Sequence Diagram

Part II

Session Outcomes

- Interaction Occurrences
  - Ref Tag
  - Gates
- Combined Fragments
  - Interaction operators
    - Alt
    - Opt
    - Interaction Operands
    - Par
    - Break
    - Loop
    - Parallel
    - Weak Sequencing
    - Strict Sequencing
    - Critical

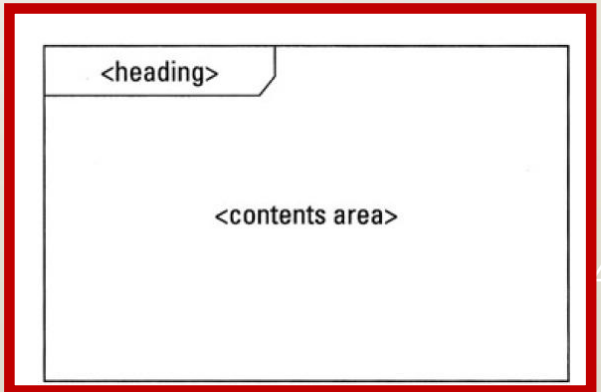
Interaction Occurrences

- In sequence diagrams, an interaction occurrence is an interaction that is referenced in another interaction frame. You can use interaction occurrences to take common content from one interaction and reuse it in another sequence diagram.
- An interaction occurrence may be used in any number of other contexts.
- To model the use of interaction occurrences we need to use the **frame concept**. Each interaction occurrence is represented by a frame with the name of the interaction.

Frame

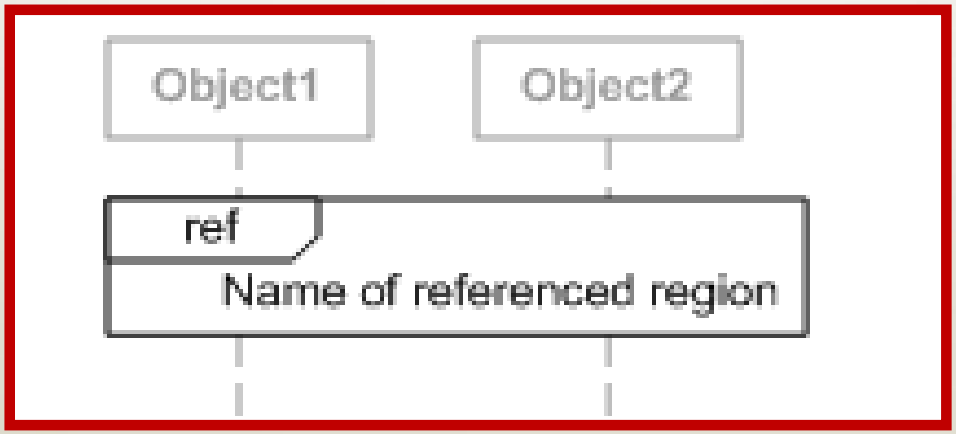
A diagram is enclosed in a frame, a boundary that consists of a *heading* and a *context area.*

- The <heading> is a string contained in a name tag, a rectangle with a cutoff corner.
- The <heading> is placed in the upper left corner of the frame and uses the following syntax:
- [<kind>]<name>[<parameters>]
- **Kind :** is optional and refers to the type of diagram; for example, the keyword sd represents a Sequence diagram, and cd represents a Class diagram.
- **Parameters:** are also optional. They list the data values passed into the frame context area when the frame is invoked.
- **Contents area** : contains the diagram.



Ref Tag

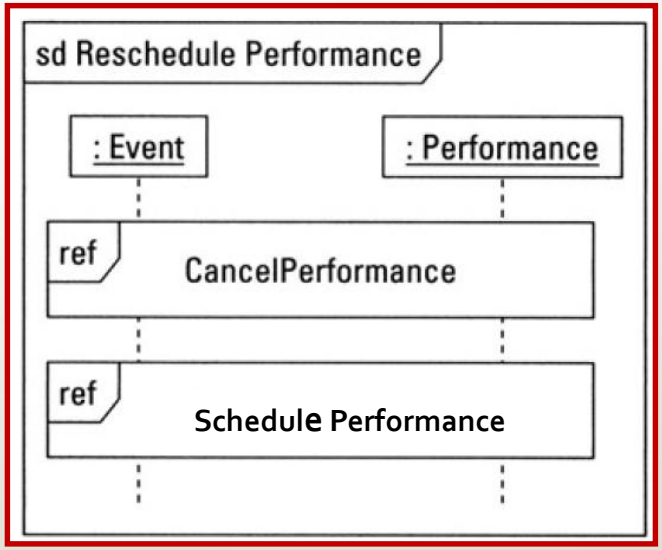
- If one sequence diagram is too large or refers to another diagram, 'ref' tag can be used.
- the keyword ref is placed in the top left name area of the frame. This alerts the reader that the content of the frame is defined elsewhere, that is, in the diagram named in the content area.
- (A reference to a named region -elsewhere in diagram or on another diagram).



Ref Tag

- Sequence diagram *sd Reschedule Performance*, which consists of two consecutive interaction occurrences, in a frame.
- That is, in order to reschedule a performance, the system first cancels the performance, and then schedules a new performance.

- Using this technique, the modeler could build a number of scenarios
- very quickly by reusing common portions of scenarios to build other scenarios.



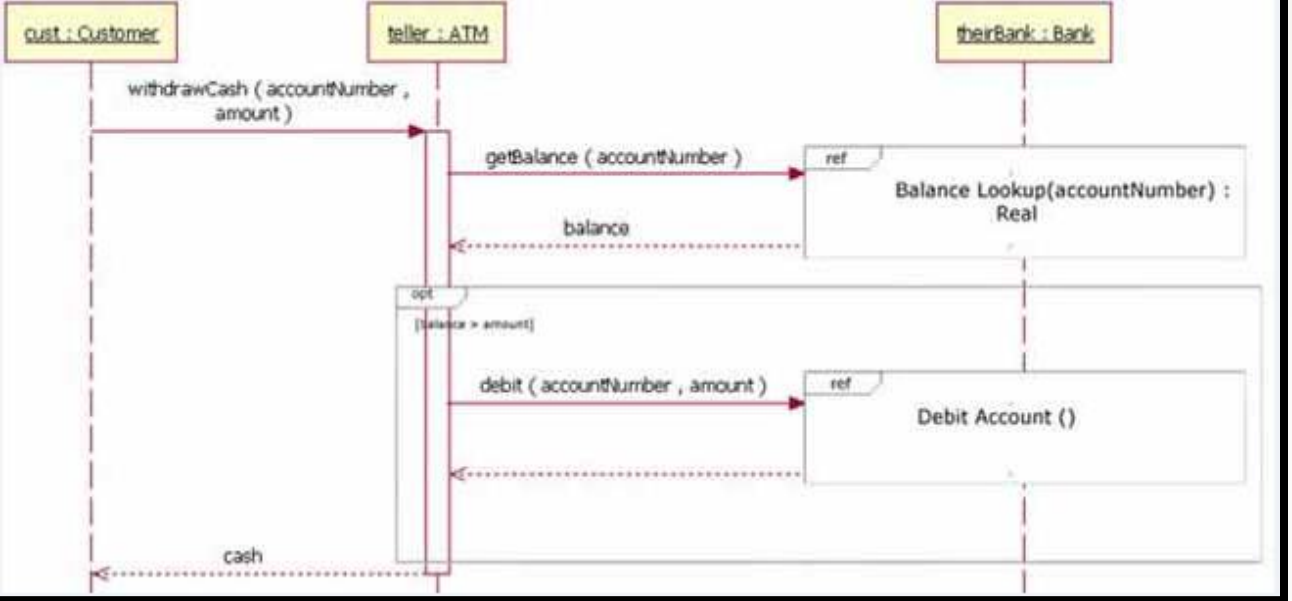
Using two interaction occurrences to construct a larger interaction/Sequence diagram.

Gates

- There is another way to pass information between sequence diagrams. Gates can be an easy way to model the passing of information between a sequence diagram and its context.

- A gate is merely a message that is illustrated with one end connected to the sequence diagram's frame's edge and the other end connected to a lifeline.

- A **gate** is a **message end**, connection point for relating a **message** outside of an **interaction fragment** with a message inside the interaction fragment.



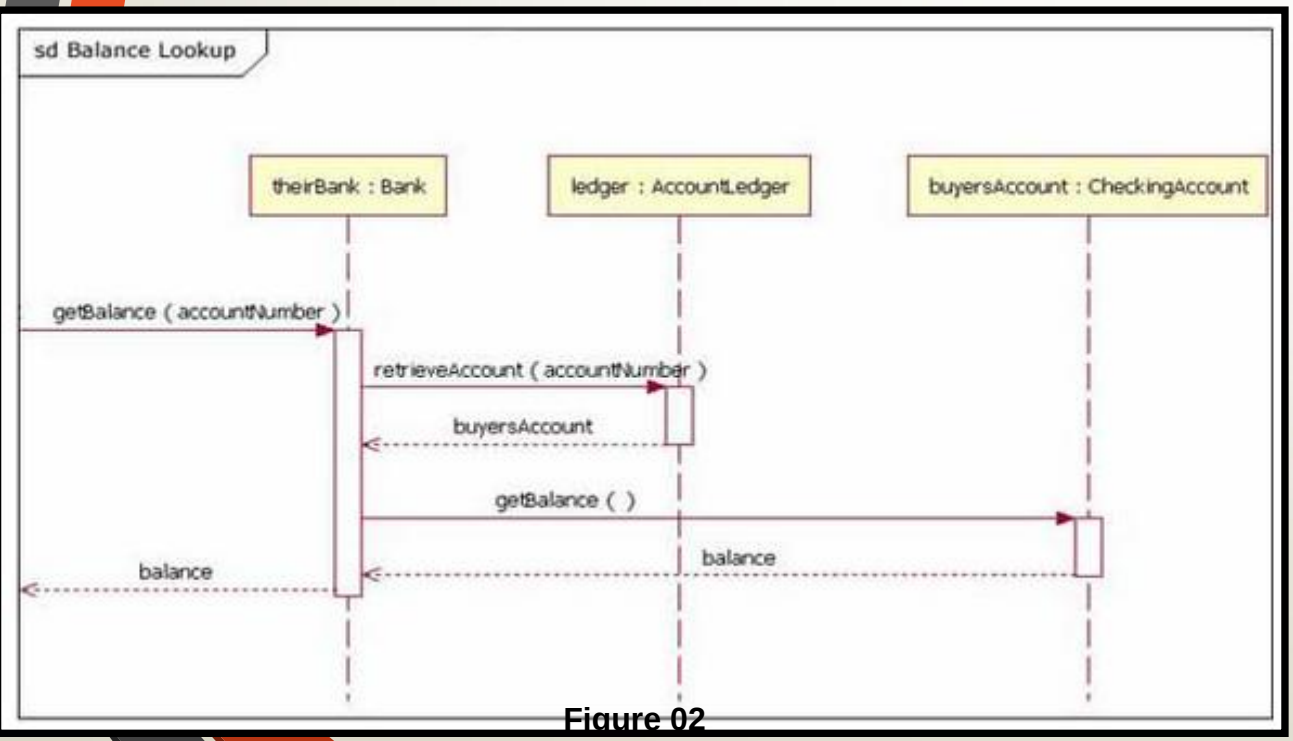


Figure 02

- The example diagram in Figure 02 has an entry gate called getBalance that takes the parameter of accountNumber.
- The getBalance message is an entry gate, because it is the arrowed line that is connected to the diagram's frame with the arrowhead connected to a lifeline.
- The sequence diagram also has an exit gate that returns the balance variable.
- The exit gate is known, because it's a return message that is connected from a lifeline to the diagram's frame with the arrowhead connected to the frame.

http://www.ibm.com/developerworks/rational/library/3101.html

Activity 1

- Draw a sequence diagram for the following scenario using Frame and ref tag.
  - A **Customer** object invokes the method of *onInsertCard()* of the **ATM** object.
  - **ATM** has to check whether the card inserted is a stolen card. For this, **ATM** object invokes the *isStolen* method of **Server** object passing, the card details.
  - The **Server** object validates the card and returns a boolean value.
- Use the gates for the following partial sequence diagram.
  - When validating the card, **Server** object calls it's own *connect()* method.
  - On successful connection, **Server** invokes *verifySecurity()* of **MainServer** object which validates that the card is not a stolen card.

Interaction Operators

- The Interaction Operator is modeled as text in a small compartment in the upper left corner of the Combined Fragment frame.

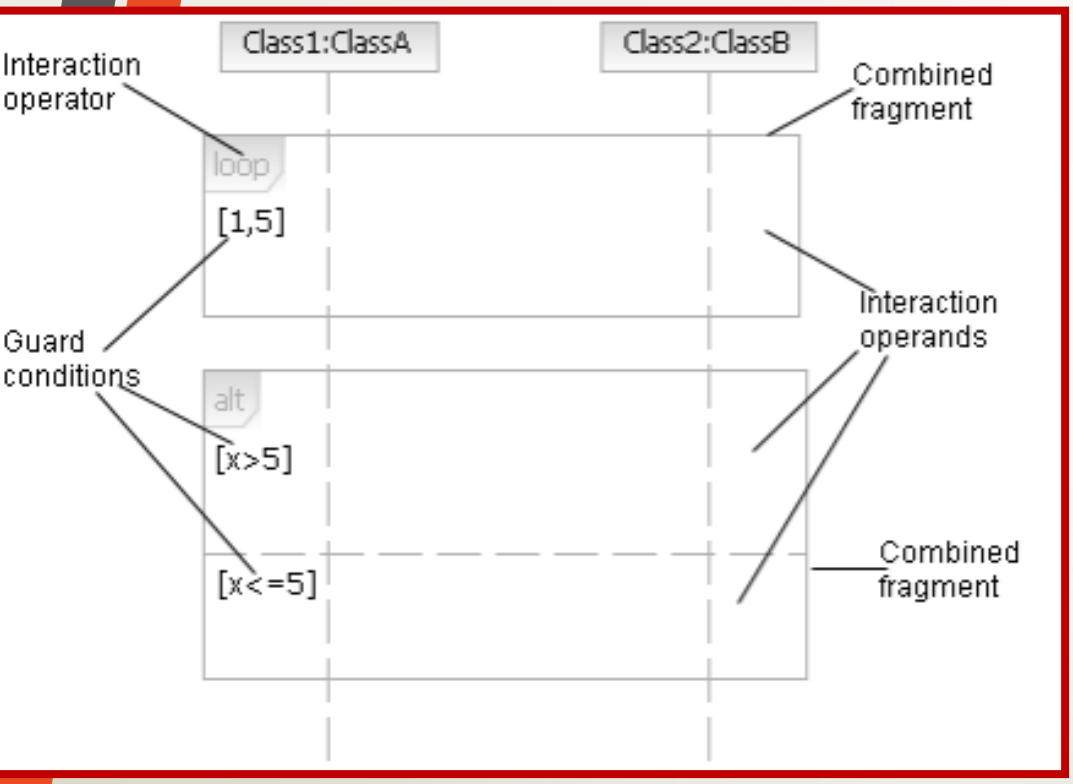
There are number of interaction operators.

- **alt** - alternatives
- **opt** - option
- **loop** - iteration
- **break** - break
- **par** - parallel
- **Weak Sequencing** - seq
- **Strict Sequencing** -strict
- **Critical** - Critical

Combined Fragment

- Each combined fragment is governed by rules that define how to execute its contained set of interaction operands
- A Combined Fragment consists of one or more **Interaction Operands.**
- Each Combined Fragment possesses an **Interaction Operator** that defines how to use the interaction operands within the context of the combined fragment.

Combined Fragments &Interaction Operand



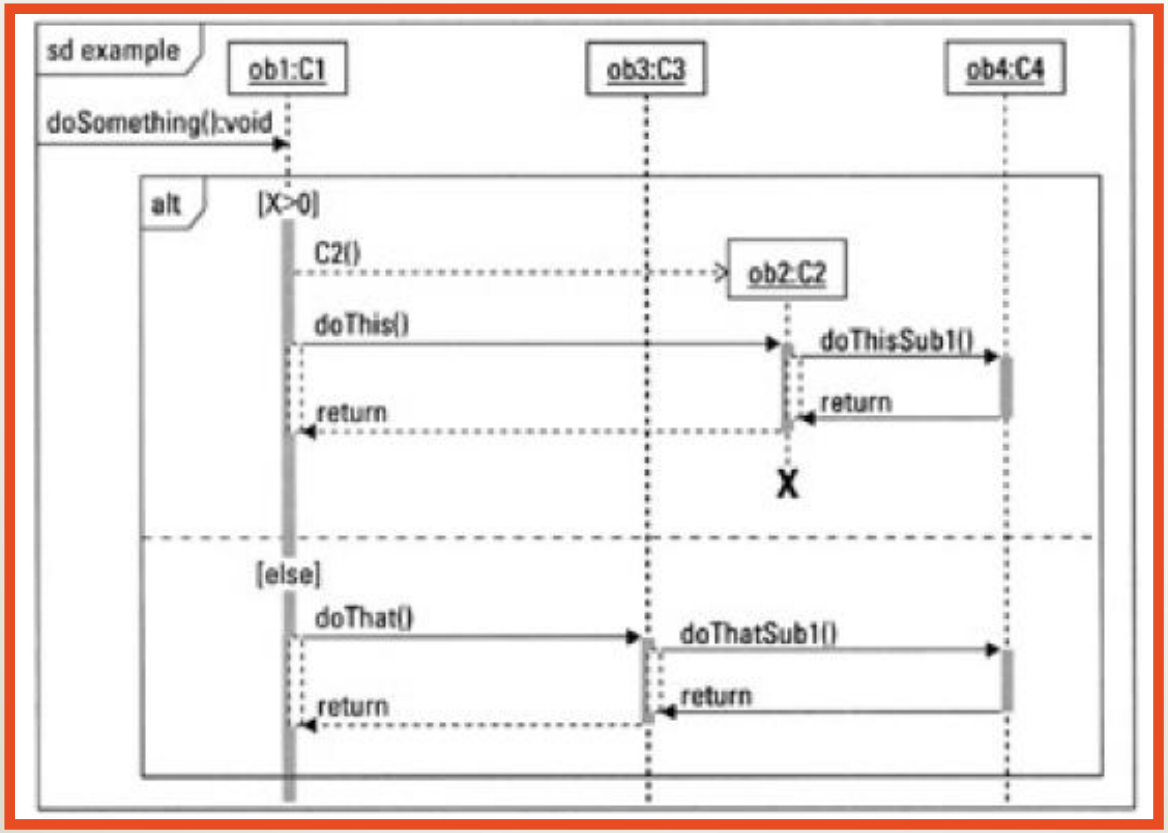
- A Combined Fragment consists of one or more **Interaction Operands.**
- Each interaction operand is a fragment of an interaction and covers the lifelines in the combined fragment.
- Each Combined Fragment possesses an **Interaction Operator** that defines how to use the interaction operands.

- Combined Fragment is a reference to a set of one or more interaction operands.
- An interaction operand contains an optional guard condition, which is also called an interaction constraint. The interaction operand runs only if the guard condition tests true.

Alternative (Keyword -alt)

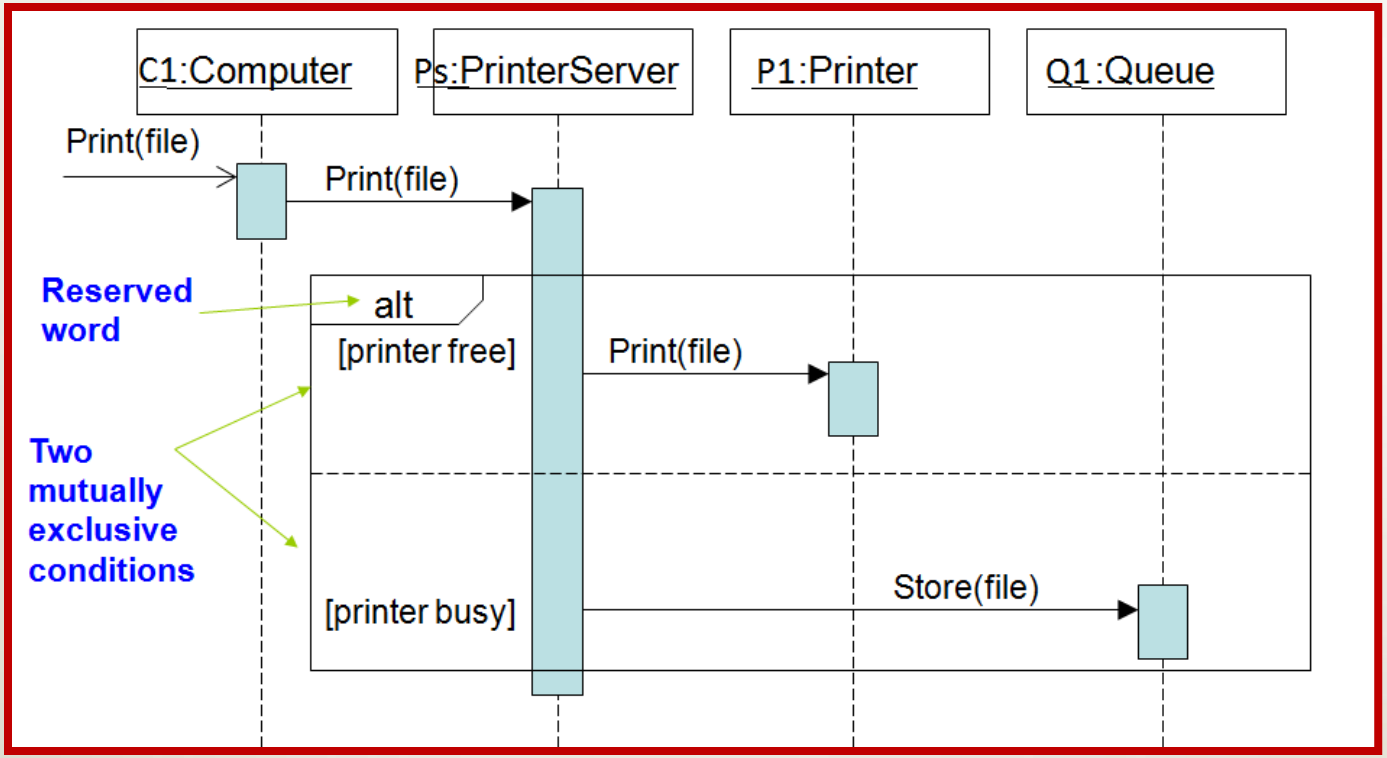
- The alt (alternatives) interaction operator identifies a set of behaviors from which the interaction can choose based on specified criteria.
- **Only one** of the offered alternatives will execute on any one pass through the interaction.
- The selected operand in the alt structure executes only if the guard condition tests true.
- The else clause of the alt combined fragment executes whenever none of the other options is selected.
- An alt combined fragment may offer any number of alternatives.

alt – Example 1



- **The alt combined fragment offers one option, and an else clause.**
- **If the guard [x>0] tests true, the first fragment will execute.**
- **If it tests false, the interaction specified by the else fragment will execute.**

alt - Example 2



Activity 2

Use case name: Add event to calendar

Main flow:

1.The user *logs* into the calendar system.

2. The user's calendar is *displayed.*

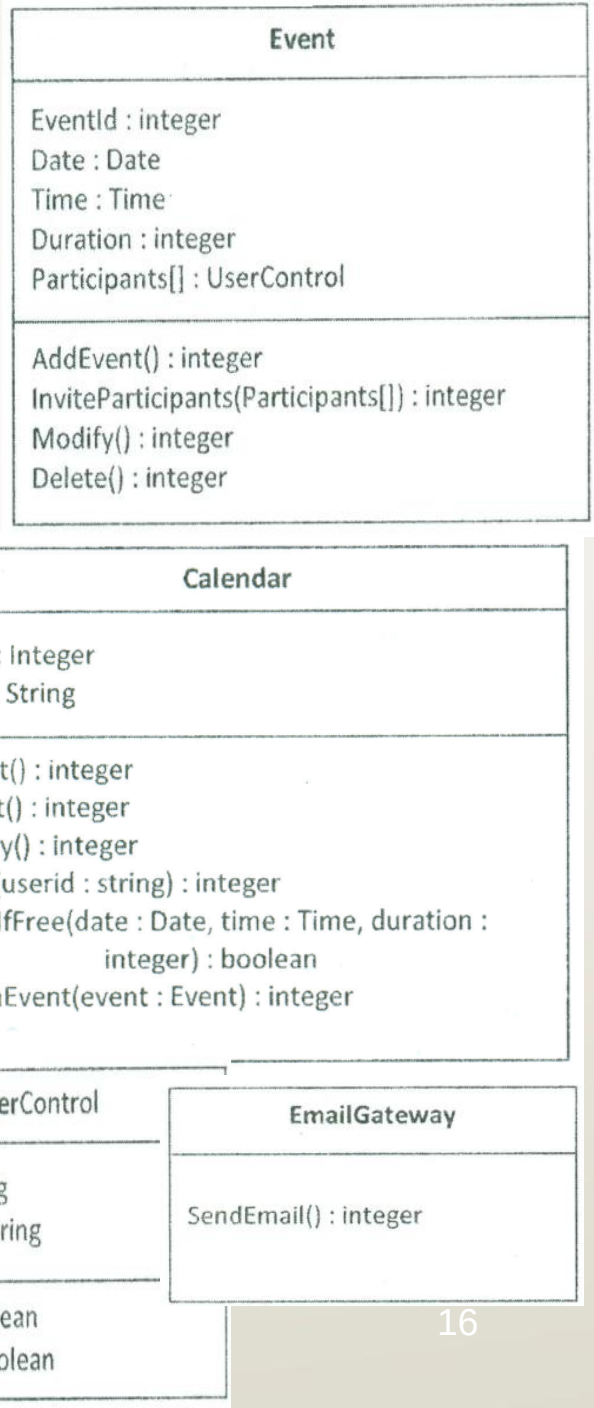
3. The user *adds an Event* in the calendar.

4. The system displays a screen to add participants.

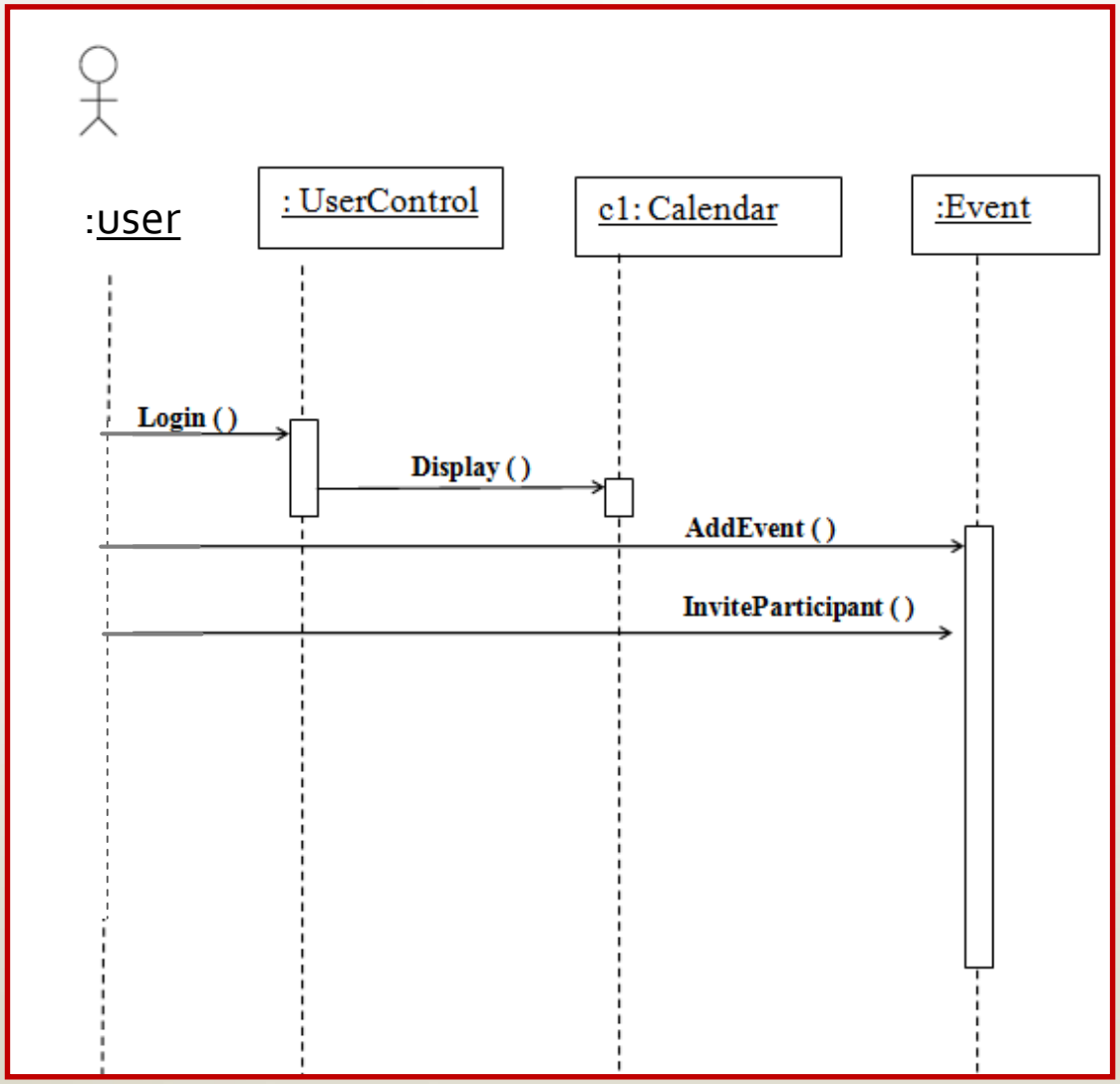
5. The user *invites* others to participate by selecting participants from a list box.

Note:

Words in italics correspond to methods in the classes given below.



Answer


:user
: UserControl
c1: Calendar
:Event
Login ( )
Display ( )
AddEvent ( )
InviteParticipant ( )

Activity 02 :-Sequence Example Contd.

Use case name: Add event to calendar

Main flow:

1.The user *logs* into the calendar system.

2. The user's calendar is *displayed*.

3. The user *adds an Event* in the calendar.

4. The system displays a screen to add participants.

5. The user *invites* others to participate by selecting participants from a list box.

6. For each participant the system *opens* the participants calendar and checks each person's calendar to see if the *timeslot is free* to assign an event.

***7.* For each participant who is free the *event is assigned* to each participants calendar.**

Extensions:

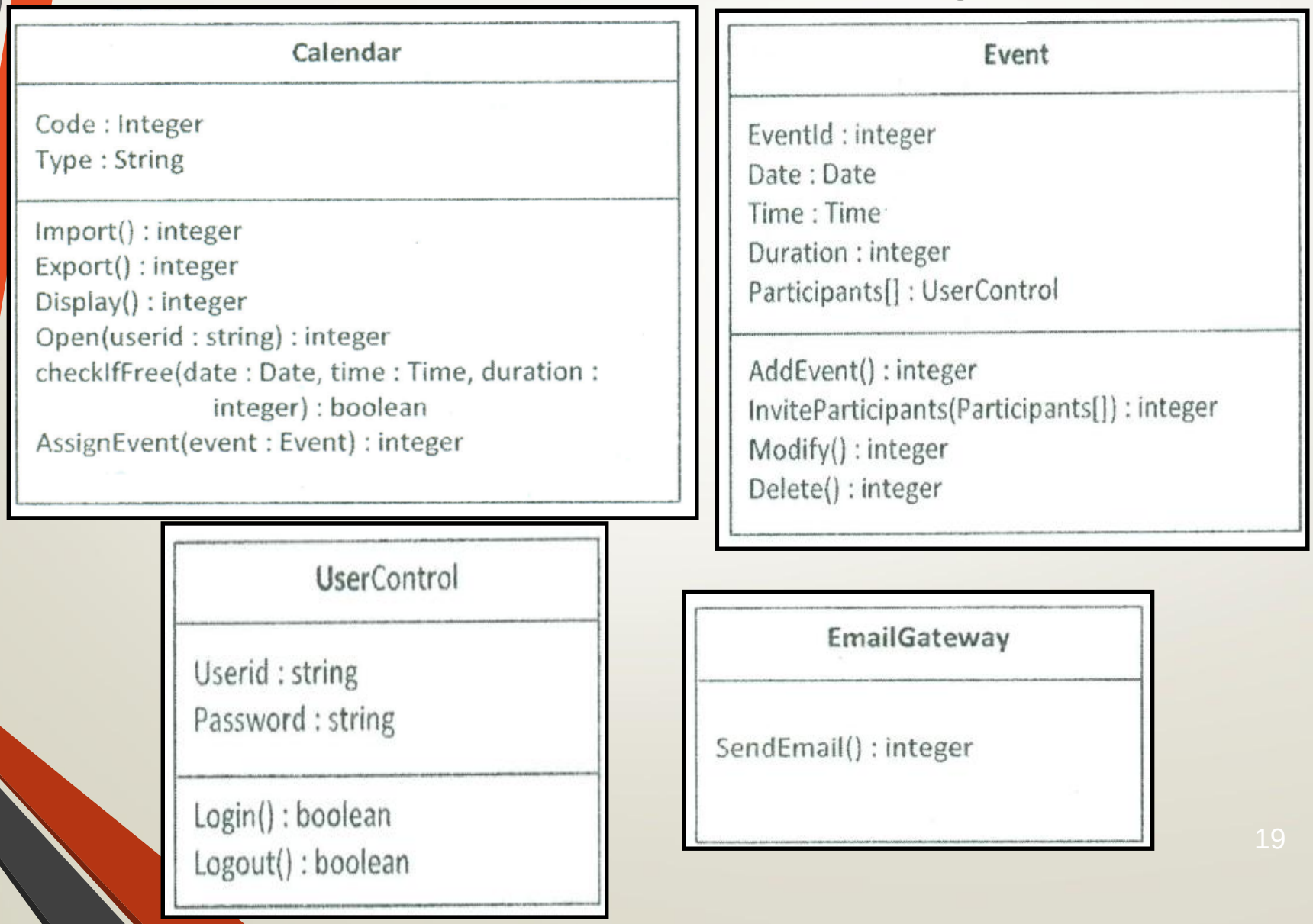
7. Participant is not free.

7a. An email is sent to participants about the event and the clash in the schedule.

Note:

For 6 use a different calendar object.

Sequence Example class diagram



Activity 2– Identify objects for step 6 & 7 and redraw the sequence diagram <u>with the condition</u>.

6. For each participant the system *opens* the participants calendar and checks each person's calendar to see if the *timeslot is* free to assign an event.

7. For each participant who is free the *event is assigned* to each participants calendar.

<u>Extensions:</u>

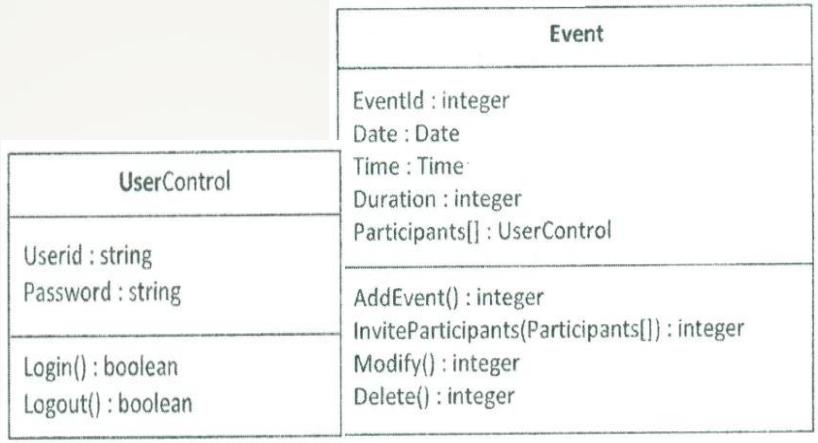
7. Participant is not free.

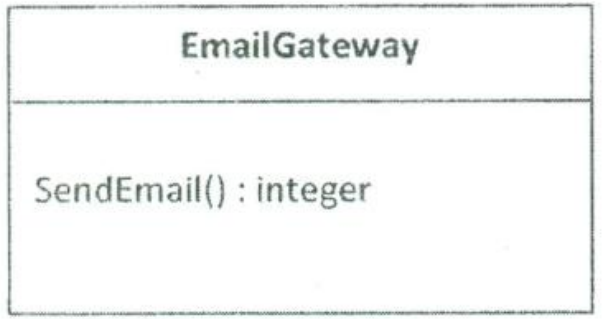
7a. An email is sent to participants about the event and the clash in the schedule.

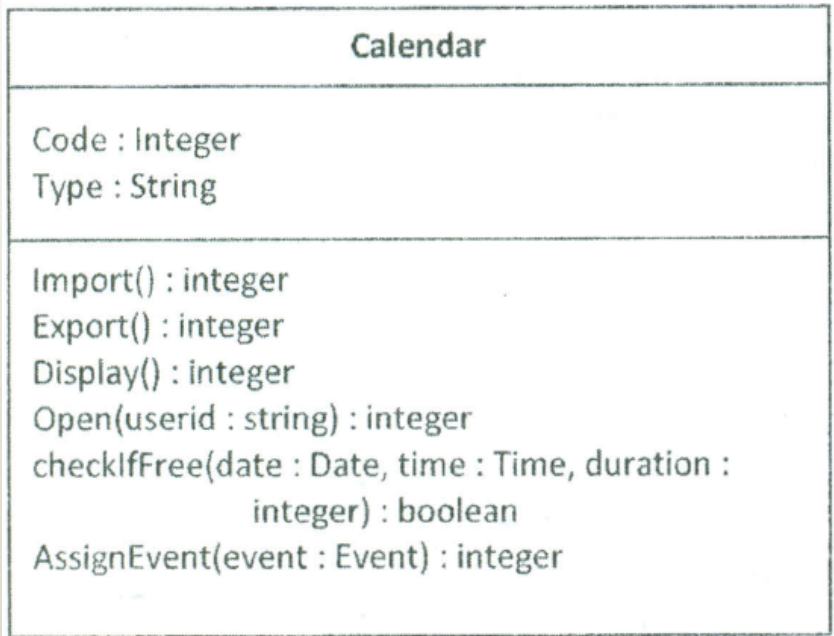
<u>Note:</u>

For 6 use a different calendar object.

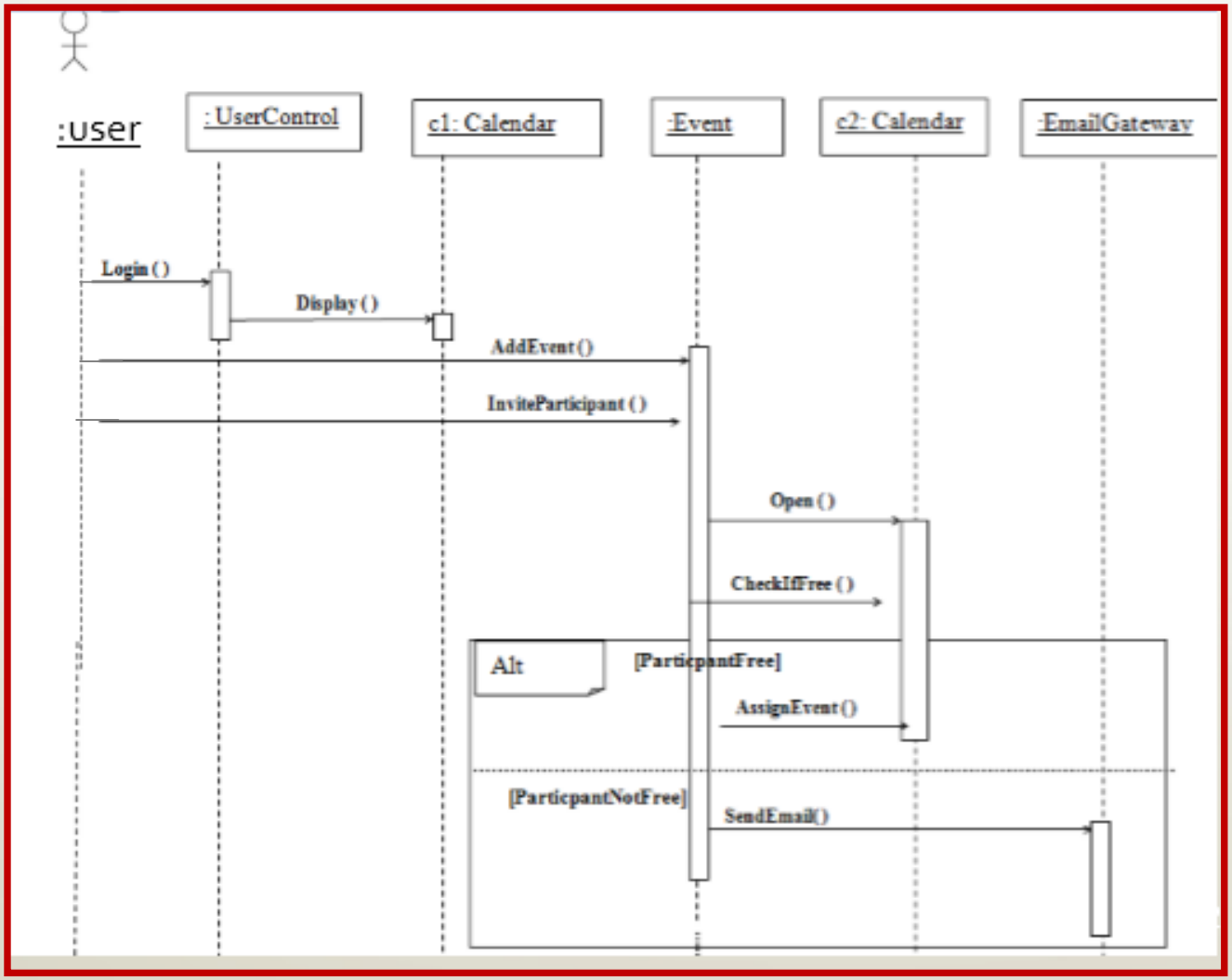
Event Object calls :open( ),checkIfFree ( ), assignEvent( ), sendEmail ( )





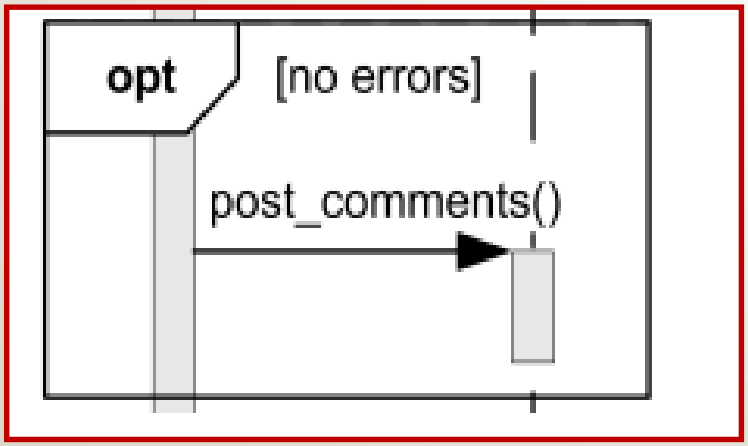


Activity 2:- answer

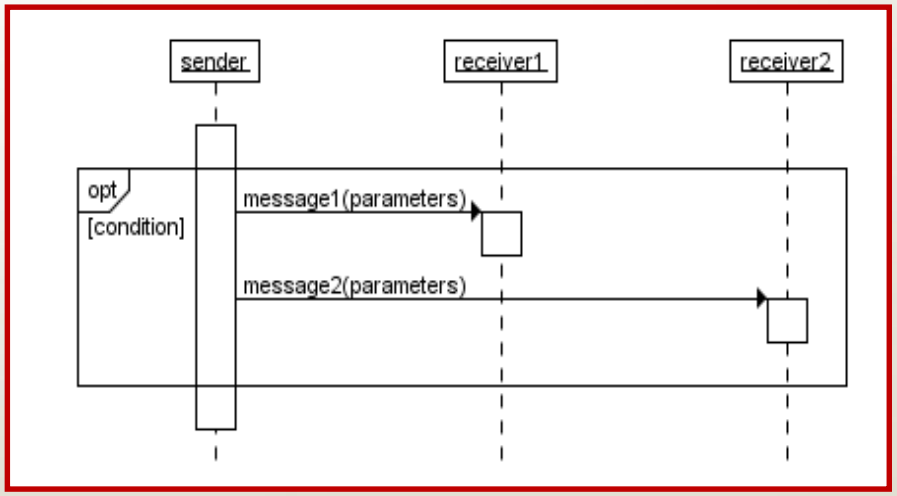


Conditions – Optional (Keyword -Opt)

- The opt (option) interaction operator represents a behavior that may or may not be used as part of the interaction.
- To be used, the guard condition must be satisfied.
- If the guard condition fails, the behavior is simply skipped.
- The model for an opt combined fragment looks like an alt that offers only one interaction.



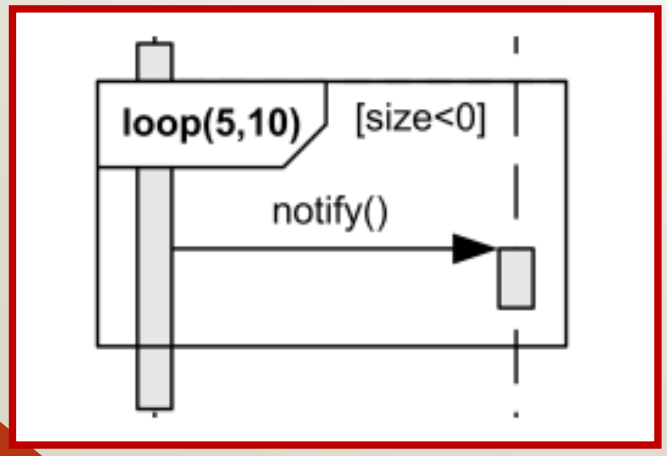
Post comments if there were no errors.



If the condition is met both messages are sent.

Loop (Keyword – Loop)

- The loop interaction operator indicates that the interaction fragment will be executed repeatedly.
- The number of times it is executed is determined by the minint and maxint parameters of the operator.
- The syntax of the loop operator is: **loop(minint, maxint).**
- Once the minimum number of iterations has been satisfied, a Boolean expression is tested on each pass.
- When the Boolean expression tests false, the loop ends.

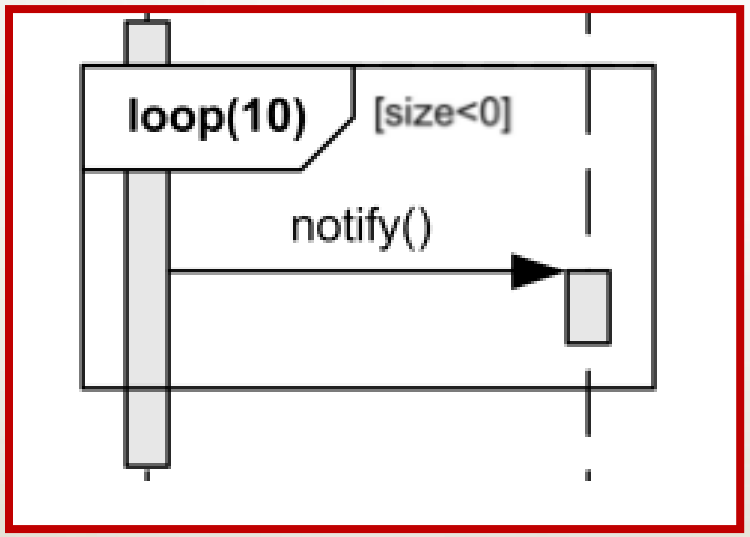


The loop is expected to execute minimum 5 times and no more than 10 times.

If guard condition [size<0] becomes false loop terminates regardless of the minimum number of iterations specified.

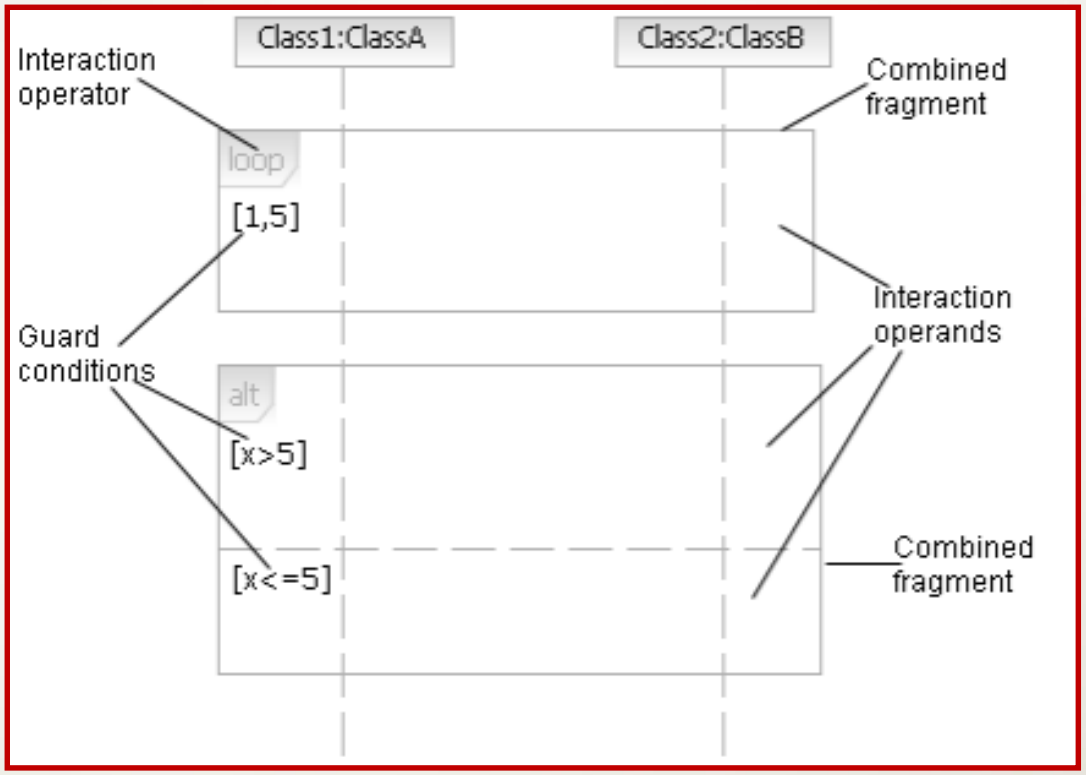
Loop cont…

If only ***min-int*** is specified, it means that upper bound is equal to the lower bound, and loop will be executed exactly the specified number of times till the condition is true.



Loop to execute exactly 10 times.

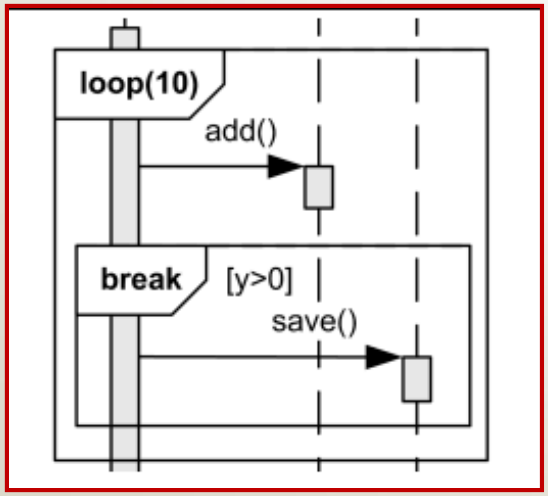
Interaction Operand



- Each interaction operand is a fragment of an interaction and covers the lifelines in the combined fragment.
- An interaction operand contains an optional guard condition, which is also called an interaction constraint.
- The interaction operand runs only if the guard condition tests true.

Break (Keyword - Break)

- If this fragment is executed, the rest of the sequence is abandoned. You can use the guard to indicate the condition in which the break will occur.
- The break interaction operator provides a mechanism similar to the break syntax in many programming languages.



Break enclosing loop if y>0.

In the course of executing an interaction, if the guard of the break is satisfied, then the containing interaction abandons its normal execution and instead performs the clause specified by the break fragment.

Activity 2 cont…

Use case name: Add event to calendar

Main flow:

1. The user *logs* into the calendar system.
2. The user's calendar is *displayed*.
3. The user *adds an Event* in the calendar.
4. The system displays a screen to add participants.
5. The user *invites* others to participate by selecting participants from a list box.
6. For each participant the system *opens* the participants calendar and checks each person's calendar to see if the *timeslot is free.*
7. For each participant who is free the *event is assigned* to each participants calendar.

Extensions:

7. Participant is not free.

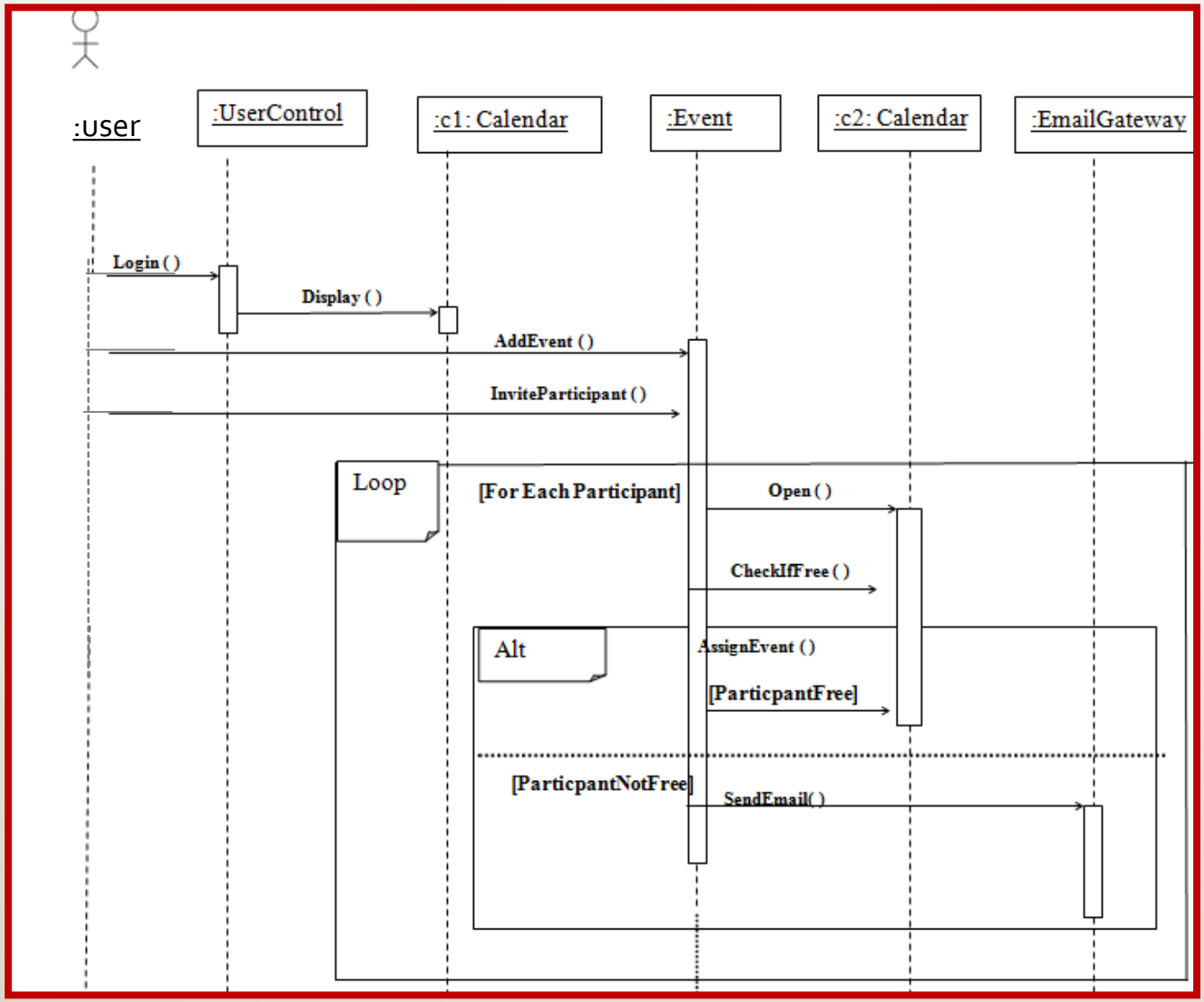
7a. An email is sent to participants about the event and the clash in the schedule.

Note:

- Words in italics correspond to methods in the classes given below.
- Event Object calls :open( ), checkIfFree ( ), assignEvent( ), sendEmail ( )
- For 6 use a different calendar object.

Both 6,7 are repeated together.

Activity 2

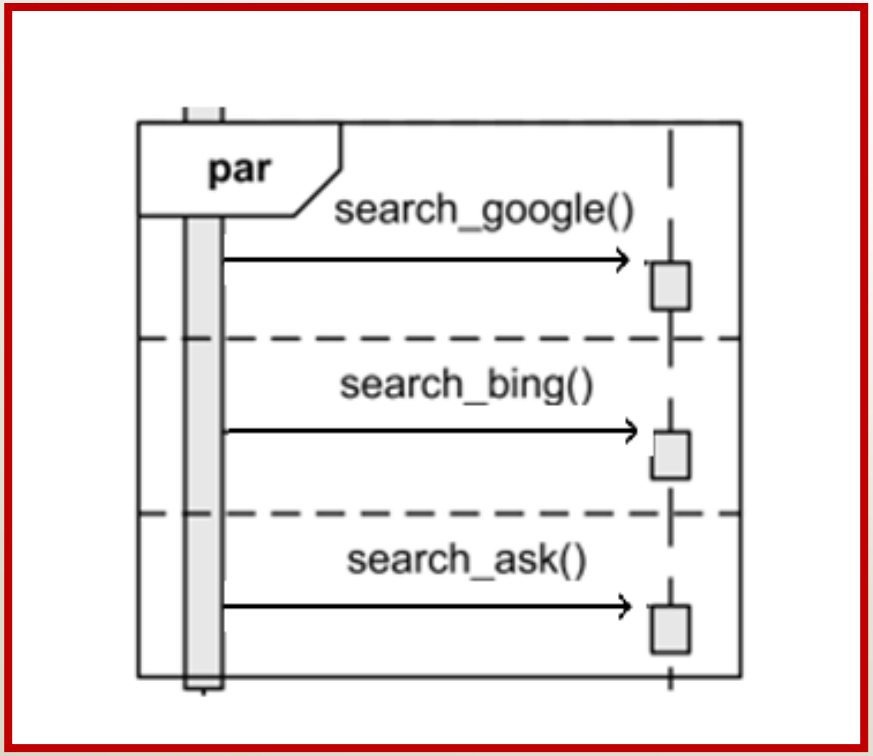

:user
:UserControl
:c1: Calendar
:Event
:c2: Calendar
:EmailGateway
Login ( )
Display ( )
AddEvent ( )
InviteParticipant ( )
Loop
[For Each Participant]
Open ( )
CheckIfFree ( )
Alt
AssignEvent ( )
[ParticpantFree]
[ParticpantNotFree]
SendEmail( )

Parallel (Keyword - Par)

- When the processing time required to complete portions of a complex task is longer than desired, some systems handle parts of the processing in parallel.
- The interaction operator **par** defines parallel execution of behaviors.
- Different operands can be interleaved in any way as long as the ordering imposed by each operand is preserved.

Parallel cont…

All three may execute at the same time, and the individual events within each interaction fragment may execute in any order within the constraints of the individual interaction fragments without regard for other events on the same lifelines.

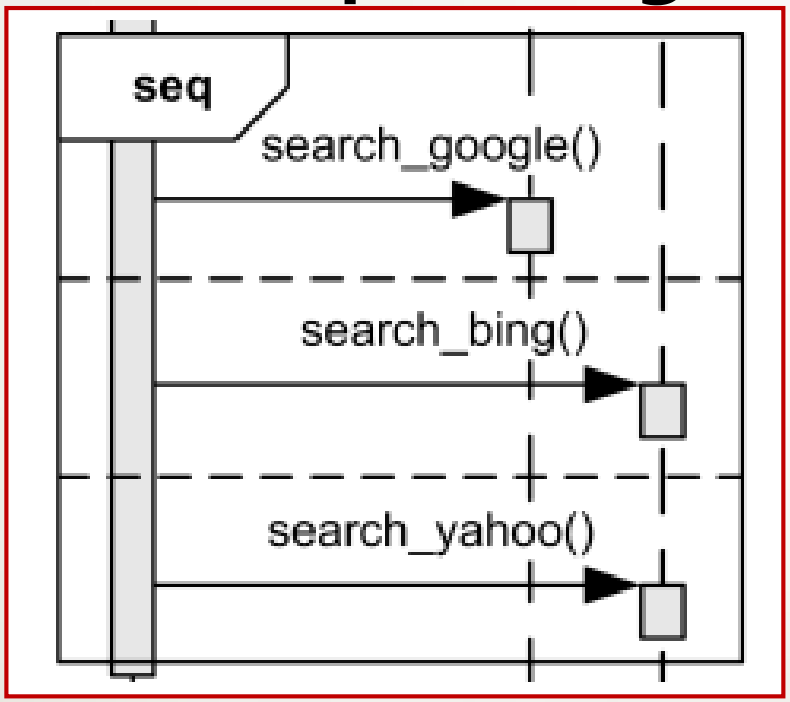


Search Google, Bing and Ask in any order, possibly parallel.

Weak Sequencing (Keyword –seq)

- The seq (weak sequence) interaction operator forces the interactions to follow a certain order.
- This interaction operator adds order to the interactions in the fragment based on their placement.
- If two events occur on the same lifeline, the event on the uppermost interaction executes first. (In same lifeline sequence does matter)
- Where they do not involve the same lifelines, messages from different fragments may be interleaved in parallel.

Weak Sequencing cont …

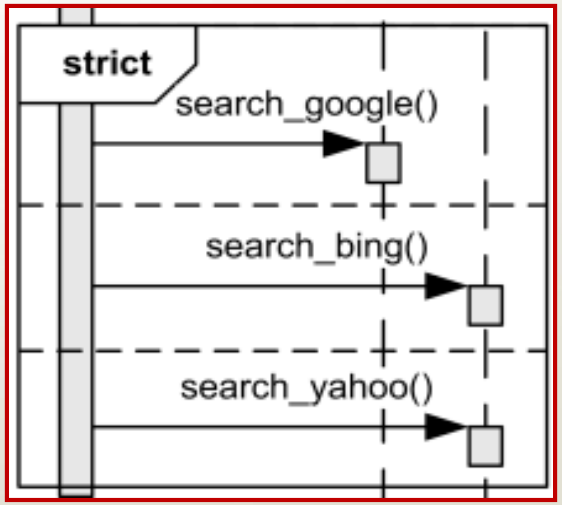


Search Google possibly parallel with Bing and Yahoo, but search Bing before Yahoo.

Event occurrences on the same lifeline from different operands are ordered such that an event occurrence of the first operand comes before that of the second operand.

Strict Sequencing (Keyword –strict)

- The strict interaction operator explicitly defines the order of execution of the interaction fragments.
- The strict operator forces the completion of the interaction before running nested or additional interactions.
- There are two or more operand fragments. The fragments must occur in the order given.
- The effect of strict is to force the completion of one interaction before proceeding to the next interaction.



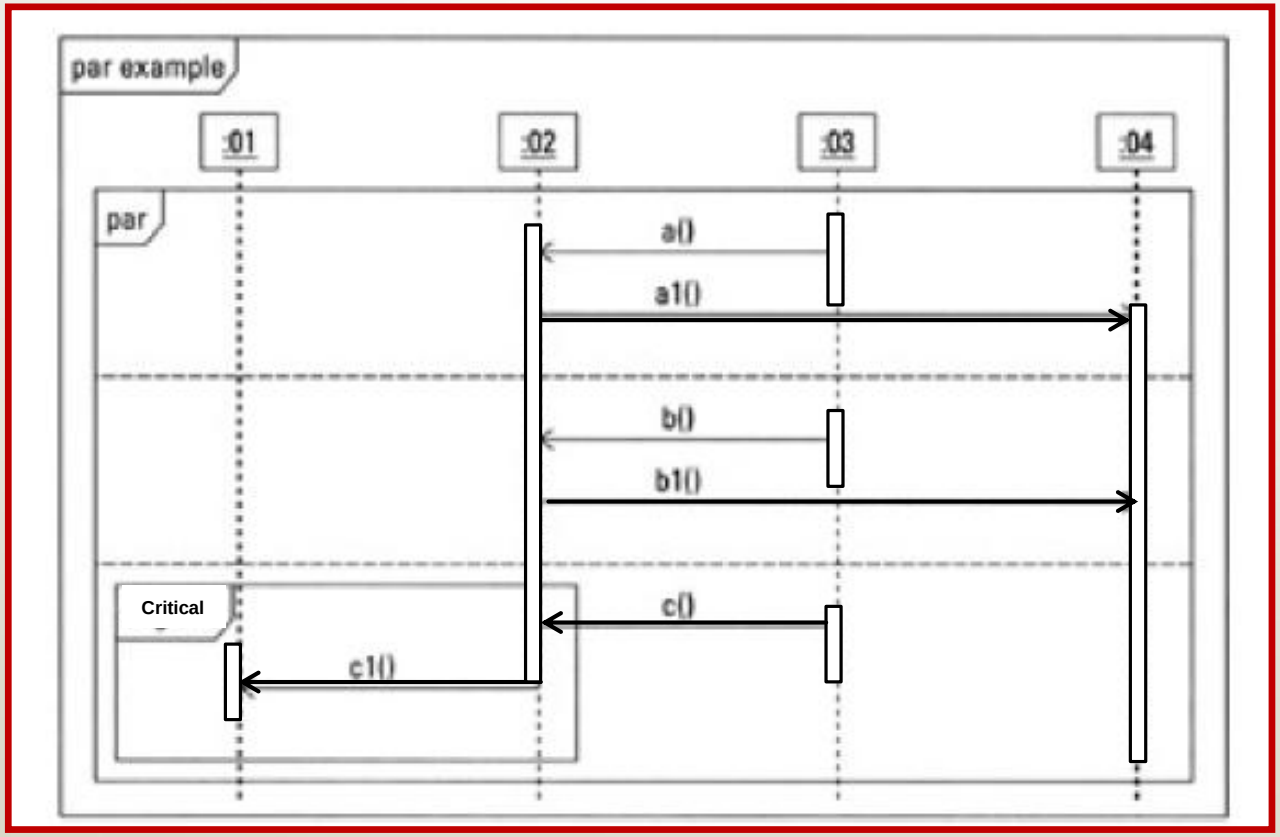
Search Google, Bing and Yahoo in the strict sequential order.

Critical (Keyword –Critical)

- Used within a Par or Seq fragment. Indicates that the messages in this fragment must not be interleaved with other messages.
- This means that the region is treated **atomically** by the enclosing fragment and can't be interleaved, e.g. by parallel operator.
- All the interaction fragments may execute in any order, unless the critical region is invoked, and then it must execute before any of the other fragments may execute.

Critical cont…

If the operation c() is invoked, it will run to completion before any new calls to a() or b() are allowed to execute.



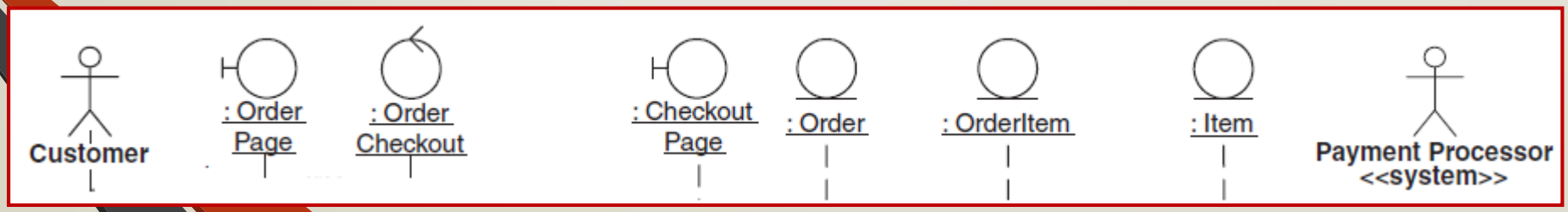
Activity 4- Draw a sequence diagram for the description given below.

- The **Registrar** opens Course Registration to student's and this is made available for students who wish to register.
- Students visit the web and Fill Up course registration details through **RegistrationUI** and submit it to the system.
- System will only allow maximum of 100 students to register for the course. So when each student is registered the count is checked.
- **RegistrationManager** will forward the registration details to **RegistrationDB** to store it.
- Once the registration is over, **Registrar** can request and view the registered student list; **RegistrationManage**r will request the registered list through querying the **RegistrationDB.**

Rules of Thumb

- **Avoid crossing links** and crowded diagrams.
- **Do not show all interactions** on a sequence diagram - only what is important for the scenario:
  - Only show those messages that are necessary for understanding;
  - Only show those classes/objects that are necessary for understanding.

- Place Proactive System Actors on the Leftmost Side while Reactive System Actors on the Rightmost Side of Your Diagram.



References

- UML 2 Bible
  - Chapters 8 & 9
- Applying UML and Patterns by Craig Larman
  - Chapter 15
- TheElementsofUML2Style
  - Chapter 7