MOHSIN IBNA HOSSAIN

AIUB, OOAD NOTES, FINAL TERM

FINAL TERM

Table of Contents

SEQUENCE DIAGRAM

Ch: 04

Sequence Diagram

- The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It captures the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent the time what messages are sent and when.

Sequence Diagram Notation

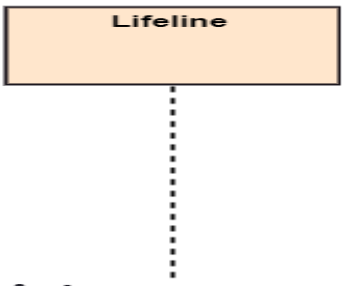
Frame:

- A rectangle with a pentagon in the upper left-hand corner called the name compartment.

- **Note**: **sd** - interaction Identifier is either a simple name or an operation specification as in a class diagram

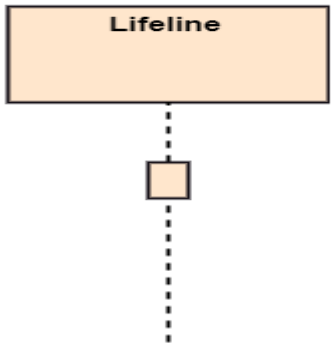
Lifeline:

- A lifeline represents an individual participant in the Interaction.



➢ Activation:

- ⇨ It is represented by a thin rectangle on the lifeline. It describes the time period in which an operation is performed by an element, such that the top and the bottom of the rectangle are associated with the initiation and the completion time, each respectively.



Types of Messages in Sequence Diagrams

➢ Synchronous Message:

- ⇨ A synchronous message requires a response before the interaction can continue. It's usually drawn using a line with a solid arrowhead pointing from one object to another.

➢ Asynchronous Message:

- ⇨ Asynchronous messages don't need a reply for interaction to continue. Like synchronous messages, they are drawn with an arrow connecting two lifelines; however, the arrowhead is usually open and there's no return message depicted.

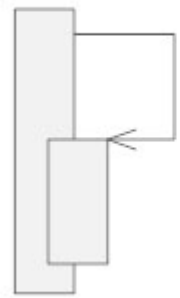
➢ Reply or Return Message:

⇨ A reply message is drawn with a dotted line and an open arrowhead pointing back to the original lifeline.

<---------- Reply or Return Message

➢ Self-Message:

⇨ A message an object sends to itself, usually shown as a U-shaped arrow pointing back to itself.



Self message

Execution Occurrence

- ❑ An operation is **executed** when some process is running its code.
- ❑ An operation is **suspended** when it sends a synchronous message and is waiting for it to return.
- ❑ An operation is **active** when it is executed or suspended. The period when an object is active can be shown using an ***execution occurrence*** (Thin white or grey rectangle over lifeline dashed line)

Structured Control Operators

– **'OPT'**: **Optional**
– **'ALT'**: **Alternate**
– **'PAR'**: **Parallel**
– **'Loop'**: **Iterative**
– **'Ref'**: **Reference**

Sequence Diagram Elements

The reference numbers on the figure denotes:

1. Object lifeline
2. Message/Stimulus
3. Iteration
4. Self-reference
5. Return
6. Anonymous object
7. Object name
8. Sequence number
9. Condition
10. Basic comment

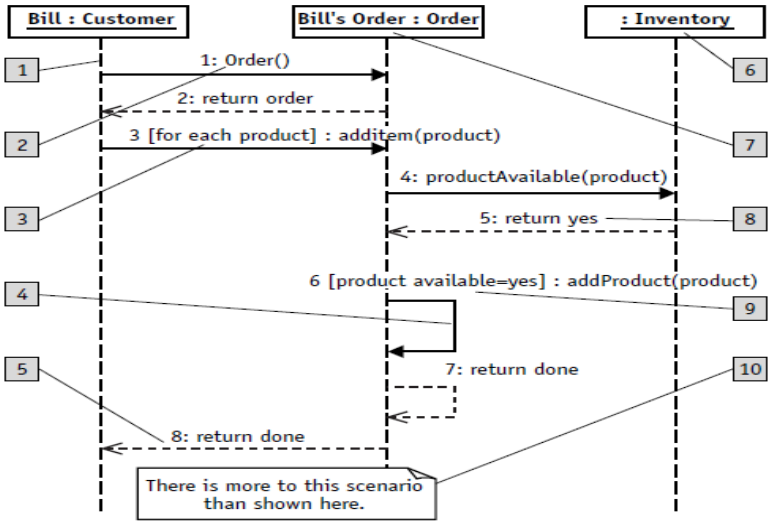
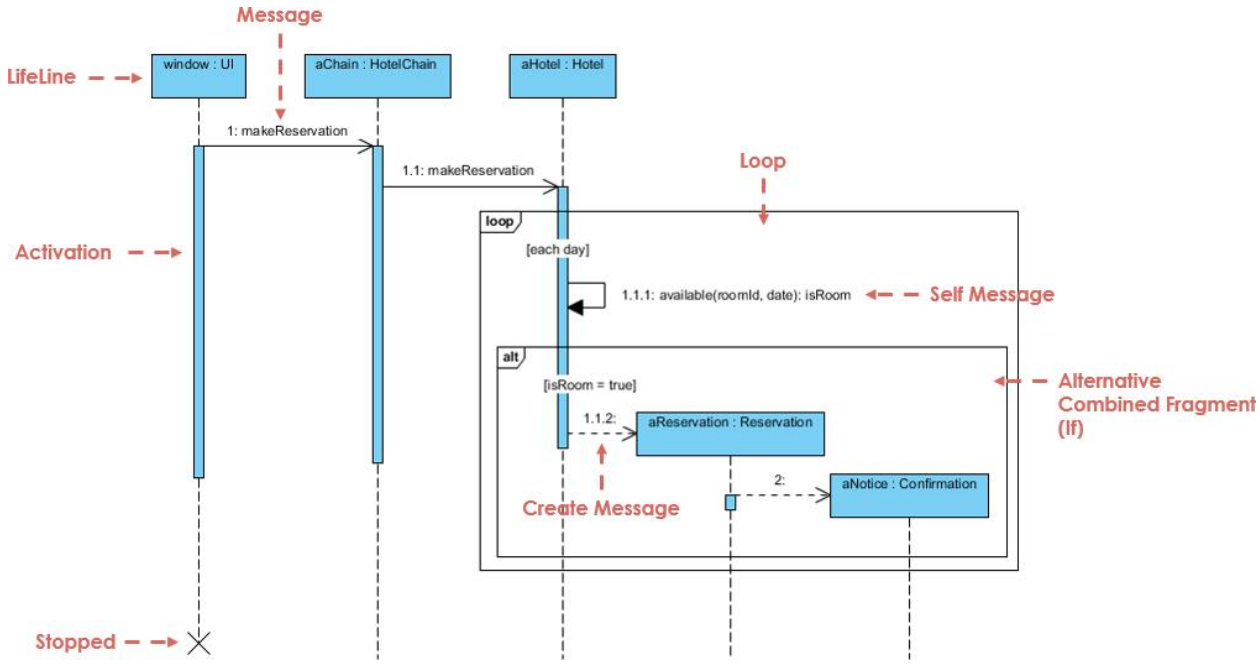


Figure 16-2 Elements of the Sequence diagram notation

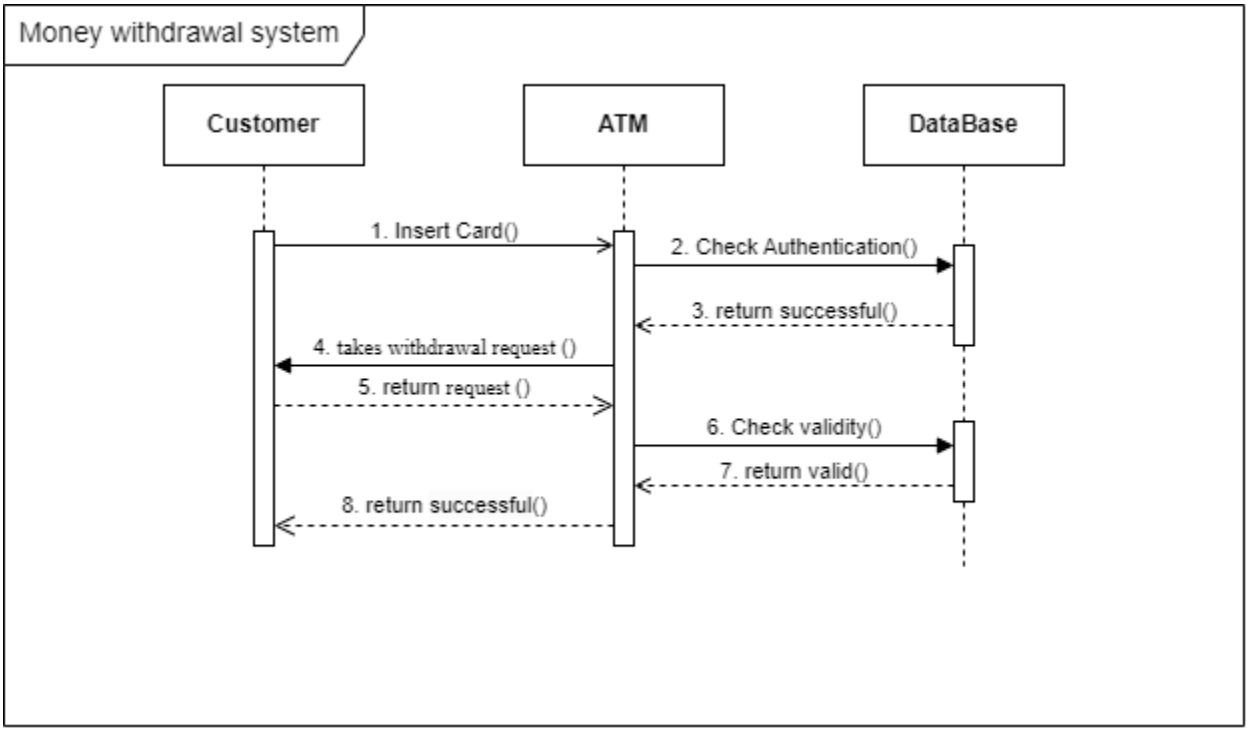
Sequence Diagram Example: Hotel System



Sequence Diagram - Case Studies

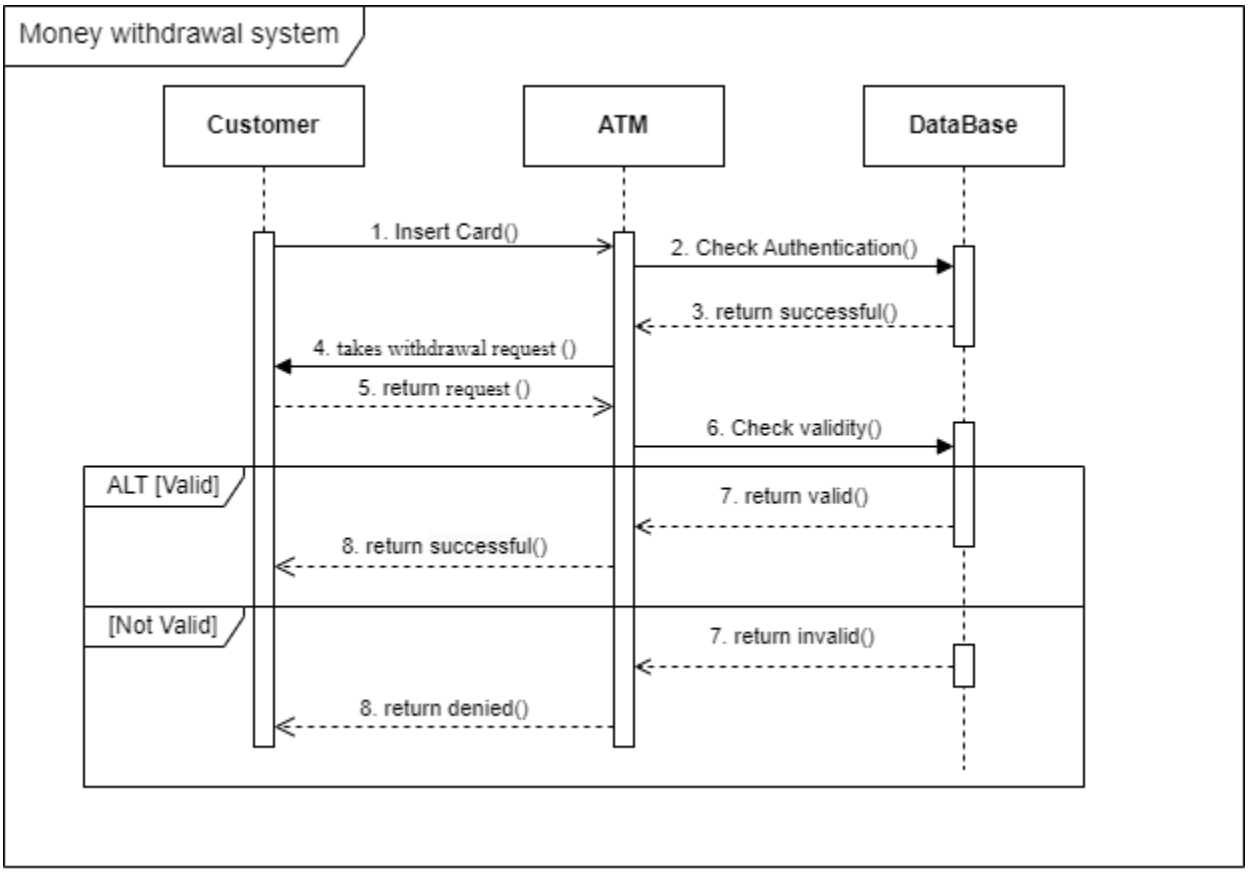
➢ Case: 01

⇨ In a withdrawal transaction of an **ATM Machine** system the **customer** inserts his ATM card in the machine. The machine then verifies the customer authentication using the information provided in the **customer account**. After successful verification the machine takes withdrawal request from the customer and checks whether the request is valid or not. A valid request is carried out by the machine.



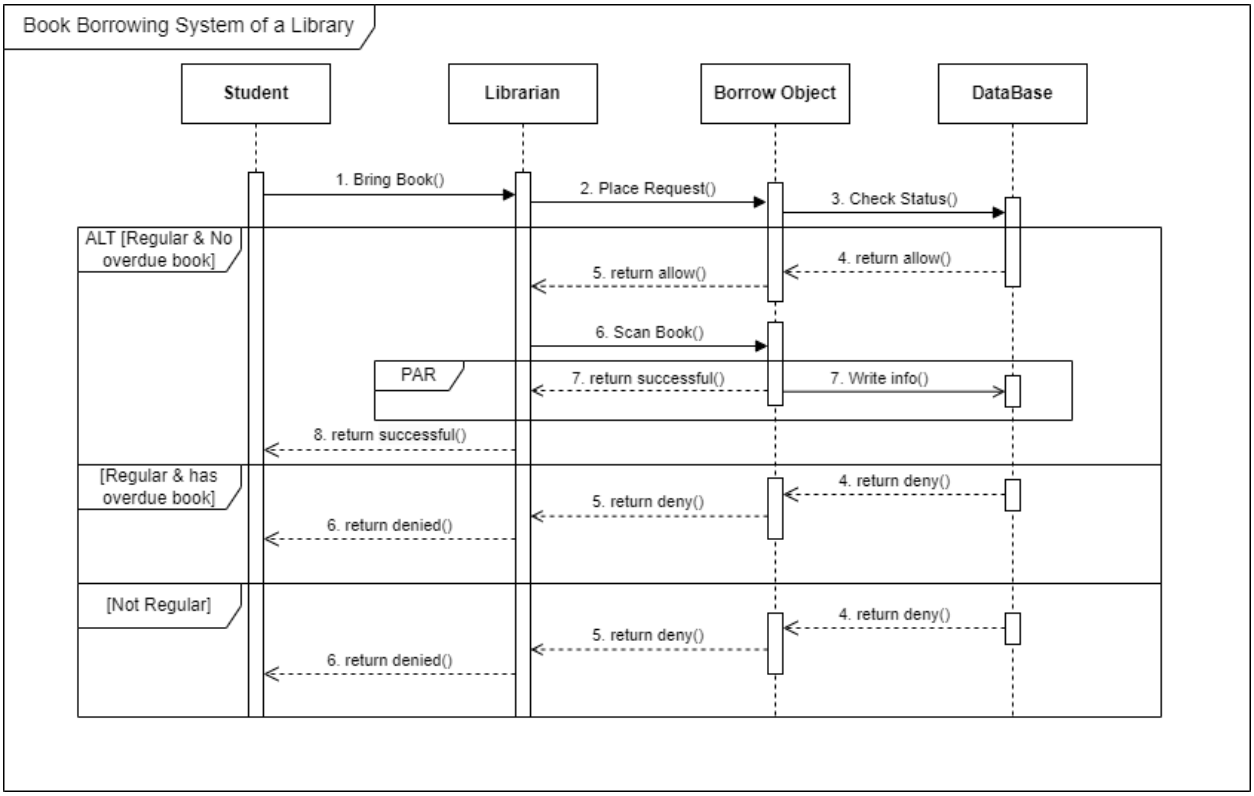
➢ Case: 02

⇨ In a withdrawal transaction of an **ATM Machine** system the **customer** inserts his ATM card in the machine. The machine then verifies the customer authentication using the information provided in **customer account**. After successful verification the machine takes withdrawal request from the customer and checks whether the request valid or not**. valid request** is carried out by the machine and an invalid request is rejected. In case of **unsuccessful verification** the customer request is denied.



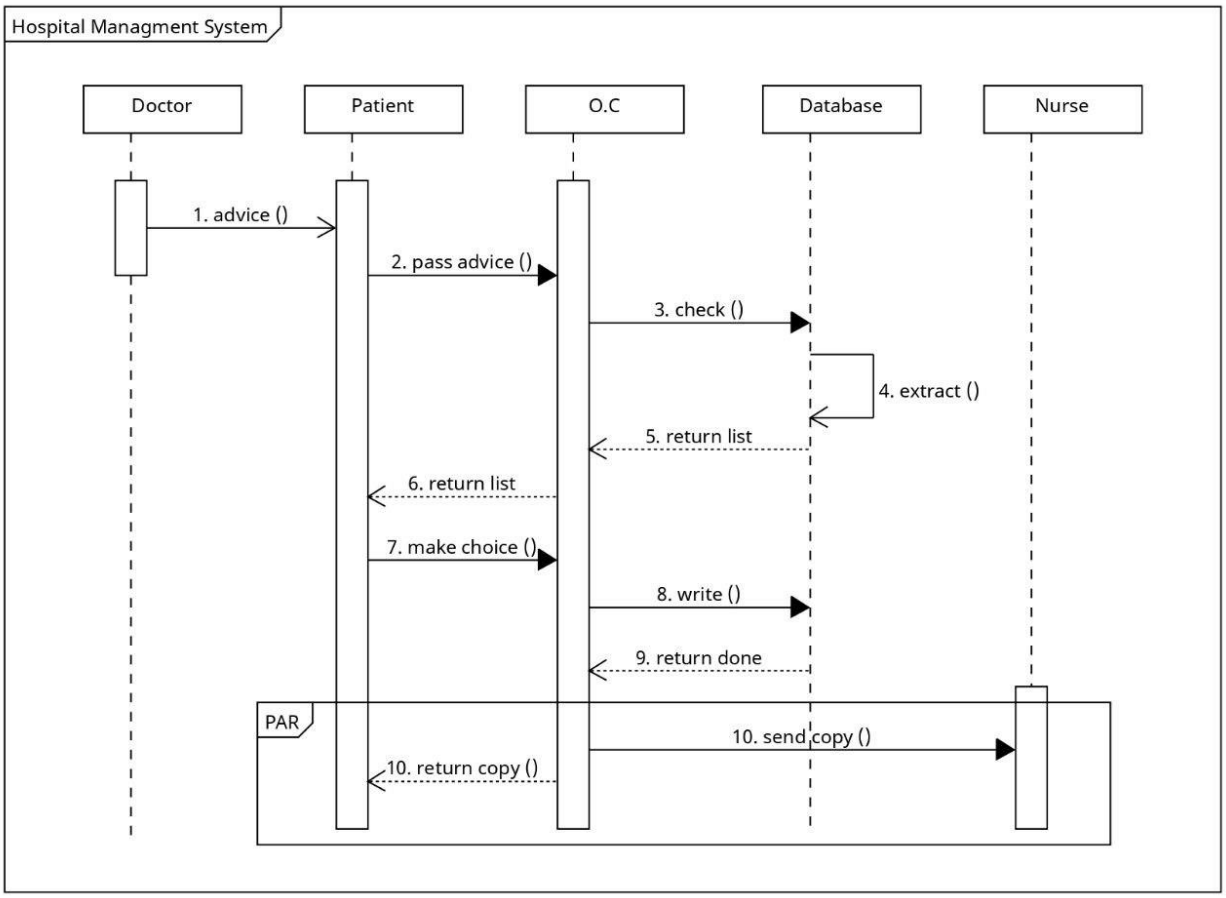
➢ Case: 03

⇨ In a library management system of a **student** can borrow books. When a student brings the books, he wants to borrow to the **librarian**, the librarian places a borrowing request by scanning the student ID. A **borrowing object** is created at that time which performs all the borrowing related tasks. It checks the student status from the **database**. If the student is a **regular student and doesn't have any overdue books,** the system allows for the books to be scanned. After the books are scanned the **data is written in the database and the librarian is informed at the same time**. If the **student is regular but already has overdue books** with him, the request is denied. If the student status is not regular, the request is also denied.



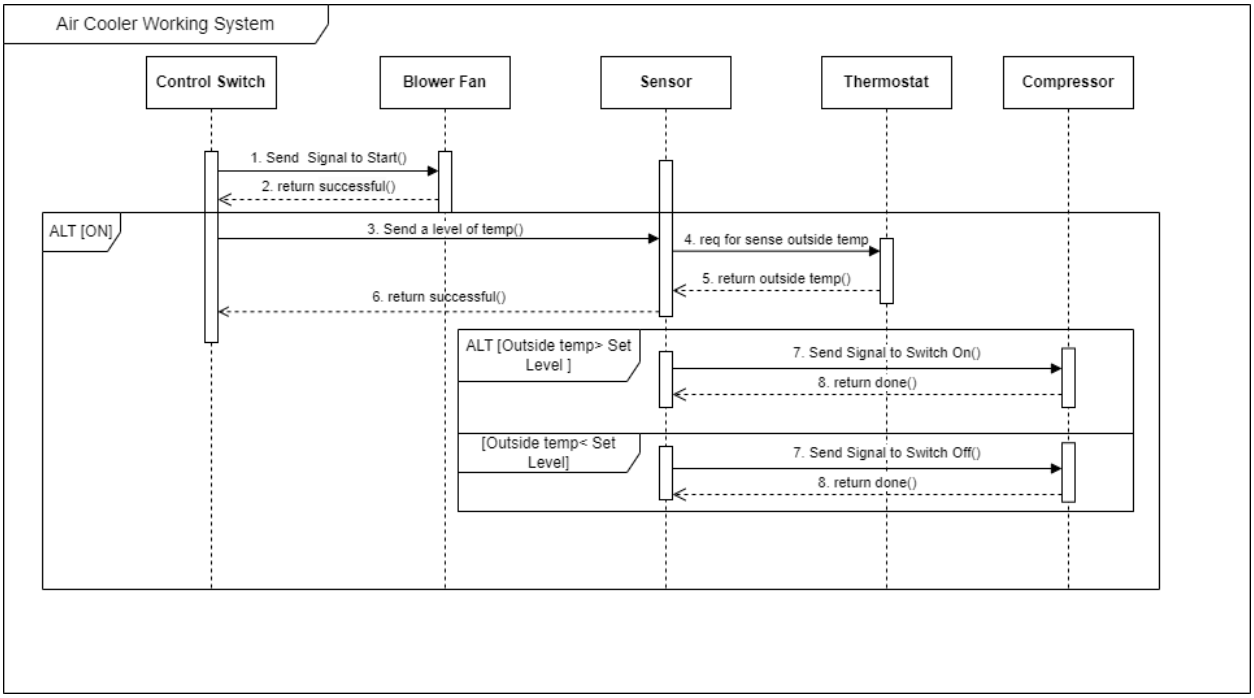
➢ Case: 05

⇨ A **doctor** includes the instruction in the patient advice when a **patient** requires a bed or room in the hospital. The advice is then passed to the **office clerk**. The office clerk checks the present booking. **database** to get the available room and bed list. The facilities of the rooms and the beds are then extracted from the room list. Available room and bed numbers and facilities are sent to the patient. The patient chooses a room or bed and then the office clerk writes patient_id, room or bed no. doctor_id and doctor advice in the booking database. Finally a copy of the admission is sent to the associated **nurse** and a copy is given to the patient.



➢ Case: 06

⇨ When an air cooler starts, the **control switch** sends signal to **the blower fan** to start at the level that is already set in the control switch. Then the control switch sends the level of temperature to the **sensor**. The sensor senses the outside temperature by sending a request to the **thermostat** and **if the outside temperature is higher then the level sent by the control switch**, the sensor send signal to the **compressor** to switch on. **When the sensor outside temperature goes down to the set level**, the thermostat sends signal to the sensor and the sensor in turn sends signal to the compressor to switch off. **The process keeps on running in a cycle as long as the air cooler stays on.**



ACTIVITY DIAGRAM

Ch: 05

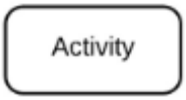
Activity Diagram

⇨ In UML, an activity diagram is used to display the sequence of activities. Activity diagrams show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity. They may be used to detail situations where parallel processing may occur in the execution of some activities. Activity diagrams are useful for business modelling where they are used for detailing the processes involved in business activities.

Activity Diagram Notation

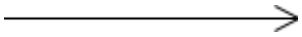
➢ Activity:

⇨ Is used to represent a set of actions.



➢ Control Flow:

⇨ Shows the sequence of execution.



➢ Initial Node:

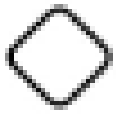
⇨ Portrays the beginning of a set of actions or activities

➢ Activity Final Node:

⇨ Stop all control flows and object flows in an activity (or action).

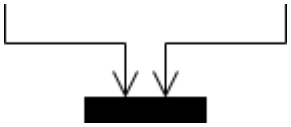
➢ Decision Node:

⇨ Represents a decision and always has at least two paths branching out with condition text to allow users to view options.



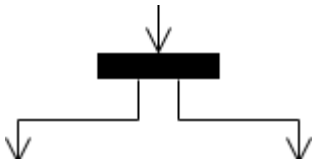
➢ Joint symbol/ Synchronization bar:

⇨ Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time. Represented with a thick vertical or horizontal line.



➢ Fork Node:

⇨ Split behavior into a set of parallel or concurrent flows of activities (or actions).



➢ Note Symbol:

⇨ Allows the diagram creators or collaborators to communicate additional messages that don't fit within the diagram itself. Leave notes for added clarity and specification.

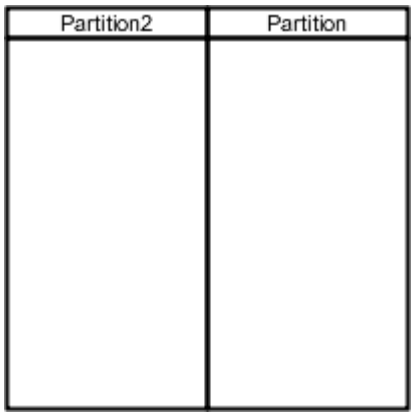
➢ Condition text:

⇨ Placed next to a decision marker to let you know under what condition an activity flow should split off in that direction.

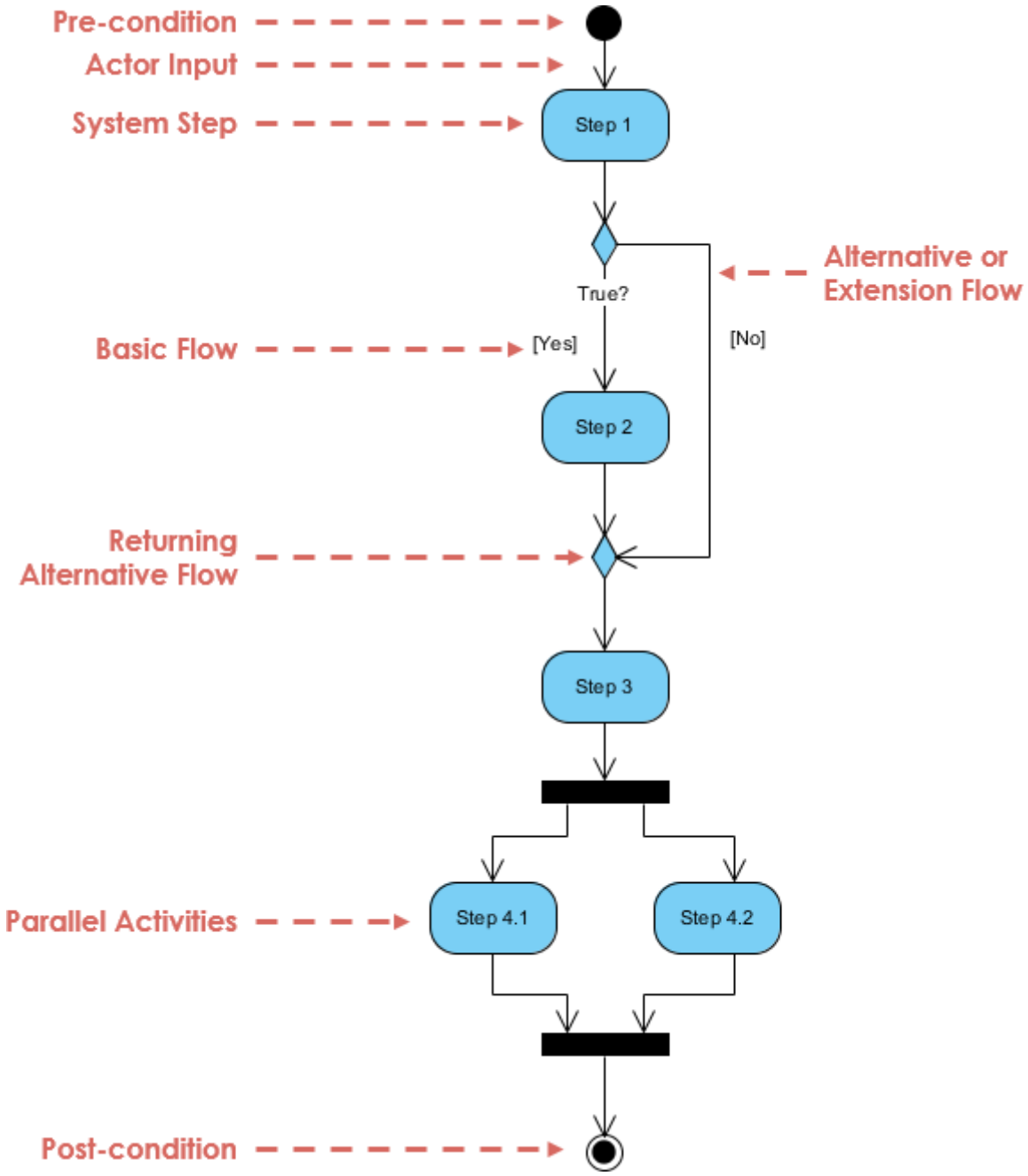
[Condition]

➢ Swimlanes/Partition:

⇨ We use Swimlanes for grouping related activities in one column. Swimlanes group related activities into one column or one row. Swimlanes can be vertical and horizontal. Swimlanes are used to add modularity to the activity diagram. It is not mandatory to use swimlanes. They usually give more clarity to the activity diagram. It's similar to creating a function in a program. It's not mandatory to do so, but, it is a recommended practice.



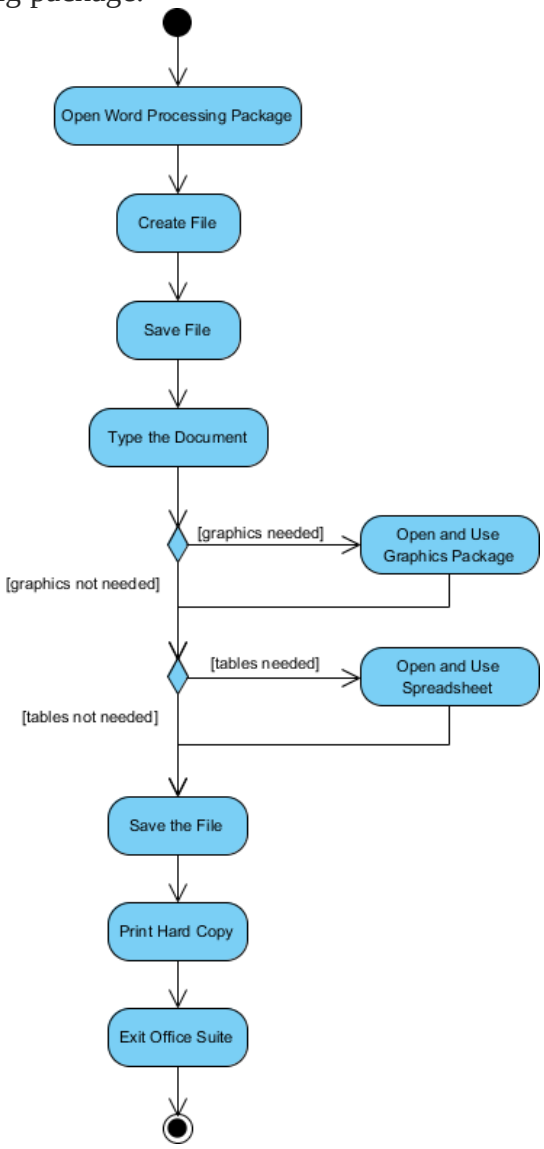
Activity Diagram - Learn by Examples



Activity Diagram - Examples

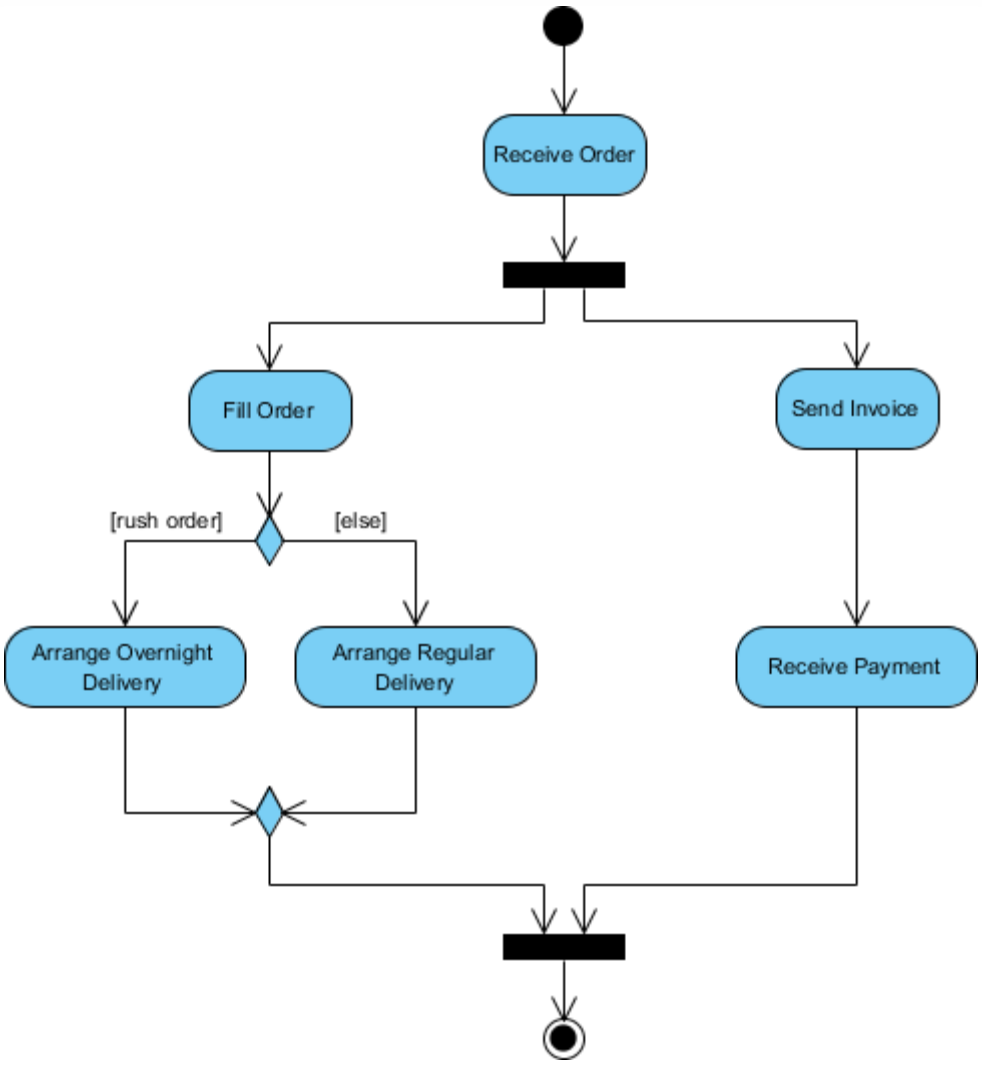
➢ Example: 01

- Open the word processing package.
- Create a file.
- Save the file under a unique name within its directory.
- Type the document.
- If graphics are necessary, open the graphics package, create the graphics, and paste the graphics into the document.
- If a spreadsheet is necessary, open the spreadsheet package, create the spreadsheet, and paste the spreadsheet into the document.
- Save the file.
- Print a hard copy of the document.
- Exit the word processing package.



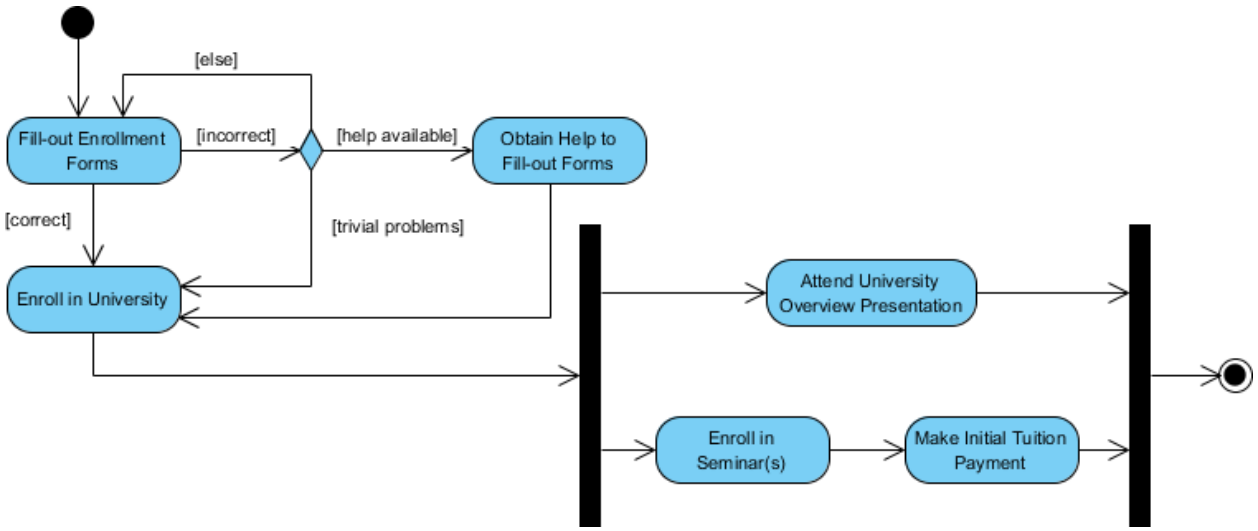
➢ Example: 02

⇨ Once the order is received, the activities split into two parallel sets of activities. One side fills and sends the order while the other handles the billing. On the Fill Order side, the method of delivery is decided conditionally. Depending on the condition either the Overnight Delivery activity or the Regular Delivery activity is performed. Finally the parallel activities combine to close the order.



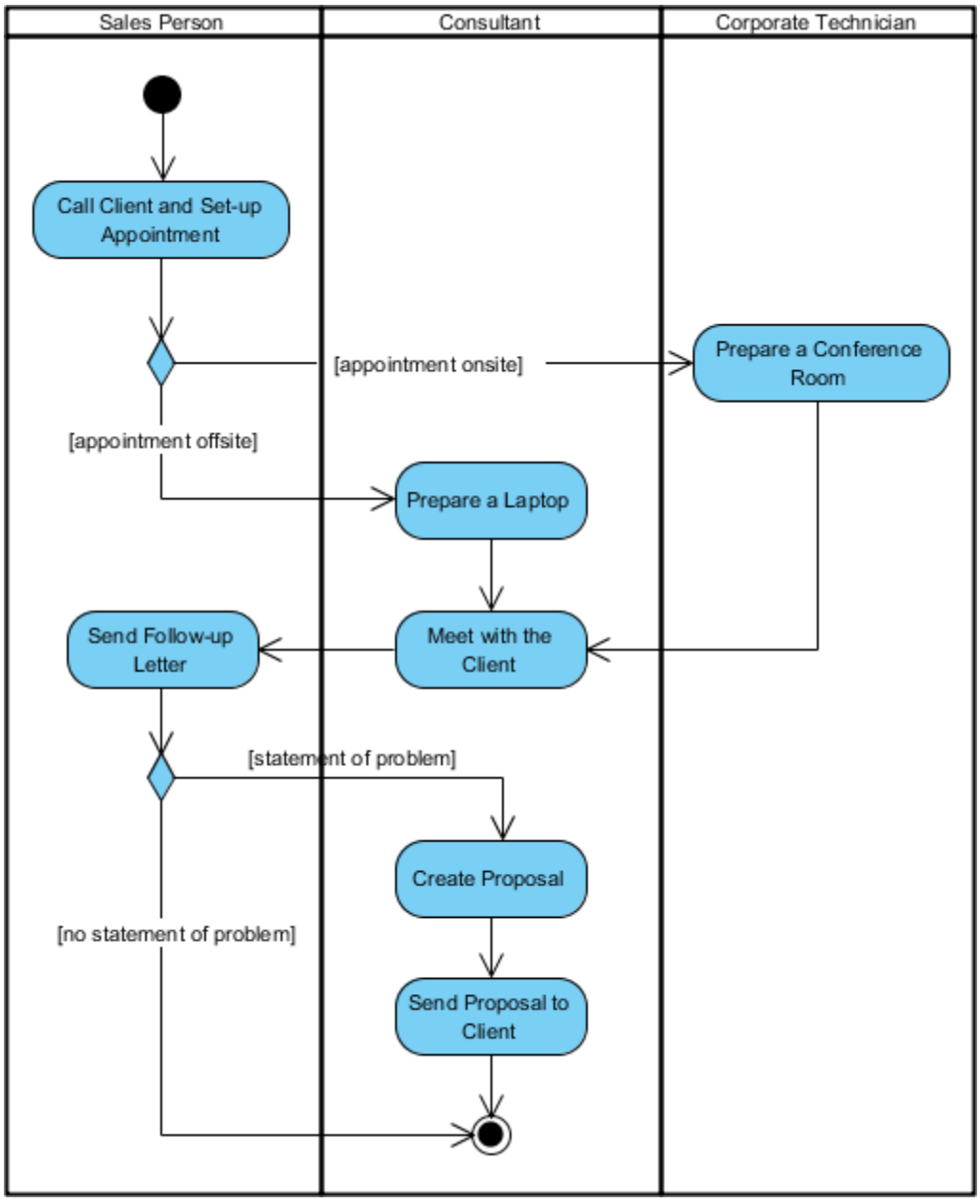
➢ Example: 03

- An applicant wants to enroll in the university.
- The applicant hands a filled out copy of Enrollment Form.
- The registrar inspects the forms.
- The registrar determines that the forms have been filled out properly.
- The registrar informs student to attend in university overview presentation.
- The registrar helps the student to enroll in seminars
- The registrar asks the student to pay for the initial tuition.

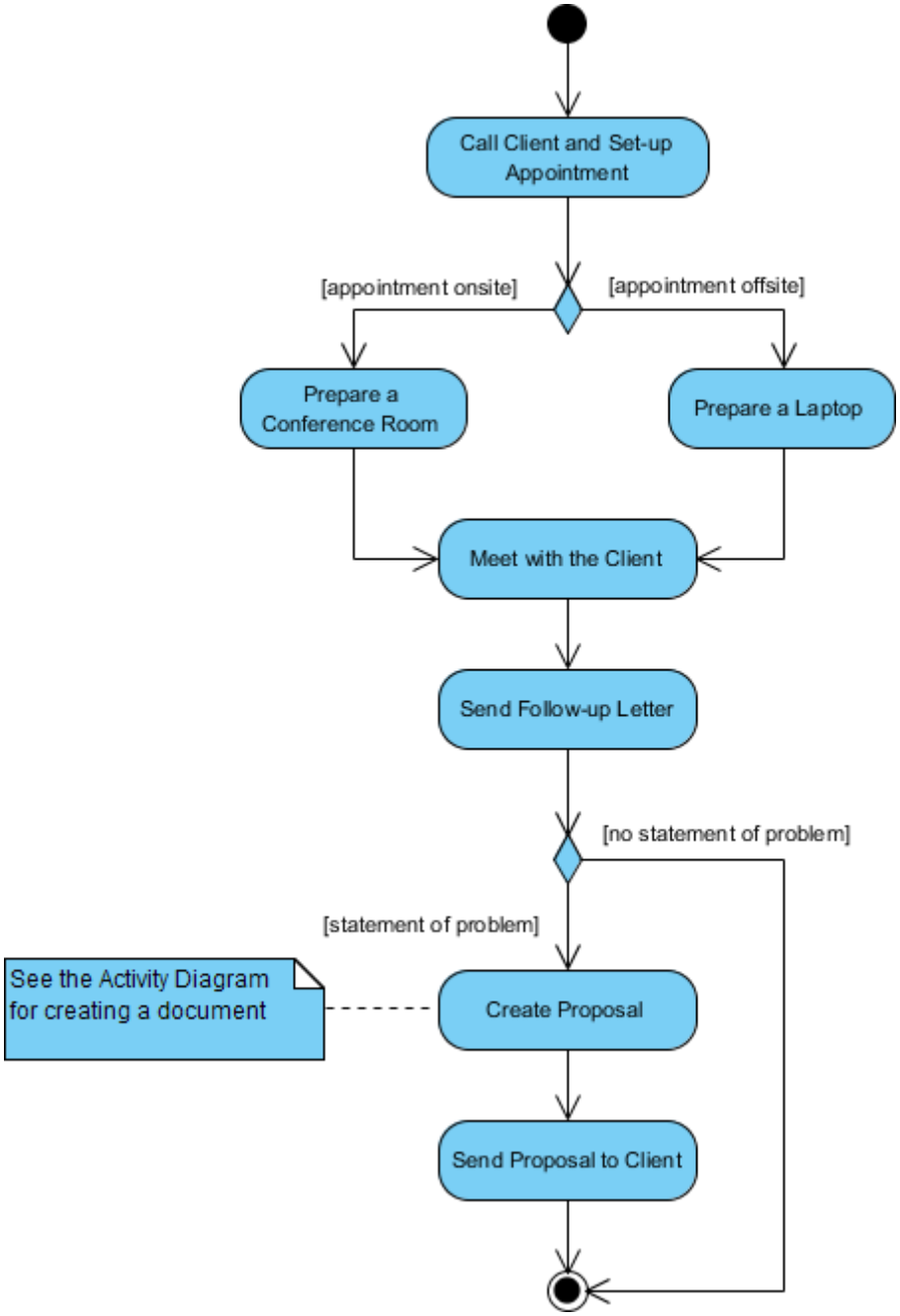


Swimlane and Non-Swimlane Activity Diagram

➢ With Swimlane:



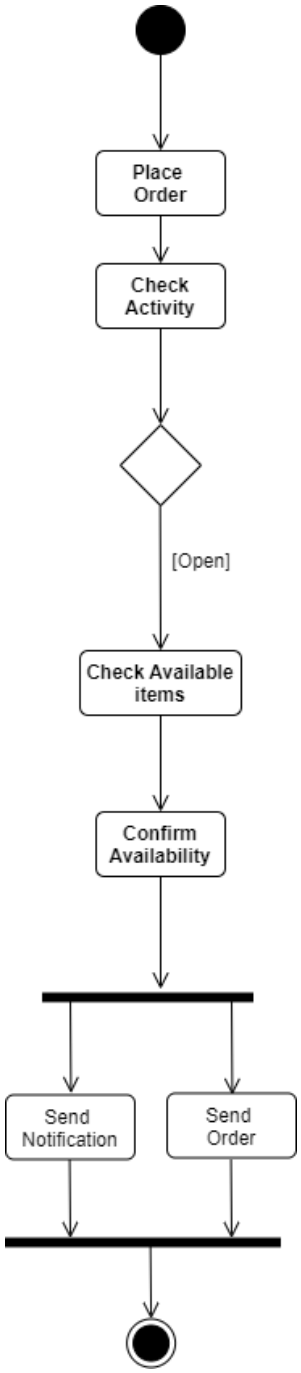
➢ With Out Swimlane:



Activity Diagram - Case Studies

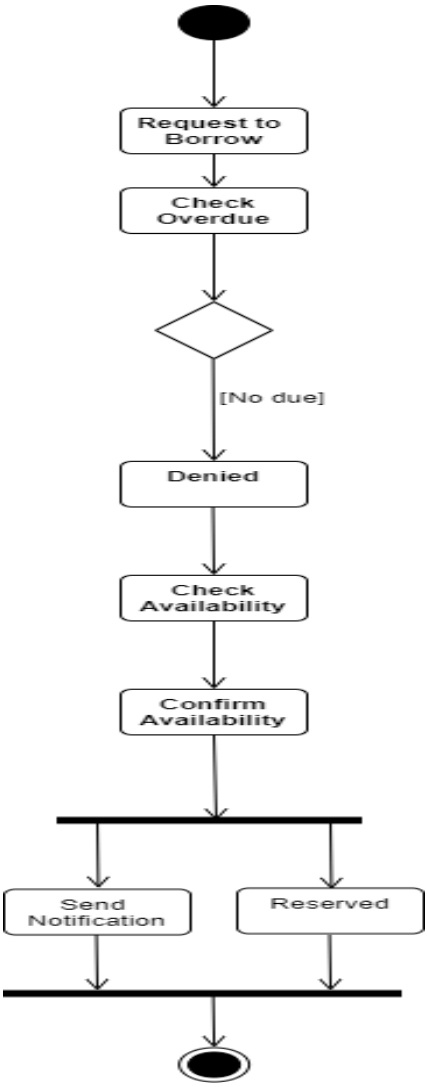
➢ Case: 01

⇨ In a food delivery app, a customer **places an order**. The system first **checks** if the selected restaurant is open. If the restaurant is open, the system **checks if the items ordered are available**. Once all items are **confirmed available**, two actions occur simultaneously, the **order is sent to the restaurant**, and **a confirmation notification** is sent to the customer.



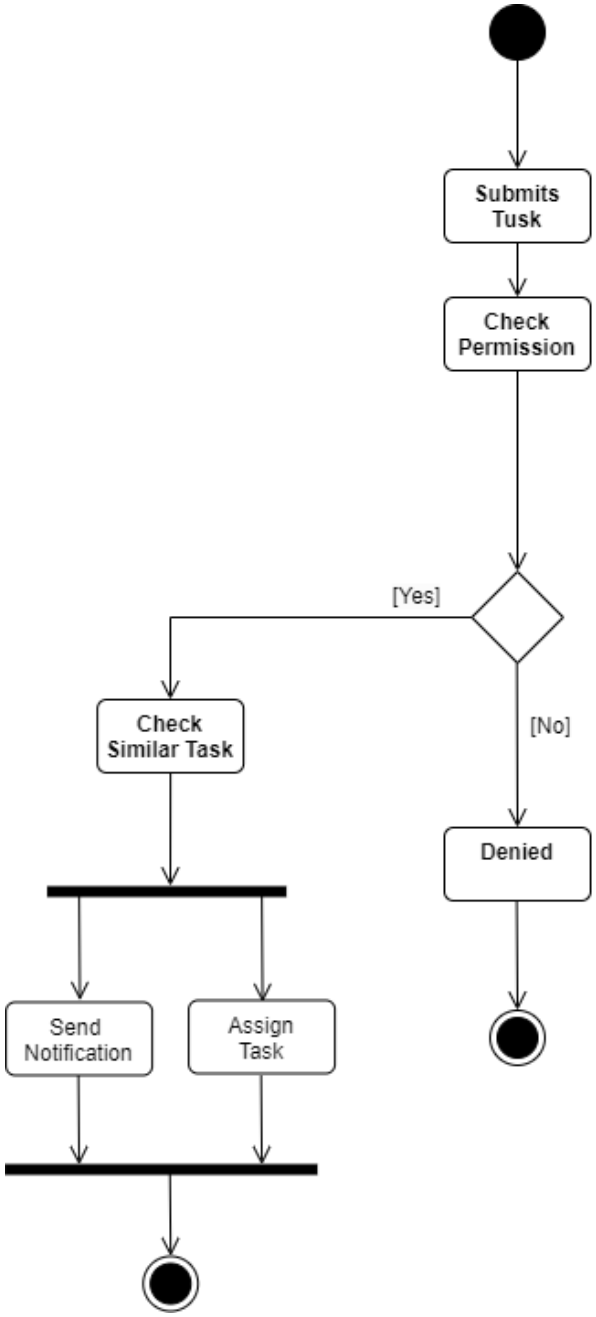
➢ Case: 02

⇨ In a simple book borrowing system, a **user requests to borrow a book**. The system **checks if the user has any overdue** books. If no overdue books exist, the system **checks the availability** of the requested book at different library branches. Once the **availability is confirmed**, two actions occur simultaneously: the book is **reserved** for the user, and a **notification is sent** to the user confirming the reservation.



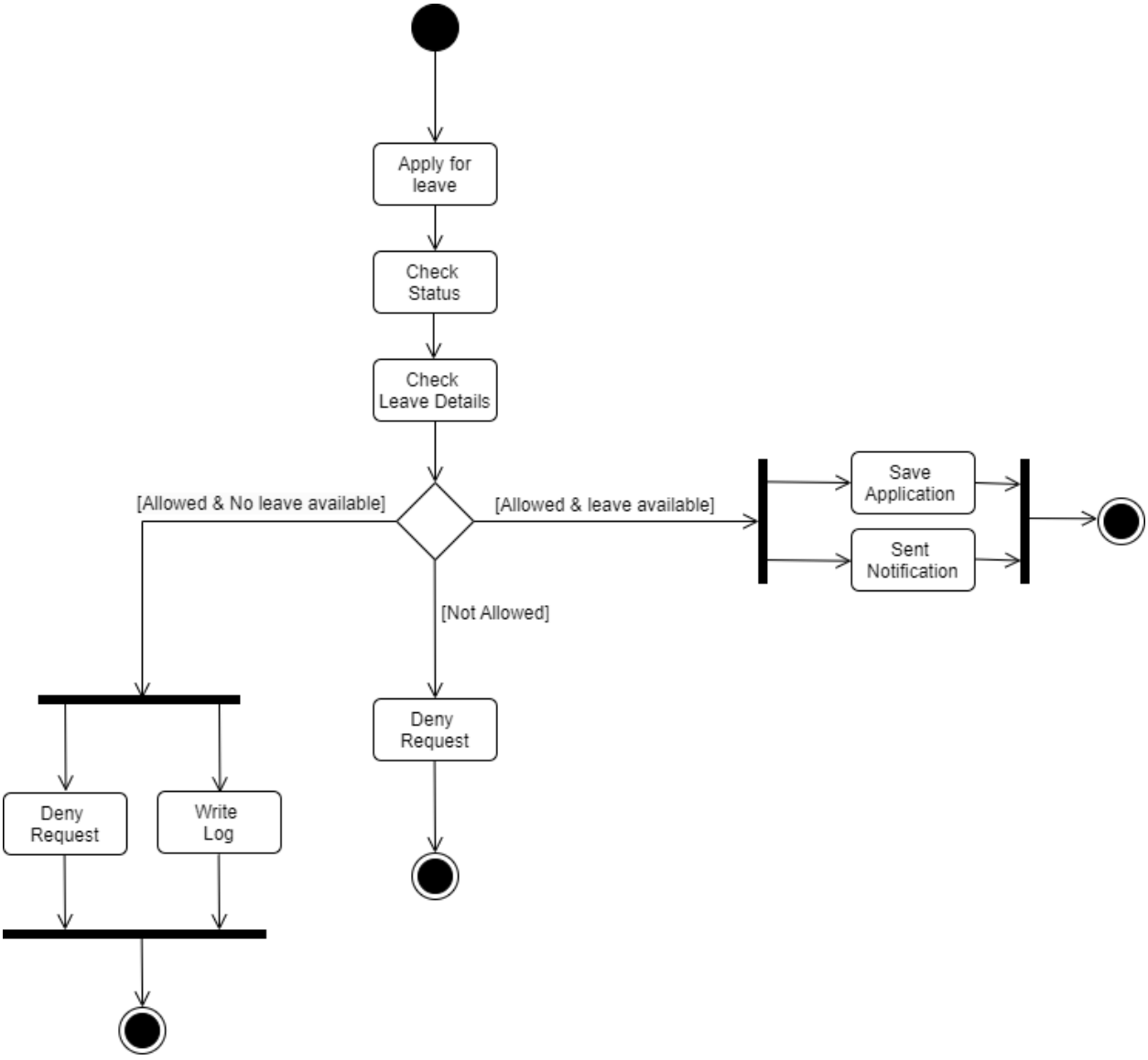
➢ Case: 03

⇨ In a task management system, an **employee submits a task** for review. The system first **checks if the employee has the required permissions** to submit the task. If the employee has permission, the system **checks for similar tasks already submitted**. Once all potential duplicates are checked, the task is submitted. Simultaneously, the system **sends a notification** to the employee and **assigns the task** to a reviewer. If the employee has no permission the system denied..



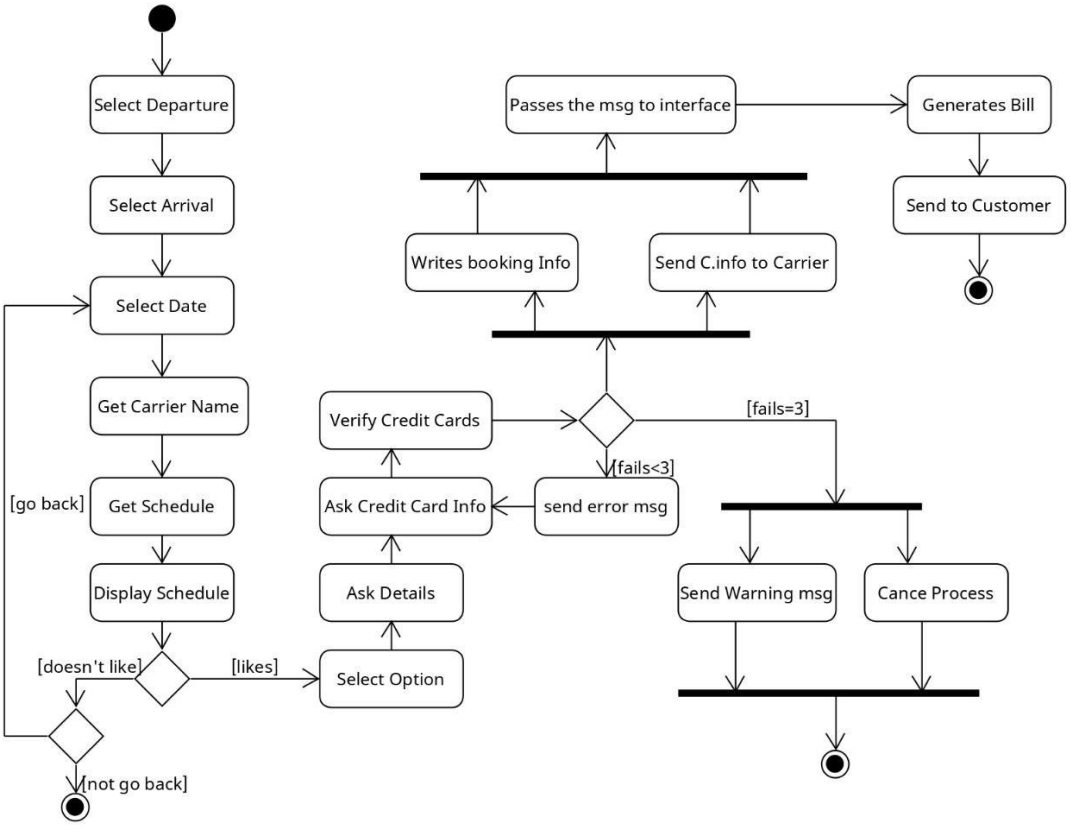
➢ Case: 04

⇨ A user can apply for leave in an employee management system. After the user **makes a request for leave,** the system **checks the user status** and **leave details** in the database. If the user is allowed and there is remaining leave available, the **leave application is saved** for supervisor approval and a **notification is sent to the user** at the same time. If the user is allowed, but there is no remaining leave available, the user request is **denied**. If the user is not allowed to apply for leave, the **request is also denied**, and a **log is written** simultaneously.



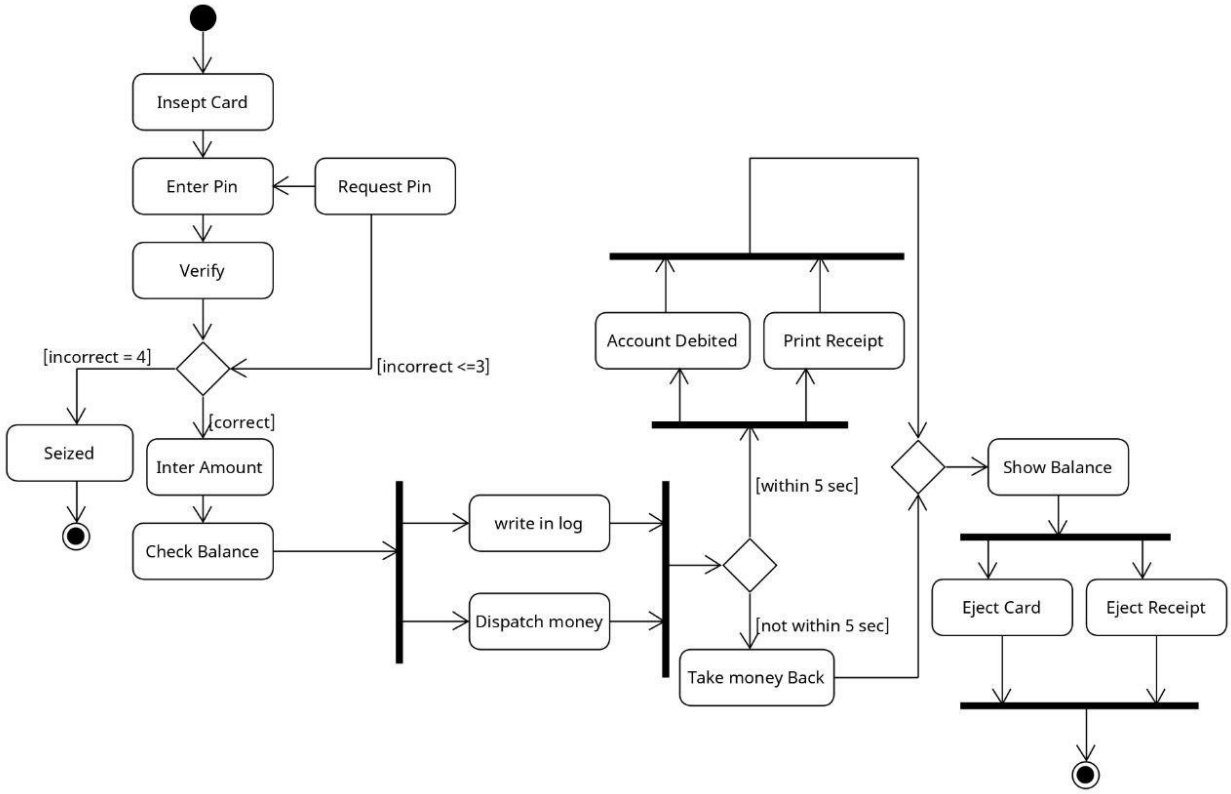
➢ Case: 05

⇨ In an online airlines ticket booking system, a customer places a request to the system **selecting the departure** and **arrival destinations** and also the **date of departure**. The system then **gets all the carrier names** and their **time schedule** from the flight database. Once the carrier names and their time schedules are received the **system displays the options** of the carriers and the schedule to the customer. If the customer likes any one of the options, he **selects the option**. But, if the customer doesn't like any option he can **go back** and select a **new date** of departure **or he can close the application**. After selecting an option, the system asks for **customer details** and **credit card information**. The system **verifies the credit card**. If the credit card is valid, the system **writes all the booking information** in the booking database and sends **customer information to the carrier** simultaneously. If the verification of the credit card fails, the system **sends an error message** to the customer and requests for **credit card information again**. The verification process is done again. If the credit card verification fails for three times the **system cancels the booking process** and **sends a warning message to the credit card agent** at the same time. Once the booking is done the system **passes the message to the interface** and the interface **generates a bill** and **sends it to the customer**.



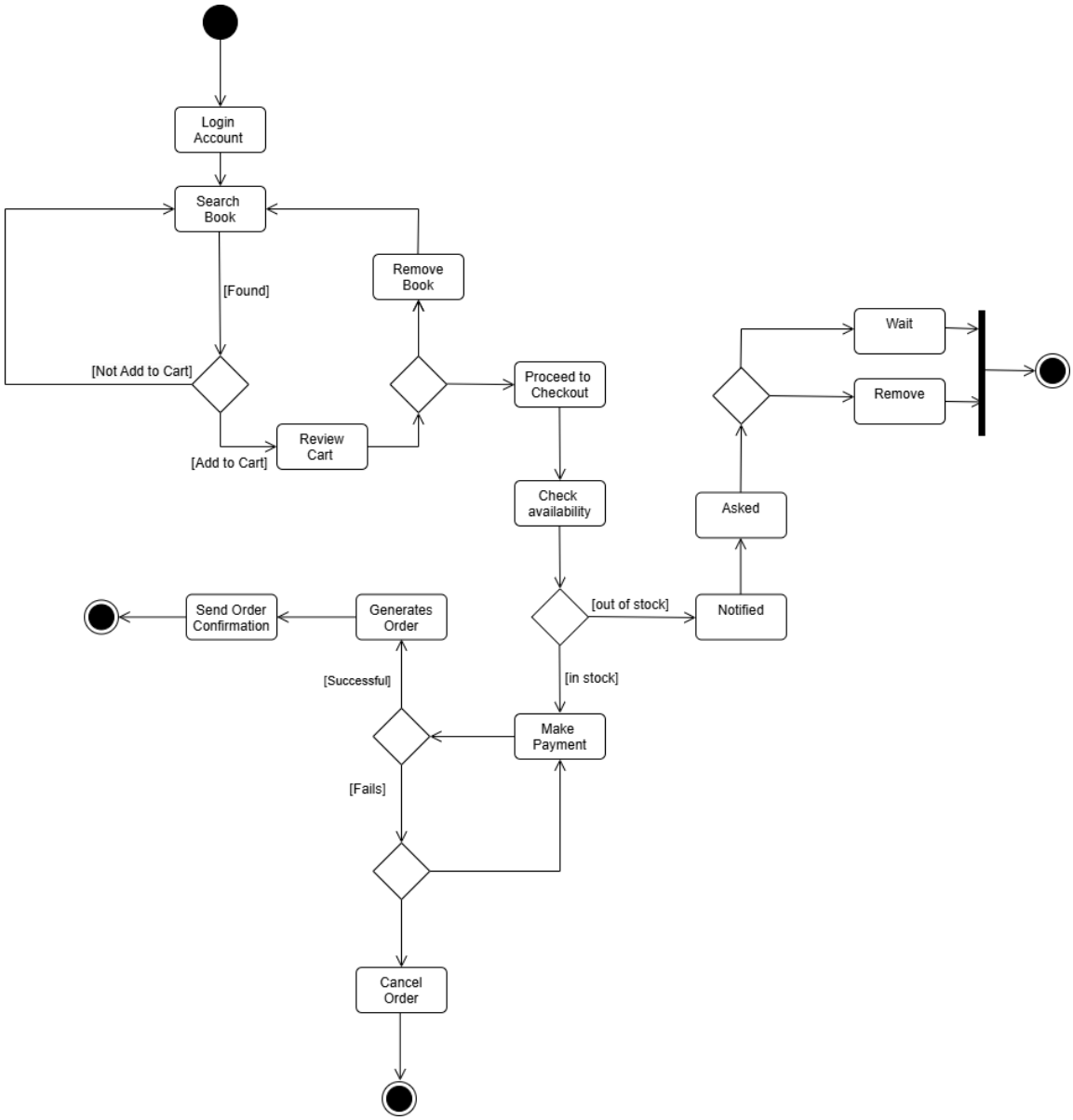
➢ Case: 06

⇨ In an ATM machine a customer starts a withdrawal transaction by **inserting the card**. Then he **enters the pin**, which is **verified** by the bank. If the pin is incorrect the **machine requests for the pin again** and the customer **enters pin number**. The verification repeated for 3 times for a pin number. If the pin is incorrect even in 4th attempt, the card is **seized** by the machine and **the transaction is closed**. If the pin is correct customer **enters the amount** he wishes to withdraw. The bank then **checks the account balance** of the customer. If the balance is greater than or equals to the withdrawal amount then money is **dispatched** through the machine and the **information is written in the log concurrently**. If the money is taken form the slot within 5 seconds, the account is **debited** and a transaction **receipt** is printed simultaneously. If money is not taken in 5 seconds, it is **taken back** by the machine. Then the **balance is shown** to the customer. The **card and the receipt** are then ejected at the same time, and the transaction is completed.



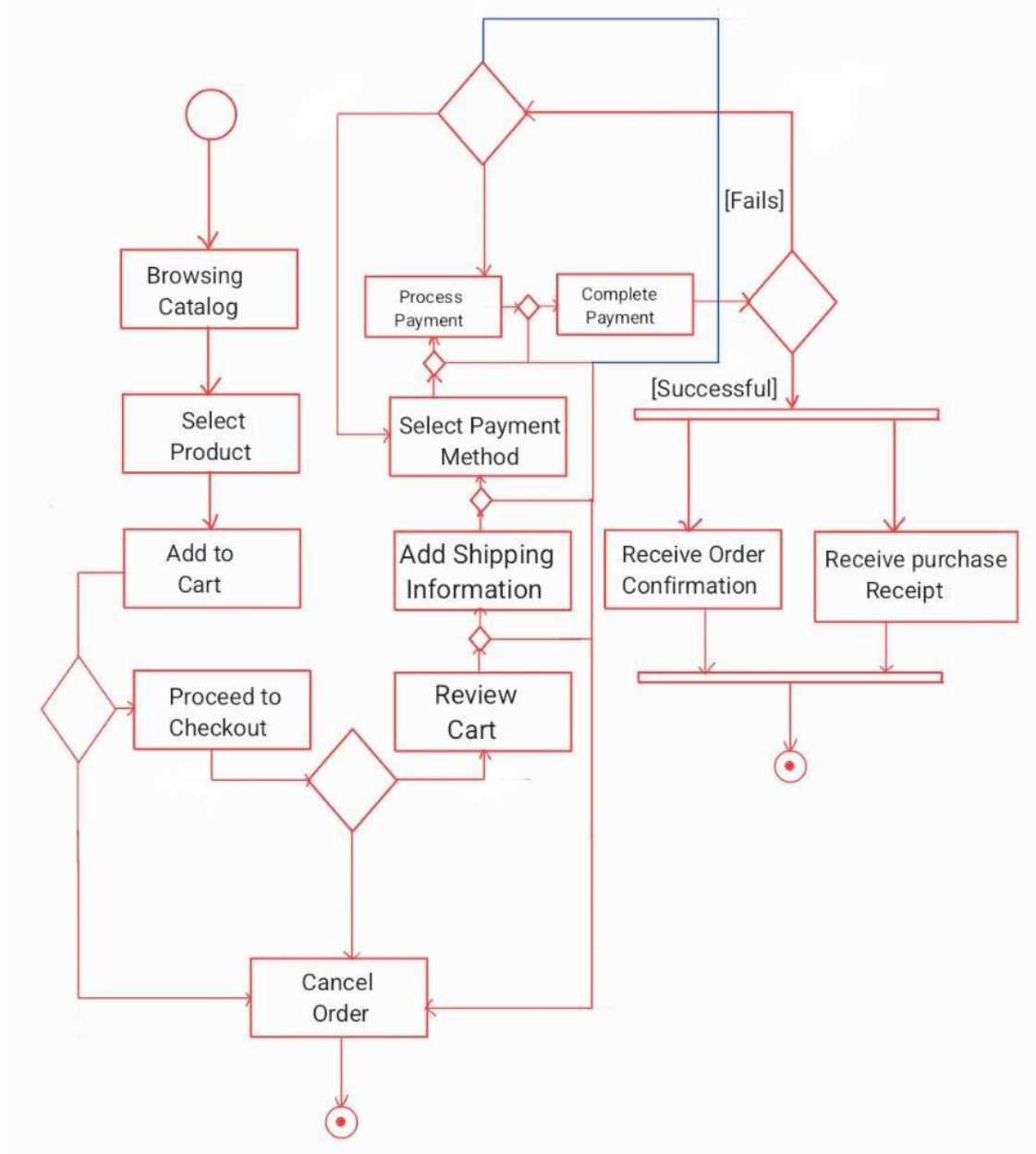
➢ Case: 07

⇨ In an online bookstore, a user **can log into** their account. Once logged in, the user can **search for a book** using a search bar. If the book is found, the user decides whether to add it to the cart or not. If not added, the user can continue searching for other books. After adding a book to the cart, the user **reviews the cart.** At this stage, the user can either **proceed to checkout** or **remove the book** and **continue searching**. If the user chooses to proceed to checkout, the system **checks the availability** of the selected books. If the books are in stock, the **user makes a payment**. If the books are out of stock, the user **is notified** and **asked** to either **wait** or **remove** the unavailable books. Once the payment is successful, the system **generates an order confirmation** and sends it to the user. If payment fails, the user is **allowed to retry** or **cancel the order.**



➢ Case: 08

⇨ In an online shopping system, a customer begins by browsing the product catalog to find items they wish to purchase. After selecting products, they add them to their shopping cart. Once they are satisfied with the selected items, they proceed to the checkout process, where they review their cart and enter their shipping information. Following this, the customer selects a payment method and completes the payment. If the payment is successful, they receive an order confirmation and a receipt for the purchase. In case the payment fails, the customer is given the option to retry the payment or select an alternative payment method. The process ends after the order is confirmed and payment is completed. At any point, the customer can decide to cancel the order and abandon the checkout process.



STATE DIAGRAM

Ch: 06

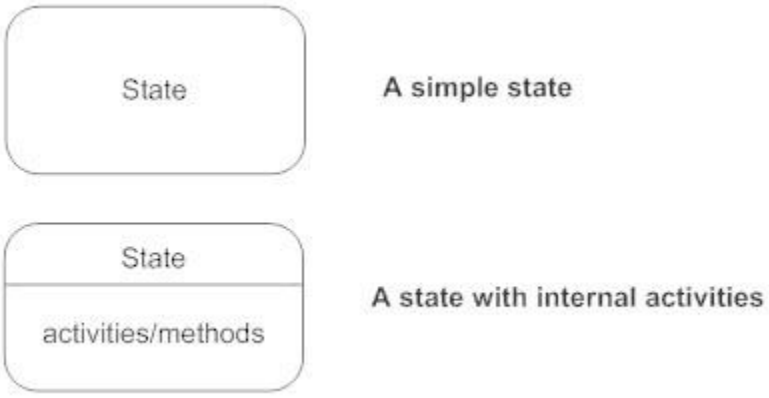
State Diagram

- In UML, state diagram shows the behavior of classes in response to external stimuli. Specifically, a state diagram describes the behavior of a single object in response to a series of events in a system. Sometimes it's also known as a Harel state chart or a state machine diagram. This UML diagram models the dynamic flow of control from state to state of a particular object within a system.

State Diagram Notation

States:

- States represent situations during the life of an object. You can easily illustrate a state by using a rectangle with rounded corners.



Transition:

- A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it. A state can have a transition that points back to itself.

➢ Initial State:

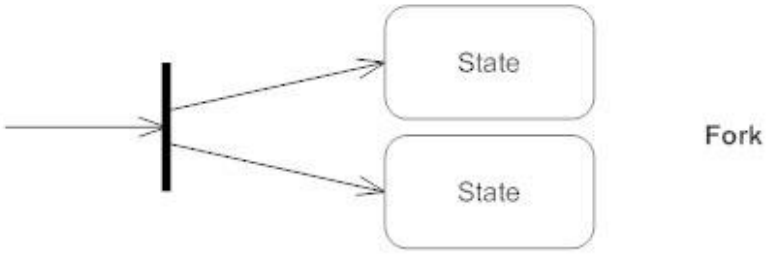
⇨ A filled circle followed by an arrow represents the object's initial state.

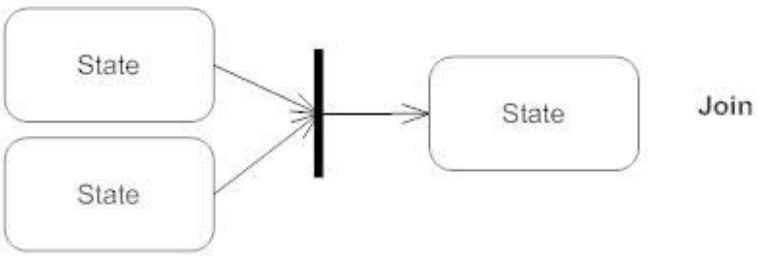
➢ Final State:

⇨ An arrow pointing to a filled circle nested inside another circle represents the object's final state.

➢ Synchronization and Splitting of Control:

⇨ A short heavy bar with two transitions entering it represents a synchronization of control. The first bar is often called a fork where a single transition splits into concurrent multiple transitions. The second bar is called a join, where the concurrent transitions reduce back to one.

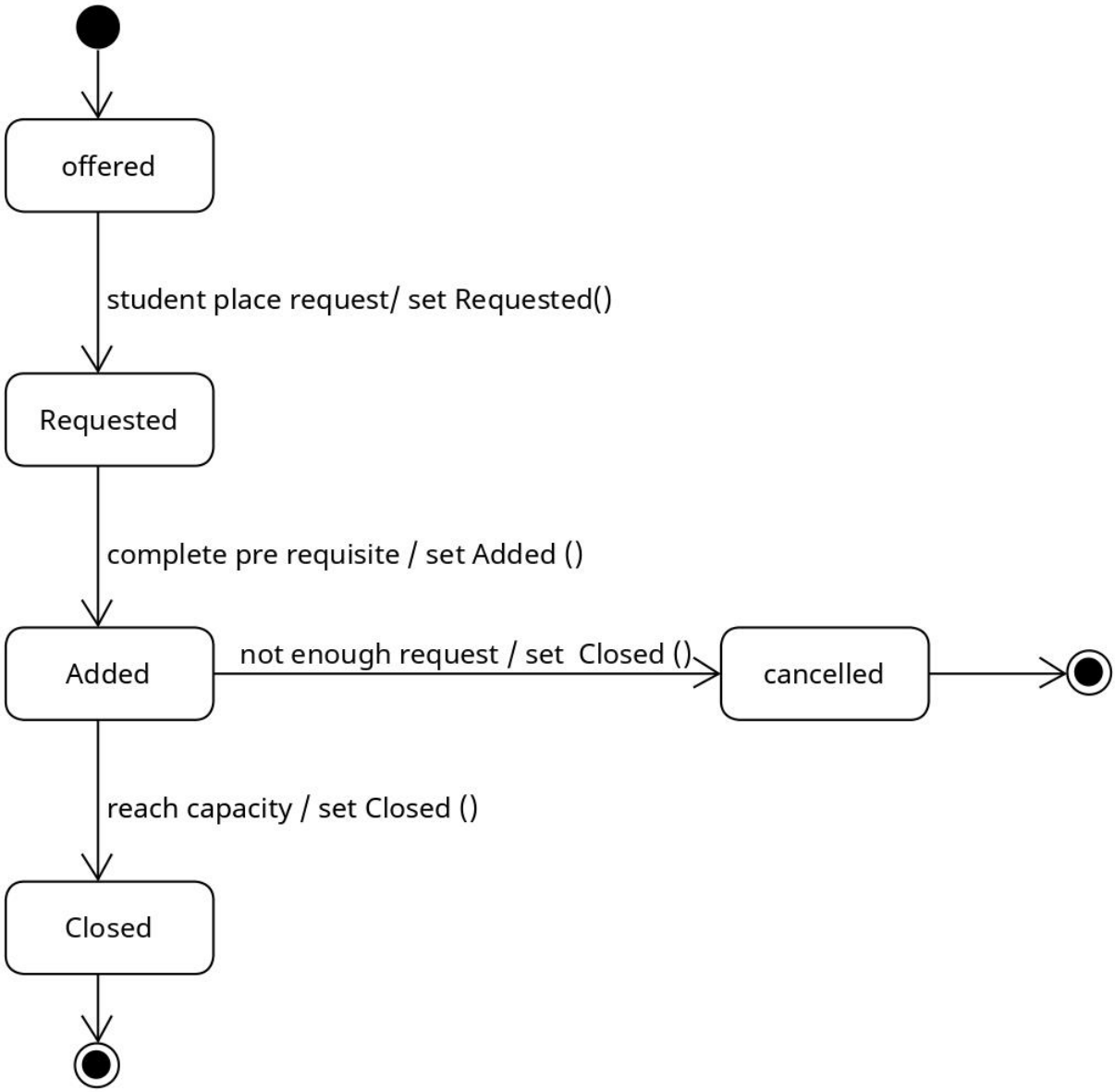




State Diagram - Case Studies

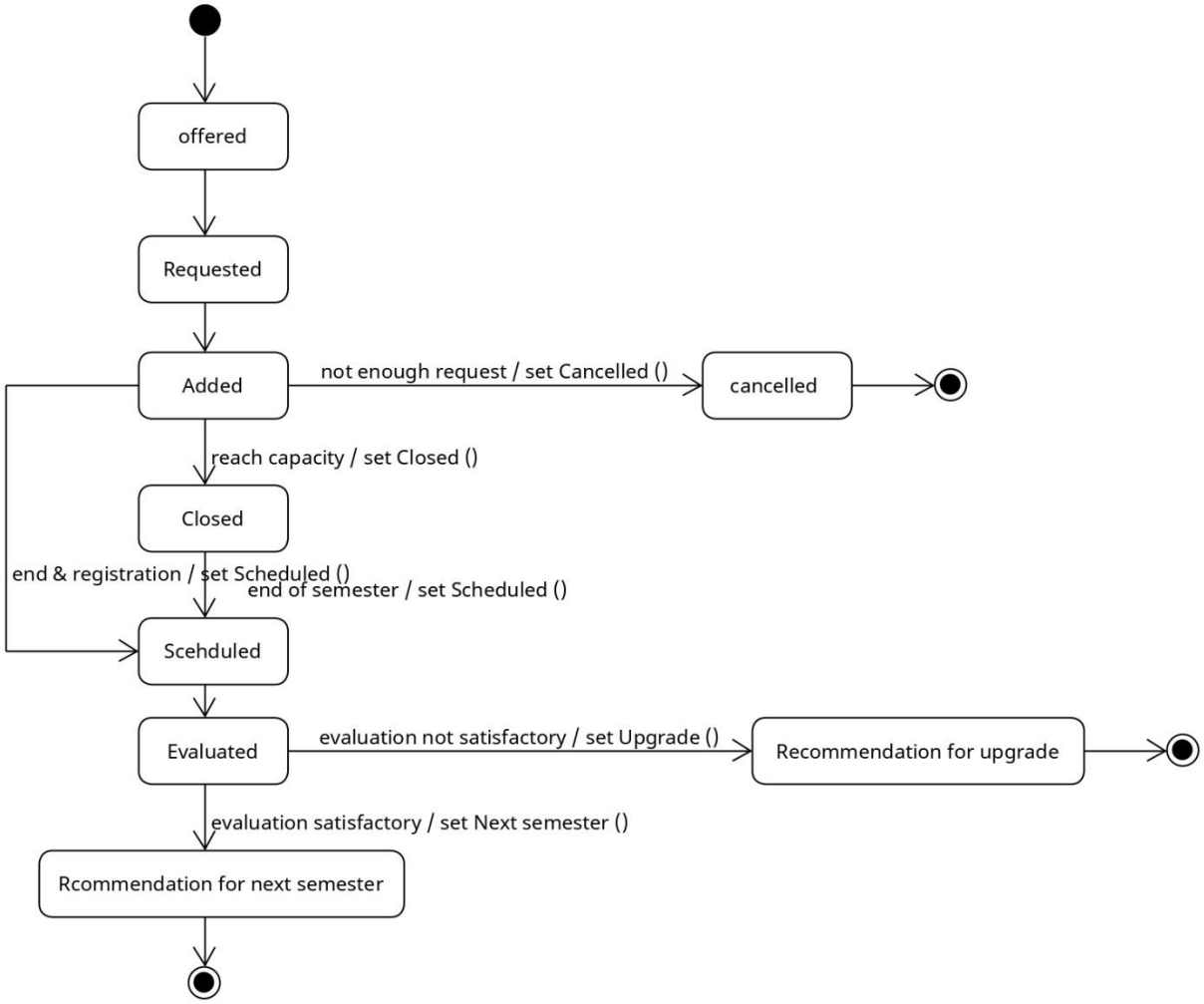
➢ Case: 01

⇨ A course is initially created in offered state. The offered courses are then included by the students in their requests and the requests are placed for registration. Some of the requested courses are then had to be checked against other completed courses by the students as their pre requisites. If the pre-requisites are fulfilled students are added in the courses. When a course reaches the limit of its capacity the course is then closed for further student requests. On the other hand if there are not enough students requesting for a course the course is cancelled



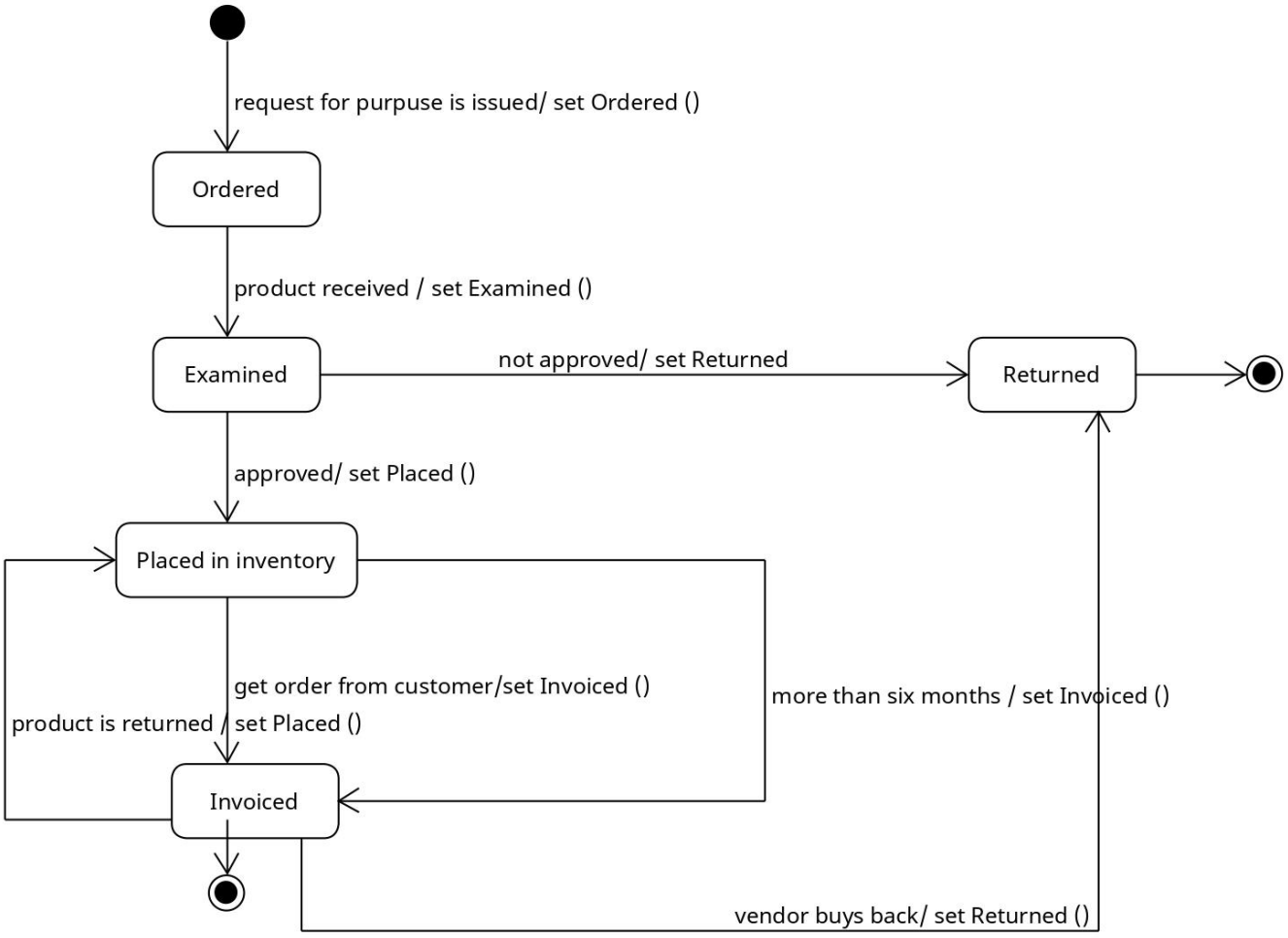
➢ Case: 02

⇨ A course is initially created in offered state. The offered courses are then included by the students in their requests and the requests are placed for registration. Some of the requested courses are then had to be checked against other completed courses by the students as their pre requisites. If the pre-requisites are fulfilled students are added in the courses. When a course reaches the limit of its capacity the course is then closed for further student requests. On the other hand if there are not enough students requesting for a course the course is cancelled. Once the student registration is finished a time schedule is attached the course along with a teacher. At the end of the semester the course is evaluated by the authority. If the evaluation is satisfactory it is recommended for the next semester, otherwise it is recommended for up gradation.



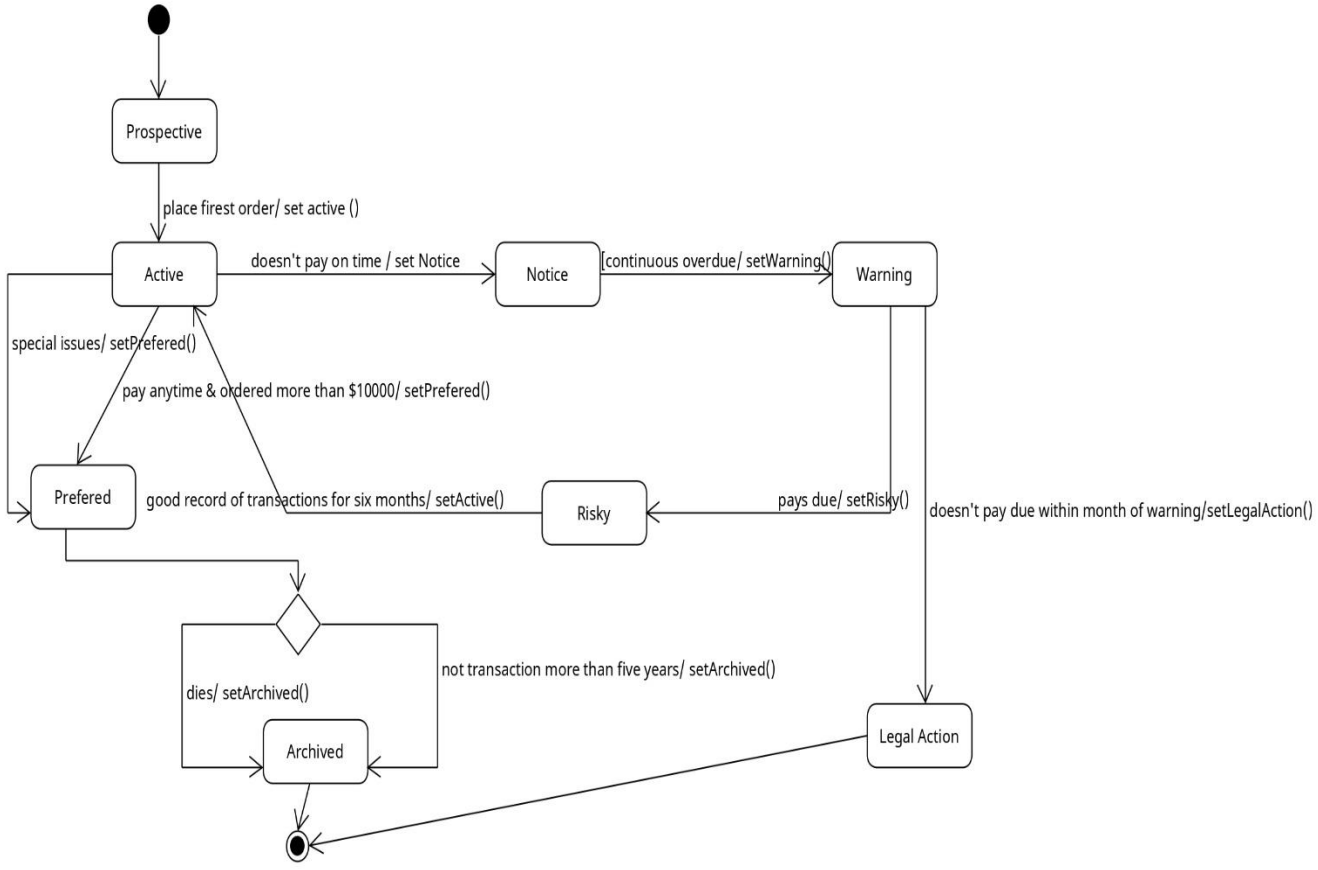
➢ Case: 03

⇨ Products are first entered into our system when a request for purchase for that order is issued. A purchase order is then generated for the product and recorded with that associated product. The purchase order is then sent to the vendor. When the product is received from the vendor, it is examined by the quality control department. Once the product is approved by the quality control it is placed into inventory by recording the location where it is placed. If it is not approved the product is returned to the vendor. When a customer buys a product it is included in an invoice. Sometimes, a product is returned. In that case, you put the product back into inventory and record the location. If a product is lying in the inventory for more than six months it is returned to the vendor, but this time the vendor buys back the product in a reduced price, which means the product is included in an invoice.



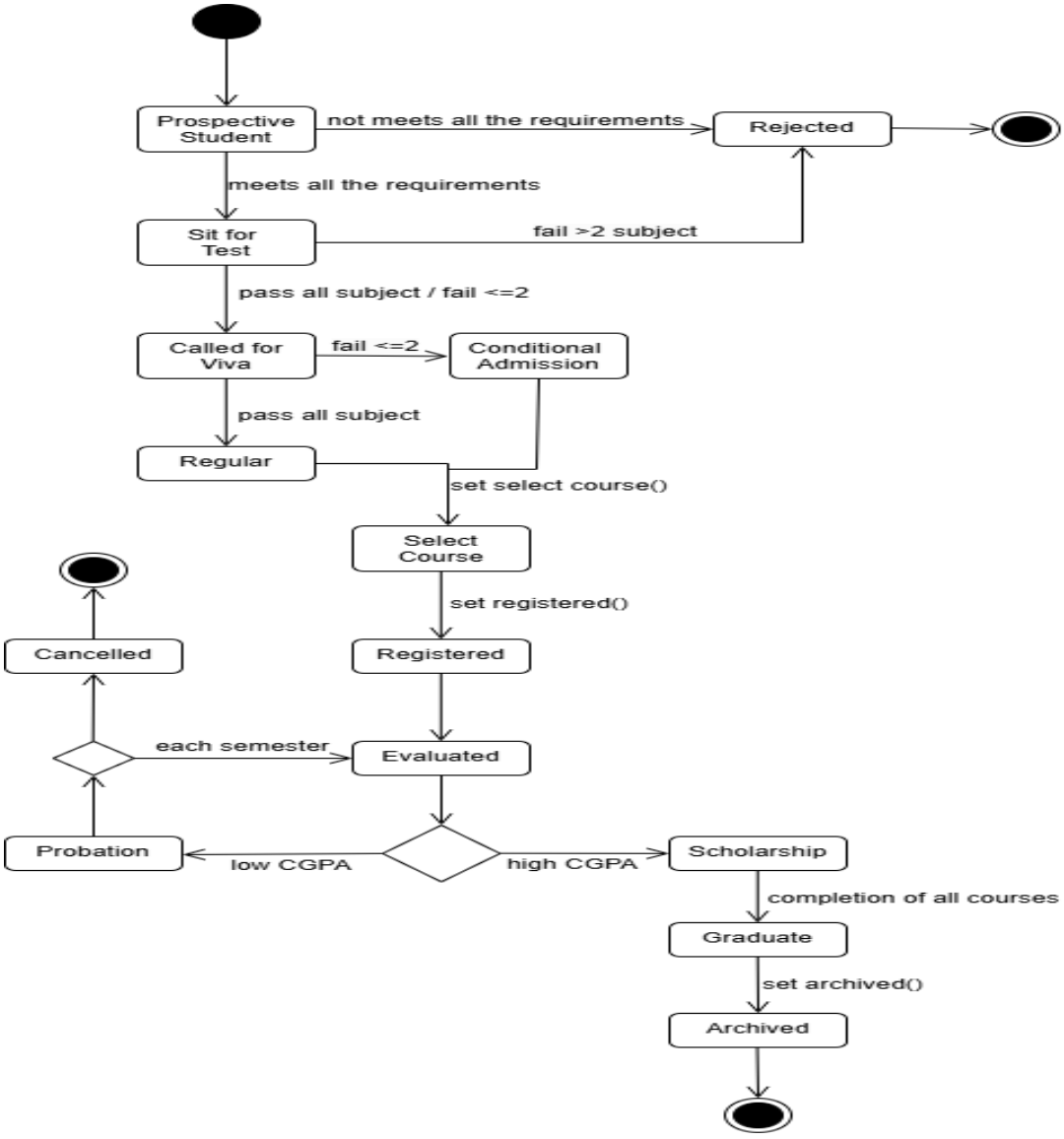
➢ Case: 04

⇨ In an order processing system all customers are initially set up as prospective customers, but when they place their first order, they are considered to be active. If a customer doesn't pay an invoice on time, he is placed on notice. If he continues to keep overdue, he is given warning. If the customer doesn't pay the dues within a month of the warning, legal actions are taken against the customer. If the customer pays after the warning, he is taken back to the system as a risky customer. Only when he has a good record of transaction for six months he is taken as regular customer. If an active customer does pay on time and has ordered more than $10,000 in the previous six months, he warrants preferred status. Preferred status may be changed only if the customer is late on two or more payments. Then he returns to active status rather than probation, giving him the benefit of the doubt based on his preferred history. Sometimes an active customer can become preferred customer if the management has special issues with the customer. The customer information is archived for good if the customer dies or if there has been no transaction with the customer for more than five years.



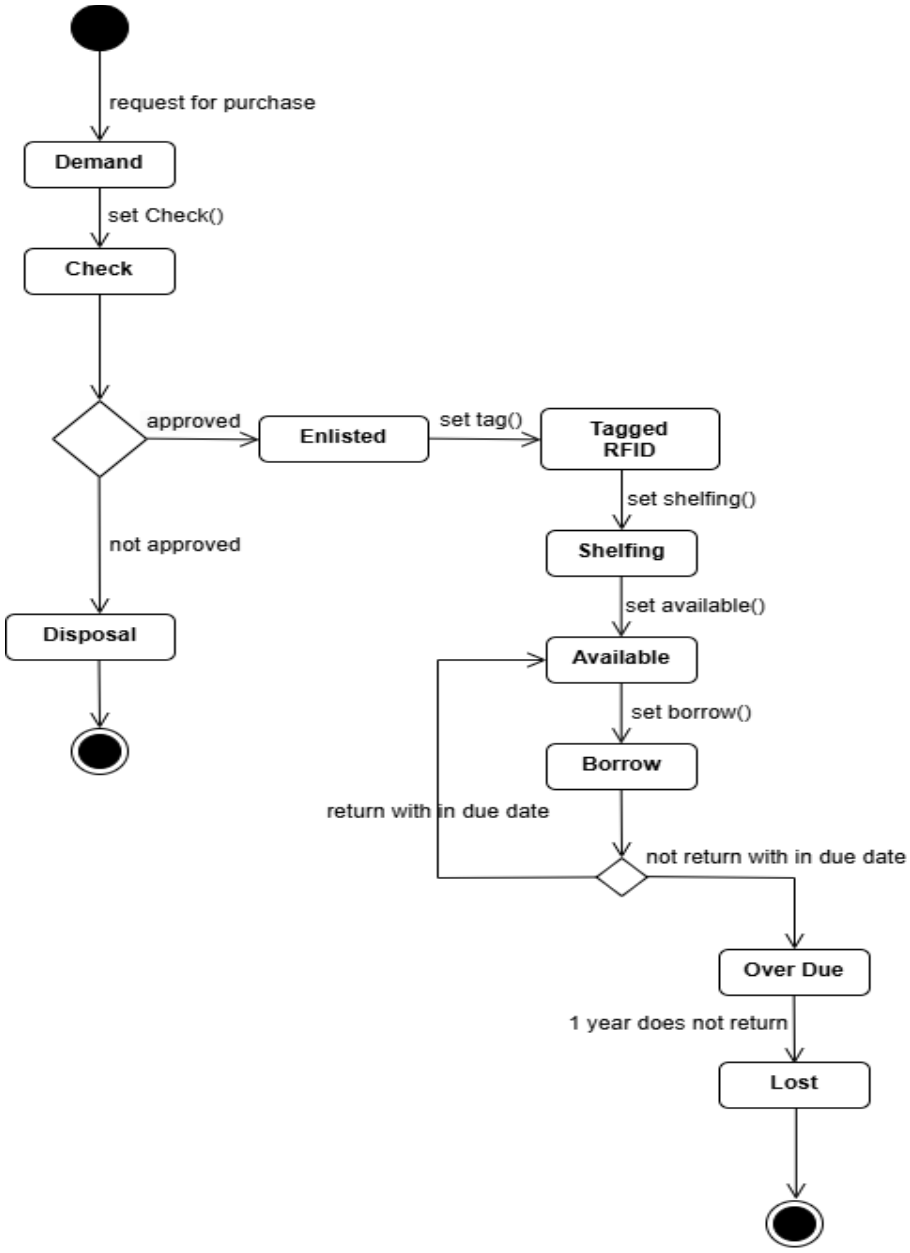
➢ Case: 05

⇨ A student object is created as prospective student when he submits the admission form to a university. Student information is verified against the admission requirements and if he meets all the requirements he is asked to sit for an admission test. But if the student doesn't meet the requirements, he is rejected and the student object is deleted permanently from the system. Any student who fails in more than two subjects in the admission exam is also rejected. Students who pass in all the subjects or fail in one or two subjects are called for viva. At the viva the students who failed one or more subjects are given conditional admission with non-credit courses. Students who pass all the subjects are given regular student status. All sorts of admitted students are required to go through pre-advising to select appropriate courses for the semester. Once courses are selected students pay for the courses they have taken and become registered students of the semester. After each semester the student grade is evaluated and in case of low CGPA, the student becomes a probation student. High CGPA students are granted scholarships provided that they apply for it and fulfill the requirements. Probation students have to go through evaluation every semester and if the CGPA of the student doesn't improve the studentship is cancelled from the university. After successful completion of the course a regular student becomes a graduate student and all the information of a graduate student is archived.



➢ Case: 06

⇨ A book object is initially created on demand status, when it is requested for purchase for a library. After it is purchased, it is set as new. All new books are checked by the Audit department. If the book is not approved, it is set for disposal. If the book is approved, it is enlisted with a Library code. Library coded books are also tagged with RFID and prepared for shelfing. After they are placed on the shelf, the book is set as available. A library member can borrow an available book. If a borrowed book is returned within the due date, it becomes available again. If it is not returned within the due date, the member is fined, and the book is set as overdue. If an overdue book is not returned within 1 year, the book is set a permanently lost and the member is warned.



Weighted Methods per Class (WMC)

- The effort in developing a class will in some sense will be determined by the number of methods the class has and the complexity of the methods.
- Suppose a class *C* has methods *M1, M2… Mn* defined on it. Let the complexity of the method *M1* be *c1, then*

$$WMC = \sum_{i=1}^{i=n} c_i$$

- If the ***complexity*** of each method is considered 1, WMC gives the total number *of* methods in the class.
- The data based on evaluation of some existing programs, shows that in most cases, the classes tend to have only a small number of methods, implying that most classes are simple and provide some specific abstraction and operations.

WMC metric has a **reasonable correlation** with **fault-proneness** of a class. As can be expected, the larger the WMC of a class the better the chances that the class is fault-prone.

Depth of Inheritance Tree (DIT)

- Inheritance is one of the unique features of the object- oriented paradigm.
- Inheritance is one of the main mechanisms for **reuse**
- The deeper a particular class is in a class hierarchy, the more methods it has available for reuse, thereby providing a **larger reuse potential**
- Inheritance increases coupling, which makes **changing a class harder**
- A class deep in the hierarchy has a lot of methods it can inherit, which makes it **difficult to predict its behavior**
- The DIT of a class *C* in an inheritance hierarchy is the depth from the root class in the inheritance tree
- In case of multiple inheritance, the DIT metric is the maximum length from a root to C
- Statistical data suggests that most classes in applications tend to be close to the root, with the maximum DIT metric value being around 10
- Most the classes have a DIT of 0 (that is, they are the root).
- Designers might be giving upon reusability in favor of comprehensibility
- The experiments show that DIT is **very significant** in predicting **defect-proneness** of a class: the higher the DIT the higher is the probability that the class is defect-prone

Number of Children (NOC)

- The number of children (NOC) metric value of a class *C* is the number of immediate subclasses of C
- This metric can be used to evaluate the degree of reuse, as a higher NOC number reflects reuse of the definitions in the superclass by a larger number of subclasses
- It also gives an idea of the **direct influence** of a class on other elements of a design
- The larger the influence of a class, the more important the *class* is **correctly designed**
- In the empirical observations, it was found that classes generally had a small NOC metric value, with a vast majority of classes having no children
- This suggests that in the systems analyzed, inheritance was not used very heavily
- The data *suggest* that the **larger** the NOC, the **lower** the probability of **detecting defects** in a class
- The higher NOC classes are **less defect-prone**. The reasons for this are not very clear or definite.

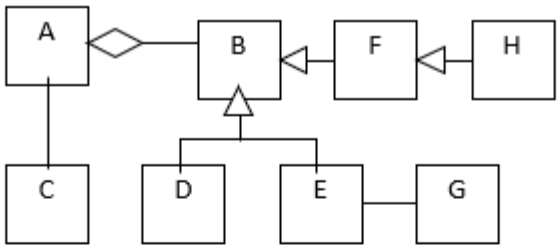
Number of Children (NOC)

- Cohesion captures how **closely bound** the different methods of the class are
- Two methods of a class *C* can be considered "cohesive" if the set of instance variables of *C* that they access have some elements in common
- High cohesion is a **highly desirable** property for modularity
- Let I_i and I_j be the set of instance variables accessed by the methods M_1 and M_2, Q be the set of all cohesive pairs of methods, that is, all (M_i, M_j) such that I_i and I_j have a non-null intersection. Let *P* be the set of all noncohesive pairs of methods, that is, pairs such that the intersection of sets of instance variables they access is null. Then LCOM is defined as

  LCOM = |P|-|Q|, if |P| > |Q|, otherwise 0.

- If there are *n* methods in a class *C*, then there are $n(n — 1)$ pairs, and LCOM is the number of pairs that are noncohesive minus the number of pairs that are cohesive
- The **larger** the number of cohesive methods, the more cohesive the class will be, and the LCOM metric will be **lower**
- A high LCOM value may indicate that the methods are trying to do different things and operate on different data entities
- If this is **validated**, the class can be **partitioned** into different classes
- The data in [BBM95] found **little significance** of this metric in predicting the **fault-proneness** of a class

Example (NOC, DIT, CBC)



CLASS	NOC	Influence on Design	DIT	Reuse Potential	CBC
A	0	Low	0	Low	Low
B	3	Highest	0	Low	Highest
C	0	Low	0	Low	Lowest
D	0	Low	1	Moderate	Low
E	0	Low	1	Moderate	Low
F	1	Moderate	1	Moderate	Moderate
G	0	Low	0	Low	Lowest
H	0	Low	2	Highest	Moderate

Example (LCOM)

CLASS A
a1 a2 a3 a4
A1(a1, a2) A2(a1) A3 (a4) A4 (a1, a4)

LCOM = |P|-|Q|, if |P| > |Q|, otherwise 0

Pairs:
(A1, A2), (A1, A3), (A1, A4), (A2, A3), (A2, A4), (A3, A4)

P = 2 (Non-Cohesive pairs)
Q = 4 (Cohesive pairs)

Q > P

LCOM = 0