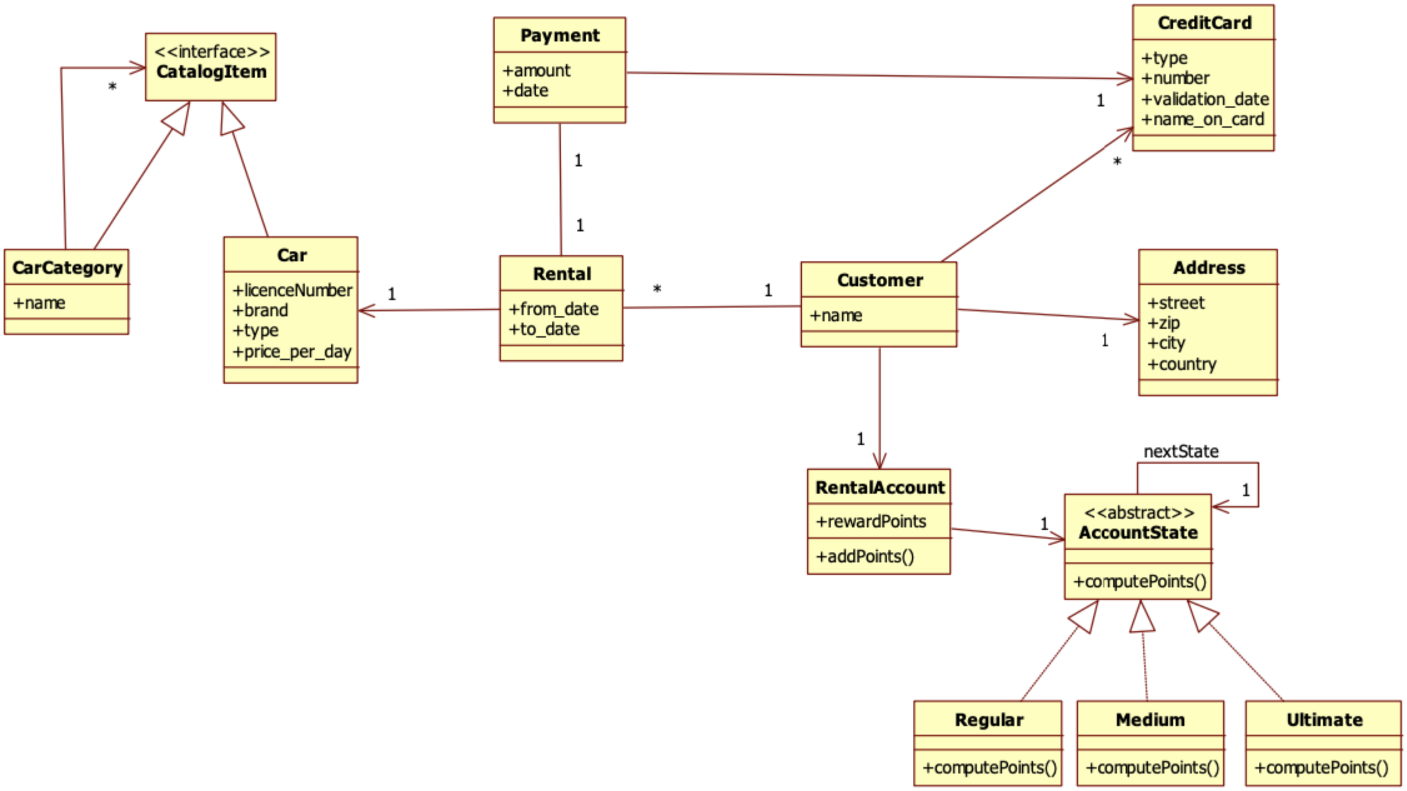
Question 1 [50 points] {55 minutes}

We have to design a car rental application for one of our customers with the following requirements:

- The application keeps track of which customers (name, address) rents which car (licenceNumber, brand, type, price_per_day) on which days.
- The application keeps track of the list of credit cards that a customer may have
- The application keeps track of payments(amount, date)
- Car rentals can only be paid with credit card.
- Cars are categorized in categories (economy, business(standard, full size, specialty), minivan, suv, etc)
- Customers can become a member of the Rental Rewards Program. Customers receive rental points for every rental. The calculation of how many points a customer receives for a rental depends on the car type and how many days the car is rented.
- Whenever a customer becomes a member of this Renter Rewards Program, the customer gets a Regular Rental Rewards account. When the customer has more than 50.000 rental points, he or she gets an upgrade to a Medium Rental Rewards account. When the customer has more than 150.000 rental points, he or she gets an upgrade to an Ultimate Rental Rewards account.
- When you have a Medium Rental Rewards account your miles you would normally receive for a car rental will be doubled.
- When you have an Ultimate Rental Rewards account your miles you would normally receive for a car rental will be tripled.

a. [25 points] {25 minutes}
Draw the **UML class diagram** of the car rental application. **Make sure you add all necessary UML elements (interfaces, abstract classes, attributes, methods, multiplicity, etc) to communicate the important parts of your design. You only need to show the domain model of the car rental application, you do not need to worry about GUI classes, database classes, etc.**



Now another customer wants us to design a tool rental application with the following requirements:

- The application keeps track of which customers (name, address) rents which tools (toolnumber, toolname, price_per_day, price_per_week, price_per_month) on which days.
- A customer can rent out multiple tools in one rental.
- The application keeps track of payments(amount, date)
- Rentals can be paid with credit card, cash, check or with bank wire. If you pay by check, the application should store the name on the check, the check number and the bank name on the check. If you pay with bank wire, the application should store the name of the bank.

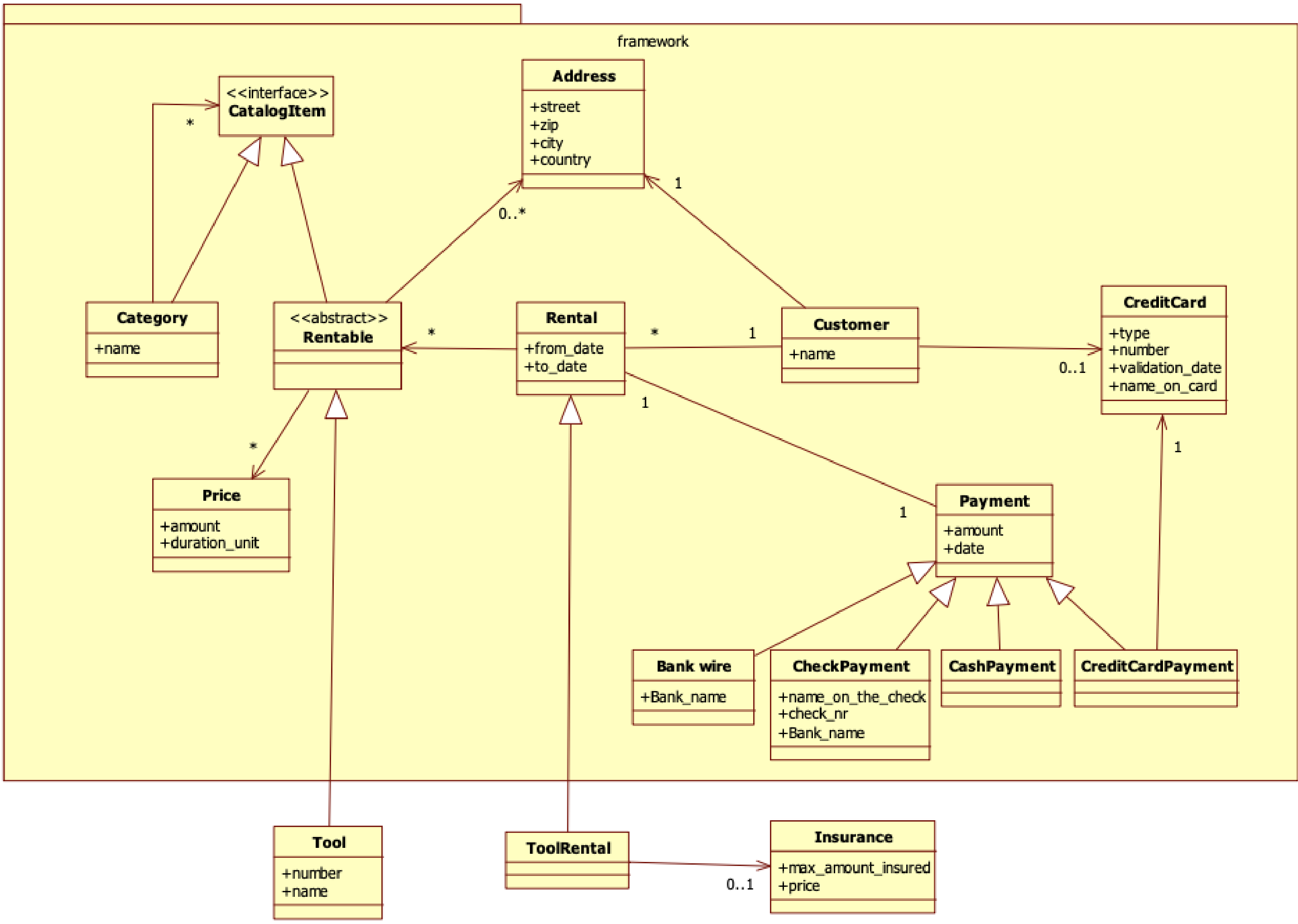
- Tools are categorized in categories (drilling tools, sawing tools, etc)
- Tool rentals can be insured but insurance is not required. If a tool rental is insured, then the application should keep track of the maximum amount that is insured and the price that has to be paid for the insurance.

Because this is our second rental application, we decide to make a rental framework.

The framework should support the following requirements:

- The framework keeps track of which customers (name, address) rents which rental-products on which days.
- A customer can rent out multiple rental-products in one rental.
- The application keeps track of payments (amount, date)
- Rentals can be paid with credit card, cash, check or bank wire. If you pay by check, the application should store the name on the check, the check number and the bank name on the check. If you pay with bank wire, the application should store the name of the bank.
- Rental-products are categorized in categories
- The framework supports all possible ways to specify the rental price: price per hour, price per half day, price per day, price per week, price per month, price per year, price per 2 weeks, etc.

b. [25 points] {30 minutes}
Draw the **UML class diagram** of the tool rental application using the framework.


framework
<<interface>>
CatalogItem
Address
+street
+zip
+city
+country
Category
+name
<<abstract>>
Rentable
Rental
+from_date
+to_date
Customer
+name
CreditCard
+type
+number
+validation_date
+name_on_card
Price
+amount
+duration_unit
Payment
+amount
+date
Bank wire
+Bank_name
CheckPayment
+name_on_the_check
+check_nr
+Bank_name
CashPayment
CreditCardPayment
Tool
+number
+name
ToolRental
Insurance
+max_amount_insured
+price
*
0..*
1
*
*
1
0..1
1
1
1
*
0..1

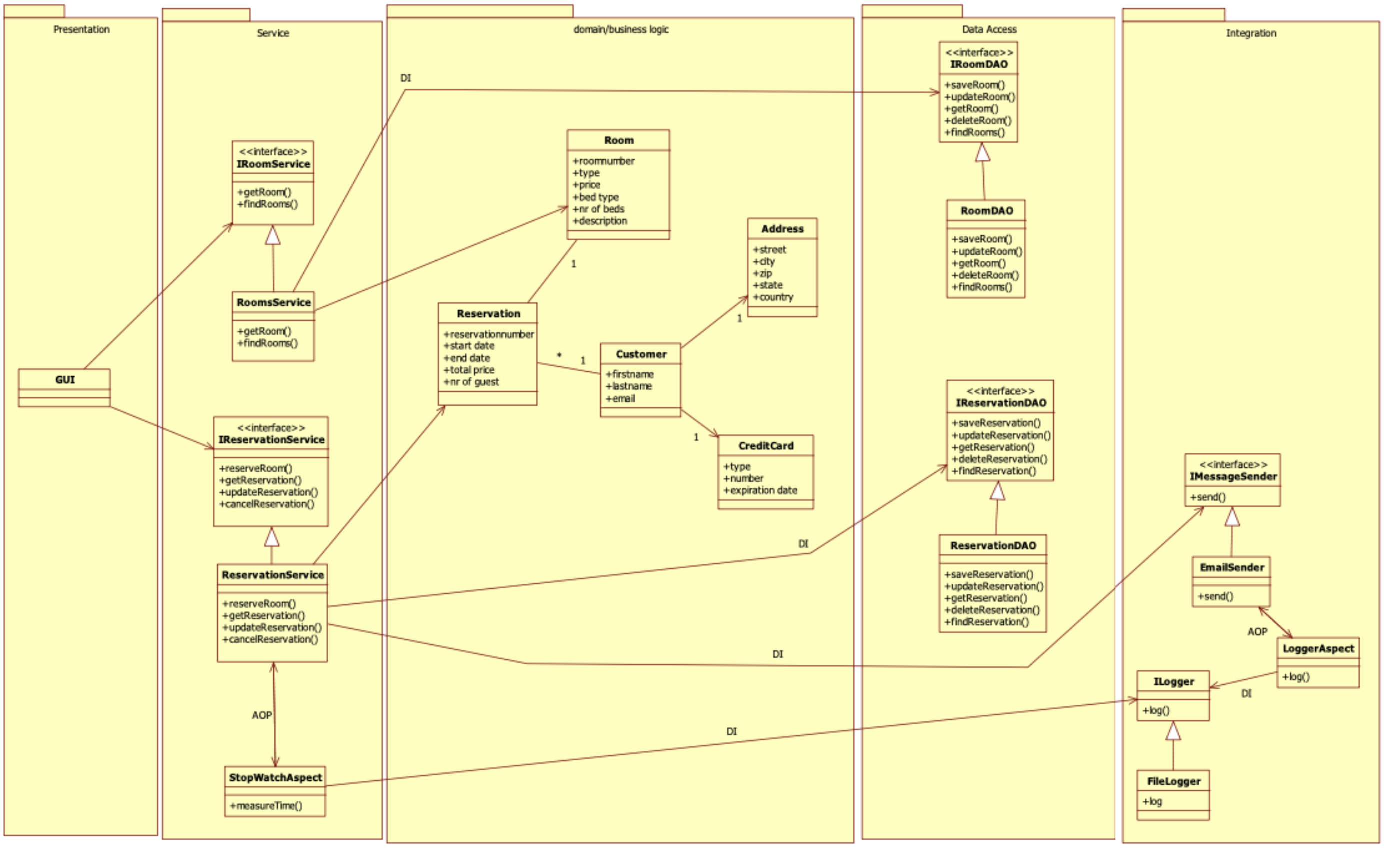
Question 2 [40 points] {55 minutes}

Design a hotel room reservation system with the following requirements:

- Customers can browse rooms
- Customers can search rooms
- Customers can reserve a room
- Customers can cancel a reservation
- Customers can change a reservation
- Customers can view their reservation
- Rooms are stored in the database
- Reservations are stored in the database
- Whenever a customer reserves a room, the system sends an email to the customer.
- It should be easy to change the technique of sending a message to the customer (SMS, WhatsApp message, etc)
- It should be easy to switch to any type of database, even non relational databases
- Whenever this application sends a message, we want to log that in a logfile
- It should be easy to log to any medium (comma separated file, database, XML file,…)
- We will be using the Spring framework to implement this application
- Every reservation has a unique reservation number
- For every reservation, the system records the start date, end date, total price, number of guest, room, customer first name, last name, email, address info and credit card info.
- For every room, the system records the room number, room type, room price, bed type, nr of beds
- Whenever the customer places a reservation, cancels a reservation or changes a reservation, we want to measure the time it takes for the actions to execute, and we want to log the result in a logfile.
- All messages we send to the customer are also logged in a logfile.

Draw the **UML class diagram** of the hotel reservation application. **Make sure you add all necessary UML elements (interfaces, abstract classes, attributes, methods, multiplicity, etc) to communicate the important parts of your design.** Make use of all the best practices we learned in this course. Show very clearly where you use Dependency Injection (DI), AOP or certain design patterns.

To show where to use DI, use the following notation:


Presentation
Service
domain/business logic
Data Access
Integration
GUI
<<interface>>
IRoomService
+getRoom()
+findRooms()
RoomsService
+getRoom()
+findRooms()
<<interface>>
IReservationService
+reserveRoom()
+getReservation()
+updateReservation()
+cancelReservation()
ReservationService
+reserveRoom()
+getReservation()
+updateReservation()
+cancelReservation()
AOP
StopWatchAspect
+measureTime()
DI
Room
+roomnumber
+type
+price
+bed type
+nr of beds
+description
1
Reservation
+reservationnumber
+start date
+end date
+total price
+nr of guest
*
1
Customer
+firstname
+lastname
+email
1
Address
+street
+city
+zip
+state
+country
1
CreditCard
+type
+number
+expiration date
DI
DI
DI
<<interface>>
IRoomDAO
+saveRoom()
+updateRoom()
+getRoom()
+deleteRoom()
+findRooms()
RoomDAO
+saveRoom()
+updateRoom()
+getRoom()
+deleteRoom()
+findRooms()
<<interface>>
IReservationDAO
+saveReservation()
+updateReservation()
+getReservation()
+deleteReservation()
+findReservation()
ReservationDAO
+saveReservation()
+updateReservation()
+getReservation()
+deleteReservation()
+findReservation()
<<interface>>
IMessageSender
+send()
EmailSender
+send()
AOP
LoggerAspect
+log()
DI
ILogger
+log()
FileLogger
+log