

# Sequence Diagram and Collaboration Diagram

---

## 5.1 INTRODUCTION TO SEQUENCE DIAGRAM

A sequence diagram is used to express use-case realizations, i.e. to show how objects interact to perform the behaviour of all or part of a use-case. A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. The sequence diagram is used primarily to show the interactions between the different objects of a system, in the sequential order of the occurrence of interactions.

A sequence diagram depicts the sequence of actions that occur in a system and captures the invocation of methods in each object, and the order in which these invocations occur. Sequence diagrams are used to model the flow of logic within the system enabling both documentation and validation of logic.

The main purpose of a sequence diagram is to define event sequences that result in some desired outcome. The focus is less on messages and more on the time order in which messages occur. However, most sequence diagrams communicate what messages are sent between objects as well as the order in which they are sent.

The diagram conveys this information along the horizontal and vertical dimensions: the vertical dimension shows, top-down, the time sequence of messages/calls as they occur, and the horizontal dimension shows, left to right, the object instances that the messages are sent to. A sequence diagram is the dynamic modelling diagram focusing on identifying the behaviour of related objects within the system. Sequence diagrams are used to model the usage scenario, the logic of methods and services.

Sequence diagrams are used to give a picture of the design because they provide a way to visually step through invocation of the operations defined by classes. One can easily understand the need to change the design and to re-distribute the load within the system by just having a glance at what messages are being sent to an object, and by looking at approximately how long it takes to run the invoked method.

Sequence diagrams are used to display the interaction between users, screens, objects and entities within the system. They provide a sequential map of message passing between objects over time. Frequently these diagrams are placed under use-cases in the model to illustrate the use-case scenario—how a user will interact with the system and what happens internally to get the work done.



## 5.2 ELEMENTS OF SEQUENCE DIAGRAM

A sequence diagram is two-dimensional in nature. On the vertical axis, it shows the life of the object that it represents, while on the horizontal axis, it shows the sequence of the creation or invocation of these objects. A sequence diagram is made up of objects and messages. Basic elements of sequence diagram are as follows:

### 5.2.1 Life Lines

Lifelines represent either roles or instances that participate in the sequence of interaction being modelled. Lifeline notation elements are placed at the top of the diagram.

Lifelines are drawn as a rectangle with a dashed line descending from the centre of the bottom edge of the rectangle and the lifeline's name is placed inside the rectangle. Figure 5.1(a) is one of the ways to represent lifelines. Here a named object "John" belonging to class "Person" is placed inside the box showing the lifeline.

Another way of representing the life line is shown in Figure 5.1(b). The lifeline can represent an anonymous or unnamed instance also. When modelling an unnamed instance on a sequence diagram, representing an anonymous instance of the Person class, the lifeline would be " :Person" as shown in Figure 5.1(b).

As the sequence diagrams are used during the design phase of projects, it is possible to have an object whose type is unspecified. This third option to draw a lifeline is shown in Figure 5.1(c) where the lifeline shows "John" as unspecified class.

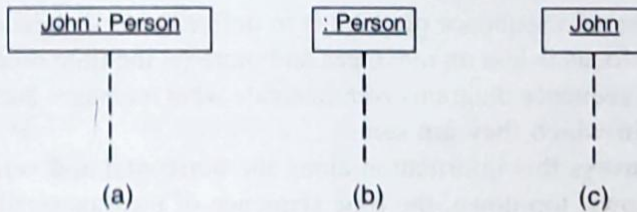


FIGURE 5.1 Lifelines.

### 5.2.2 Messages

A message defines a specific kind of communication between instances in an interaction. It represents a method call and invocation between objects. A message specifies the sender and the receiver, and defines the kind of communication that occurs between lifelines. For example, a communication can invoke, or call, an operation using a synchCall or asynchCall, and create or destroy a participant.

As shown in Figure 5.2 a line with a solid arrowhead that points towards the receiving lifeline represents a synchronous call operation in which the system waits for the flow of control to complete before continuing with the outer flow.

A line with an open arrowhead represents an asynchronous call in which the source object sends the message and immediately continues with the next step.

A dashed line with a solid arrowhead that points towards the originating lifeline represents a return message from a call to a procedure (Figure 5.2).

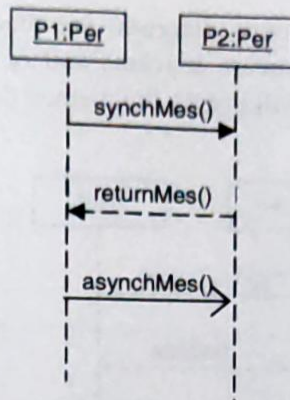


FIGURE 5.2 Message.

### 5.2.3 Activation

Activation boxes represent the time an object needs to complete a task. Activation is created automatically when we create a synchronous or an asynchronous message. Each activation represents an execution in a behaviour. Activation boxes also called, *focus of control* shown as a tall, thin rectangle on a lifeline, represents the period during which an element is performing an operation. The top and the bottom of the rectangle are aligned with the initiation and the completion time respectively. These are represented using rectangular boxes in Figure 5.3.

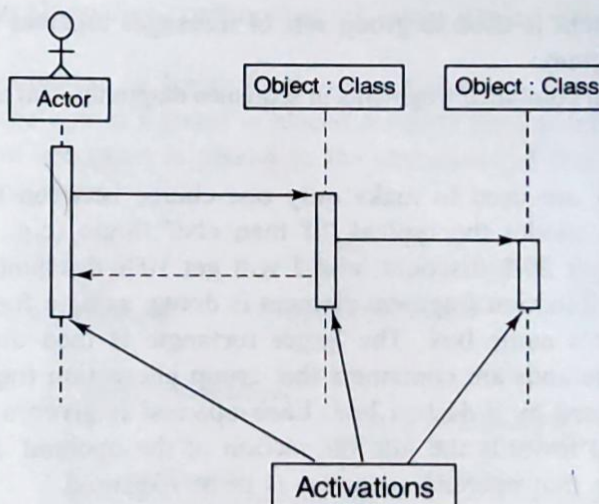


FIGURE 5.3 Activations.

### 5.2.4 Guards

Guards are the conditions used throughout UML diagrams to control flow. When modelling object interactions, there will be times when a condition must be met for a message to be sent to the object.



To display a guard on a sequence diagram, we place the guard element above the message line being guarded within square brackets and in front of the message name, e.g. `[balance < 100] debitCharges()` will invoke the method `debitCharges()` if `balance < 100` as shown in Figure 5.4.

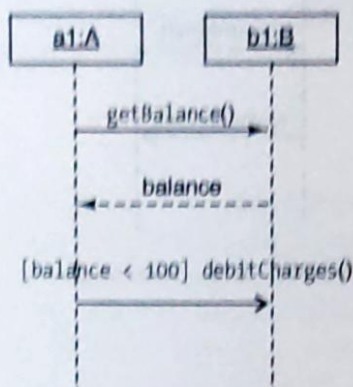


FIGURE 5.4 Guards.

### 5.2.5 Combined Fragments

Initially, before UML 2, sequence diagrams were lacking in notation to show complex procedural logic. In UML 2 combined fragments notation was introduced for adding procedural logic to sequence diagrams.

A combined fragment is used to group sets of messages together to show a conditional flow in a sequence diagram.

There are three main combined fragments in sequence diagrams—(a) alternatives, (b) options, and (c) loops.

**(a) Alternatives:** They are used to make only one choice between two or more message sequences. Alternatives model the typical “if then else” logic (e.g., if I purchase above Rs. 10000, then I will get 20% discount; else I will get 10% discount).

An alternative combination fragment element is drawn using a frame. The word “alt” is placed inside the frame’s name box. The larger rectangle is then divided into interaction operands. Interaction operands are containers that group interaction fragments in a combined fragment and are separated by a dashed line. Each operand is given a guard to test against, and this guard is placed towards the top left section of the operand. If an operand’s guard condition is “true”, then that operand sequence is to be executed.

Apart from only “if then else” condition, alternative combination fragments can also have many alternative paths as required. To have one more alternative, an operand with that sequence’s guard and messages must be added. Alternatives are shown in Figure 5.5(a).

**(b) Options:** The option combination fragment models a sequence that will occur for a certain condition, otherwise, the sequence does not occur.

An option fragment models a simple “if then” statement (i.e. if a customer purchases above Rs. 5000, give 10% discount, otherwise no discount). The option combination fragment

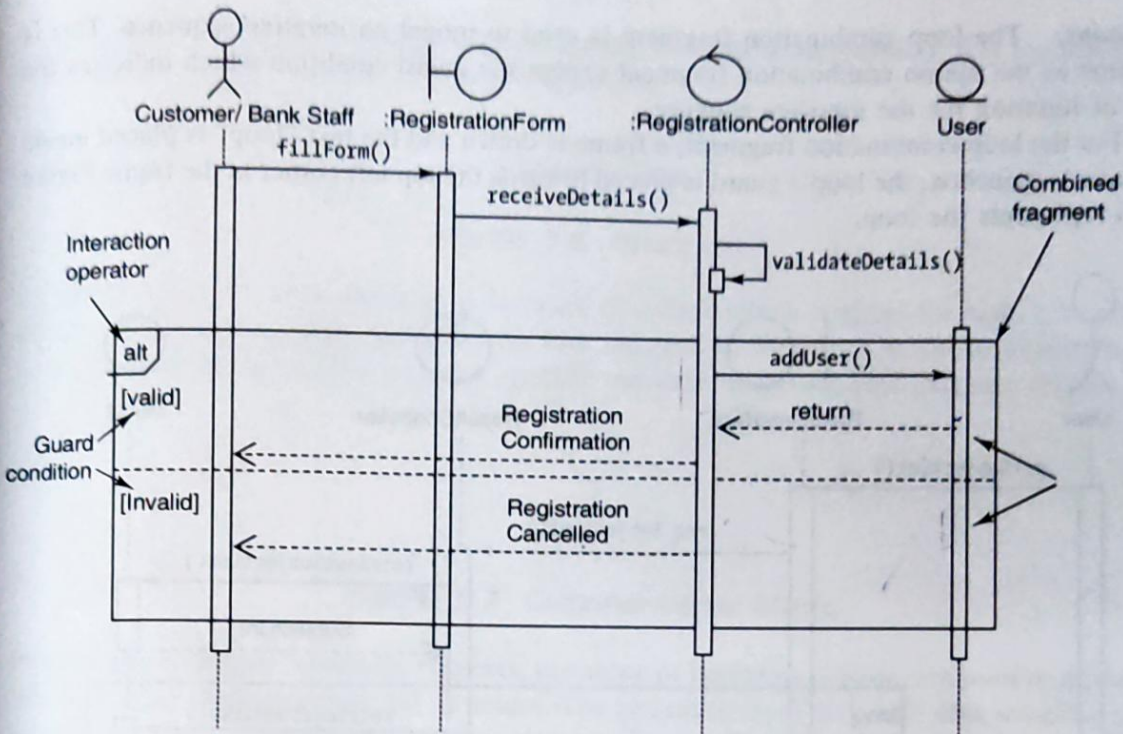


FIGURE 5.5(a) Representing alternatives.

notation is similar to the alternative combination fragment having only one operand and no other condition.

For an option combination, a frame is drawn and the text "opt" is placed inside the frame's name box, and the option's guard is placed towards the top left corner of the frame. The option's sequence of messages is placed in the remainder of the frame's content area. Figure 5.5(b) represents options.

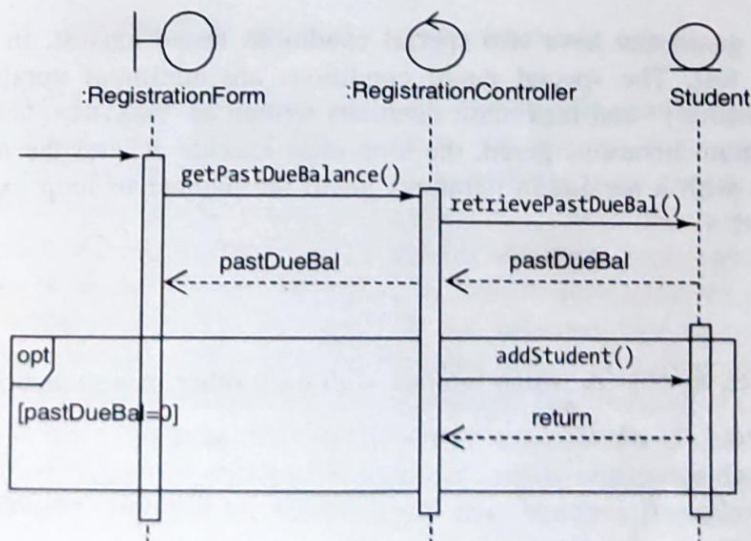


FIGURE 5.5(b) Representing options.



(c) **Loops:** The loop combination fragment is used to model an iterative sequence. This is the same as the option combination fragment except the guard condition which indicates the basis of iteration for the message sequence.

For the loop combination fragment, a frame is drawn and the text "loop" is placed inside the frame's namebox, the loop's guard is placed towards the top left corner in the frame Figure 5.5(c) represents the loop.

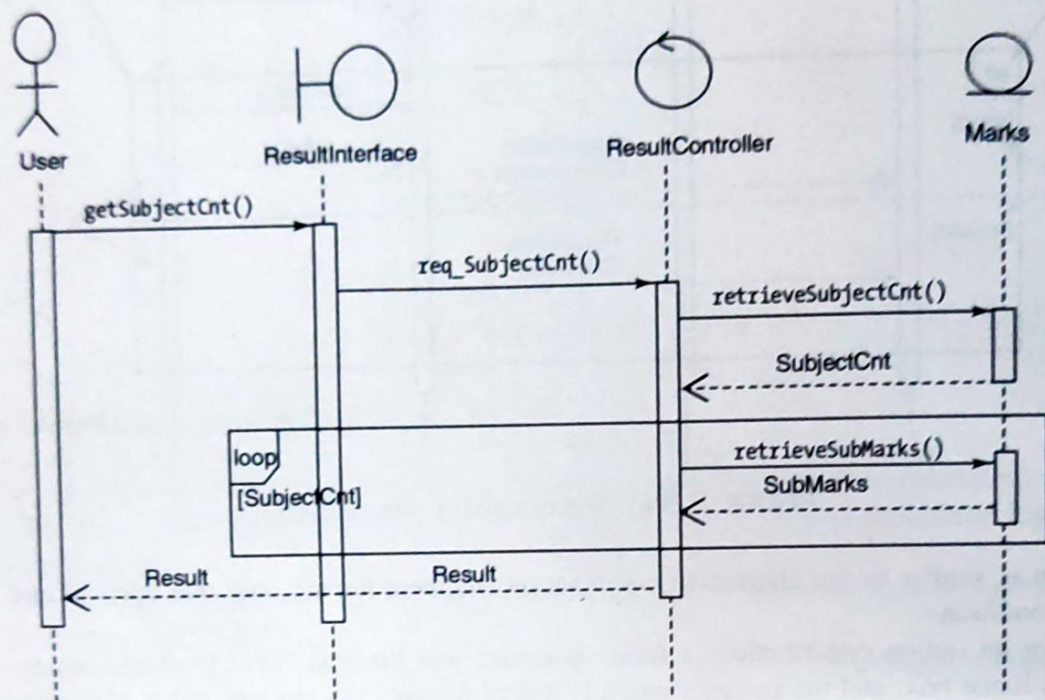


FIGURE 5.5(c) Representing loop.

In a loop, a guard can have two special conditions tested against, in addition to the standard Boolean test. The special guard conditions are minimum iterations written as "minint=[the number]" and maximum iterations written as "maxint=[the number]".

With a minimum iterations guard, the loop must execute at least the number of times indicated, whereas with a maximum iterations guard the number of loop executions cannot exceed the number.

### 5.2.6 Objects

There are four types of objects which interact with each other in a sequence diagram.

- Actor object
- Boundary object
- Controller object
- Entity object

If any other type of objects is to be shown, then the notation used, is a rectangle with *object name : class name* as shown in Figure 5.6.

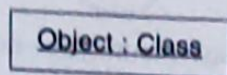


FIGURE 5.6 Object icon.

(a) **Actor object:** Actor object is an instance of a class which initiates the task. It represents an external person or entity that interacts with the system. The actor in the sequence diagram is the same actor interacting with that specific use-case in the use-case diagram (Figure 5.7).



FIGURE 5.7 Customer—actor object.

(b) **Boundary object:** Boundary objects, instances of boundary classes, are used to model the interaction between a system and its actors. The boundary class may be a user interface class, system interface class or device interface class. A user interface class is used for communication of human users with the system.

A system interface class is used for communication of other systems with the system. A device interface class provides the interface to device.

e.g. In a web application, the user interface boundary classes represent the web pages such as in ASP.NET, \*.aspx pages (Figure 5.8).

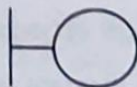


FIGURE 5.8 Application form—boundary object.

(c) **Controller object:** Controller objects are used to model the control behaviour specific to one or a few use-cases. Controller objects represent the use-case logic and coordinates the other classes. It separates the interface classes from business logic classes.

In ASP.NET the code-behind classes can be regarded as controller classes. Each controller object is closely related to the realization of a specific use-case. Some controller objects can participate in more than one use-case realization if the use-case tasks are strongly related.

Several controller objects can participate in one use-case. Not all use-cases require a control object. Some use-cases may be without controller objects (just using entity and boundary objects)—particularly use-cases that involve only the simple manipulation of stored information.

More complex use-cases generally require one or more control classes to coordinate the behaviour of other objects in the system. Examples of control objects include programs, such as transaction managers, resource coordinators and error handlers (Figure 5.9).





FIGURE 5.9 Login controller—controller object.

(d) **Entity object:** An entity object is used to model information and associated behaviour that must be stored. Entity objects (Figure 5.10) are used to hold and update information about some phenomenon, such as an event, a person or some real-life object.

They are usually persistent, having attributes and relationships needed for a long period, sometimes for the life of the system and can be used to generate database schema directly. An actor often provides the values of its attributes and relationships. Entity objects represent the key concepts of the system being developed. Typical examples of entity classes in a banking system are, the Account and Customer.

Entity objects represent stores of information in the system. Entity objects are frequently passive and persistent. Their main responsibilities are to store and manage information in the system (Figure 5.10).



FIGURE 5.10 User—entity object.

### 5.3 GUIDELINES FOR DESIGN OF SEQUENCE DIAGRAMS

The following are guidelines for the design of sequence diagrams.

1. For each use-case, one sequence diagram is drawn.
  - For each use-case, identify the object (person, external system, or device) which initiates the task and that will be the actor for a sequence diagram.
  - Find out the interface (form or screen) through which an actor interacts with the system and that will be the boundary object for a sequence diagram.
  - Find out the object which does not have the interface and controls the use-case and that will be the controller object for sequence diagram.
  - Find out the object which stores and manages information, usually the database table and that will be the entity object for sequence diagram.
2. Always have one boundary class per actor/use case pair, one control class per use-case.
3. Ordering of message sequence is always shown from left to right.
4. An actor name must be the same as specified in a use-case diagram.
5. An actor can have the same name as a class.
6. Primary actors must be specified on the left-most side of the diagram.
7. Reactive system actors must be specified on the right-most side of your diagram.



8. Proactive system actors must be specified on the left-most side of your diagram.
9. An object can call itself recursively. An arrow commencing and ending at itself denotes this.

## 5.4 PROBLEM STATEMENT: MILTON JEWELS PVT. LTD.

Milton Jewels has specialized in online jewellery retail since 1998, selling wonderful ranges of both children's and women's jewellery. A customer can register online so that he/she can check the status of the placed order. A customer can purchase any jewellery item online either by using his/her existing account or as an anonymous user specifying shipping address and contact information. Customer can only check the status of his/her order if he/she creates an account. The customer will pay online through credit card or debit card and the order will be delivered on the shipping address within one week.

### 5.4.1 Analysis of Milton Jewels Pvt. Ltd.

A sequence diagram needs to be drawn during the design phase of the system.

During analysis, we have got:

- Use-case diagram
- Class diagram

We will make use of both these diagrams for design.

- For each use-case in the use-case diagram, we will draw one sequence diagram at least. If the use-case is much complex, then there can be more than one sequence diagram.
- While drawing the sequence diagram from the use-case diagram, we will draw the sequence diagram only for the base use-cases embedding the flow for <<include>> and <<extend>> use-cases in the same.

In the above case, actors and use-cases identified are summarized in Table 5.1.

TABLE 5.1 Actors and their corresponding use-cases—Milton Jewels Pvt. Ltd.

<i>Sr. No.</i>	<i>Actors</i>	<i>Use-cases</i>
1	Customer	Registration Login Purchase jewellery Check order status Search jewellery View jewellery
2	Store Manager	Registration Login Add/Edit/Remove jewellery Add/Edit/Remove Categories Specify discounts



- During design, identify four types of objects interacting with each other to perform the use-case such as:
  1. Actor objects—Person, external system or device that initiates the use case, e.g. Customer, Store Manager.
  2. Boundary objects—All the interfaces through which the actor interacts with the system, e.g. Login Screen, Application Form, etc.
  3. Controller objects—All the objects which co-ordinate the task, e.g. Security Manager, etc.
  4. Entity objects—All domain entities identified during analysis of the class diagram, e.g. Users, products etc.

#### 5.4.2 Sequence Diagrams: Milton Jewels Pvt. Ltd.

From the identified use-cases and actors as above, consider use-case Login performed by both actors, the Customer and Stores Manager, which is the most common functionality in nearly every application. For the Login use-case, ask the following four questions and answers to those questions will be the objects interacting with each other for login process.

1. Who will login? (Actor)  
Customer/Store Manager
2. Through which web page (interface)? (Boundary)  
Login Screen
3. Who will validate the user? (Control)  
Security Manager
4. Which object holds valid user details? (Entity)  
Users

After identifying interacting objects, it is required to find out what sequences of message communication happen among them. Message communication occurs through method calls, also listed in Table 5.2.

**TABLE 5.2** Classes their types and methods—Milton Jewels Pvt. Ltd.

<i>Class Type</i>	<i>Classes</i>	<i>Methods</i>
Actor	Customer	<i>Not required to specify in this context.</i>
Boundary	LoginScreen	login()
Control	SecurityManager	validateUser()
Entity	Users	retrieveUserDetails

Figure 5.11 shows one sequence diagram corresponding to the use-case for login utility. There will be many more such sequence diagrams corresponding to the total number of use-cases and depending upon the complexity of the use-cases. The other sequence diagrams are not shown here and left to the reader as an exercise.



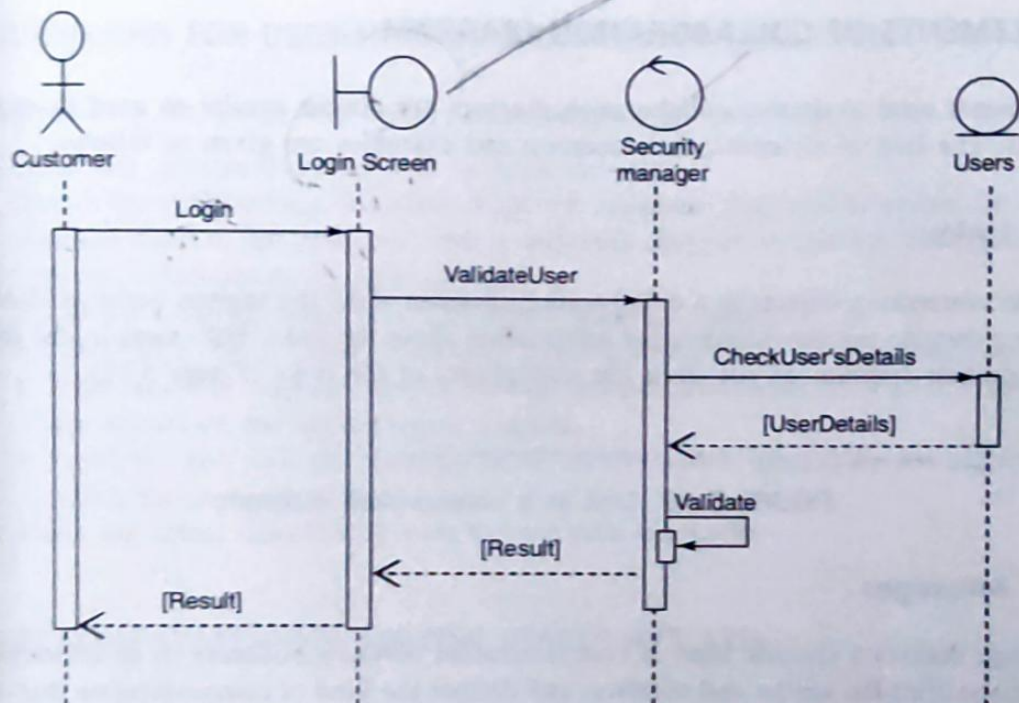


FIGURE 5.11 Sequence diagram of Milton Jewelers Pvt. Ltd. for Login Use-Case.

## 5.5 COLLABORATION DIAGRAM

A Collaboration diagram is a kind of interaction diagram, called as Communication diagram in UML 2.0. They describe the interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use-case diagrams describing both the static structure and dynamic behaviour of a system.

Like UML sequence diagrams, they are used to explore the dynamic nature of the system which emphasizes the data links between the various participants in the interaction. Instead of drawing each participant as a lifeline and showing the sequence of messages by the vertical direction as the sequence diagram does, it allows free placement of participants, allowing drawing links to show how the participants connect, and use numbering to show the sequence of messages.

Unlike a sequence diagram, a collaboration diagram shows the relationships among the objects. A collaboration diagram does not show time as a separate dimension, so sequence numbers determine the sequence of messages and the concurrent threads.

Collaboration diagrams show the message flow as well as relationships between objects and are often used to provide a bird's-eye view of a collection of collaborating objects. These diagrams are used to model the logic of the implementation of a complex operation, particularly one that interacts with a large number of other objects. They are better to draw when you want to emphasize the links between the objects, whereas sequence diagrams are better when you want to emphasize the sequence of calls.



## 5.6 ELEMENTS OF COLLABORATION DIAGRAM

The elements used to draw a collaboration diagram are almost similar as used in sequence diagrams. The lists of elements, their notation and examples are given as follows.

### 5.6.1 Links

The links connecting objects in a collaboration diagram show the various paths available for messages; they do not provide detailed information about the links. For example, the links in a collaboration diagram do not show the multiplicity of the links (Figure 5.12).




FIGURE 5.12 Link in a collaboration diagram.

### 5.6.2 Messages

A message defines a specific kind of communication between instances in an interaction. A message specifies the sender and receiver, and defines the kind of communication that occurs between lifelines.

A message flow carries a message from one object to another along the link. Each message flow in a collaboration diagram is characterized by direction and is numbered starting with 1. The number with the message represents the order/sequence of this interaction (Figure 5.13).

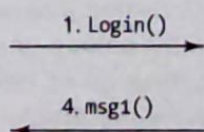


FIGURE 5.13 Messages in a collaboration diagram.

### 5.6.3 Objects

There are four types of objects which interact with each other in a collaboration diagram very similar to sequence diagram and already explained in section 5.2.6.

- (a) Actor objects
- (b) Boundary objects
- (c) Controller objects and
- (d) Entity objects

If any other type of objects is to be shown, then the notation used is a rectangle as shown in Figure 5.6.



## 5.7 GUIDELINES FOR DESIGN OF COLLABORATION DIAGRAMS

The guidelines for the design of a collaboration diagram are as follows:

1. Draw one collaboration diagram for each use-case.
2. Derive the collaboration diagrams from the sequence diagrams or create the object message trace in the same way that a sequence diagram is created, but represent it with the collaboration diagram notation.
3. To derive a collaboration diagram from a sequence diagram,
  - Draw each object (actor, boundary, control, entity) from the sequence diagram.
  - If the sequence diagram shows a message between the objects, draw a line connecting the objects on the collaboration diagram.
  - Label the line with the message name and a number identifying the sequence in which the message appears.
4. Name the actors consistently with the use case diagrams.

## 5.8 PROBLEM STATEMENT: MILTON JEWELS PVT. LTD.

Milton Jewels has specialized in online jewellery retail since 1998, selling wonderful ranges of both children's and women's jewellery. A customer can register online so that he/she can check the status of the placed order. A customer can purchase any jewellery item online either by using his/her existing account or as an anonymous user specifying shipping address and contact information. The customer can only check the status of his/her order if he/she creates an account. The customer will pay online through credit card or debit card and order will be delivered on the shipping address within one week.

### 5.8.1 Analysis of Milton Jewels Pvt. Ltd.

Since the sequence diagram of Milton Jewellers Pvt. Ltd. is drawn the collaboration diagram is easy to derive. Only one sample collaboration diagram is shown as follows. The rest of the collaboration diagrams are left for the reader as an exercise.

### 5.8.2 Collaboration Diagram: Milton Jewels Pvt. Ltd.

In the first part of this chapter, we have a sequence diagram ready for one of the several use-cases performed in the system, i.e., sequence diagram for Login use-case. We will derive the collaboration diagram from the sequence diagram (Figure 5.14). The objects, links and related methods are shown in Table 5.3.



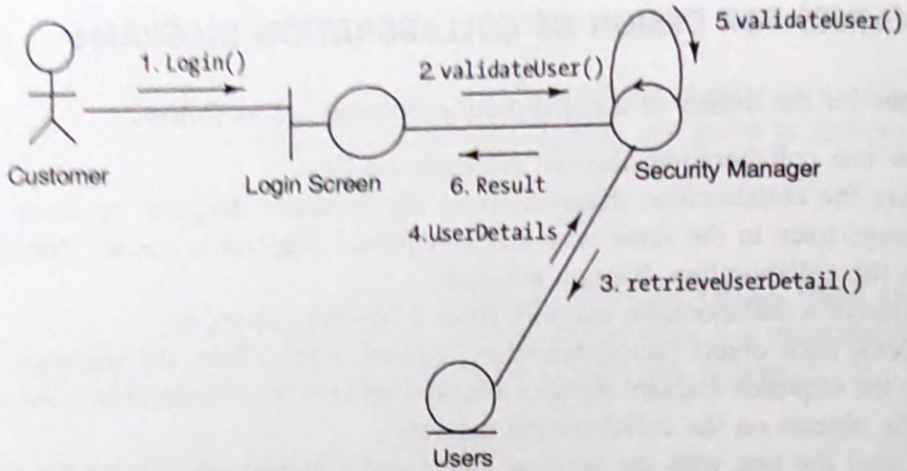


FIGURE 5.14 Collaboration diagram—Milton Jewels Pvt. Ltd. for Login use-case.

TABLE 5.3 Objects, links and methods—Milton Jewels Pvt. Ltd.

<i>Objects</i>	<i>Links</i>	<i>Methods (with sequence number)</i>
Customer	Customer—LoginScreen	1. Login()
LoginScreen	LoginScreen—SecurityManger	2. validateUser() 6. Result
SecurityManager	SecurityManager—SecurityManager	5. validatUser()
Users	SecurityManager—Users	3. retrieveUserDetail() 4. UserDetails

## EXERCISES

1. Differentiate between a sequence diagrams and a collaboration diagram.
2. Explain the different types of objects in a sequence diagram.
3. Refer to section 5.4 and 5.8, and draw the sequence diagrams and collaboration diagrams for the remaining use-cases.
4. Draw the sequence and collaboration diagrams to withdraw money from a bank.
5. Draw the sequence and collaboration diagrams to deposit money into a bank.
6. Draw the sequence and collaboration diagrams for shopping cart application.
7. Draw the sequence and collaboration diagrams for an online Air Ticket Reservation System.
8. Draw the sequence and collaboration diagrams for a Hotel Reservation System.



9. The main function of a self-service machine is to allow a customer to buy a product from the machine (candy, chocolate, juice ...).

The customer wants to buy some of the products offered by the self-service machine. First of all he/she inserts money into the machine, selects one or more products, and the machine presents a selected product to the customer. It is possible that the self-service machine is out of one or more products, or the machine has not the exact amount of money to return to the customer.

When the customer inserts money into the machine and enters his or her selection, if the machine is out of brand, in this case it's preferable to present a message to the customer that the machine is out of brand and allow him or her to make another selection or return money back. If incorrect-amount-of-money scenario has happened, then the self-service machine is supposed to return original amount of money to the customer. A supplier has to restock the machine, and a collector has to collect the accumulated money from the machine. Draw the sequence diagram.

10. A new Student Registration System is to be developed which students could use to register for courses at the beginning of each semester and view grade report cards for courses at the end of each semester. The system would also let professors indicate courses they would be teaching in a semester and see students who have signed up for their course offerings. Professors would also be able to enter student grades into the system.

The school has an existing system used to maintain course catalogue information, such as course titles, prerequisites, weekly meeting times and locations. The registration system would use the course catalogue data for its purposes. Also, after the registration period's completion, the new registration system must send a notification to the school's billing system so that each student can be appropriately billed.

At the semester's beginning, the system lets a student select a maximum of 18 credits—six primary courses. Further, each student can indicate two alternative three-credit courses that could be substituted for primary course selections. If a course fills up while a student is registering, the system must notify the student of the course's unavailability before it accepts the schedule for processing.

A student could access the system to add or delete courses before the add/delete period expires. Once the system completes a student's registration, it notifies the school's billing system. The registrar uses the system to maintain student information, close registration, and cancel course offerings with fewer than five students. Draw the sequence and collaboration diagrams for the system.