Student ID__________________Student Name____________________________________

Software Architecture

Midterm Exam July 5, 2018

PRIVATE AND CONFIDENTIAL

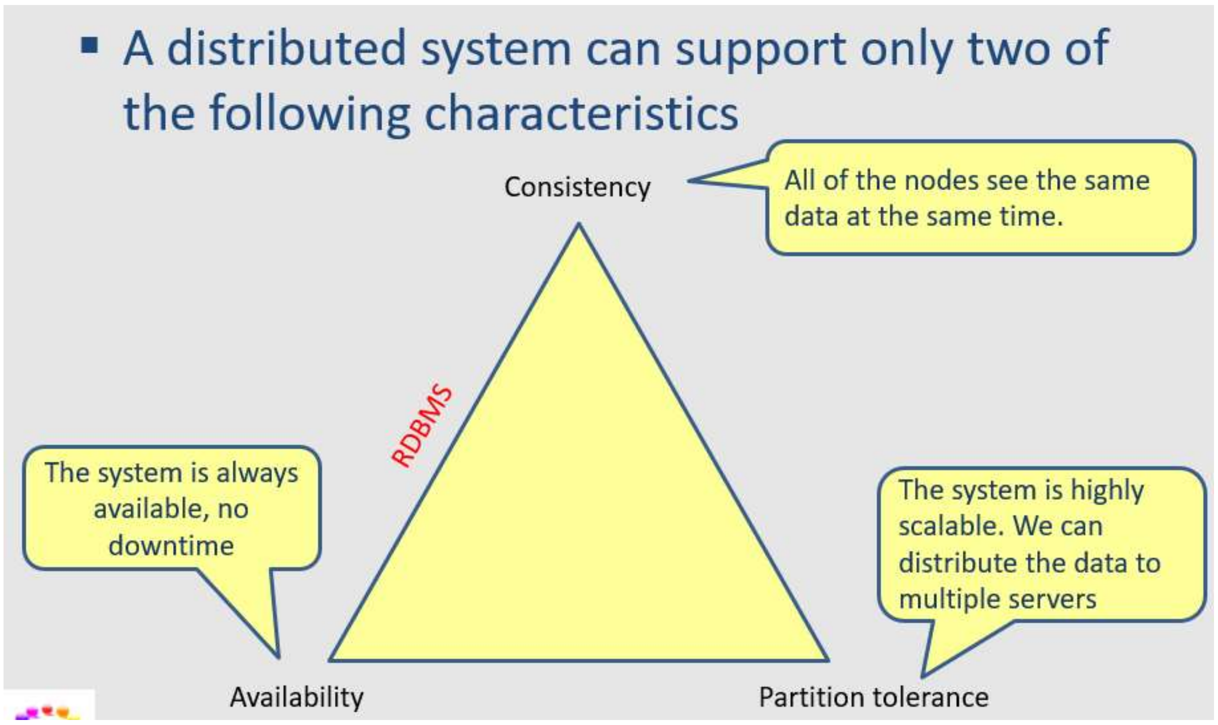
1. **Allotted exam duration is 3 hours.**
2. **Closed book/notes.**
3. **No personal items including electronic devices (cell phones, computers, calculators, PDAs).**
4. **No additional papers are allowed. Sufficient blank paper is included in the exam packet.**
5. **Exams are copyrighted and may not be copied or transferred.**
6. **Restroom and other personal breaks are not permitted.**

Write your name and student id at the top of this page.

<u>Write your answers clearly. If I cannot read it, you don't get points for it</u>.

Question 1 [10 points] {10 minutes}

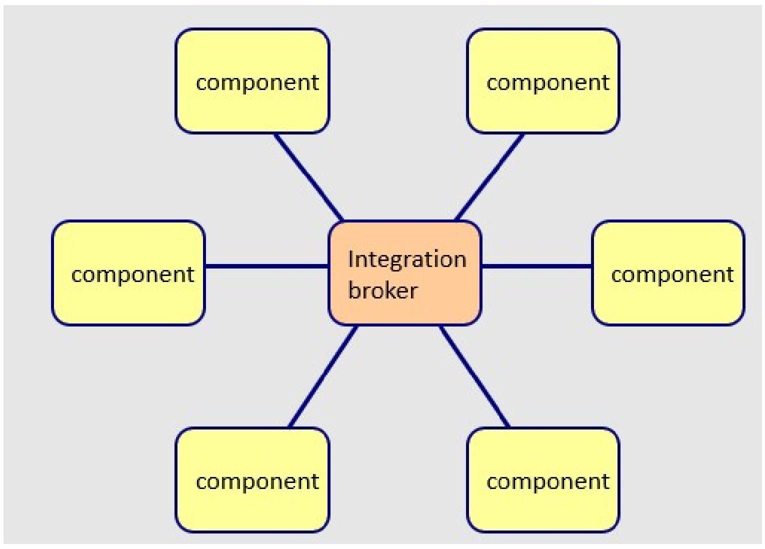
Explain the brewers cap theorem and explain what this means for distributed systems / databases.


A distributed system can support only two of the following characteristics
Consistency
All of the nodes see the same data at the same time.
RDBMS
The system is always available, no downtime
The system is highly scalable. We can distribute the data to multiple servers
Availability
Partition tolerance

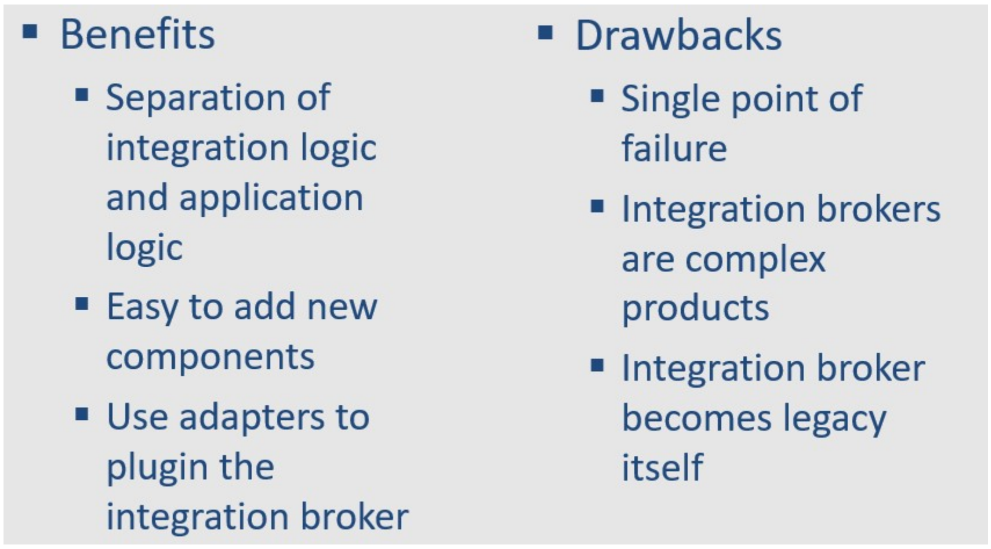
This means that if we want to scale out and be partition tolerance, we either have eventual consistency together with availability, or we have strict consistency with less availability, because we have to wait till all systems become consistent before we can use them again.

Question 2 [5 points] {10 minutes}

Explain the hub and spoke architecture style.
When would you use this style?
What problem(s) does it solve?
Explain the advantages and disadvantages.



You would use this if you need to connect a lot of systems together.
It solves the integration problem, where every system needs to talk to many other systems, and these systems use different techniques, languages and data structures.



- Benefits
  - Separation of integration logic and application logic
  - Easy to add new components
  - Use adapters to plugin the integration broker
- Drawbacks
  - Single point of failure
  - Integration brokers are complex products
  - Integration broker becomes legacy itself

Question 3 [5 points] {5 minutes}

When we implement an integration broker / ESB, we typically send messages to channels.
There are 3 different types of messages we can send.
Give the 3 types of messages we can send, and give an example of each.

Type of message	**Example of this type of message**
1 Command message	getLastTradePrice
2 Document message	aPurchaseOrder
3 Event message	priceChangeEvent

Question 4 [10 points] {10 minutes}

Give the different techniques we studied for point-to-point connections between 2 applications.
For each of the different techniques, give the advantage(s), disadvantage(s) and when you would use this technique.

technique	advantage	disadvantage	When to use?
RMI		Only Java to Java High coupling	Never
Messaging	Buffer Low coupling Asynchronous	You need messaging middleware	Between applications within the same organization When you need asynchronous requests
SOAP	Standards for security, transactions, etc. Standard for interface description	Complex	When you need standards
REST	Simple	No standards for security, transactions, etc No standard for interface description	Everywhere you need synchronous requests
Serialized objects over HTTP	Simpler as RMI Webcontainer functionality	Only Java to Java High coupling	For fast Java to Java integration between applications managed by the same project
Database integration	Simple	High coupling	Never
File based integration		High coupling	If you have to communicate large files

Question 5 [15 points] {10 minutes}

Suppose you need to design a bank account application that allows users to perform the following actions:

- Deposit money to an account
- Withdraw money from an account
- View the details of an account
- Transfer money from one account to another account.

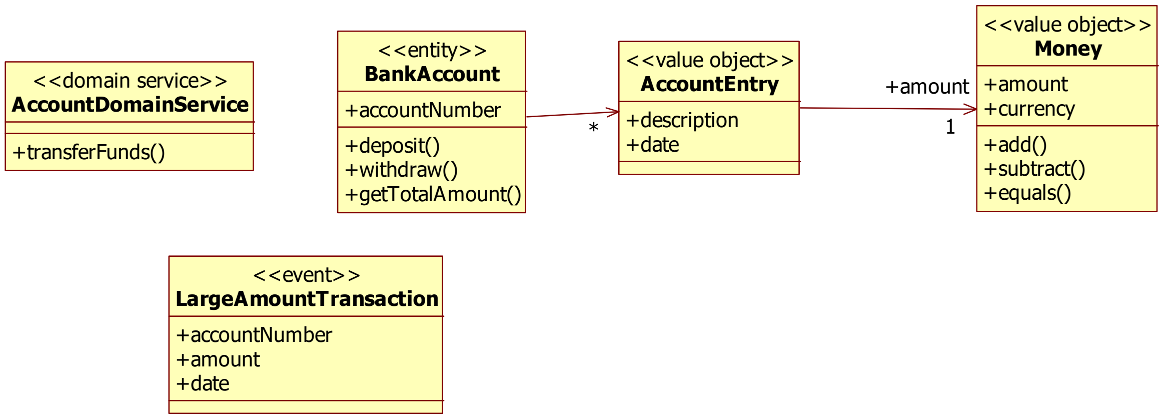
Our bank account application supports different currencies, so you can deposit in dollars, but also other currencies.
We need to implement the following business rules:

- You cannot withdraw more money than you have on your bank account
- Whenever the amount of a transaction is larger than $20.000, then the bank account application and maybe other applications need to know about this so they can check if this is not a fraudulent transaction

We need to design the bank account application using **Domain Driven Design**.

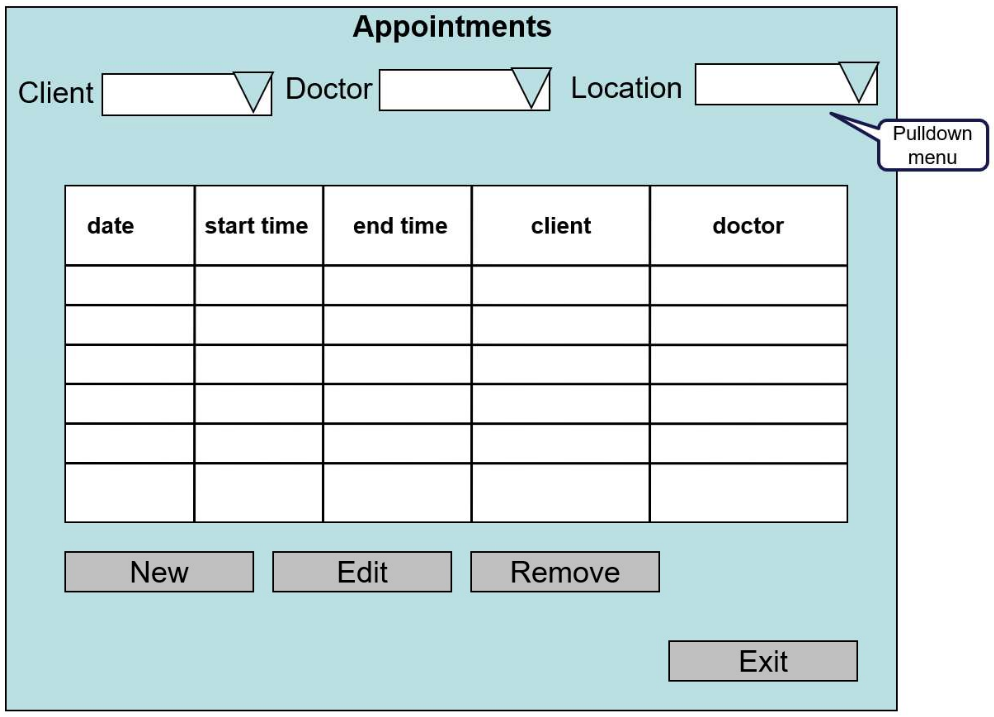
Draw the class diagram of **<u>only the domain classes</u>** of the bank account application. For every class show what type of class it is. Also show all attributes, methods and relations.

Your answer:

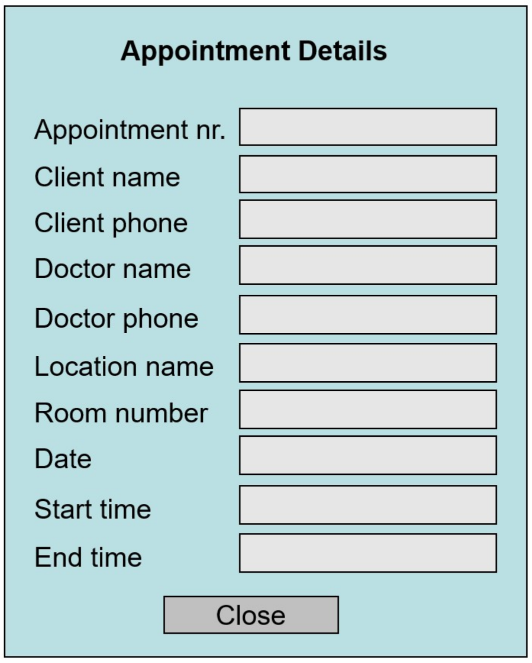


Question 6 [30 points] {55 minutes}

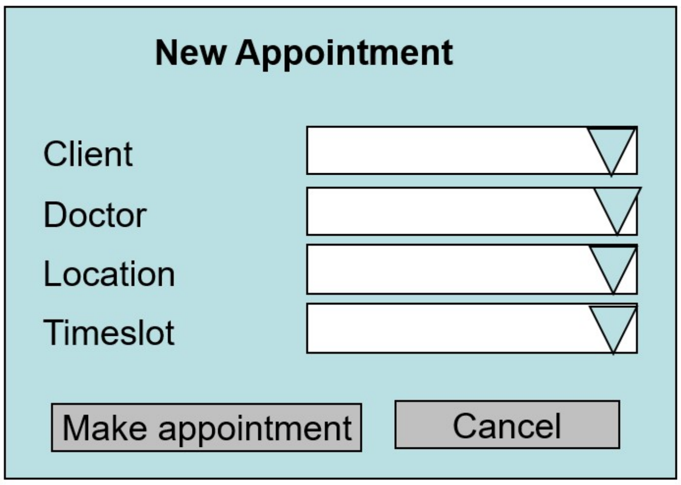
Suppose we need to design a **doctor's appointment application** using DDD. The user interface looks as follows:



The user can see the appointments for a selected client, doctor or location. The user can also add a new appointment, edit an existing appointment or remove an appointment. In the table that shows all appointments, you can double-click on an appointment which will show the following appointment details window:

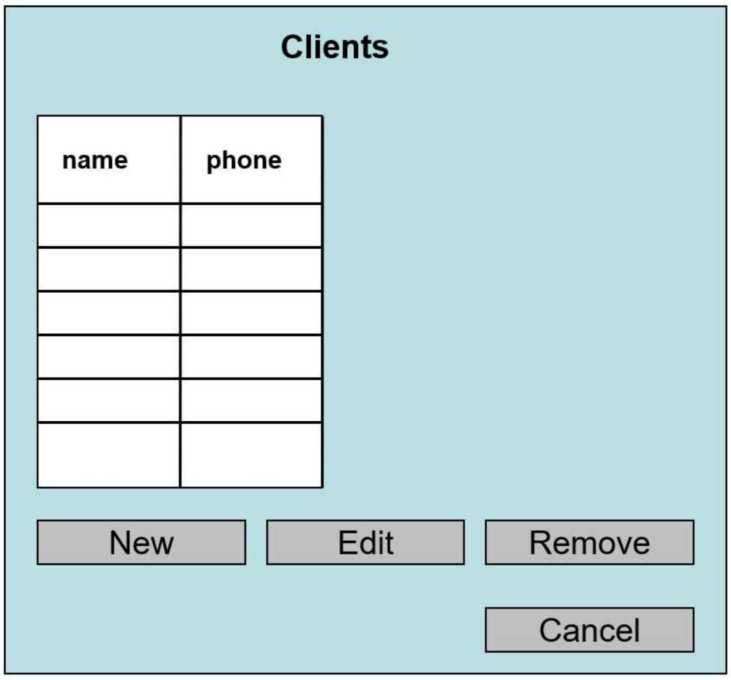


When you click the Add button you can add a new appointment using the following window:



The time is divided into fixed timeslots.
A client can make only 3 appointments in total. For every appointment the system needs to check if the location and the doctor is available at that timeslot.

The application also allows you to add, edit, remove or view **clients**, **doctors** and **locations**:

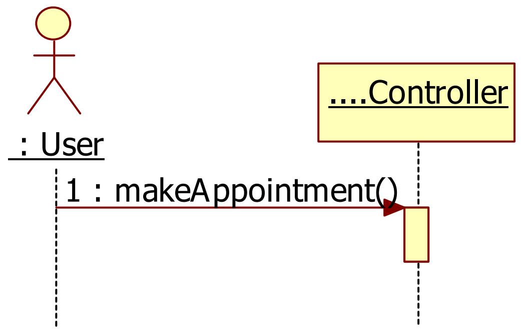


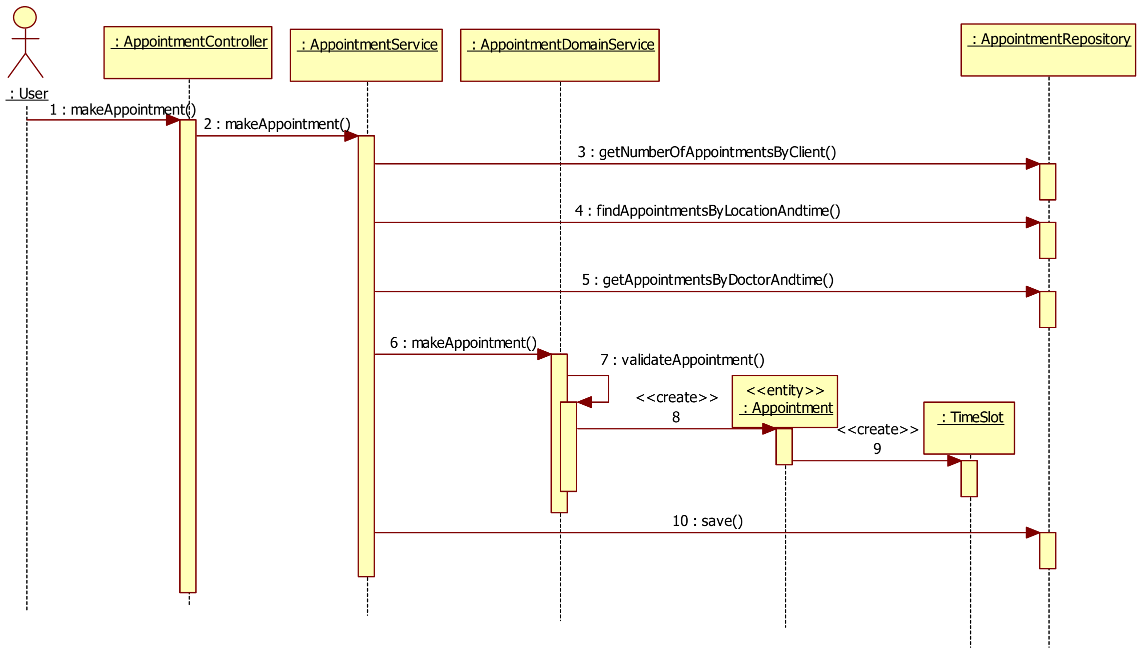
We have the following additional requirements:

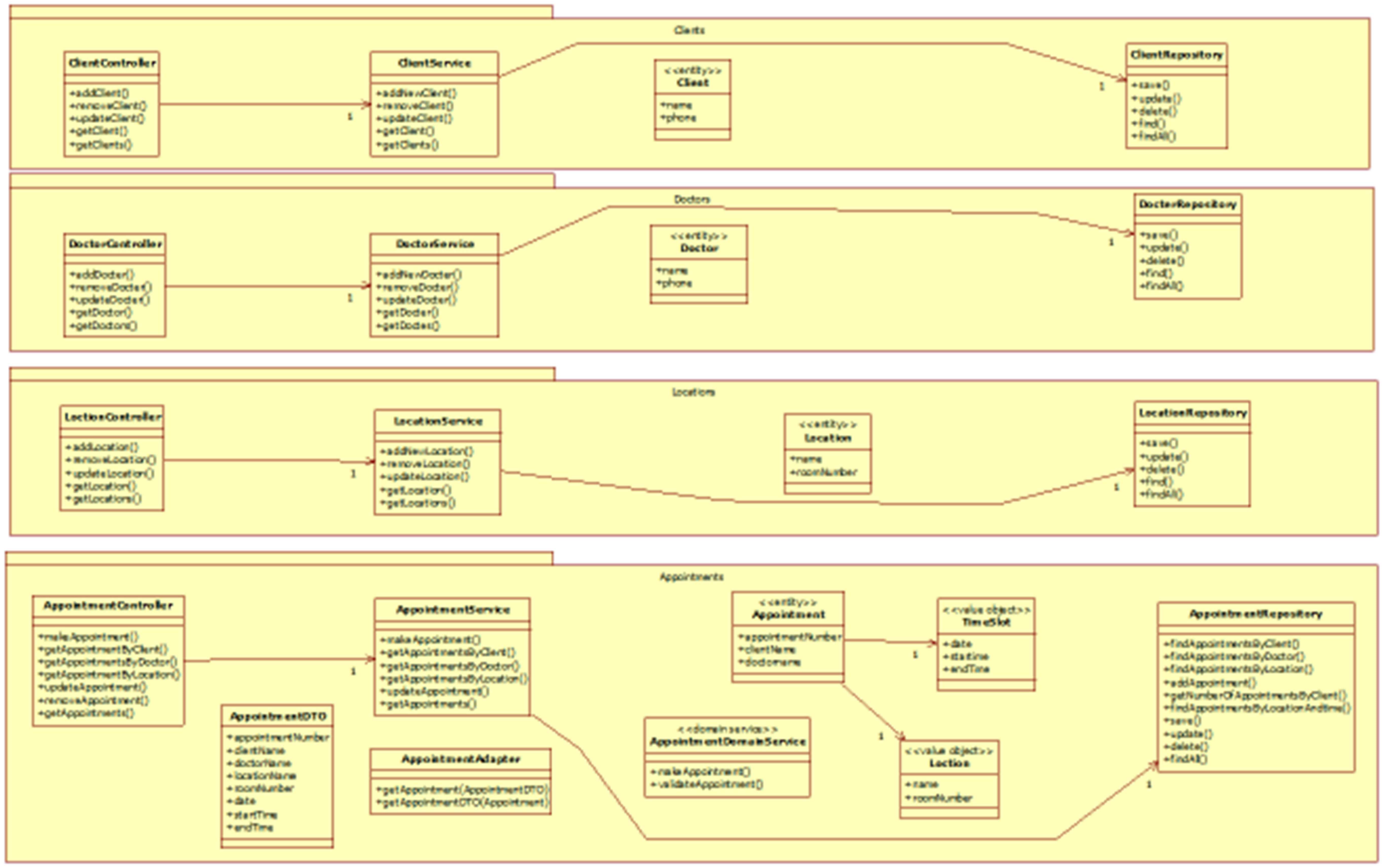
- We have to apply **Domain Driven Design**.
- It should be easy to add new functionality with minimal impact on the other parts of the system
- Because this will eventually a very large system, we need to be able to work with different agile scrum teams on this application.
- We are **NOT** allowed to use an ESB.

a. Draw the class diagram of your application. Show ALL classes that you need. Show clearly all attributes, methods, relationships, multiplicity, class type (entity, …)

b. Draw the sequence diagram of the scenario when the user makes a new appointment. Show all objects involved on your sequence diagram.
Your sequence diagram should start like this:



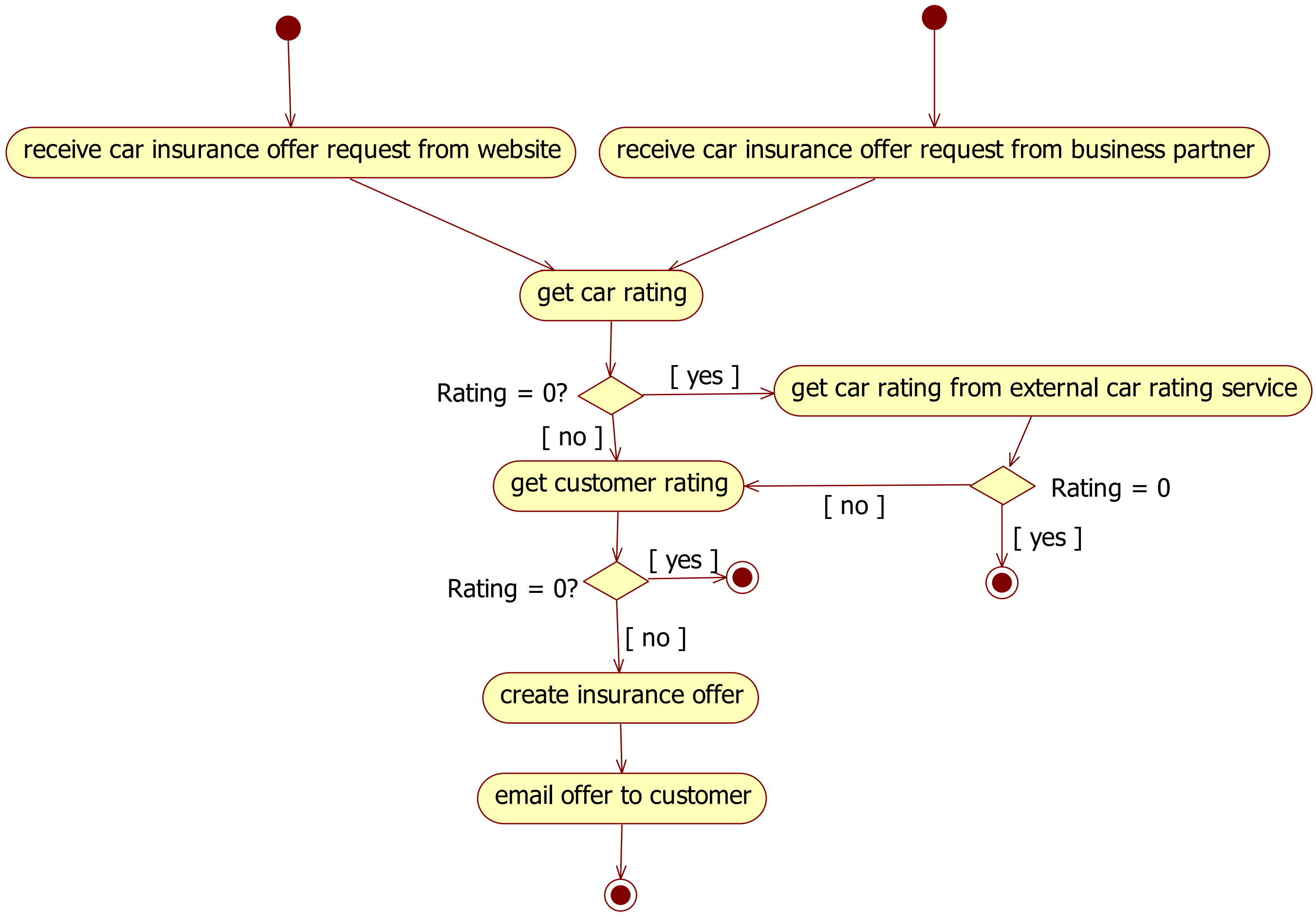



Clients
ClientController
+addClient()
+removeClient()
+updateClient()
+getClient()
+getClients()
ClientService
+addNewClient()
+removeClient()
+updateClient()
+getClient()
+getClients()
<<entity>>
Client
+name
+phone
ClientRepository
+save()
+update()
+delete()
+find()
+findAll()
Doctors
DoctorController
+addDoctor()
+removeDoctor()
+updateDoctor()
+getDoctor()
+getDoctors()
DoctorService
+addNewDoctor()
+removeDoctor()
+updateDoctor()
+getDoctor()
+getDoctors()
<<entity>>
Doctor
+name
+phone
DoctorRepository
+save()
+update()
+delete()
+find()
+findAll()
Locations
LoctionController
+addLocation()
+removeLocation()
+updateLocation()
+getLocation()
+getLocations()
LocationService
+addNewLocation()
+removeLocation()
+updateLocation()
+getLocation()
+getLocations()
<<entity>>
Location
+name
+roomNumber
LocationRepository
+save()
+update()
+delete()
+find()
+findAll()
Appointments
AppointmentController
+makeAppointment()
+getAppointmentByClient()
+getAppointmentsByDoctor()
+getAppointmentByLocation()
+updateAppointment()
+removeAppointment()
+getAppointments()
AppointmentService
+makeAppointment()
+getAppointmentsByClient()
+getAppointmentsByDoctor()
+getAppointmentsByLocation()
+updateAppointment()
+getAppointments()
AppointmentDTO
+appointmentNumber
+clientName
+doctorName
+locationName
+roomNumber
+date
+startTime
+endTime
AppointmentAdapter
+getAppointment(AppointmentDTO)
+getAppointmentDTO(Appointment)
<<entity>>
Appointment
+appointmentNumber
+clientName
+doctorname
<<domain service>>
AppointmentDomainService
+makeAppointment()
+validateAppointment()
<<value object>>
TimeSlot
+date
+starttime
+endTime
<<value object>>
Loction
+name
+roomNumber
AppointmentRepository
+findAppointmentsByClient()
+findAppointmentsByDoctor()
+findAppointmentsByLocation()
+addAppointment()
+getNumberOfAppointmentsByClient()
+findAppointmentsByLocationAndtime()
+save()
+update()
+delete()
+findAll()
1
1
1
1
1
1
1
1
1
1

Question 7 [20 points] {40 minutes}

Your boss asks you to implement **Car Insurance Offer Application** as a **Service Oriented Architecture** with the following business process:

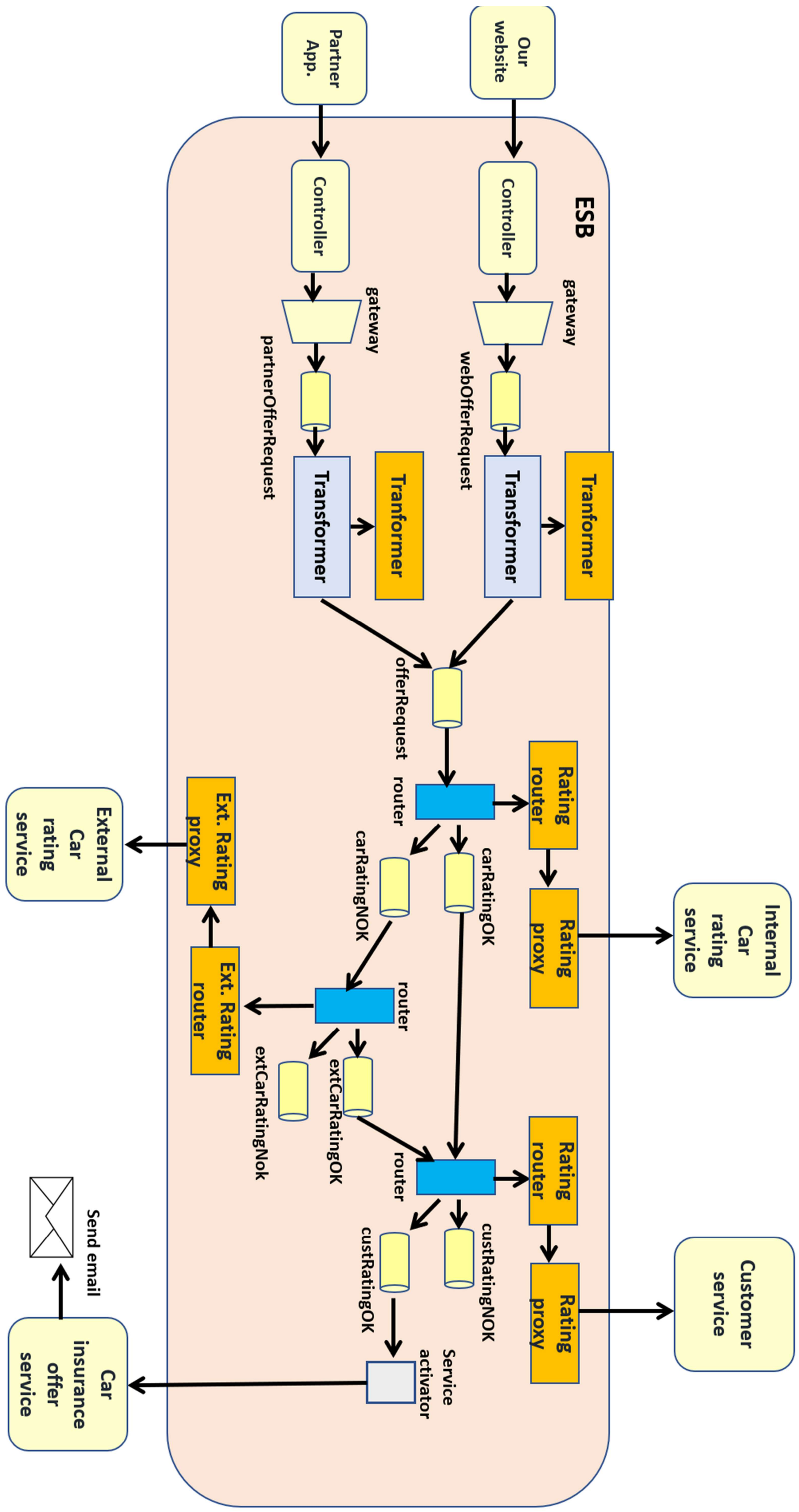
- We receive car insurance requests from our website through a REST interface.
- We also have a business partner that forwards car insurance offer requests to us through a different REST endpoint than the endpoint used by our own website.
- The data structure we receive from our website is also different from the data structure we receive from our business partner.
- After we receive a car insurance requests, we go to our own car rating functionality to get a rating for this particular car. A car rating a number between 0 and 100. Zero means that our car rating software cannot create a rating for this car.
- If the car rating is 0, we will call an external car rating service.
- If the external car rating service also returns 0, our process stops.
- After we get a car rating, we calculate a customer rating. A customer rating a number between 0 and 100. Zero means that our customer rating software cannot create a rating for this customer and our process stops.
- With both a car rating and a customer rating, we can calculate an car insurance offer that we send in an email to the customer.


receive car insurance offer request from website
receive car insurance offer request from business partner
get car rating
Rating = 0?
[yes]
get car rating from external car rating service
[no]
get customer rating
[no]
Rating = 0
[yes]
Rating = 0?
[yes]
[no]
create insurance offer
email offer to customer

Architectural requirements:

- It should be easy to add new functionality with minimal impact on the other parts of the system
- Because this will eventually a very large system, we need to be able to work with different agile scrum teams on this application.
- We have to use an ESB and we choose to use Spring Integration to implement the ESB.

Draw your solution in one diagram.
Show the integration design using the integration patterns we studied.


Our website
Partner App.
ESB
Controller
Controller
gateway
gateway
webOfferRequest
partnerOfferRequest
Transformer
Tranformer
Transformer
Tranformer
offerRequest
router
Rating router
Rating proxy
Internal Car rating service
carRatingOK
carRatingNOK
router
Ext. Rating router
Ext. Rating proxy
External Car rating service
extCarRatingOK
extCarRatingNok
router
Rating router
Rating proxy
Customer service
custRatingNOK
custRatingOK
Service activator
Car insurance offer service
Send email

Question 8 [5 points] {10 minutes}

Describe how an Enterprise Service Bus (ESB) relates to one or more of the SCI principles you know. Your answer should be about half a page, but should not exceed one page (handwritten). The number of points you get for this question depends on how well you explain the relationship between an Enterprise Service Bus (ESB) and the principles of SCI. Write clearly, I cannot give points if I cannot read it.

Your answer: