Question 1. (8 marks) Scenarios and Use Case Descriptions

This question tests your proficiency at identifying scenarios and at writing use case descriptions.

Below is a partial use case description for using a proposed automated Student Information System (SIS) to manage a student's term enrollment: it describes the main scenario for Adding a Course to a student's enrollment. On the next page, help to complete this use case by providing two (2) alternative flows and two (2) exceptions. Each alternative flow and exception should *start from a different point* in the main scenario.

Typical Process Description

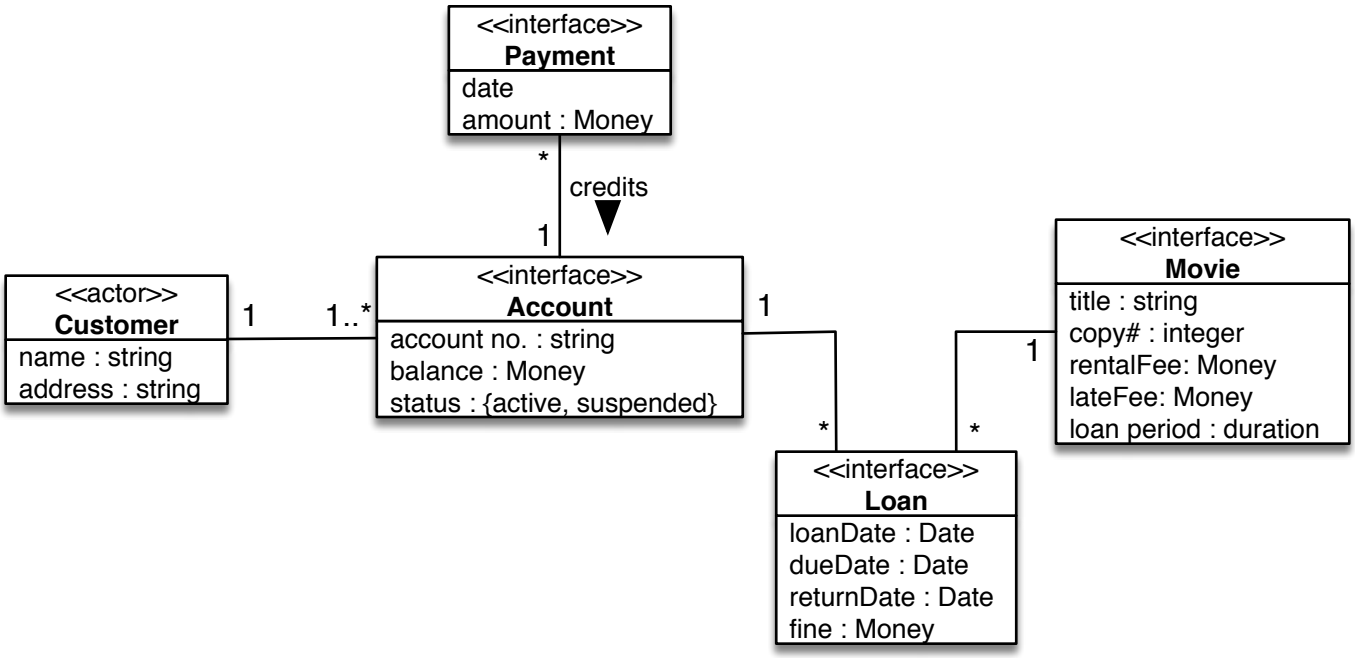
Student	System response
1. Login into SIS	
	2. Validate login
	3. Display SIS options: - enrollment - transcript - course catalogue - schedule of classes - examination schedule
4. Select enrollment	
	5. Display terms enrolled
6. Select term	
	7. Display selected term's course schedule and allowable operations: - add course - drop course - swap course enrollments - print course schedule
8. Select "add course" operation	
	9. Display "add course" form
10. Enter the class number of the course to be added	
	11. Display selected course, including information about related course meets (e.g., tutorials, laboratory, discussion meets)
12. Enter the class number of the related course meets	
13. Submit request	
	14. Display information of successfully added course.

Question 2. (25 marks) Domain Modelling

This question tests your proficiency at domain modelling and with using UML class diagrams.

InternetFlix wants to build a web-based DVD rental business whereby customers interact with the company via the Internet. Customers use a web interface to browse the DVD inventory, ask about the availability of movies, and request to borrow movies (up to 2 movies at a time). The company mails requested DVDs out to the customer, and the customer mails the DVDs back when he is done with them. The rental period is two weeks. There is a fee for renting a movie (movies have different rental fees based on whether they are new releases, whether the movie has won any special awards, etc.). There is a late fee (assessed daily) for not returning a movie before the end of the rental period. When an account's balance falls below the value MinBal, the account is suspended and the customer cannot initiate new loans until payment is made and the balance becomes nonnegative. Information about both current and past loans is maintained.

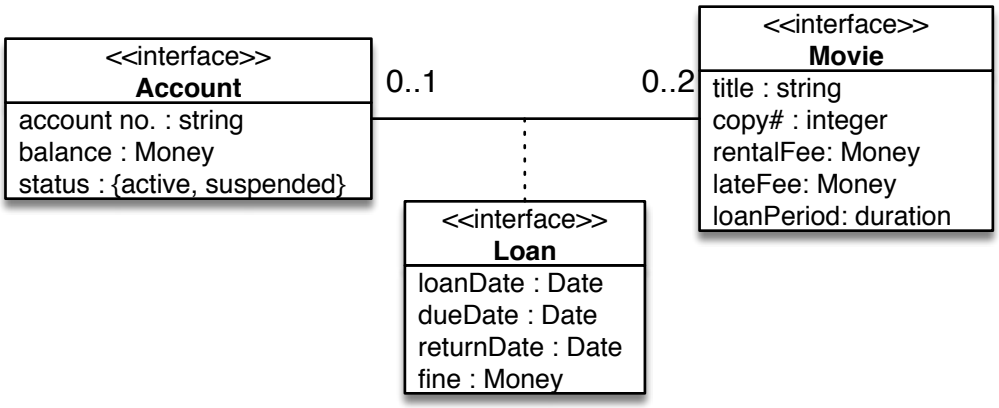
Below is a partial Domain Model for the *specification* of the proposed InternetFlix system, although it is not necessarily the best model of the problem's domain.



Question 2. [continued]

2a) (3 marks) Association Classes

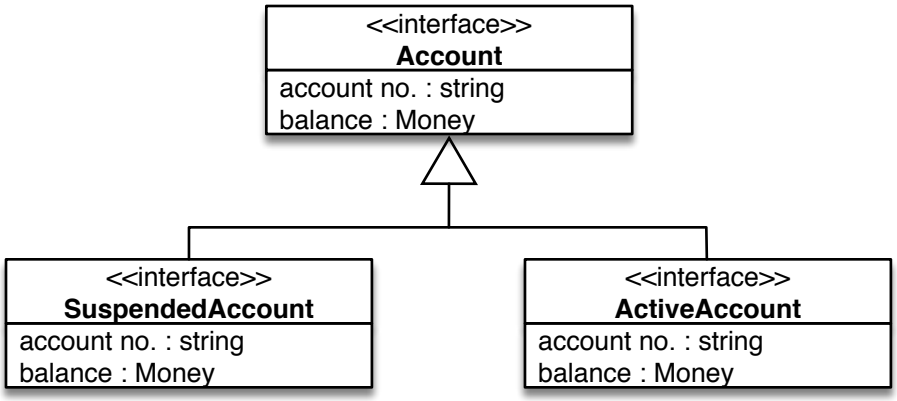
In the initial model, a Loan is modelled as an ordinary class. Alternatively, we could have modelled Loan as an association class of the relationship between the Account and a Loan.



Which modelling decision is better for this domain model and why?

2b) (3 marks) Generalization

In the initial model, the status of an Account is modelled as an attribute. Alternatively, we could have modelled active and suspended Accounts as subclasses of the Account class.

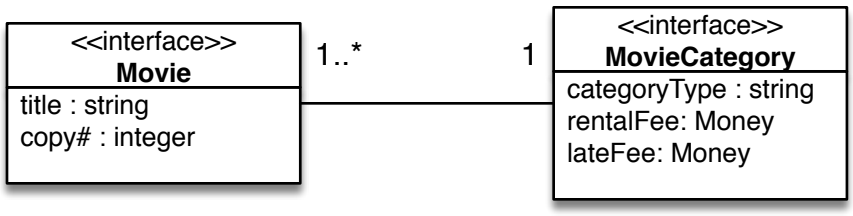


Which modelling decision is better for this domain model and why?

Question 2. [continued]

2d) (4 marks) Attributes vs. Associations

In the initial model, a Movie's rental fee and late fee are modelled as ordinary attributes of the Movie class. Alternatively, we could have created a new Movie Category class that records rental fees and late fees as attributes of the Movie Category class, and we could have associated each Movie with a Movie Category.



Which modelling decision is better for this domain model and why?

2e) (12 marks) Domain Modelling

Revise the initial model to *replace* the ad-hoc rental service with a subscription service. If a customer subscribes to this service, then instead of asking to borrow movies one at a time, the customer, as part of this service, maintains an *ordered* queue of movies that he is interested in renting listed in priority of interest. Whenever the customer has fewer than two movies out on loan (e.g., whenever a movie is returned), a new movie is automatically sent out to him. The movie that is sent out is the *highest priority movie* on the customer's request queue for which there is an *available copy*. With this service, there is no rental fee per loan, no loan period, no late fee, and so forth. Instead, the customer pays a fixed monthly fee for the privilege of always having two movies out on loan.

As part of your revisions to the initial UML model, you should cross out any classes, attributes, and associations that are not needed to model the new subscription service, and you should add any new classes, attributes, and associations that model the new subscription service. Use multiplicities, aggregation, generalization, etc. to best express the service. Make your revisions by marking up the initial model.

Question 4. (10 marks) Use Case Diagrams

Consider a system for administering the lending of books at a university library. A person must be a member of the university's community and must be in good standing – that is, not have any outstanding fines or overdue books – to borrow books. A book may be borrowed for up to two weeks at a time. A book loan may be renewed if the book is returned before the loan's due date and if no other library member has expressed an interest in borrowing the book. If a book is returned after the loan's due date, the borrower will be a charged a fine of $1 for each late day. Fines are paid to the library staff at the circulation desk, where books are returned. Heavily-used books may be put on reserve, meaning that members can read them only in the library and cannot borrow them.

4a) (5 Marks) Provide a context diagram for the library lending system.

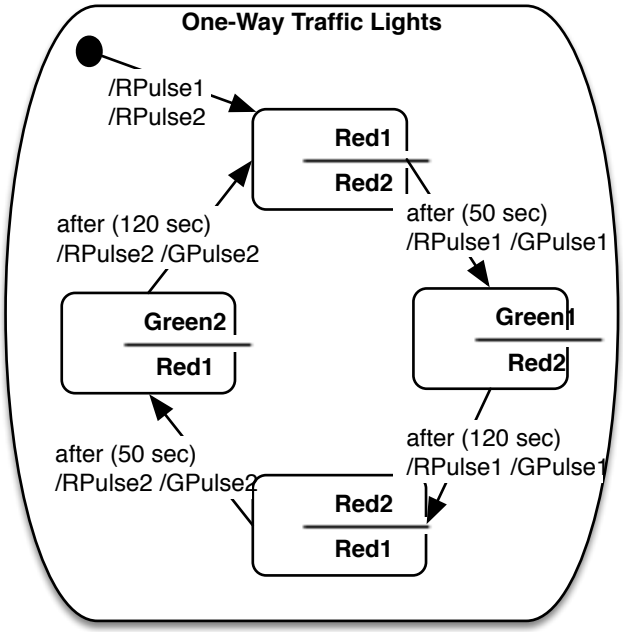
4b) (5 Marks) Provide a use-case diagram that depicts **only** those use cases that are initiated by library members.

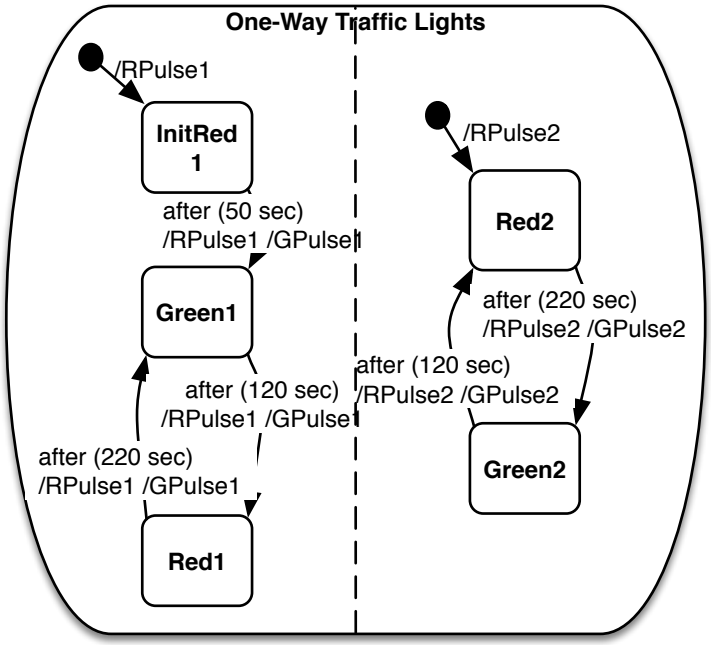
Question 5. (8 marks) State Machine Modelling

5a) (4 Marks)

Consider the following problem. When a section of road is being repaired, it is often necessary for traffic to share one lane of road while the construction crew works on the other lane. This segment of road becomes a temporary one-way road, where the direction of traffic alternates. The traffic is controlled by a pair of simple portable traffic light units connected by wire to a computer controller. A unit is placed at each end of the one-way section of road. Each unit has a stop (red) light and a go (green) light. The computer controls the lights by emitting RPulses and GPulses, to which the units respond by turning the lights on and off. Each pulse toggles the associated light on or off; when the units are first connected, both lights are off. The computer controls the lights to follow a fixed cycle of four phases: initially, both units show stop lights for 50 seconds. Then, for 120 seconds, one unit shows a go light and the other unit shows a stop light. Then for 50 seconds, both units show stop lights, to let traffic drain from the road segment. And then for 120 seconds, the other unit shows a go light while the first unit shows a stop light. Then the cycle repeats.

Below are two alternative StateMachine models of this problem. The only difference between them is that one model uses concurrency and the other does not:

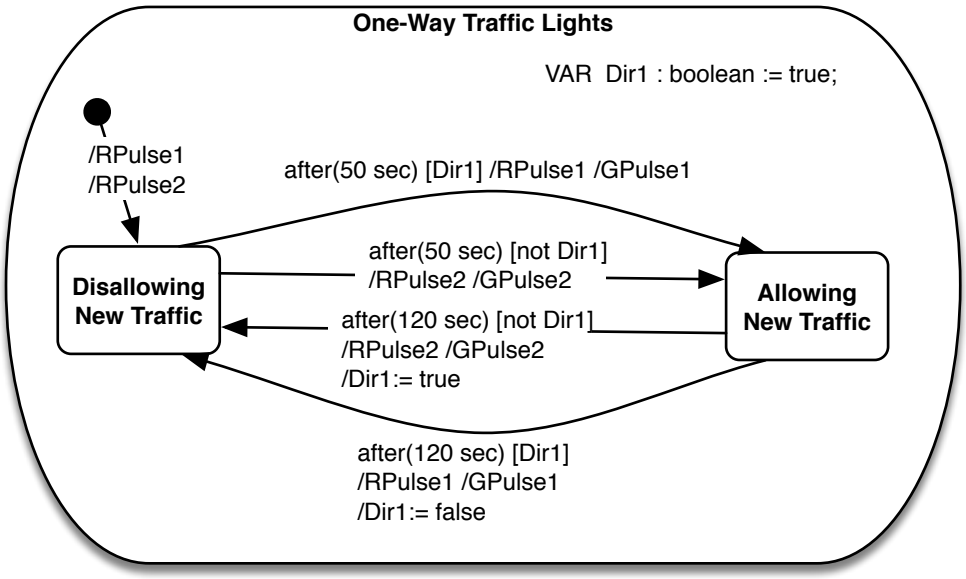




Which model is better (i.e., easier to understand; exhibits better use, or nonuse, of the concurrency modelling construct)? Defend your answer.

5b) (4 Marks)

A third alternative model of the One-Way Traffic Lights problem uses variables to keep track of which units are showing stop vs. go lights:



Which model is better (i.e., easier to understand; exhibits better use, or nonuse, of the concurrency modelling construct)? Defend your answer.

Question 6. (15 marks) Short Answer

These questions evaluate your understanding of software engineering and requirements engineering.

6b) (3 marks) Requirements vs. Specifications

What is the difference between a *requirement* and its corresponding *specification*?

6d) (5 marks) Stakeholders

What contributions (in the way of expertise) do each of the following types of stakeholders make towards eliciting and understanding the requirements of a proposed system?

- Customer
- Software Engineer
- Domain Expert

Question 7. (20 marks) Scenarios and Use Case Descriptions

This question tests your proficiency at identifying scenarios and at writing use case descriptions.

Below is a partial use case description for Library Work that interacts with a library's existing computerized catalogue. It describes the main scenario for requesting a book.

Typical Process Description

Library Patron	Library Work	Library Catalogue
1. Enters search parameters to identify book (includes one or more of - book title - author - subject headling - call number)		
	2. Constructs search query and sends it to the library catalogue.	
		3. Finds all catalogue entries that match the query's search parameters, and sends result to the Web Interface.
	4. Displays search results to the user, providing means for the user to select one of the displayed entries or to modify the search.	
5. Selects one of the displayed entries.		
	6. Constructs query based on selected item.	
		7. Retrieves all information about queried publication, including - book title - author - publisher - call number - number of copies - loan status (i.e., on loan?)
	8. Display item's details to the user, providing means for the user to request the item, view another entry from the search results, or to modify the search parameters.	
9. Requests the item		
	10. Asks the user whether he wants the library to hold the requested item at the circulation desk for pick up (if item is not out on loan), or to recall the item (if item is out on loan).	
11. Asks to put item on hold, to be picked up later.		
	12. Displays instructions to Library Circulation Desk console to retrieve requested item (book title, author, call number) from the library shelves and put it on hold at the Circulation Desk.	

Question 7. [continued.]

7a) (7 marks) Alternative Flows and Exceptions

Below, help to complete the use case description from the previous page by providing two (2) alternative flows and two (2) exceptions. Each alternative flow and exception should *start from a different point* in the main scenario.

Question 7. [continued.]

7b) (4 marks) Actors

The initial use-case description may or may not be the best model of the interactions between the Library Work and the actors in its environment. In the provided use-case description, the Library Catalogue is modelled as an actor. Instead, we could have modelled the Library Catalogue as part of the Work, and showed only the interactions between the Library Patron and the Work. Which modelling decision is most appropriate for *this problem* and why?

7c) (4 marks) Actions

In the initial use-case description, the Library Patron's search parameters are conveyed in a single step (step 1). Instead, we could have modelled this interaction in more detail, with the patron sending each piece of information (book title, author, etc.) as a separate interaction:

Library Patron	Library Work	Library Catalogue
1. Enters book title (if known)		
2. Enters author (if known)		
3. Enters subject heading (if known)		
4. Enters call number (if known)		
	5. Constructs search query and sends it to the library catalogue.	

Which version of the use case description is better, and why?

Question 8. (15 marks) Use Case Diagrams

This question tests your proficiency at identifying use cases and at drawing use case diagrams.

Draw a use case diagram for the Library Work, introduced in Question 7, that models the *main* library services that are to be offered as part of the Library's Work. The full Work provides services to both patrons and librarians. The main functionality to be provided is listed below, but you decide how to group the functionality into use cases:

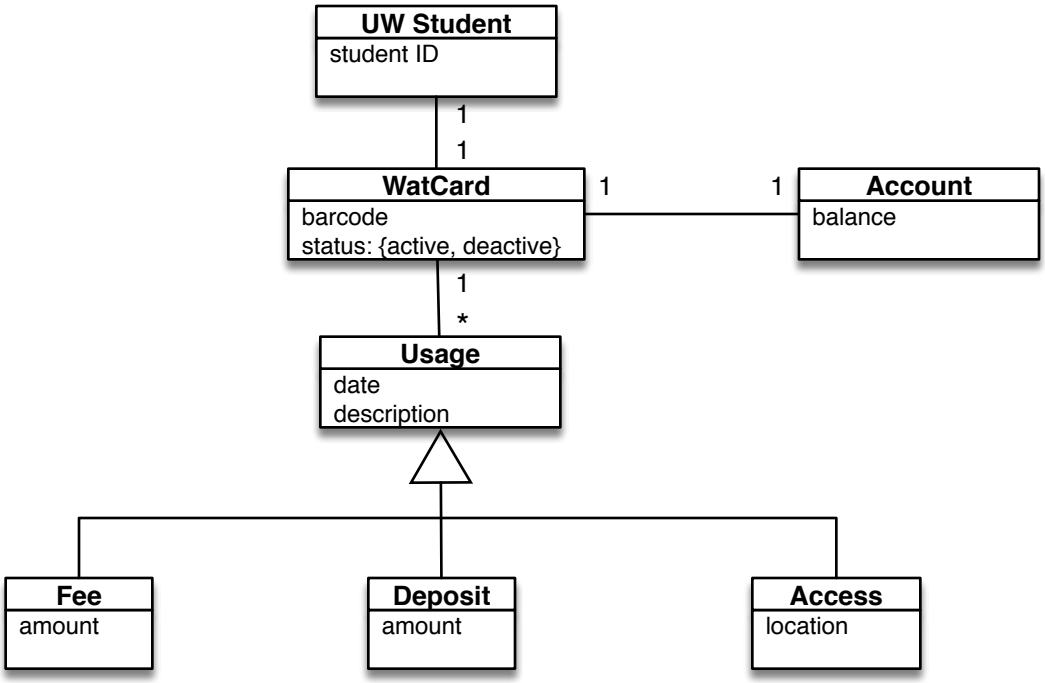
- lookup a book
- put an in-library book on hold, for pickup at the library desk
- recall a book that is currently out on loan
- borrow a book
- return a book
- pay a fine
- send out notices to patrons regarding overdue books
- validate a patron via his or her WatCard (assume that the WatCard system is an adjacent system)

Question 9. (20 marks) Domain Modelling

This question tests your proficiency at domain modelling and with using UML class diagrams.

The university has decided to develop its own WatCard system, rather than rely on software from a vendor. The initial release of the software allows a WatCard holder to use his or her card to gain access to secure areas on campus (e.g., their residence after hours, computer rooms). He or she can also use the card as a debit card, meaning that he/she can deposit money into an associated account, and can purchase items in certain locations by using the card to withdraw money from the account. In the beginning, students will only be able to use their WatCards to pay various university-related fees: parking fees (at parking permit dispensers), library fees, equipment rental fees at the Physical Activity Centre, fees for official transcripts, etc.. Also, in the beginning, only online deposits will be supported. The system keeps a Usage record of every use of the WatCard.

Below is a domain model for the proposed WatCard system, although it is not necessarily the best model of the problem's domain.



9a) (4 marks)

In the model provided above, there is a 1-1 association between the UW Student and WatCard classes, and a 1-1 association between WatCard Classes and Accounts. Is this the best way of modelling this information (about students and WatCards and Accounts), or is there a better way? Defend your answer.

Question 9. [continued]

9b) (5 marks) Aggregation and Composition

In the model provided on the previous page, the relationship between a WatCard and a Usage, recording some use of the card, is a simple *association*. Alternatively, we could have modelled the relationship as an *aggregation* or *composition*. Which modelling decision would be better for this domain problem, and why?

9c) (4 marks) Associations

In the model provided on the previous page, there is no explicit association between the Deposit class and the Account class to which a deposit would made. Similarly, there is no explicit association between the Fee class and the Account class from which a fee payment would be made. Must an association between Deposit class and Account class (and between Fee class and Account class) be added to make the model reflect these relationships, or can these relationships be inferred from the existing associations? Defend your answer.

9d) (7 marks) Domain Modelling

The second release of the WatCard system will support multiple types of accounts. For example, Food Services would like to use the WatCard system to keep track of each student's meal-plan allowance, and to allow students to use their WatCards to purchase food and snacks at all Food Services eateries. As another example, IST would like to use the WatCard system to keep track of each student's balance in his or her printing account, and to allow students to use their WatCards to access various self-service printers across campus. A student's meal-plan balance is separate from his or her printer-account balance, and both are separate from the debit-card balance.

The second release of the WatCard system will also support the use of the WatCard as a debit card in many more locations where a student might make a purchase (e.g., stores on campus or in the vicinity of the University). Purchases and deposits (to the debit-card balance only) can be made at any of these locations. The system will keep track of the location of each debit-card purchase and deposit.

Revise the UML model on the previous page to reflect the expanded domain of the second release of the WatCard system. Cross out any modelling elements that are no longer needed, and add any modelling elements that are introduced by the expanded domain. Use multiplicities, qualifiers, aggregation, generalization, etc. to best express the expanded domain. If you need to change your answer, you can use the blank page at the end of the exam to rewrite your full revised model.

Question 10. (9 marks) Short Answer

These questions evaluate your understanding of elicitation, modelling paradigms, quality requirements.

10a) (5 marks) Elicitation

You are eliciting requirements for a new release of an existing product. For each of the elicitation problems below, name one elicitation technique that would *best* address that problem.

i. You want to understand the requirements of the existing version of the product.

ii. You want to solicit several opinions about how to improve the current version of the product.

iii. You want to check your understanding of the collected information and of elicited requirements by re-expressing them in a different language.

iv. You want a systematic means of ensuring that you elicit all relevant aspects of the product's requirements.

v. You want answers to specific questions about late-phase requirements details.

10b) (4 marks)

List two (2) reasonable quality requirements of the WatCard system from Question 9.

Car Rental Agency (used in Questions 11 and 12)

A proposed information system for a car rental agency keeps track of the rental agency's

- customers (both customers that currently have cars out on loan and past customers)
- vehicles that the agency has in its inventory
- contracts that record current and past rental agreements
- reservations for future rentals

A customer has a name, address, driver's licence, age, and credit card information. Customers may have one or more credit card numbers on file with the car rental agency. Customers must be at least 18 years of age to rent a car. Any customer who is 25 years or younger is subject to a "young-age fee".

Vehicles are classified by make and model (e.g., Toyota Matrix); other information includes the year they were produced, the most recent odometer reading, and vehicle identification number (i.e., a unique 20-character ID). Rental rates depend on the type of vehicle (e.g., compact (small) car, a standard size car, or a luxury car). Rental rates depend also on whether the car is being rented by the day, by the week, or by the month. (Assume that any rental agreement whose duration is more than 6 days is a weekly rental, and that any agreement whose duration is more than 29 days is a monthly rental.) A vehicle is available for rent for a specified range of dates if it is not already assigned to a rental contract during that period.

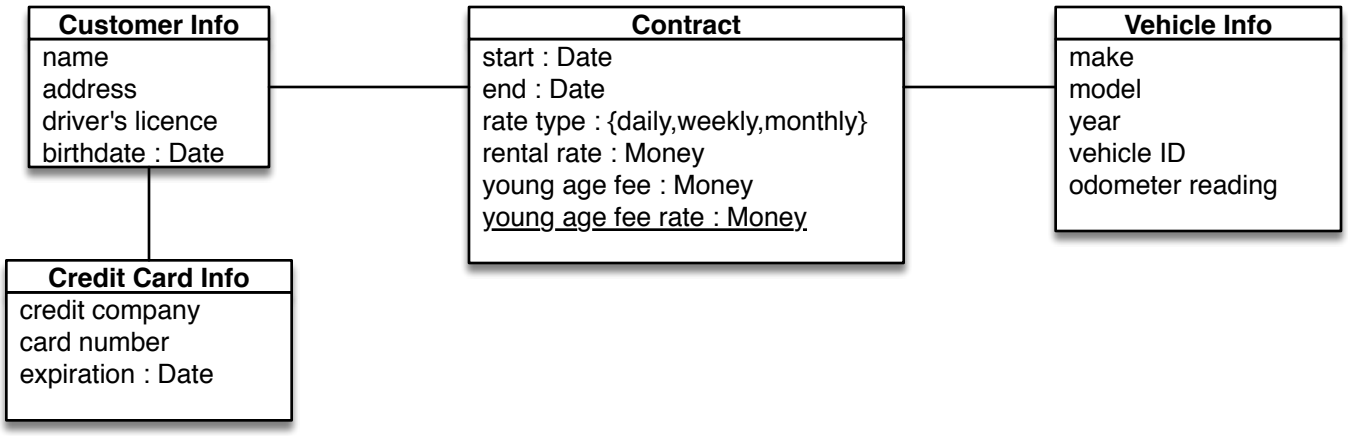
A contract records information regarding one car rental by one customer: the start date of the rental, the end date, and a young-age fee, if applicable. Rental rates are subject to change, so the contract records the rental rate that applied at the time the contact was negotiated.

A reservation records the planned start and end dates of the rental, the requested vehicle type (i.e., compact, standard-size, luxury), and the quoted rental rate. Reservations are requests. The car rental agency will do its best to have a vehicle of the requested type available at the time of the rental, but it might substitute a more expensive vehicle and still charge the quoted rate.

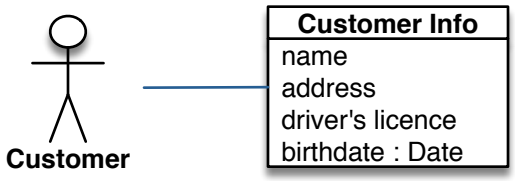
Question 11 (25 marks) Domain Models

The following questions ask about the domain model for the car rental agency described in the above problem statement. You are to comment on or extend a provided ***partial*** domain model that ***may be incorrect.***

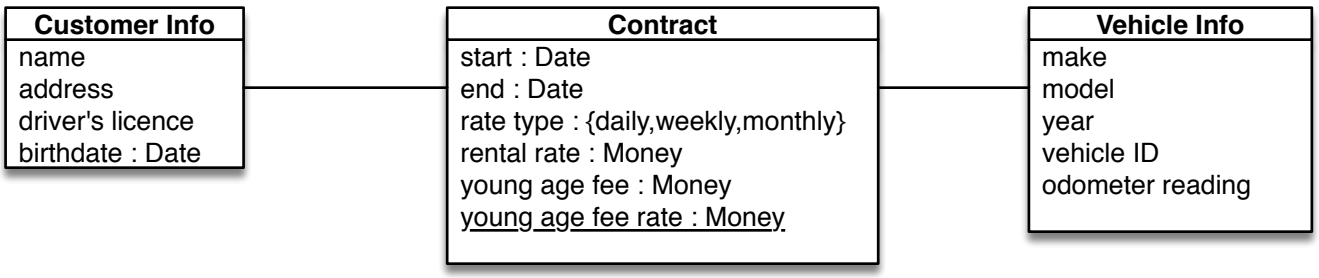
11a) (3 marks) Annotate the partial domain model given below with *multiplicities*.



Question 11 Domain Models (cont.)



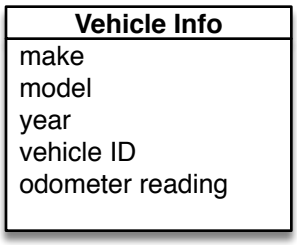
11b) (5 marks) Why do some domain models include both an actor (e.g., customer) and a class representing information about that actor (e.g., Customer Info)? Are both entities needed in a *requirements* model? Are both entities needed in a *specification* model?



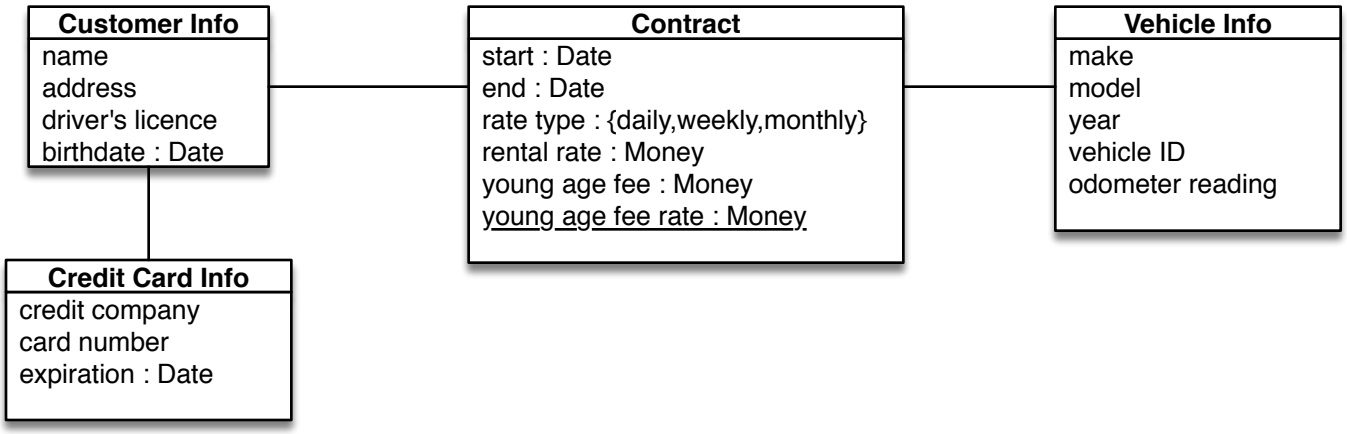
11c) (5 marks) In the above partial model, **Contact** is modelled as a regular class rather than as an association class. Which modelling decision (class vs. association class) is more appropriate and why?

Question 11 Domain Models (cont.)

11d) (5 marks) Extend the model below to include rental rates based on vehicle type (compact, standard size, luxury) and duration of the contract (daily rate, weekly rate, monthly rate). All cars of the same vehicle type have the same rental rate. Use the most appropriate UML Class Diagram constructs to model this information.



11e) (7 marks) Extend the model below to include the concept of a car rental *reservation*. Include associations, attributes, multiplicities, etc.



Cellphone Information System (used in Questions 13-16)

You are to model different views of the requirements for a cellphone information system, including customer accounts, calling plans, records of calls, bills, etc. On the next page is In the following, a *caller* is a person or cellphone that initiates a call, and a *callee* is a person or cellphone that receives a call.

A customer has some number of cellphones, each of which has a calling plan (for simplicity, assume that a cellphone is associated with exactly one calling plan). The contact gives the duration of the calling plan; the terms of the calling plan do not change during this time period. The calling plan determines how much it costs to use the cellphone:

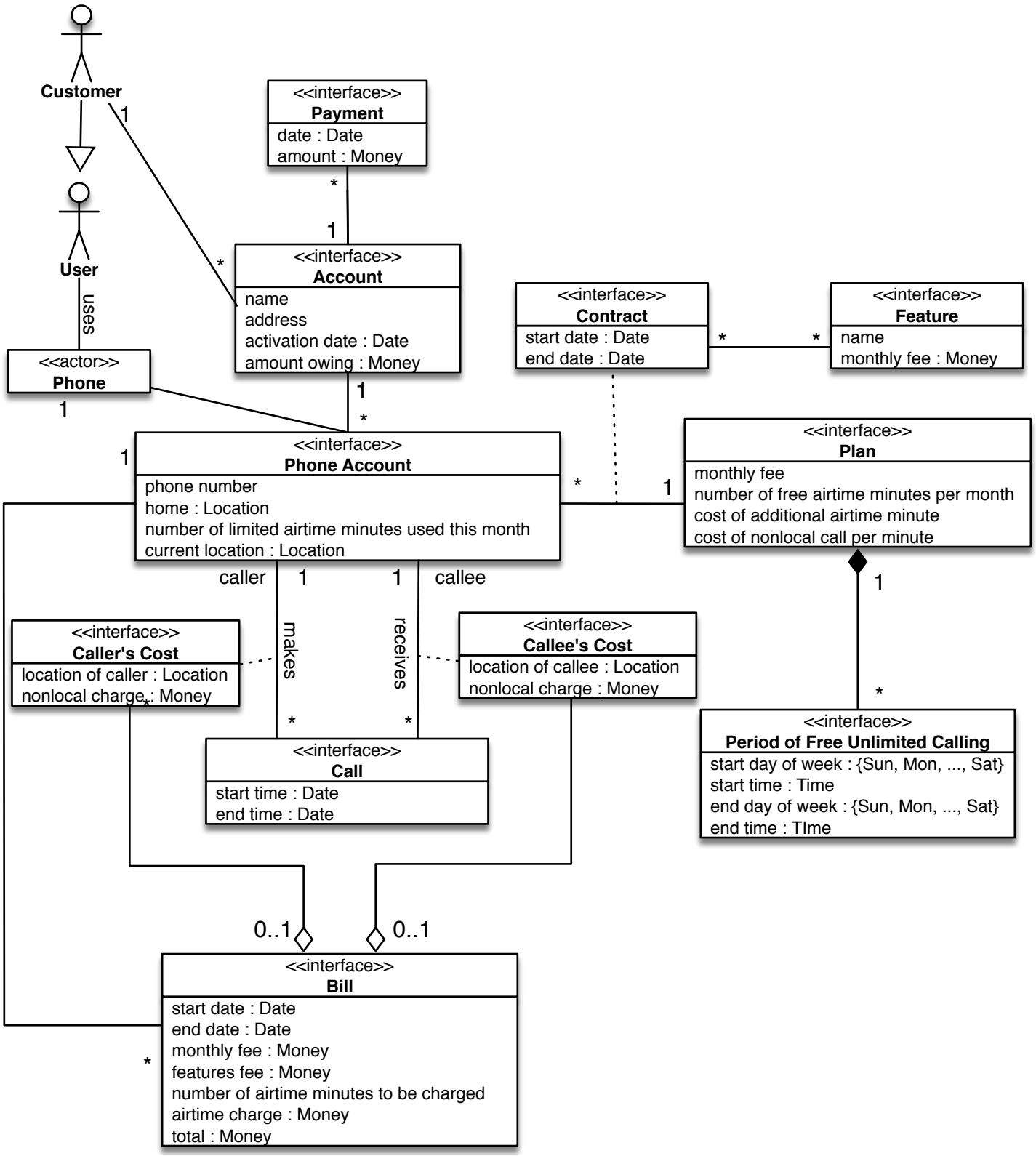
- There is a flat monthly fee for the plan plus some additional flat fees for features (if the customer has added any features to the contract).
- The customer is charged for any nonlocal calls that he or she makes (i.e., for calls where the customer is the caller, and the caller and callee are not in the same location -- the home locations of the caller and the callee are not important in determining whether a caller is charged for a call).
- The customer is charged for any nonlocal calls that he or she receives (i.e., for calls where the customer is the callee, and the callee is not in his or her home location when he or she receives the call -- the home or current locations of the caller are not important in determining whether a callee is charged for a call).
- The customer may also be charged for airtime. The calling plan will provide some number of free minutes per month (though the number could be 0). Also, the calling plan will not deduct from these minutes any call that starts during a period of unlimited calling (e.g., during evenings and weekends)

The system keeps track of all successful calls (i.e., all calls that are answered) -- both past calls and currently occurring calls -- so that monthly bills can contain an itemized list of the calls for which the customer is being charged.

Bills are issued once a month for each phone with an active calling plan (i.e., whose contract has started but has not yet ended). Bills list the fees, the extra charges for additional minutes of calling time beyond the free minutes provided by the calling plan, and the charges for nonlocal calls.

The customer's account keeps track of the amount of money the customer owes the cellphone company. This amount is based on the totals of the bills that have been issued and the amounts of the payments that the customer has made.

Domain Model for the Cellphone Information System (used in Questions 13-16)



Question 13. (10 marks) Use Case Diagrams

This question tests your proficiency at identifying and modelling use cases.

Draw a use-case diagram for the proposed Cellphone Information System. For each use case, provide a 1-sentence description of the capabilities represented by that use case.

Question 14. (10 marks) Use-Case Descriptions

This question tests your proficiency at identifying scenarios and at writing use-case descriptions.

Below is a partial use-case description for establishing a phone call between two actors: it describes the main scenario for making a call. In this description, Caller and Callee are instances of User that are involved in the call. You are to help to complete this use-case description by providing one alternative flow and one exception flow. Use your knowledge of scenarios of phone calls to identify reasonable alternative and exception flows. *Don't provide more than one alternative flow and one exception flow—we will mark only the first of each.*

Typical Process Description

Caller	Cellphone System	Callee
1. Caller initiates a call to the callee (designating a phone number)		
	2. System determines that the dialled phone number is associated with an active call plan	
	3. System establishes a connection between the caller and the callee	
	4. System notifies the callee of the call request	
		5. Callee answers the call
	6. System establishes a voice connection between the caller and the callee. This is the start of the call for billing purposes.	
7. Caller ends the call		
	8. System dismantles the connection between the caller and callee, and updates billing information	
		9. Callee hangs up

Question 15. (15 marks) Domain Model

This question tests your proficiency at domain modelling using UML.

Extend the provided domain model to support text messaging. By default, the Cellphone Company charges 20¢ for each sent text message, and received messages are free. However, a customer can subscribe to a text-messaging feature for his or her phone that will make text messaging cheaper.

- Feature **Messaging 250** is $5 a month, and allows the customer to send 250 messages a month for free. Each additional message is 15¢ a message.
- Feature **Messaging 500** is $10 a month, and allows the customer to send 500 messages a month for free. Each additional message is 15¢ a message.
- Feature **Messaging 2500** is $15 a month, and allows the customer to send 2500 messages a month for free. Each additional message is 15¢ a message.
- Feature **Messaging unlimited** is $20 a month, and allows the customer to send an unlimited number of messages a month for free.

Draw your extended domain model below. Copy only the portions of the provided domain model that you need in order to show how your new environmental phenomena fit into the original model.

Question 17. (35 marks) Domain Model

This question tests your proficiency at domain modelling using UML.

Provide a domain model for the Parking Information System described below. Use UML syntax, and use appropriate UML constructs (e.g., association names or role names, multiplicities, association classes, composition and aggregation, generalization, class scope attributes) to make your domain model as *complete and precise as possible*.

Parking Information System

All students, employees, and visitors to the university must pay for parking. Students and employees may buy permits that allow them to park in particular lots. Or they may park as visitors, and pay a fee to park in visitors' parking.

The university's parking lots are divided into reserved employee parking lots, unreserved student parking lots, and visitor lots:

Reserved Employee Parking Lots (Lots A, B, H, K, L, O, R)
Permits for these lots are sold to employees. Each permit is associated with a particular parking lot and allows the holder of the permit to park in that lot, anytime between 6:00AM and 3:00AM. For each of these lots, Employees buy a long-term permit for which they are charged a monthly fee. The monthly fee is the same for all reserved parking lots. A permit has a start date and an open-ended end date; the permit ends when the permit is cancelled.

The reserved parking lots are "card controlled", meaning that the permit holder has a physical card that he or she uses to get into the lot. The entrance to each reserved parking lot is protected by one or more gates (maximum three gates). A permit holder slides the card into a lot's gate's card reader; if the card is associated with that lot, and if the card is valid (i.e, if the permit associated with the card has not been cancelled), then the gate rises temporarily to allow a single car into the lot.

Reserved Employee Parking Lots can be used by visitors during the evenings; the starting time for visitor parking is different for each lot. During this time, the gates accept coins, and visitors can pay a fee to gain entry to the reserved lot. The fee for visitor's parking is the same for all lots that allow visitor parking.

Unreserved Student Parking Lots (Lots C, N, W, X)
Permits for these lots are sold to students. Holders of such permits may park in any of the unreserved student parking lots anytime between 6:00AM and 3:00AM. Permits are sold to students on a per term basis.

The unreserved student parking lots are "permit controlled", meaning that the lot is always open, but that cars must display a valid permit at all times. There is no visitor parking in the unreserved student parking lots.

Visitor Parking Lots (Y, Z)
There are two parking lots that are dedicated to visitor parking. Anyone can park in the visitor lots anytime between 6:00AM and 3:00AM. These lots are "fee-controlled", meaning that the entrance to the lot is protected by a gate that accepts coins, and visitors pay a fee to gain entry to the lot. The fee for visitor's parking is the same for all lots that allow visitor parking.

ACM International Collegiate Programming Contest (used in Questions 18-19)

Below is a *partial* domain model for an information system that will support the ACM International Collegiate Programming Contest (ICPC). The system will record historical data about contests, teams, and their performances. Below is a description of the contest and its rules, adapted from the Wikipedia description. (http://en.wikipedia.org/wiki/ACM_International_Collegiate_Programming_Contest).

The ICPC is a two-tiered competition held annually among teams of students, with each team representing the students' university. Teams first compete in Regional Contests, held around the world. The winning team from each Regional Contest advances to the ICPC World Finals.
The ICPC is a team competition. New teams are formed each year. A team consists of three students, each of whom is enrolled in university and has had less than five years of university education at the time that the team registers for a Regional Contest. Students who have previously competed in two World Finals or five Regional Contests are ineligible to compete again.

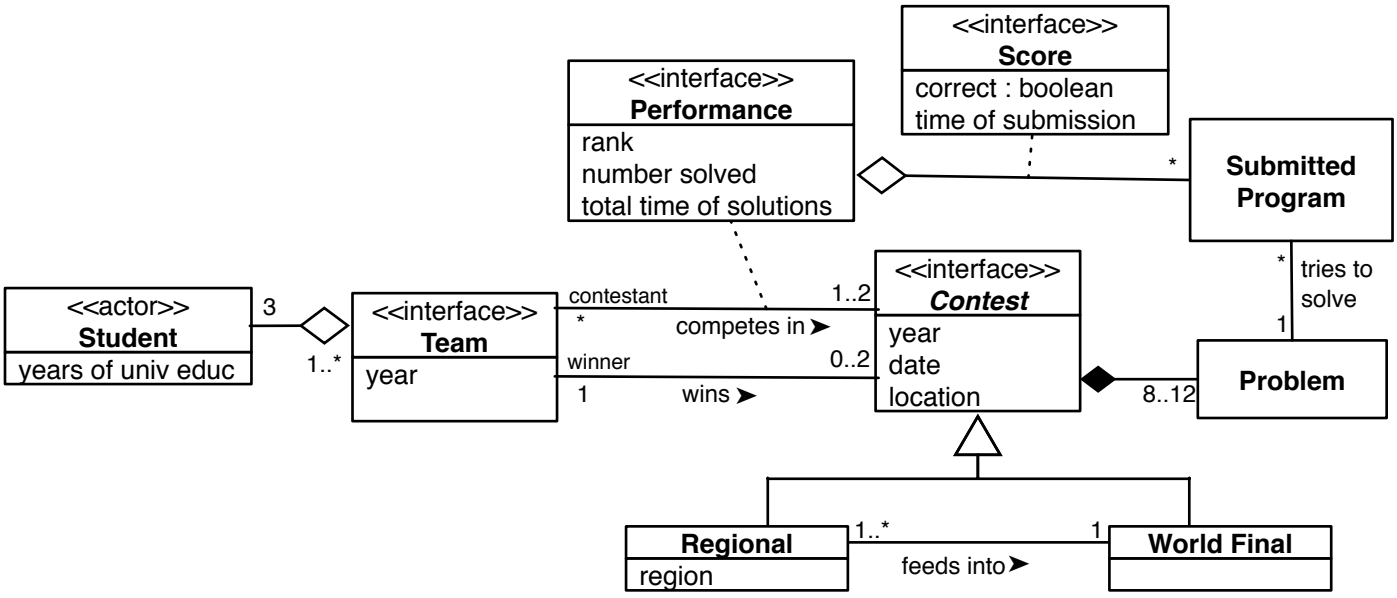
During a contest, the teams are given 5 hours to solve between 8 and 12 programming problems. A team submits solutions as programs in C, C++, or Java. The submitted programs are run on test data. If a program fails to give a correct answer, the team is notified and can submit another program.

The winner is the team that solves the most problems correctly. If teams solve the same number of problems, their relative ranking is determined by the time (since the start of the contest) that it took a team to submit each correct solution, plus a 20-minute penalty for each incorrect submission for a problem ultimately solved.

For example, consider a situation where two teams, Red and Blue, both solve two problems. One hour after the start of the contest, Team Red submitted a correct program for problem A (submission time was 1:00). Two hours and 45 minutes after the start of the contest, it submitted a correct program for problem B (submission time was 2:45). It also submitted an incorrect solution to problem C, but this is ignored since the team never solved C. Team Blue submitted correct solutions to problems A and C, with respective submission times of 1:20 and 2:00. It also submitted one incorrect program for problem C. Then, the teams' timed performances are

- Team Red: 1:00 + 2:45 = 3:45
- Team Blue: 1:20 + 2:00 + 0:20 = 3:40

Lower times are better, so Team Blue performed better than Team Red. But any team that solved more than two problems performed better than both Red and Blue, regardless of how long it took.



Question 19. (20 marks) RE Reference Model

This question tests your ability to derive specifications from requirements.

For each of the following requirements:
(1) State what *information* (if any) needs to be added to the domain model in order to provide the vocabulary to restate the requirements as a specification (a textual answer is sufficient).
(2) Re-express the requirement as a *specification*.
(3) State any *assumptions* that are needed in order to argue that any system that satisfies the specification would also satisfy the requirement.

19a) The winner of a contest must have correctly solved as many or more problems than any other team in the contest.

19b) All of the students on a team are enrolled at the same university.

19c) Every programming contest has exactly one winner.

19d) Each team is affiliated with a university, and all of the students on the team are enrolled in that university.

Expresso Shop (used in Questions 20–24)

Questions 20-24 are based on the Expresso Shop problem described below.

Customers want to buy espresso coffee pods from the Expresso Shop and have the pods shipped to their homes (or to another address).

The Expresso Shop sells a variety of espresso coffee pods made from different types of coffee. Pods of the same coffee type are packaged and sold 10 to a pack. A customer order is some number of 10-packs (total price of the order, sales tax, shipping fee, etc.). If the Shop does not have the customer's entire order in stock, then it breaks the order into multiple shipments, it ships the packs that are in stock, and it ships the remaining packs when the inventory is restocked. (Regardless of the number of shipments needed to fill an order, the customer is charged only one shipping fee.) The Shop pays a shipping company to make the deliveries. Customers pay by credit card. The Shop collects the customer's credit-card information and shipping address each time the customer places an order. The Shop will charge the customer's credit card immediately before every shipment, for the pods being shipped. (The customer's credit card is charged the shipping fee as part of the charge for the first shipment of an order.) The Shop owners would like to maintain information about the inventory of Expresso pod packs. Once a month, a report is produced that summarizes the number of 10-packs of each type of coffee that were ordered in the last month — regardless of whether they have been shipped yet.

Question 20. (10 marks) Use-Case Model

This question tests your proficiency at use-case modelling.

Provide a use-case model for the *requirements* of the Expresso Shop. You do *not* to need provide brief descriptions of the use cases.

Question 21. (30 marks) Domain Model

This question tests your proficiency at domain modelling using UML.

Provide a Domain Model for *the specification* of the Expresso Shop. Your domain model must distinguish interface phenomena from domain (environmental) phenomena. Use UML syntax, and use appropriate UML constructs (e.g., attributes, association names or role names, multiplicities, association classes, composition and aggregation, generalization, <<actor>> and <<interface>> stereotypes) to make your domain model as *complete and precise as possible*.

Question 22. (6 marks) Specifications

Rewrite each of the following requirements as specifications, based on interface phenomena that appear in your domain model in Question 21.

R1: The Expresso Shop eventually sends to a customer's specified shipping address all of the coffee pod packs that the customer wants.

R2: The Shop will charge the customer's credit card at the time of each the customer's shipments.

Question 23. (4 marks) Assumptions ($S, D \models R$)

List four distinct domain assumptions that you need to make in order to "prove" that your specification for R1 of the Express Shop software system (given in Question 22) meets the requirement R1.

Question 25. (13 marks) Short Answer

25a) (3 Marks) Elicitation

What does it mean to "invent" requirements, and why would one want to do this?

Question 26. (12 marks) Short Answer

26a) (4 Marks) Stakeholders

List four (4) types of stakeholders. For each, provide a one-sentence description of the unique type of information or requirements that can be elicited from that stakeholder.

26b) (4 Marks) Elicitation

You are eliciting requirements for a new release of an existing product. For each of the elicitation problems described below, list a distinct elicitation technique that would best address that problem.

(a) You want to understand how users really use the existing system, as opposed to how they tell you they use the system.

(b) You want to invent new requirements or features to be added to the new release.

(c) You want to understand the original requirements of the existing system.

(d) You want to get a quick sense of the most popular features of the existing system.

26c) (2 Marks) Requirements vs. Specifications

What is the difference between a proposed system's requirements and its specification?

26d) (2 Marks) Use Case Diagrams

Under what circumstances should Time be an actor in a use-case diagram?

26e) (4 Marks) Elicitation

Suppose you have been asked to help develop a new student-transcript system for the university. The system will record students' course marks and (if appropriate) co-op evaluations; will calculate and report each student's major and overall averages, their standing in their programs, their class rank, and other statistical information; and will present a record of a student's academic record as a written transcript.

(a) Give a complete a list of the system's stakeholders (e.g., registrar's office), and the role (e.g., customer, user, etc.) that each stakeholder plays.

(b) Who is the most important stakeholder?

Question 27. (15 marks) Elicitation

Pick three of the nine requirements elicitation techniques covered in lecture, and fill in the table rows for those techniques. Place a ✓ in a table entry if the technique is particularly good at addressing the requirements problem associated with that entry. Place an X in the table entry if the table entry is not particularly good at addressing the entry's associated problem. Elicitation techniques may or may not be suitable for eliciting requirements of brand New Software Systems or of Existing Software Systems that are being extended. A technique may or may not be suitable for use during Early Elicitation phases or during later Detailed Elicitation phases.

Briefly defend each of your table marks.

Elicitation Technique	New System	Existing System	Early Elicitation	Detailed Elicitation
Document Analysis				
Questionnaire				
Interviewing				
Ethnographic Analysis				
Modelling				
Mockups				
Prototyping				
Brainstorming				
Creativity Workshop				

Vancouver Olympic Tickets (used in Questions 28–31)

In the next few questions, you will analyze the requirements and model the domain for Phase 1 of the Ticket Sales process for the Vancouver 2010 Winter Olympics (adapted from source: www.vancouver 2010.com). You do not need to model eventual seat assignments.

Tickets to the Vancouver 2010 Olympic Winter Games will be sold in phases. Phase 1 starts October 3, 2008. 60% of all tickets of all event sessions (where a *session* might be a heat, a semi-final, a final, an opening or closing ceremony) are available for purchase during this phase. Phase 1 of the 2010 Olympic Winter Games ticket program will move through four stages: Request Period, Lottery, Notification Period and Priority Access Period.

Request Period (October 3 – November 7): Tickets and ticket prices for every olympic event session and every predefined package of sessions are advertised. Each applicant submits one or more requests, where each request is for a specified number of tickets for some session or some package of sessions. If you require accessible seating for some event, indicate so on the request (you'll be contacted separately about your needs if you are allocated tickets for the event). Be sure to submit your requests by November 7, 2008 to be eligible – all requests will be weighted equally, regardless of when in the request period they are submitted. Only VISA payment cards will be accepted. Your VISA credit card will be checked for authenticity but no payments will be charged until the lottery period.

Lottery (November 8–23): In cases where ticket demand exceeds the available supply, our official supplier of ticketing services, Tickets.com, will use an automated random selection process (lottery) to ensure the fairest possible ticket distribution on a session by session basis (for example each session you have requested, if oversubscribed, will be subject to a separate lottery). Your Visa card will be charged immediately upon allocation of tickets if you are randomly selected.

Notification Period (By December 5): All applicants will be notified via e-mail regarding the status of their request and which sessions and/or packages they have secured. NOTE: Notification will be sent to the e-mail address provided in your account. Please ensure that you update your account information with any changes.

Priority Access Period (December 8- 22): The Priority Access Period is only open to Canadian residents who submit a request before November 7. At this time applicants will have the opportunity to purchase remaining tickets before they are released to the general public in early 2009. Applicants can add new sports or sessions to their own Olympic experience or even purchase tickets as a holiday gift for friends or family. Confirmation of your purchase will be immediate and payment will be charged at that time.

Question 28. (25 Marks)

Using UML Class Diagram notation, provide a Domain Model for the *specifications* of Phase 1 of the Vancouver Olympic Tickets Sales. List and briefly describe the interface phenomena in your domain model. (You may have to devise interfaces if the above problem description does not contain all of the phenomena that you need to complete questions 28-31.)

Your Domain Model will be evaluated in terms of its correctness, and its use of the UML modelling notation.

Question 29. (12 Marks) Specifications and Assumptions

Consider the following four requirements of the Vancouver Olympic Tickets Sales problem.

1. Applicants must be Canadian residents, regardless of citizenship, and must have a current and valid Canadian mailing address (P.O. boxes are not considered valid).
2. Applicants must have a valid VISA credit card.
3. If a session is undersubscribed (i.e., where ticket availability meets ticket demand), then all requests for that session will be fulfilled.
4. If a session is undersubscribed (i.e., where ticket demand exceeds ticket availability), the all available tickets will be distributed uniformly among requests.

29a) Recast the above requirements as *specifications* for a system to be built to automate Phase 1 of ticket sales.

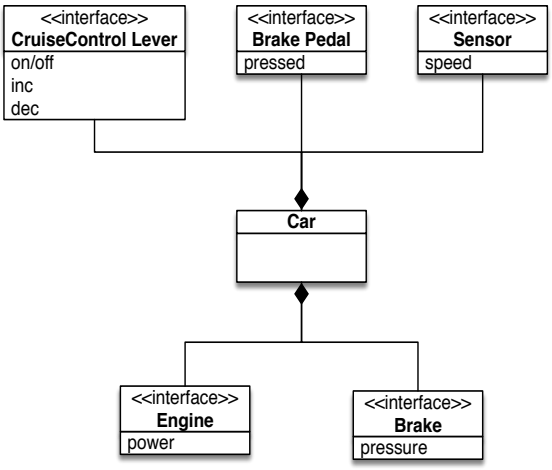
29b) List all *assumptions* that you need to make to argue that your specifications for the Vancouver Olympic Tickets Sales system meet the problem requirements.

Question 30. (10 Marks) Use Case Diagram

Provide a use-case diagram that provides an overview of all the functionality of the proposed Vancouver Olympic Tickets Sales system and the relevant actors, as described in the problem statement.

Question 32. (15 Marks) State Machine Modelling

Below is a Domain Model for the specification of a automobile Cruise Control System.



You are to write a State Machine model for an automobile cruise control system, where the inputs and outputs of your model are expressions over the vocabulary in the above Domain Model. When active, the cruise control system keeps a car travelling steadily at a driver-specified speed, without the driver having to keep his foot on the accelerator pedal. The cruise control system can be activated only if the car is travelling at least 60 km/hr and if the brake is not being pressed - at which point the driver-specified speed is set to be the car's current speed. The driver can increase or decrease the driver-specified speed using the inc and dec levers, respectively. An active cruise control system is automatically deactivated if the driver engages the brake (presumably to react to road or traffic conditions); but the value of the driver-specified speed is retained. The driver can resume a deactivated cruise control system with the on/off lever - but only if the car's current speed is within 10 km/hr of the driver-specified speed and the brake pedal is not depressed. The cruise control system terminates when the car's ignition is turned off.

Room Lighting System (used in Questions 33-35)
Consider a new safe and efficient system for lighting a room. The new system avoids waste by automatically turning lights off when a room is unoccupied.

The room is equipped with

- A collection of ceiling lights that are controlled as a group.
- A motion detector that senses the room every T seconds and determines whether the room is occupied
- A dimmer dial that a user can use to set the brightness of the ceiling lights. The dimmer's values range from 10% to 100% illumination

The requirements of the system are:

- The room is initially not illuminated.
- If a person occupies the room, there must be at least safe illumination in the room.
- If a user has set the light level in the room, the system should maintain that light level as long as the room is occupied (assuming that the set-light level is at least as bright as safe illumination)
- The lights should be off if the room as been unoccupied for T1 seconds
- If the room is reoccupied within T2 seconds since the last person left the room, then the most recent set light level must be restored.
- If the room is reoccupied after more than T2 seconds since the last person left the room, then the default light setting must be established. The default light setting is brighter than safe illumination.

Question 33. (15 Marks) Domain Models

Using UML Class Diagram notation, provide a Domain Model for the *specification* of the Room Lighting System.

Question 34. (9 marks) Requirements and Specifications

This question tests your understanding of the relationships between requirements and specifications.

34b) (3 marks) Specifications

Consider the following requirement of the lighting system:

> REQ: If a person occupies the room, there must be at least *safe illumination* in the room (there could be greater illumination if the set light level is greater than *safe illumination)*.

Re-express the above requirement as a specification that uses vocabulary from your domain model in question 33.

34c) (3 marks) Assumptions

List three (3) assumptions that you have to make about the lighting system domain in order to argue, or informally prove, that the specification you gave in Question 34b is sufficient to satisfy the requirement provided in Question 34b.

Question 35. (25 Marks) StateMachine Models

Draw a State Machine model of the described lighting system, where the inputs and outputs of your model are expressions over the vocabulary in the Domain Model that you provided in Question 35. You may use any state machine modelling construct presented in lecture. Declare any constants, variables and macros that you use.

Netflix (used in Questions 36-40)

Netflix is an internet service for streaming movies and TV shows to personal computers and TVs. Anyone can browse the Netflix library (by title, actor, director, genre), but one must have a subscription to be able to stream videos. A user can activate (i.e., create), suspend, or cancel their membership. An account is active as long as it has not been suspended (and not re-activated) or cancelled. The subscription fee is $7.99 per month, charged on the monthly anniversary of the day that the subscription was activated. If a user has an active subscription and accesses the Web site from within Canada, then the user can stream as many videos (from the Netflix library) as he or she wants, whenever he or she wants. A user can pause, rewind, fast-forward or stop a stream as often as they like.

Question 36. (10 marks) Use Case Diagrams

Create a use-case diagram for the Netflix service. Provide a short 1 sentence description of each use case.

Question 37. (8 marks) Scenarios

You are to create a "fully dressed" (i.e., tabular) use-case description for the *main scenario*, two *alternative* scenarios, and one *exceptional* scenario associated with streaming a video. The main scenario is triggered by the user selecting a particular video (this could happen in a variety of ways, but how it happens is not part of this use-case description). The user can switch among viewing (at normal speed), pausing, rewinding, or fast-forwarding at any time, and as often as they like. Stopping a stream ends the use case, as does losing internet service (if the user wants to watch the rest of the video, the user would have to initiate the streaming use case again and fast forward to where he or she want to resume viewing).

Question 38. (15 marks) Domain Model

You are to create a Domain Model for the *specification* of the Netflix service. Your domain model must include any phenomena to which your constraints in question 40 and your specifications in question 41 refer. Use appropriate UML constructs (e.g., multiplicities, association classes, composition and aggregation, generalization) to make your domain model as complete and precise as possible.

Question 40. (12 marks) Specifications and Assumptions

Consider the following requirements of the Netflix streaming service:

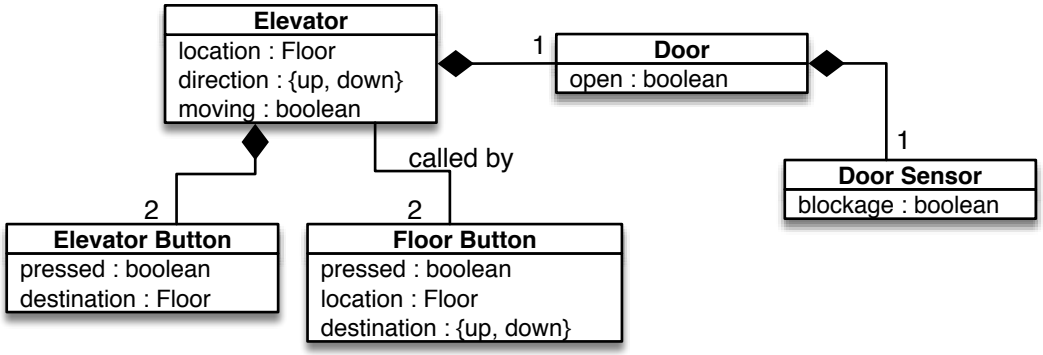
- R1: A user must have an active account in order to stream videos.
- R2: A user must access the Netflix Web site from within Canada in order to stream videos.
- R3: A user can pause a stream at any time.
- R4: Every user is charged $7.99 a month, on the monthly anniversary of the day that the subscription was activated.

40a) Transform each of the above requirements into a *specification* that uses vocabulary from your Domain Model from question 38.

40b) List all *assumptions* that you need to make to argue that your specifications for the Netflix system meet the problem requirements.

Question 41. (20 marks) State Machine Modelling

You are to create a state-machine model for an elevator that has the following domain model.



There is one button on each floor just outside of the elevator that passengers can press to summon the elevator to come pick them up. There two buttons inside of the elevator, one button for each floor, that passengers can press to instruct the elevator to deliver them to the associated floor.

When idle, the elevator sits stationary at the last floor serviced, with its doors closed. When there is a request to service a particular floor (i.e., to deliver a passenger to that floor, or to pick up a passenger from that floor), the elevator moves to that floor (if necessary) and opens its doors. A request to service a floor remains outstanding until the elevator reaches the requested floor and opens its doors; at which point, all outstanding requests for that floor are considered serviced. If the elevator is stationary, then requests to service the floor that the elevator is currently on have priority over requests to service the other floor.

For simplicity, your model should have only one door for the elevator (i.e., don't model the outside doors on each floor). When the elevator door is opened, it remains open for 5 seconds. If the door sensors detect some blockage on closing the doors, the doors will reopen and again try to close after 5 seconds.

For full credit, your model should make the most effective use of UML state-machine modelling constructs

JobMine (used in Questions 42-46)

Consider a simplified version of JobMine, a co-op job application system, that focuses on job applications, interview schedules, and matching students with jobs.

Employer's View of JobMine

An employer submits to JobMine a Job Registration Form for each available job position, indicating the job title, location, start and end dates, job summary, desired academic level of applicants (junior, intermediate, or senior), and desired academic disciplines of applicants (e.g., computer science, computer engineering, etc.). The employer receives an email confirmation when the job is posted and applications are available for viewing. The employer then reviews applications and selects which applicants are to be interviewed. The employer submits to JobMine an interview schedule that specifies dates and times of interview slots. The employer interviews students at the Tatham Centre, by telephone, or by video conference. Following the interviews, the employer submits to JobMine a ranking of the interviewees: exactly one first-choice interviewee is ranked with the value "1"; other interviewees whom the employee is willing to hire are ranked with values "2" to "9", where a lower value indicates a higher preference to hire; and any interviewee whom the employee is not willing to hire is left unranked. If the employer submits ranking information by Thursday at noon, then at 4:00pm the next day, JobMine will have information about whether any of the ranked applicants has been matched with the job; if the employer submits ranks after Thursday at noon, the employer will have to wait until the following Friday at 4:00pm for hiring results.

Student's View of JobMine

Students are asked to keep an updated resume loaded in JobMine at all times. Resumes can be changed at any time. Every Monday morning at 7:00am, new job postings are available for viewing on JobMine, and those postings are removed the next day at 11:59pm. During this time period, students who do not yet have a job are expected to view the job postings and to apply for any job of interest. Whenever a student applies for a job, an application is created using the student's current resume and information from Quest (e.g., the student's academic record and the student's co-op record, if there is one). An application remains active until the employer turns down the student, the student withdraws the application, or the system matches the student with some job. At any time, a student can have up to 50 active applications. A student who has active applications is expected to check JobMine daily to see if the employer of one of the active applications has selected the student to be interviewed. Students can view the interview schedules of employers who are interviewing them, and the student can choose a time from among the employer's open interview slots. The student can set or change the time of a scheduled interview up to 48 hours before the first day of the employer's interviews. If the student neglects to sign up for an interview slot, then JobMine will automatically assign a slot to the student (automatic assignment happens 24 hours before the first day of the employer's interviews). The ranking period for students opens every Thursday at 3:00pm and closes the next day at noon. If, since the previous week's ranking period, any employer has indicated interest in hiring the student (by ranking the student -- see above for details), the student is expected to express his or her interest in the employer's job by ranking it: rank values range from "1" to "9", where a lower value indicates a higher preference (so ranking a job with value "1" indicates that the job is one of the student's top choices). If a student neglects to rank a job and the employer has expressed interest in hiring the student, the system will automatically assign a ranking of "9" to that job. After Friday at 4:00pm, the student can check JobMine to see if he or she has been matched with a job.

JobMine Problem Description (cont.)

Matching Students with Jobs

On Fridays at 2:00pm, the algorithm that matches employers' rankings (of desired applicants) with students' rankings (of desired jobs) is executed. The matching algorithm works as follows:

1. For each student-job combination, the employer's and student's ranking numbers are summed together. (Recall that if an employer ranks a student, but the student does not rank that job, the system automatically assigns the student's ranking to be "9".)
2. A random number is assigned to each student-job ranking combination.
3. The algorithm looks for the lowest sum (1+1=2) that has the lowest random number. If both the student and the job are still available, the student is then assigned to that job.
4. The matching algorithm then continues to look for the next lowest sum with the lowest random number, and so on, until all potential matches take place.

On Fridays at 4:00pm, the results of the matching algorithm can be viewed by students and employers.

Question 42. (10 marks) Use Case Diagram

Create a use-case diagram for this system. Model only what has been described above (i.e., do not try to identify all actors or all use cases). Provide a "brief" use-case description (1-2 sentences) of each use case.

Question 43. (8 marks) Scenarios

You are to create a "fully-dressed" (i.e., tabular) use-case description for the *main scenario*, two *alternative* scenarios, and one *exceptional* scenario associated with editing students' interview schedules. The use case is triggered by the student signing on to JobMine and asking to view his or her interviews. Selecting a particular interview allows the student to see the date and time of the interview, if it has already been scheduled, and to see the dates and times of other available interview slots for this job. The student can simply view the interview schedule, schedule or reschedule the interview, or select a different interview to view/edit.

Question 44. (15 marks) Domain Model

You are to create a Domain Model for the *specification* of the simplified JobMine. Your domain model must include any phenomena to which your constraints in question 45 and your specifications in question 46 refer. Use appropriate UML constructs (e.g., multiplicities, association classes, composition and aggregation, generalization) to make your domain model as complete and precise as possible.

Question 46. (9 marks) Specifications and Assumptions

Consider the following requirements of JobMine:

R1: A student must be enrolled in a co-op program in order to use JobMine.

R2: A student who is matched with a job should have no active applications and no scheduled interviews.

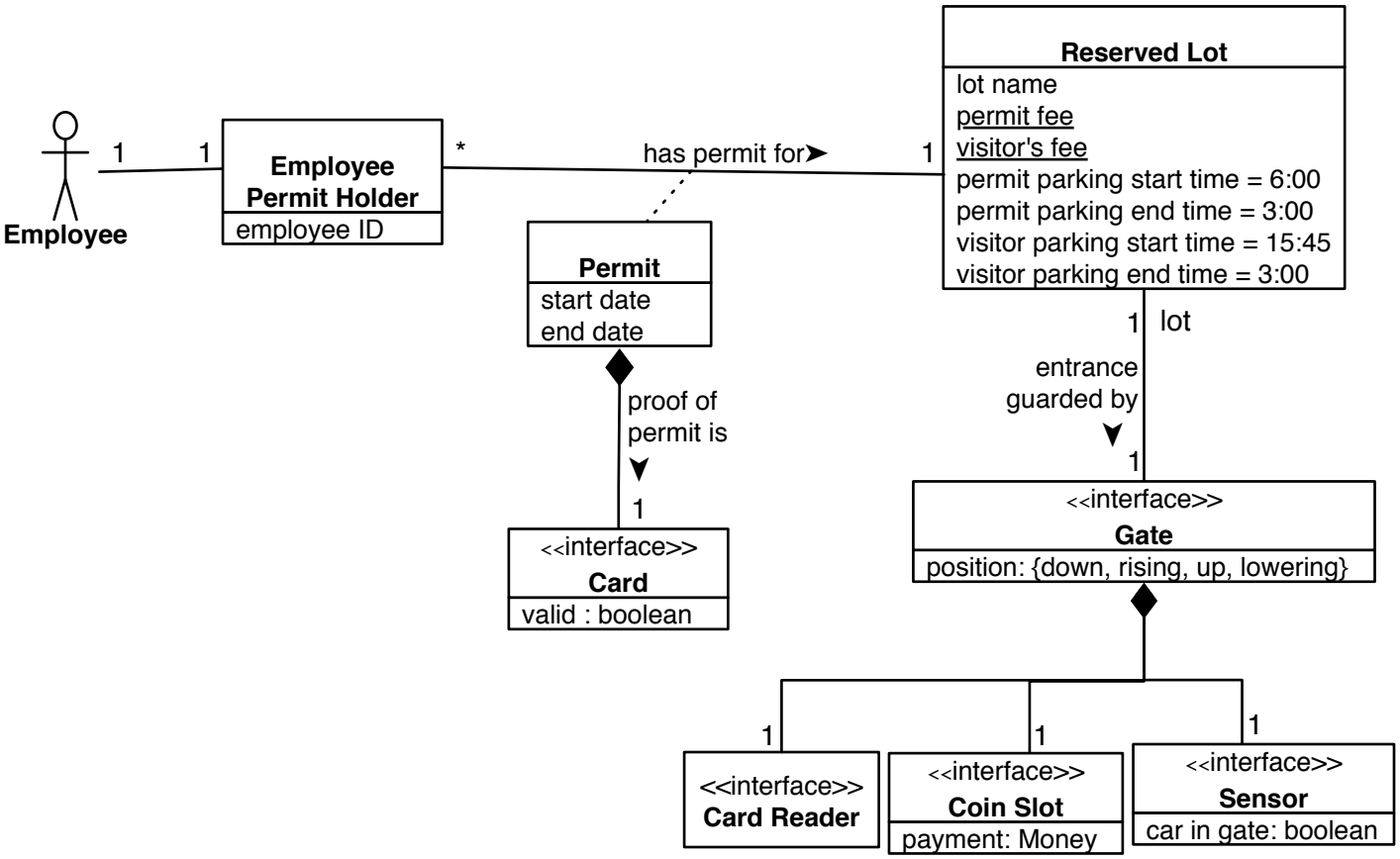
R3: Every student is matched with a job before the start of the relevant co-op term.

46a) Transform each of the above requirements into a *specification* that uses vocabulary from your Domain Model from Question 44.

46b) List all *assumptions* that you need to make to argue that your specifications for the JobMine system meet the problem requirements.

Question 47. (25 marks) StateMachine Models

You are to create a state-machine model for the software controller of a gate to a UW reserved employee parking lot. Below is the domain model for this system.



A UW employee can buy a long-term permit for a particular reserved parking lot. The permit holder is given a physical card that he or she uses to get into the lot. A permit has a start date and an open-ended end date. When the permit is cancelled, it has a defined end date, which is the last date that the physical card is valid.

Permit holders can park in the lot anytime between 6:00AM and 3:00AM. The entrance to the reserved parking lot is protected by a gate. A permit holder slides his or her permit card into a lot's gate's card reader. If the card is associated with that lot and the card is valid and the current time is within allowed permit-parking time, then the gate rises and stays up for 5 seconds before lowering again.

Anyone can park in a reserved parking lot during the evenings; the starting time for visitor parking is 3:45PM and the end time is 3:00AM. During this time, the gates accept coins and a visitor can pay a fee to gain entry to the reserved lot: once sufficient coinage has been inserted into the coin slot, the gate rises and stays up for 5 seconds before lowering again. No change is issued if a visitor pays more than the visitor fee.

When the gate is not down, it will not read cards or accept coin payments. Once the gate is lowered, it can accept the next visitor's fee or the next permit car. If the car is under the gate when the gate starts to lower, the gate will stay raised for another 5 seconds. Note that a permit card will not be accepted outside of permit-parking hours, and coinage will not be accepted outside of visitor-parking hours.

For full credit, your model should make the most effective use of UML state-machine modelling constructs. Be sure to declare any events that you use and any variables (outside of those in the domain model).

Grand River Transit - Real Time (G(RT)^2 for short) (used in Questions 48-52)

Consider a proposed software system called Grand River Transit - Real Time (or G(RT)^2 for short) that provides real-time information about the schedules of Grand River Transit (GRT) buses, including route planning.

A user interacts with the software using his or her smartphone. The user can provide an address (or use their current location), and the software will display nearby bus stops and the bus routes that service those bus stops. (In this assignment, buses that travel in different directions are considered to be different bus routes. Also, the bus stops "near" an address are either (1) all bus stops within two blocks of the given address, or (2) if there is no bus stop within two blocks, it is the closest bus stop to the given address.) If the user selects a particular bus stop and a particular bus route, the software will highlight the chosen route superimposed on a basic street map. The software will also highlight the current location of the next bus (on the chosen route) that will stop at the chosen bus stop, and display the predicted arrival time of that bus. The predicted arrival time for a bus is not based on the published bus schedule; it based on the bus's current location and on historical data on how long it takes buses on the chosen route to travel from one stop to the next at different times of the day.

Alternatively, the user can provide two addresses, an origin A and a destination B, and the software will provide real-time routing for a multi-hop bus trip from a bus stop near A to bus stop near B. The software will determine the best possible routes from bus stops near A to bus stops B, will then find candidate buses that are currently servicing those routes, and will display up to three trips:

- the next trip from a bus stop near A to a bus stop near B with the shortest duration
- the next trip from a bus stop near A to a bus stop near B with the earliest arrival time at stop B
- the next trip from a bus stop near A to a bus stop near B with the fewest number of bus transfers

The three trips are superimposed on a basic street map, showing the full routes from the trips' starting points to their end points. If the user views a particular trip, the software will display the sequence of buses that make up that trip, and the start and end times for that trip. The user can view the different trips before selecting a particular trip. Once a trip is selected, the software will highlight the current location of the next bus (on the first leg of the selected trip) to arrive at the starting bus stop near address A. On each leg of the trip, when the user arrives at a bus stop where the trip indicates a transfer to a different bus route, the software highlights the current location of the next bus (on the next bus route) that will stop at that bus stop. As before, the candidate buses and time predictions are not based on the published bus schedule; they are based on the current locations of buses and on historical data on how long it takes buses to travel from one stop to the next at different times of the day.

G(RT)^2 maintains 4 weeks of historical data of how long it took each bus to travel from each bus stop to the next bus stop on each bus route. Periodically, the bus company updates the bus schedule for all bus routes.

Question 48. (8 marks) Use Case Diagram

Create a use-case diagram for the proposed G(RT)^2 system. Model only what has been described above (i.e., do not try to identify all actors or all use cases). Provide a "brief" use-case description (1-2 sentences) of each use case.

Question 49. (8 marks) Scenarios

You are to create a "fully-dressed" (i.e., tabular) use-case description for the *main scenario*, two *alternative* scenarios, and one *exceptional* scenario associated with real-time route planning (from user-specified point A to user-specified point B) in G(RT)2. Alternative and exceptional behaviours can include modifying the origin or destination bus stops during route planning, cancelling a selected trip, recalculating a selected trip if the user deviates from the selected trip (e.g., getting on a different bus than the next bus of the selected trip, getting off of a bus at the wrong bus stop), and so on.

Question 50. (20 marks) Domain Model

You are to create a Domain Model for the *specification* of G(RT)2. Your domain model must include any phenomena to which your constraints in question 51 and your specifications in question 52 refer. Use appropriate UML constructs (e.g., multiplicities, association classes, composition and aggregation, generalization) to make your domain model as complete and precise as possible.

Question 52. (9 marks) Specifications and Assumptions

Consider the following requirements of G(RT)2:

- R1: When the user identifies a bus route and a bus stop on that route, the software highlights the current location of the next bus (on the chosen route) to stop at the chosen bus stop.
- R2: When the user identifies a bus route and a bus stop on that route, the software displays the predicted arrival time of the next bus (on the chosen route) that will stop at the chosen bus stop.
- R3: After a multi-hop bus trip has been selected, when the user arrives at a bus stop where the trip indicates a transfer to a different bus route, the software highlights the current location of the next bus (on the next bus route) that will stop at that bus stop.

52a) Transform each of the above requirements into a *specification* that uses vocabulary from your Domain Model from Question 50.

52b) List all *assumptions* that you need to make to argue that your specifications for the G(RT)2 system meets the problem requirements.

Question 53. (25 Marks) State Machine Model

You are to create a UML State-Machine model of the proposed system to enforce the parking policies at the Bauer complex. Bauer provides parking for two types of users: customers and employees. There are two levels of parking: surface-level parking and underground parking. Surface-level parking is for customers only. They pay a $2 flat fee for parking. Employees park for free, but only if they park underground. Underground parking is also open to customers in the evenings and on weekends, but of course customers must still pay ($2) to use it.

Gates control access to both levels of parking. Each gate controls traffic flowing in a single direction (e.g., traffic entering the complex), and it opens only for cars that approach the gate from the appropriate direction. Employees' cars have transponders that the gates can sense. Some gates open only for cars that have transponders. In addition, the system uses transponder data (IDs) to keep track of where employees' cars park. There are four gates that are controlled independently:

- Gate **A** controls entry into the Bauer complex. It opens whenever it senses a car approaching from outside the Bauer complex. Cars that pass through gate A enter the surface-level parking lot.
- Gate **B** controls access from surface-level parking to underground parking. Normally, it opens only if an approaching car has a transponder. But during evenings (18:00-6:00) and weekends (Friday 18:00 - Monday 6:00), it remains open to all cars.
- Gate **C** opens whenever it senses a car approaching from the underground parking lot.
- Gate **D** controls exits from the Bauer complex and collects parking fees. The gate opens whenever a $2 fee is paid (no change is provided). In addition, if an approaching car has a transponder, the gate opens if the car had parked underground (i.e., if the transponder triggered gate B since the car entered the complex).

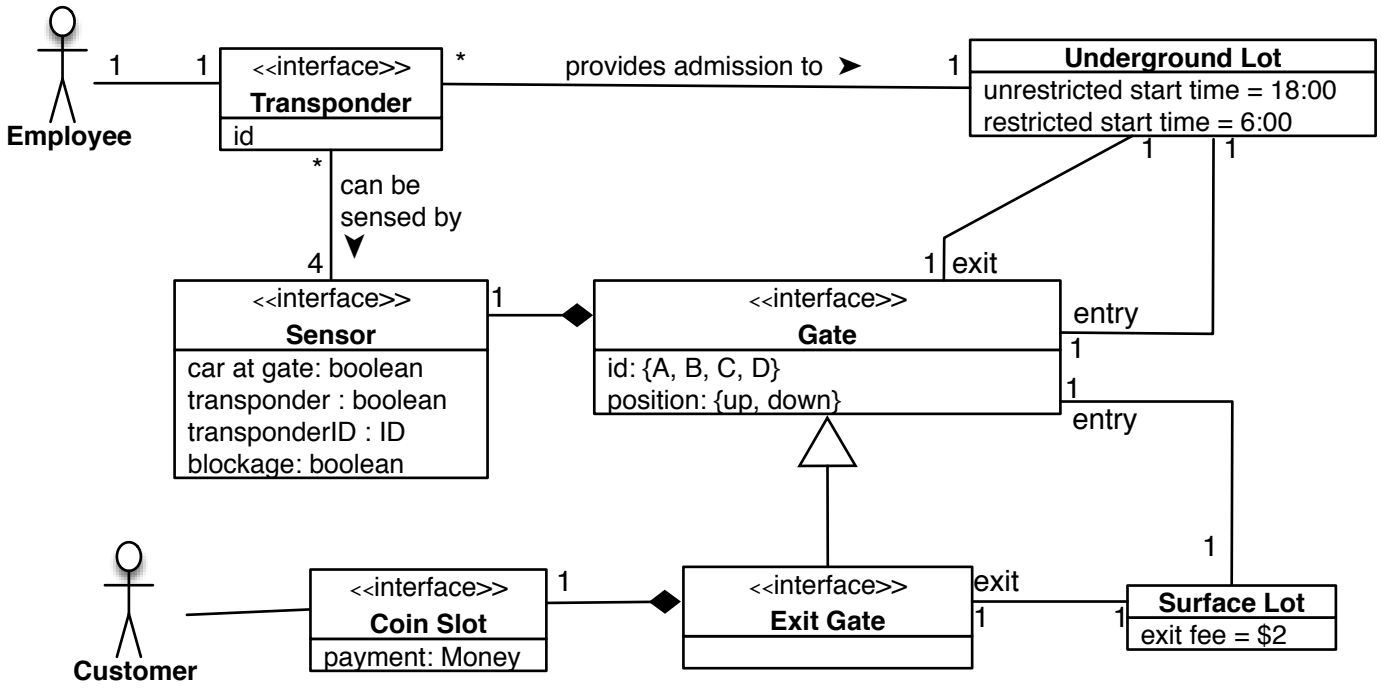
Each of the gates closes 5 seconds after it opens — unless a blockage is detected, in which case it remains open and tries to close again after another 5 seconds. (Don't worry about modelling the delay of the gate opening and closing.)

On the next page is a Domain Model for the system based on the interface choices already made. The events, conditions, and actions in your state-machine model should be expressed in terms of elements in the domain model. You can declare events and abbreviations based on these elements, to simplify the expressions in your model.

For full credit, your model should make the most effective use of UML state-machine modelling constructs. You must use a software modelling tool or drawing tool to create the state machine model that you include in your submission.

Question 53. (25 Marks) State Machine Model (cont.)

Below is the Domain Model for the *specification* of the system that enforces the Bauer parking policies.



COVID Tracker (used in Questions 54-57)

COVID Alert is a Government of Canada app that helps to combat COVID-19 by using technology to speed up community-driven contact tracing. Specifically, COVID Alert aims to identify when a person has been exposed to someone who is infected with COVID and to notify the person of the exposure. (Notifying a person that they have been exposed to the disease enables them to self-isolate to protect others, and to get tested for early diagnosis and timely treatment.)

The system consists of an app that runs on a smartphone that users carry with them (e.g., in their pocket, or in their bag) and an internet-connected backend server that stores only non-identifying IDs of users who have reported that they have tested positive.

Basically, a user downloads the app onto their smartphone and runs the app continuously in the background. The app uses Bluetooth to find other nearby users of the app. When the user gets close to another person who is also carrying a smartphone running the COVID Alert app (such that the two apps detect each other's Bluetooth signals), the two apps then exchange messages and information so that both apps document the encounter. The app stores locally on the user's phone an encrypted history of its user's encounters from the last 14 days.

If the user is infected with COVID (i.e., tests positive), then they obtain from their provincial health care provider a one-time key that the user enters into the app. The app validates the key and then uploads to the backend server the IDs that the app used in its message exchanges over the last 14 days.

Once a day, the app downloads from the server all of the IDs representing COVID-positive users. The app then searches its log of recorded encounters and looks for encounters associated with COVID-positive users; the app sends a push notification it finds in its logs a past **close encounter** with a reported COVID-positive person.

Initialization
A user downloads the COVID Alert app onto their smartphone, opens the app, and keeps the app open and running in the background. It is important that the phone's Bluetooth settings remain "on".

The app generates 15 unique IDs to use in its communications with other devices running the app; these IDs are unique with respect to all IDs generated by all users of the app. The app will use a different ID each day and rotate through its 15 IDs every 15 days.

Encounter
The app uses the phone's Bluetooth chip to repeatedly emit (every 250 ms) a Bluetooth signal and it periodically scans (every 5 minutes) for nearby Bluetooth signals. When the app detects a signal from another person's phone running the app, the two apps exchange messages that enable both apps to create an Encounter Record that records

(1) the app ID (of the day) of the *other* device
(2) a measure of the Bluetooth signal strength (used to estimate the distance between the devices)
(3) a timestamp

COVID Tracker (used in Questions 54-57)(cont.)

Each app stores locally an encrypted history of the user's Encounter Records that have been recorded over the last 14 days. Records that are older than 14 days are destroyed.

Testing Positive
If the user tests positive for COVID, they obtain from a provincial health official a one-time key, which the user enters into the app and which the app validates. (The one-time key helps to ensure that only COVID-positive users upload their data to the server.)

If the one-time key is validated, then the app uploads to the server all of the daily IDs that the app used in its message exchanges with other instances of the COVID Alert app over the last 14 days, and the date it used each ID.

Exposure Notifications
Once a day, the app downloads from the server all of the IDs that have been uploaded to the server by COVID-positive users.

The app then automatically searches its log of Encounter Records and looks for matches between the IDs in its Records and downloaded IDs of COVID-positive users. Looking at all of the Encounter Records that match the same COVID-positive ID on the same day, the app determines that a **close encounter** occurred, if

- the **length** of an encounter was at least 15 minutes (indicated by the number of Encounter Records recorded at consecutive timestamps matching the same ID), and
- the **proximity** of an encounter was within 2 metres (estimated using the recorded Bluetooth signal strength)

If the app determines that the user had a close encounter with a reported COVID-positive user, the app issues a push notification to warn the user that "someone you've been near has reported a COVID-19 diagnosis" and provides a link for advice on whether to get tested. The notification provides no information about who the "someone" is, and no information about the date, time, or location of the encounter.

Question 54. (15 marks) Use Case Diagram

Create a use-case diagram for the COVID contract-tracing system. The use-case diagram should represent all of the functionality described above. Provide a "brief" use-case description (1-2 sentences) of each use case.

Question 55. (25 marks) Specification Domain Model

Draw a specification-level domain model for the *specification* of the COVID contact-tracing system described above. The domain model should incorporate all environmental, actor, and interface phenomena that are mentioned in the provided description; and should cover all of the actors that appear in your use-case diagram in Question 53.

For full credit, use advanced UML class-diagram constructs where appropriate (e.g., attributes, aggregations, compositions, generalizations, association classes, multiplicities). Complete the model with stereotypes that indicate what type of phenomenon each entity is.

Question 56. (25 marks) RE Reference Model

This question tests your ability to derive specifications from requirements.

(a) Identify in the above description the main requirement that the COVID Tracker aims to address. Provide a complete statement of the requirement below. Be sure to make reference to all required actors and the required phenomena (e.g., events, actions, services, properties of actors, constraints). Label your requirement R:

(b) Express one or more specifications of the COVID Tracker solution that, taken together, aim to satisfy the above requirement. Label your specifications S:

(c) List as many distinct assumptions about the environment as you can. More credit will be given to assumptions that bridge the gap between the specification(s) and the requirement, in making the argument that "S,D⊢R" (versus assumptions about working hardware). Label your assumptions A:

Question 57. (5 marks) Solution-Fit Hypothesis

Identify and describe below one (1) solution-fit hypothesis of the COVID contact-tracing system that reflects a key solution-fit risk of the system.