

Unit 1: Transition Analysis to Design using Interaction Diagrams

Most Essential Learning Competencies: *At the end of the course, you must be able to:*

1. Discuss and define communication diagram
2. Explain the elements of communication diagram
3. Create an example of UML communication diagram



Reading Activity

UML Communication Diagrams Overview

Communication diagram (called **collaboration diagram** in UML 1.x) is a kind of UML interaction diagram which shows interactions between objects and/or parts (represented as lifelines) using sequenced messages in a free-form arrangement.

Communication diagram corresponds (i.e. could be converted to/from or replaced by) to a simple sequence diagram without structuring mechanisms such as interaction uses and combined fragments. It is also assumed that **message overtaking** (i.e., the order of the receptions are different from the order of sending of a given set of messages) will not take place or is irrelevant.

Purpose of Communication Diagram

- Model message passing between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the passed messages between objects and roles within the collaboration scenario
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes), and their attributes (parameters of message) and operations (messages) that participate in use cases

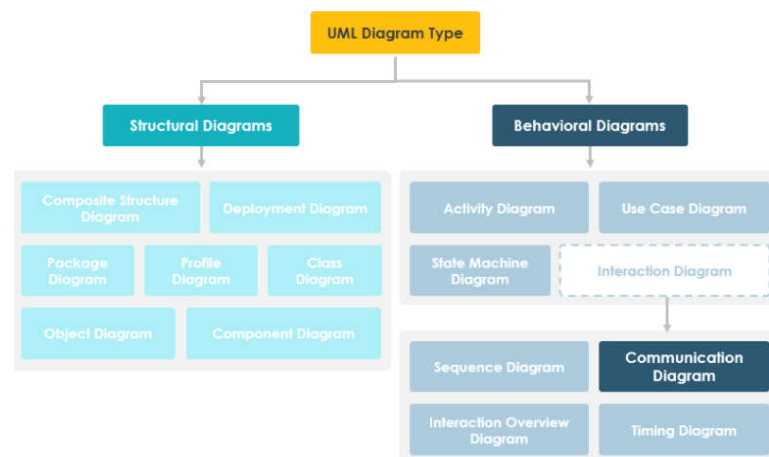


Figure 4.0: UML Diagram Type

The following nodes and edges are drawn in a UML communication diagrams: **frame**, **lifeline**, **message**. These major elements of the communication diagram are shown on the picture below.

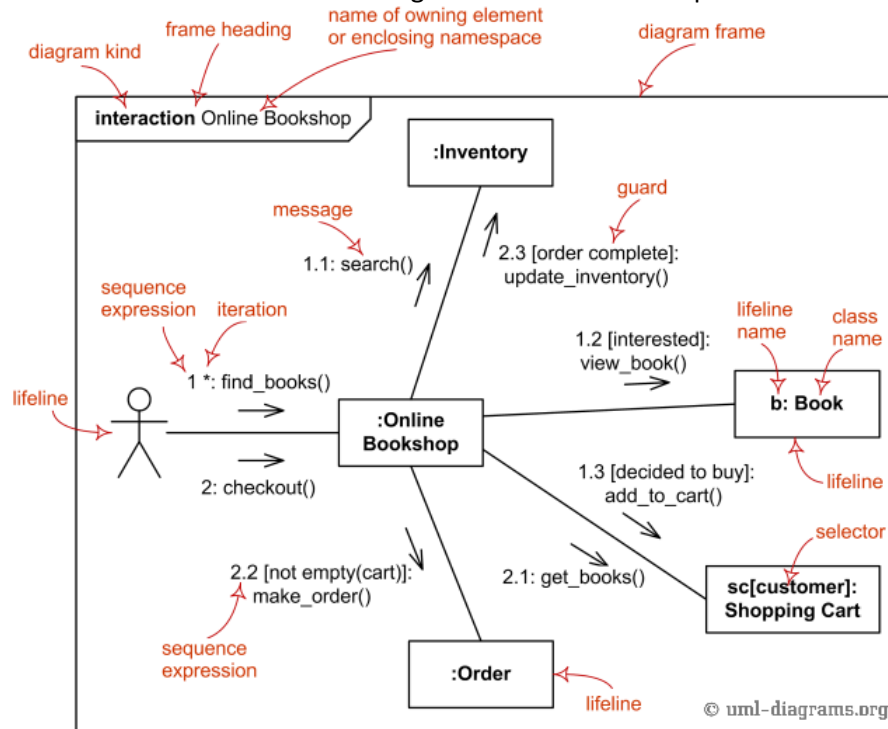


Figure 4.1: The major elements of UML communication diagram.

Frame

Communication diagrams could be shown within a rectangular frame with the name in a compartment in the upper left corner.

There is no specific long form name for communication diagrams heading types. The long form name interaction (used for interaction diagrams in general) could be used.

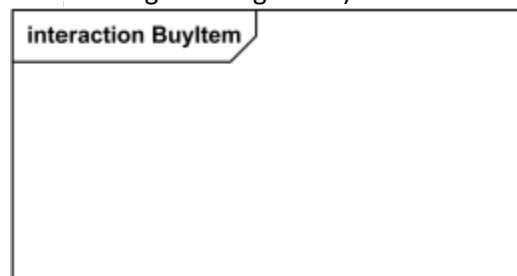


Figure 4.2: Interaction Frame for Communication Diagram BuyItem

There is also no specific short form name for communication diagrams. Short form name sd (which is used for interaction diagrams in general) could be used. This sd is bit confusing as it looks like abbreviation of sequence diagram.

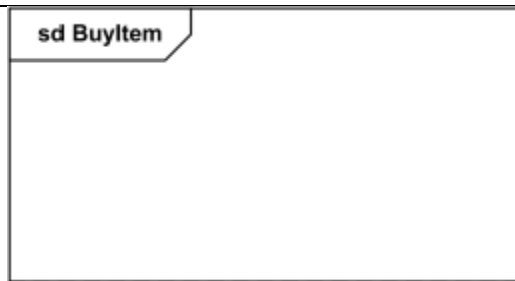


Figure 4.3: sd Frame for Communication Diagram BuyItem

Lifeline

Lifeline is a specialization of named element which represents an **individual participant** in the interaction. While parts and structural features may have multiplicity greater than 1, lifelines represent **only one** interacting entity.

If the referenced connectable element is multivalued (i.e, has a multiplicity > 1), then the lifeline may have an expression (**selector**) that specifies which particular part is represented by this lifeline. If the selector is omitted, this means that an **arbitrary representative** of the multivalued connectable element is chosen.

A **Lifeline** is shown as a rectangle (corresponding to the "head" in sequence diagrams). Lifeline in sequence diagrams does have "tail" representing the **line of life** whereas "lifeline" in **communication diagram** has no line, just "head".

Information identifying the lifeline is displayed inside the rectangle in the following format:

lifeline-ident ::= ([connectable-element-name ['[' selector ']']] [: class-name] [decomposition]) | 'self'

selector ::= expression

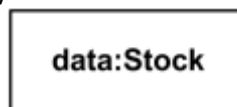
decomposition ::= 'ref' interaction-ident ['strict']

where class-name is type referenced by the represented connectable element. Note that, although the syntax allows it, lifeline-ident cannot be empty.

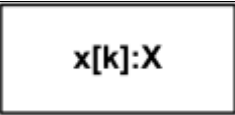
The lifeline head has a shape that is based on the classifier for the part that this lifeline represents. Usually the head is a white rectangle containing name of the class after colon.



Anonymous lifeline of class User.



Lifeline "data" of class Stock



x[k]:X

Lifeline "x" of class X is selected with **selector** [k].

Figure 4.4: Illustrates a lifeline.

If the name is the keyword **self**, then the lifeline represents the object of the classifier that encloses the Interaction that **owns** the Lifeline. **Ports** of the enclosure may be shown separately even when self is included.

Message

Message in **communication diagram** is shown as a line with sequence expression and **arrow** above the line. The arrow indicates direction of the communication.

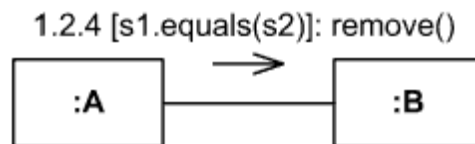


Figure 4.5: Instance of class A sends remove() message to instance of B if s1 is equal to s2

Sequence Expression

The **sequence expression** is a dot separated list of **sequence terms** followed by a colon (":") and message name after that:

sequence-expression ::= sequence-term '.' ... ':' message-name

For example,

3b.2.2:m5

contains sequence expression 3b.2.2 and message name m5.

Each sequence term represents a level of procedural nesting within the overall interaction. Each sequence-term has the following syntax:

sequence-term ::= [integer [name]] [recurrence]

The integer represents the sequential order of the message within the next higher level of procedural calling (activation). Messages that differ in one integer term are sequential at that level of nesting.

For example,

- message with sequence 2 follows message with sequence 1,
- 2.1 follows 2
- 5.3 follows 5.2 within activation 5
- 1.2.4 follows message 1.2.3 within activation 1.2.

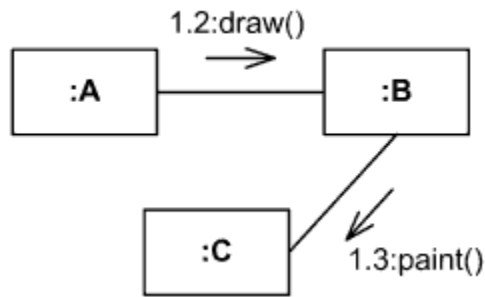


Figure 4.6: Instance of A sends draw() message to instance of B, and after that B sends paint() to C

The name represents a concurrent thread of control. Messages that differ in the final name are concurrent at that level of nesting.

For example,

- messages 2.3a and 2.3b are concurrent within activation 2.3,
- 1.1 follows 1a and 1b,
- 3a.2.1 and 3b.2.1 follow 3.2.

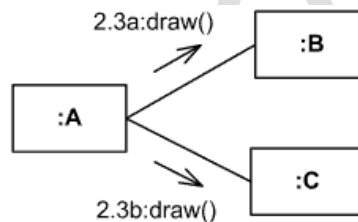


Figure 4.7: Instance of A sends draw() messages concurrently to instance of B and to instance of C

The recurrence defines conditional or iterative execution of zero or more messages that are executed depending on the specified condition.

recurrence ::= branch | loop

branch ::= '[' guard ']

loop ::= '*' ['|'] ['[' iteration-clause ']']

A guard specifies condition for the message to be sent (executed) at the given nesting depth. UML does not specify guard syntax, so it could be expressed in pseudocode, some programming language, or something else.

For example,

- 2.3b [x>y]: draw() - message draw() will be executed if x is greater than y,
- 1.1.1 [s1.equals(s2)]: remove() - message remove() will be executed if s1 equals s2.

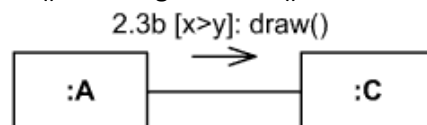


Figure 4.8 Instance of class A will send message draw() to the instance of C, if x > y

An iteration specifies a sequence of messages at the given nesting depth. UML does not specify iteration-clause syntax, so it could be expressed in pseudocode, some programming language, or something else. Iteration clause may be omitted, in which case the iteration conditions are unspecified.

The * iteration notation specifies that the messages in the iteration will be executed sequentially.

The *|| (star followed by a double vertical line) iteration notation specifies concurrent (parallel) execution of messages.

For example,

- 4.2c *[i=1..12]: search(t[i]) – search() will be executed 12 times, one after another
- 4.2c *||[i=1..12]: search(t[i]) – 12 search() messages will be sent concurrently,
- 2.2 *: notify() – message notify() will be repeated some unspecified number of times.

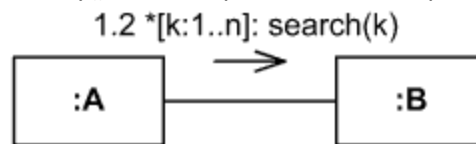


Figure 4.9: Instance of class A will send search() message to instance of B n times, one by one

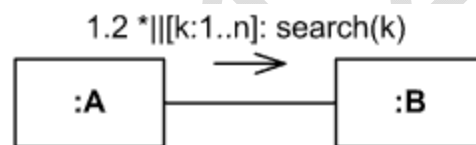


Figure 4.10: Instance of class A will send n concurrent search() messages to instance of B

Recurrence is not repeated at inner levels in a nested control structure. Each level of structure specifies its own iteration within the enclosing context.



Read Additional Resources

1. **UML Communication Diagram**
Week4_Communication Diagram.pdf
2. **UML Communication Diagram**
Week4_Communication Diagram Tutorial.pdf



Watch Video Resources

1. **UML Communication Diagram**
Week4_Communication Diagram.mp4
2. **UML Communication Diagram**
Week4_Communication Diagram2.mp4

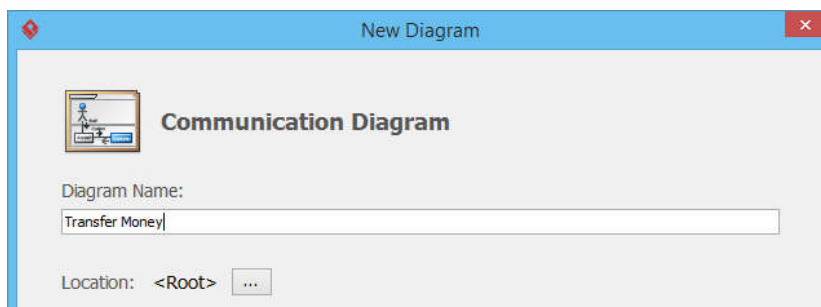


Laboratory Activity

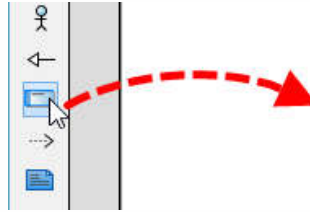
Laboratory Activity 4.1

Drawing a Communication Diagram (Using Visual Paradigm)

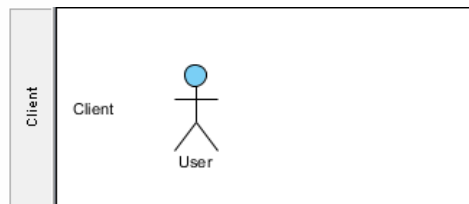
1. Create a new project. Name it as Online Banking.
2. To create a Communication Diagram, select Diagram > New from the toolbar.
3. In the New Diagram window, select Communication Diagram and click Next.
4. Enter Transfer Money as diagram name and click OK to confirm.



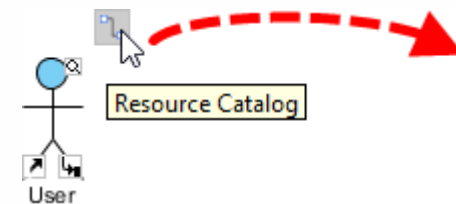
5. Add swimlanes to model the partitioning of system. Select Swimlane from diagram toolbar and click on the diagram to create it. Name it as Client.



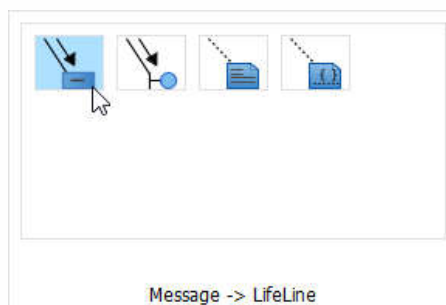
6. Repeat the previous step to create another swimlane Main frame below swimlane Client.
7. Start the story from the actor. Select Actor from the diagram toolbar. Click within the swimlane Client to create an actor, and name it as User.



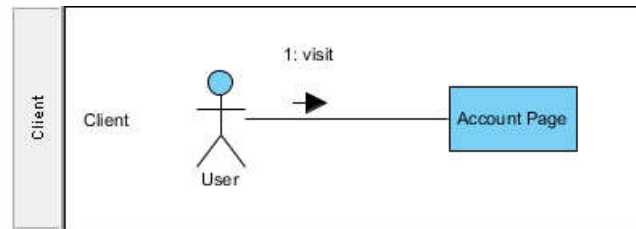
8. User will operate the system via the account page. To present this, we need to add a lifeline for Account Page, and link it up with the actor User. Move the mouse pointer over actor User. Drag out the Resource Catalog icon at top right.



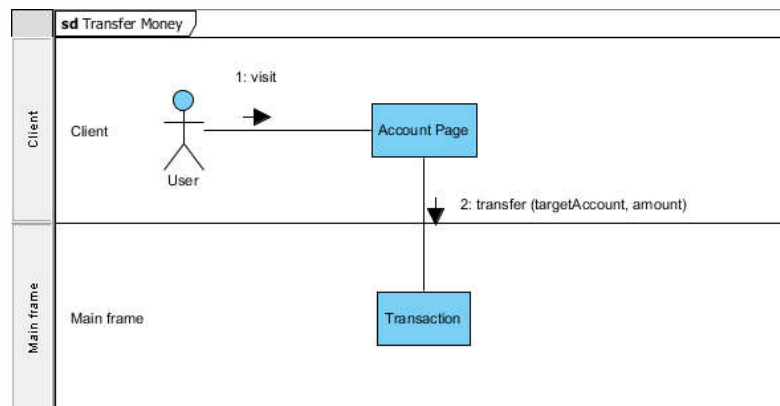
9. Select Message -> Lifeline from Resource Catalog. Name the lifeline Account Page.



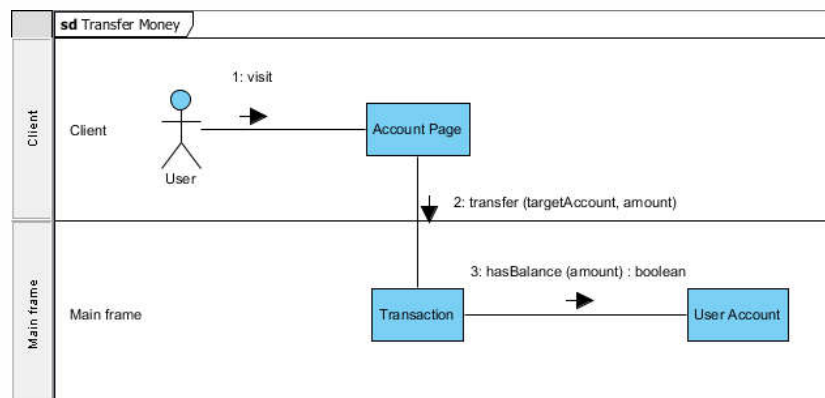
10. We want to model the message pass from user to account page. Double click on the arrow marked 1, enter visit, then click on the background of diagram to confirm the change. Note that the number next to the message, 1 in this case, represent the order of message flow within the interaction being modeled.



11. The account page will direct user's request of transferring money to the main frame for validation and execution. Again, move the mouse pointer over lifeline Account Page and make use of the Resource Catalog to create lifeline Transaction in swimlane Main frame.
12. Name message 2 as transfer (targetAccount, amount).

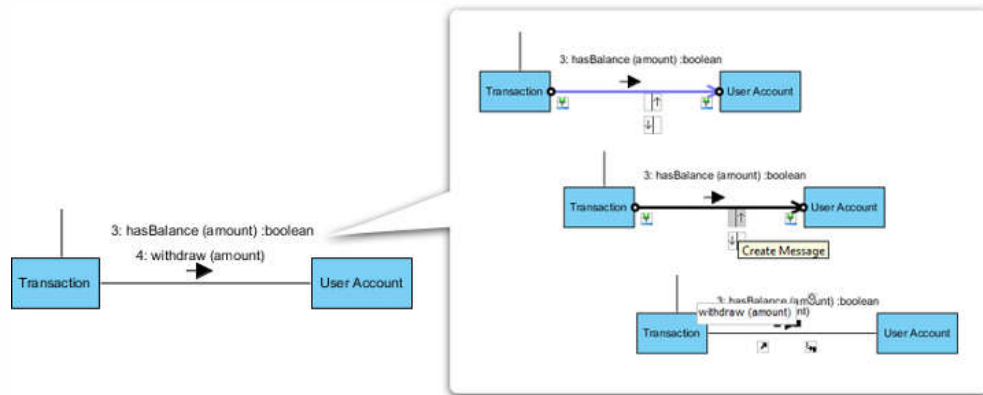


13. The money transferral process involve withdrawing money from user's account, and depositing money to target account. But before these, we need to make sure user's account has enough money to undergoing the transferral. Make use of the Resource Catalog to create a lifeline User Account from lifeline Transaction. Name message 3 as hasBalance (amount) : boolean.

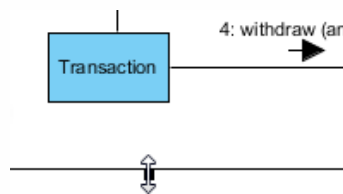


14. Once the account balance is checked, we can withdraw money from user's account. To add the 4th message between lifeline Transaction and User Account, move the mouse pointer over

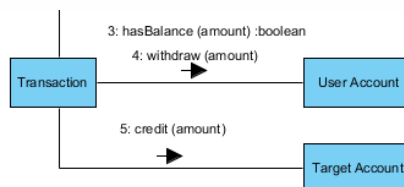
the link between Transaction and User Account, click on the resource icon Create Message and name the message as withdraw (amount).



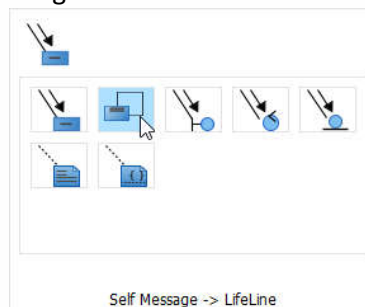
- Before we continue, expand the swimlane Main frame first. Select the header of swimlane Main frame, press on the resize handler at the bottom of swimlane and drag downwards.



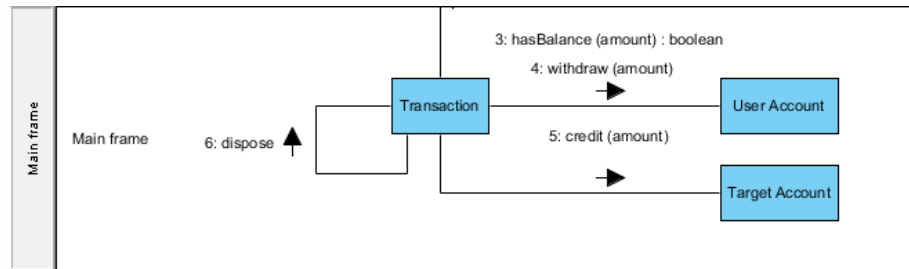
- Create a lifeline Target Account from lifeline Transaction. Name the 5th message as credit (amount).



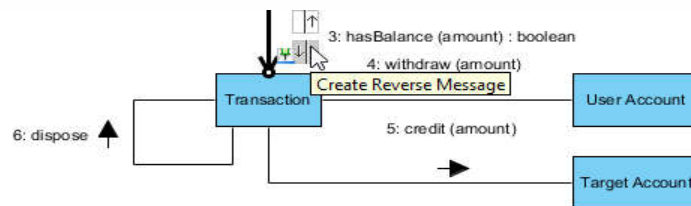
- The transaction is complete. Let's make the lifeline Transaction dispose itself. This time, move the mouse pointer over the lifeline Transaction, then click once on the Resource Catalog icon. Select Self Message -> Lifeline from the Resource Catalog.



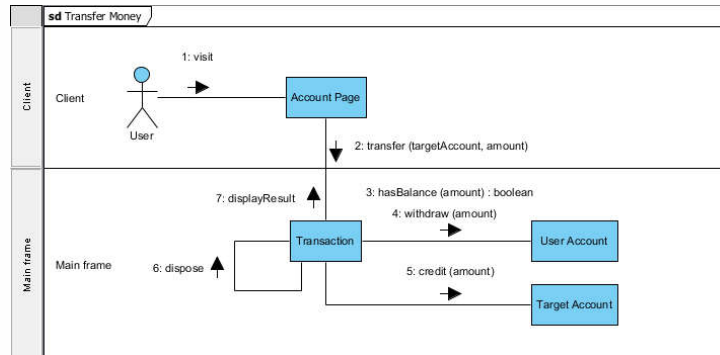
- Name the 6th message as dispose().



19. Finally, we shall display a message on screen, telling user that transaction is complete. Move the mouse pointer over the link between lifeline Account Page and Transaction. Click on the resource icon Create Reverse Message. Name the 7th message as displayResult().



Finally, the diagram should look like this:



20. Save the project by selecting Project > Save from the toolbar. This is the end of the tutorial.



Self-Check

Quiz 4.1

Instructions: Write your answer on the Answer Sheet (AS) provided in this module.

1. Explain what is communication diagram (5-points).
2. Enumerate the purpose of communication diagram (5-points).
3. Create an example of communication diagram in Java programming. (20-points).



Internet References

1. <https://www.youtube.com>
2. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-communication-diagram/>
3. <https://www.uml-diagrams.org/communication-diagrams.html>
4. <https://www.visual-paradigm.com/tutorials/how-to-draw-communication-diagram.jsp>
5. https://www.youtube.com/watch?v=TL4ABTx_RtE
6. <https://www.youtube.com/watch?v=8CBnAmYnwK0>