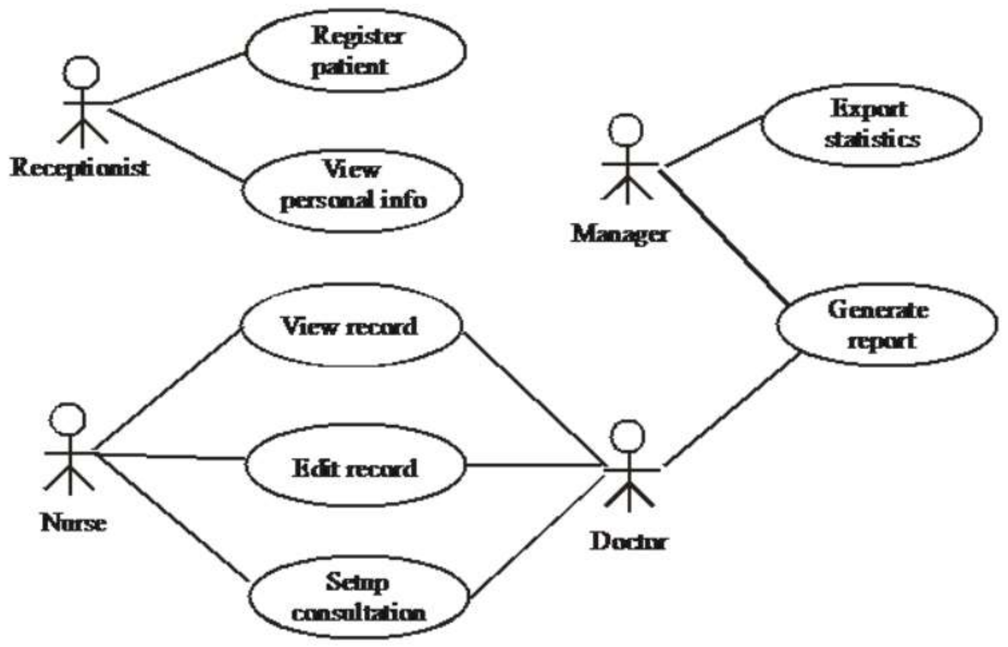
Chapter 7

7.1 Using the structured notation shown in Figure 7.3, specify the weather station use cases for Report status and Reconfigure. You should make reasonable assumptions about the functionality that is required here.

System:	Weather station
Use case:	Report status
Actors:	Weather information system, weather station
Data:	The weather station sends a status update to the weather information system giving information about the status of its instruments, computers and power supply.
Stimulus:	The weather information system establishes a satellite link with the weather station and requests status information.
Response:	A status summary is uploaded to the weather information system
Comments:	System status is usually requested at the same time as the weather report.

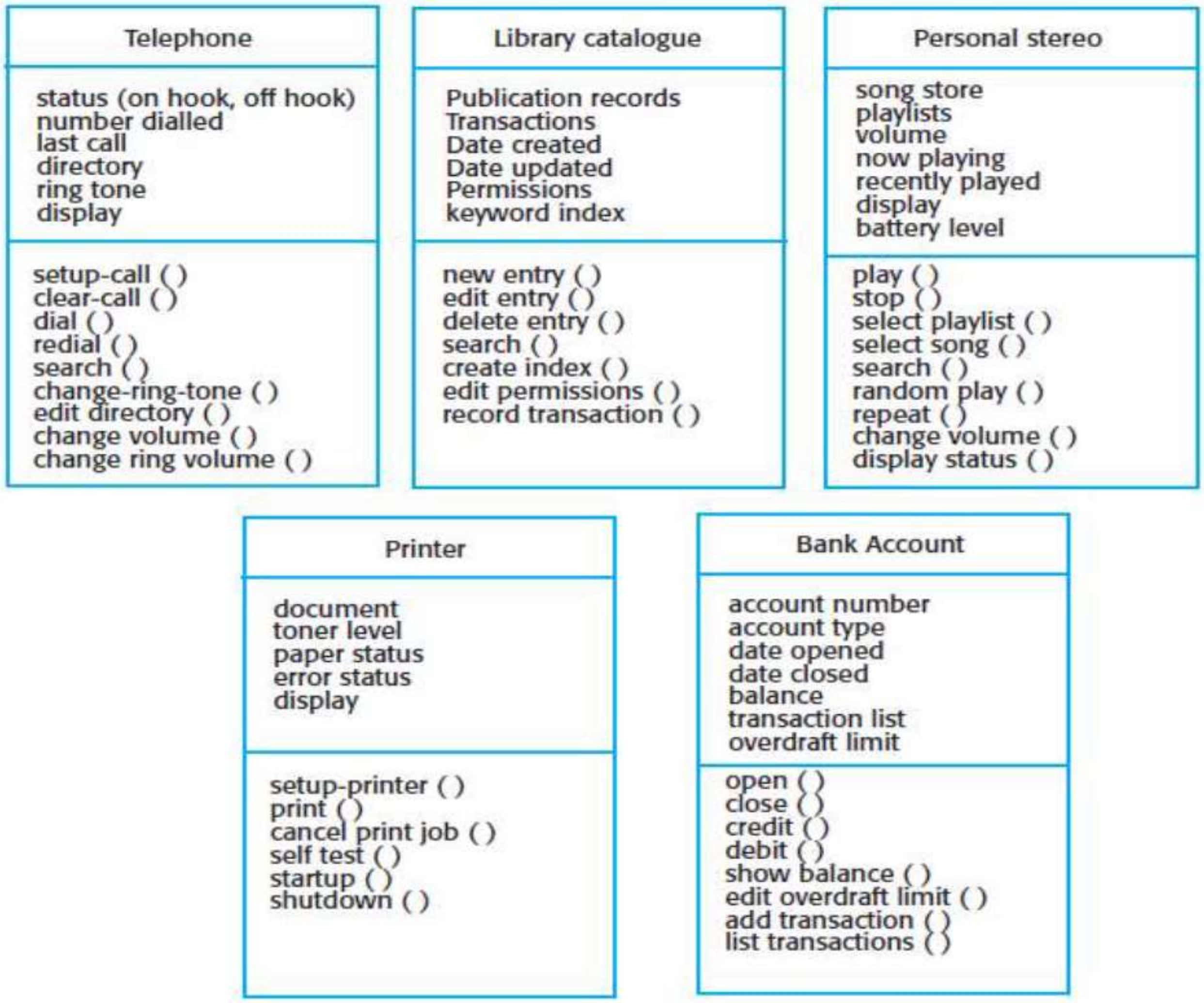
System:	Weather station
Use case:	Reconfigure
Actors:	Weather information system, weather station
Data:	The weather information station sends a reconfiguration command to the weather station. This places it into remote control mode where further commands may be sent from the remote system to update the weather station software.
Stimulus:	A command from the weather information system.
Response:	Confirmation that the system is in remote control mode
Comments:	Used occasionally when software updates have to be installed.

7.2 Assume that the mentcare system is being developed using an object-oriented approach. Draw a use case diagram showing at least six possible use cases for this system

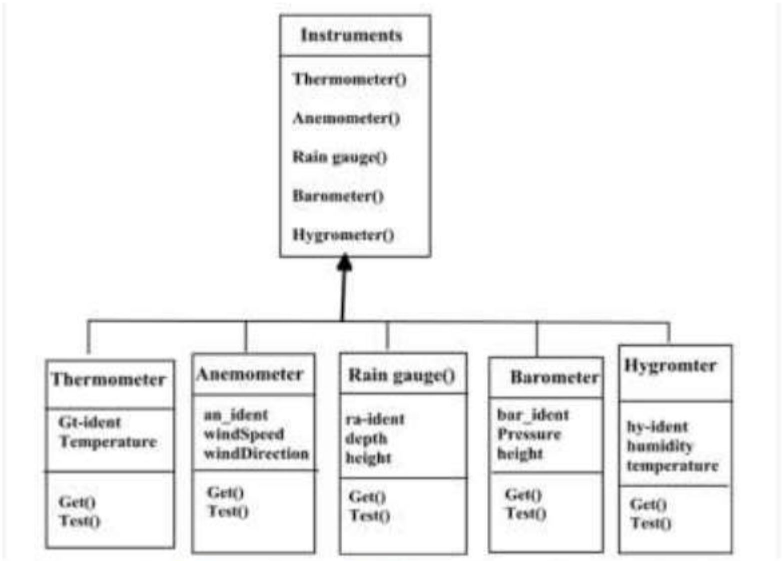

Register patient
View personal info
Receptionist
Export statistics
Manager
Generate report
View record
Edit record
Setup consultation
Nurse
Doctor

7.3 Using the UML graphical notation for object classes, design the following object classes, identifying attributes and operations. Use your own experience to decide on the attributes and operations that should be associated with these objects.

1. a telephone
2. a printer for a personal computer
3. a personal stereo system
4. a bank account
5. a library catalogue



7.4 A shape can be classified into 2-D and 3-D. Design an inheritance hierarchy that will include different kinds of 2-D and 3-D shapes. Make sure you identify at least five other classes of shapes.



1. To calculate the weather report, different types of instruments are provided.

2. In the above figure, **Instruments** object is taken as superclass.

3. The domain objects are inherited from the superclass. The domain objects are Thermometer, Anemometer, Rain gauge, Barometer, and Hygrometer.

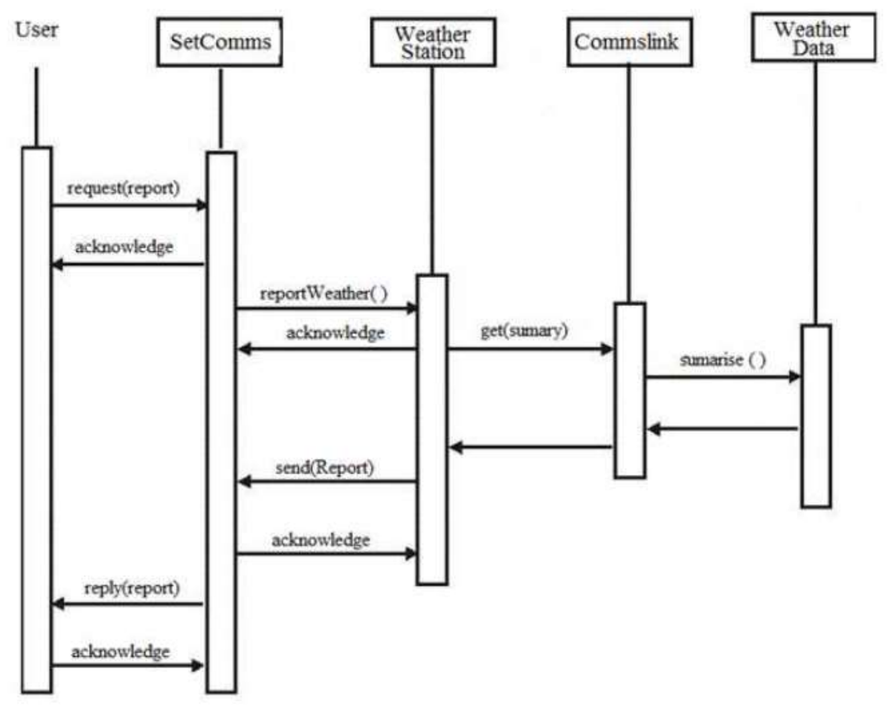
4. In **Thermometer** object have identifiers like unique identifier **gt-ident** and amount of **temperature** and the function are get() and test() are used to calculate and test the report.

5. In **Anemometer** object have identifiers like unique identifier **an-ident** and amount of **windSpeed** and **windDirection** and the function are get() and test() are used to calculate and test the report.

6. In **Rain gauage** object have identifiers like unique identifier **ra-ident** and amount of **depth** and **height** of the wind and the function are get() and test() are used to calculate and test the report.

7. In **Barometer** object have identifiers like unique identifier **bar-ident** and amount of **pressure** and **height** of the wind and the function are get() and test() are usedto calculate and test the report

7.5 Develop the design of the weather station to show the interaction between the data collection subsystem and the instruments that collect weather data. Use sequence diagrams to show this interaction

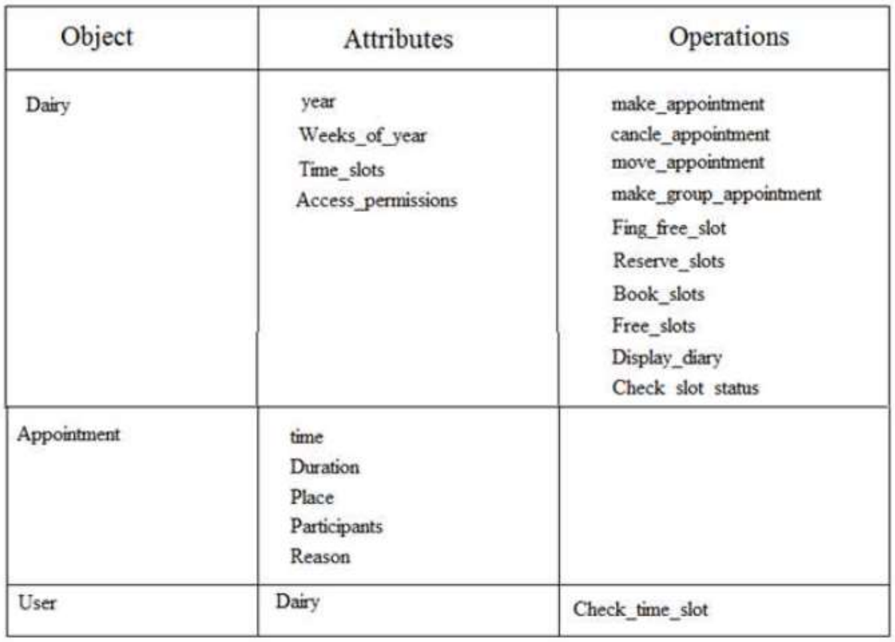


7.6 Identify possible object in the following systems and develop an object-oriented design for them. You may make any reasonable assumptions about the systems when deriving the design.

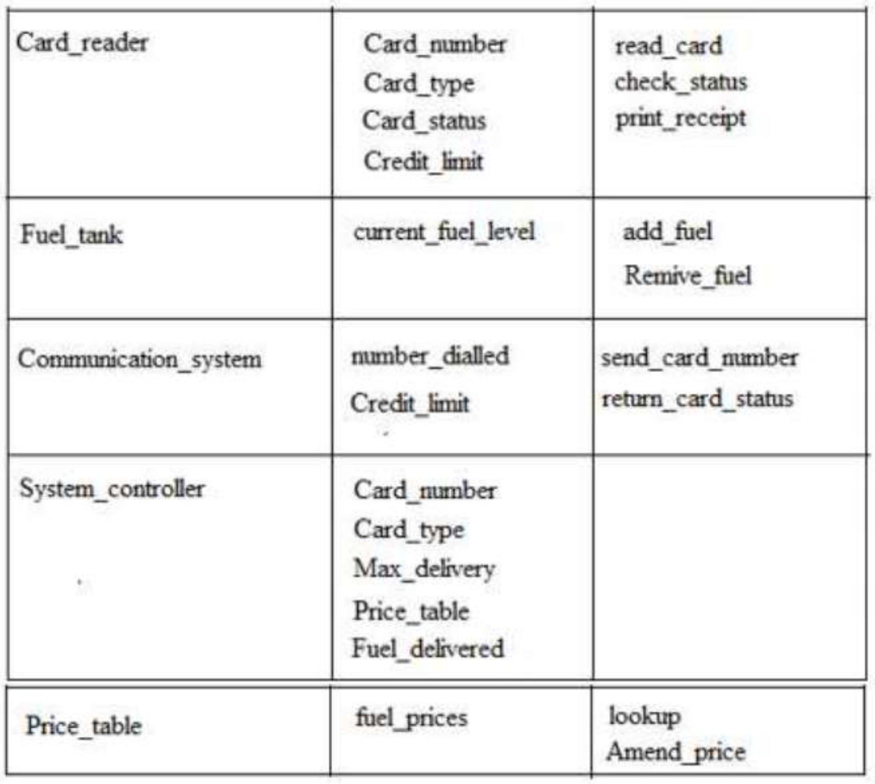
1. A group diary and time management system is intended to support the timetabling of meetings and appointments across a group of co-workers. When an appointment is to be made that involves a number of people, the system finds a common slot in each of their diaries and arranges the appointment for that time. If no common slots are available, it interacts with the user to rearrange his personal diary to make room for the appointment
2. A filling station is to be set up for fully automated operation. Drivers swipe their credit card through a reader connected to the pump; the card is verified by communication with a credit company computer, and a fuel limit is established. The driver may then take the fuel required. When fuel delivery is complete and the pump hose is returned to its holster, the driver's credit card account is debited with the cost of the fuel taken. The credit card is returned after debiting. If the card is invalid, the pump returns it before fuel is dispensed.

a.

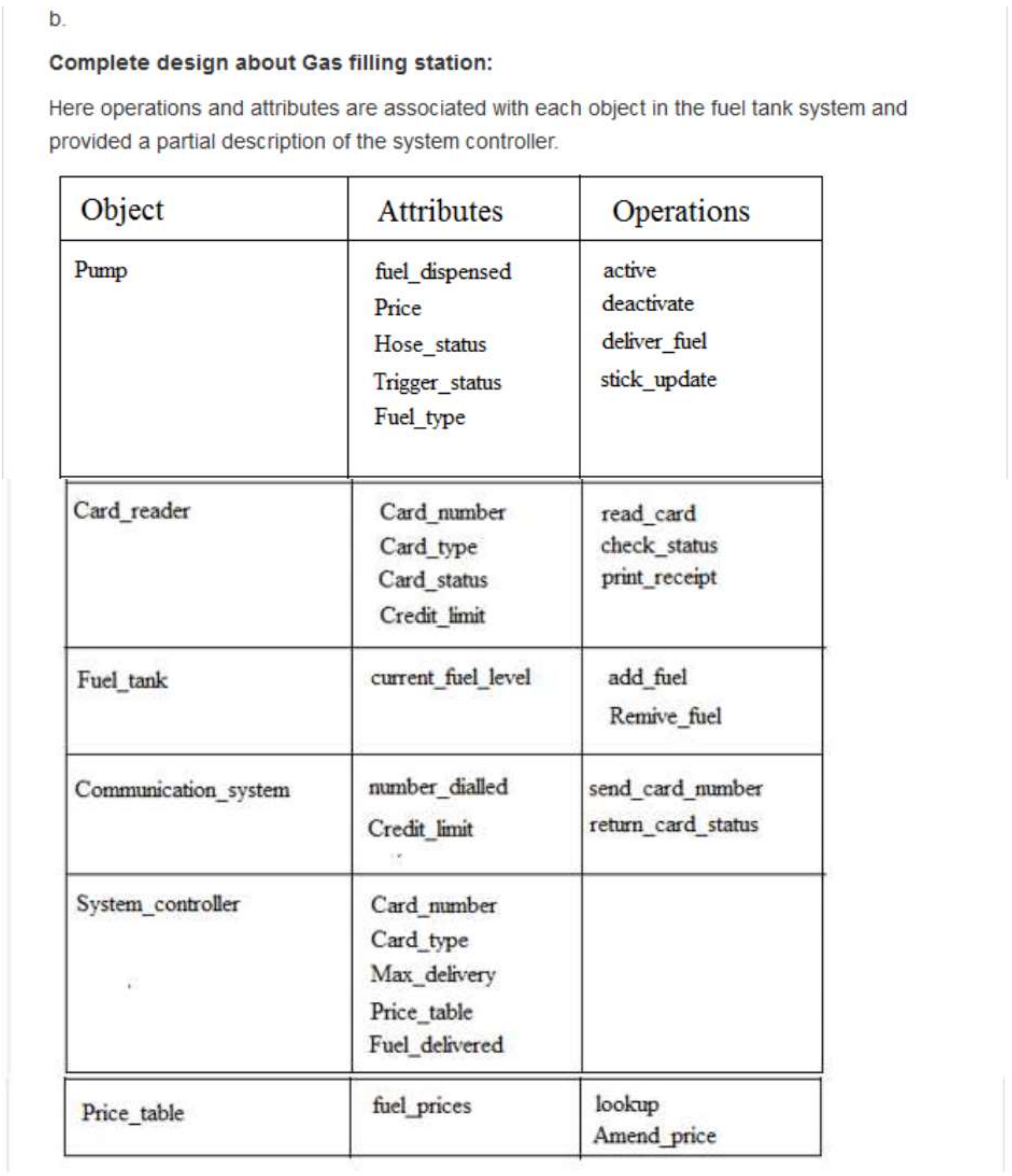
Possible principal objects of diary and time management system with their operations and attributes. Here there is a single diary object with different operations for group appointments and personal appointments.



Object	Attributes	Operations
Dairy	year Weeks_of_year Time_slots Access_permissions	make_appointment cancle_appointment move_appointment make_group_appointment Fing_free_slot Reserve_slots Book_slots Free_slots Display_diary Check slot status
Appointment	time Duration Place Participants Reason	
User	Dairy	Check_time_slot



Card_reader	Card_number Card_type Card_status Credit_limit	read_card check_status print_receipt
Fuel_tank	current_fuel_level	add_fuel Remive_fuel
Communication_system	number_dialled Credit_limit	send_card_number return_card_status
System_controller	Card_number Card_type Max_delivery Price_table Fuel_delivered	
Price_table	fuel_prices	lookup Amend_price



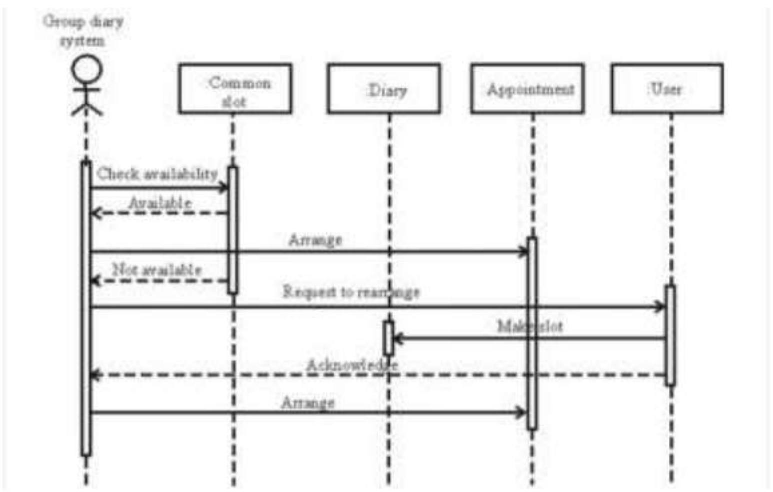
b.

Complete design about Gas filling station:

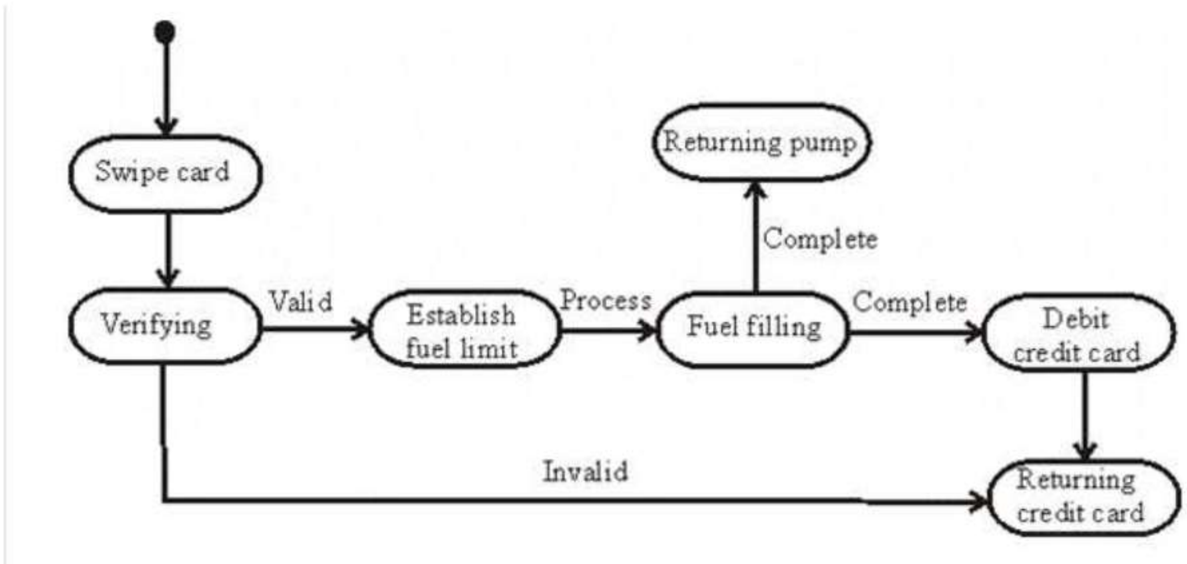
Here operations and attributes are associated with each object in the fuel tank system and provided a partial description of the system controller.

Object	Attributes	Operations
Pump	fuel_dispensed Price Hose_status Trigger_status Fuel_type	active deactivate deliver_fuel stick_update
Card_reader	Card_number Card_type Card_status Credit_limit	read_card check_status print_receipt
Fuel_tank	current_fuel_level	add_fuel Remive_fuel
Communication_system	number_dialled Credit_limit	send_card_number return_card_status
System_controller	Card_number Card_type Max_delivery Price_table Fuel_delivered	
Price_table	fuel_prices	lookup Amend_price

<u>7.7 Draw a sequence diagram showing the interactions of objects in a group dairy system when a group of people are arranging a meeting</u>



7.8 Draw a UML state diagram showing the possible state changes in either the group diary or the filling station system



The filling station system performs the actions in various states as shown:

1. When the swipe card state is completed, the card details are verified in the verifications state.

2. If the credentials are valid then the fuel limit is established and the fuel filling state is processed.

3. After the fuel filling state, the returning pump state occurs and the card is debited.

4. When the card is debited or if the credentials are not matched, the card returning state occurs.

7.9 When code is integrated into a larger system, problems may surface. Explain how configuration management can be useful when handling such problems

Configuration management aims:

- Changes made by different developers do not interfere with each other
- Always possible to create a specific version of a system

Without it, it's easy to lose track of changes each dev. Makes to code and for changes made by one programmer.
Essentially CM tracks and keep changes made to ensure previous changes are kept and not overwritten.

7.10 A small company has developed a specialized product that it configures specially for each customer. New customers usually have specific requirements to be incorporated into their system, and they pay for these to be developed. The company has an opportunity to bid for a new contract, which would more than double its customer base. The new customer also wishes to have some involvement in the configuration of the system. Explain why, in these circumstances, it might be a good idea for the company owning the software to make it open source.

The key benefits of open source are is that it opens up development to a wide range of developers and so accelerates the development and debugging of the product. This reduces cost of expansion when customer base increases. The company can then make the money back from providing "free" software can be made back by charging for the specialisation of the software for it's customers. Thereafter any developer can make changes to said system, reducing the load on the original developers