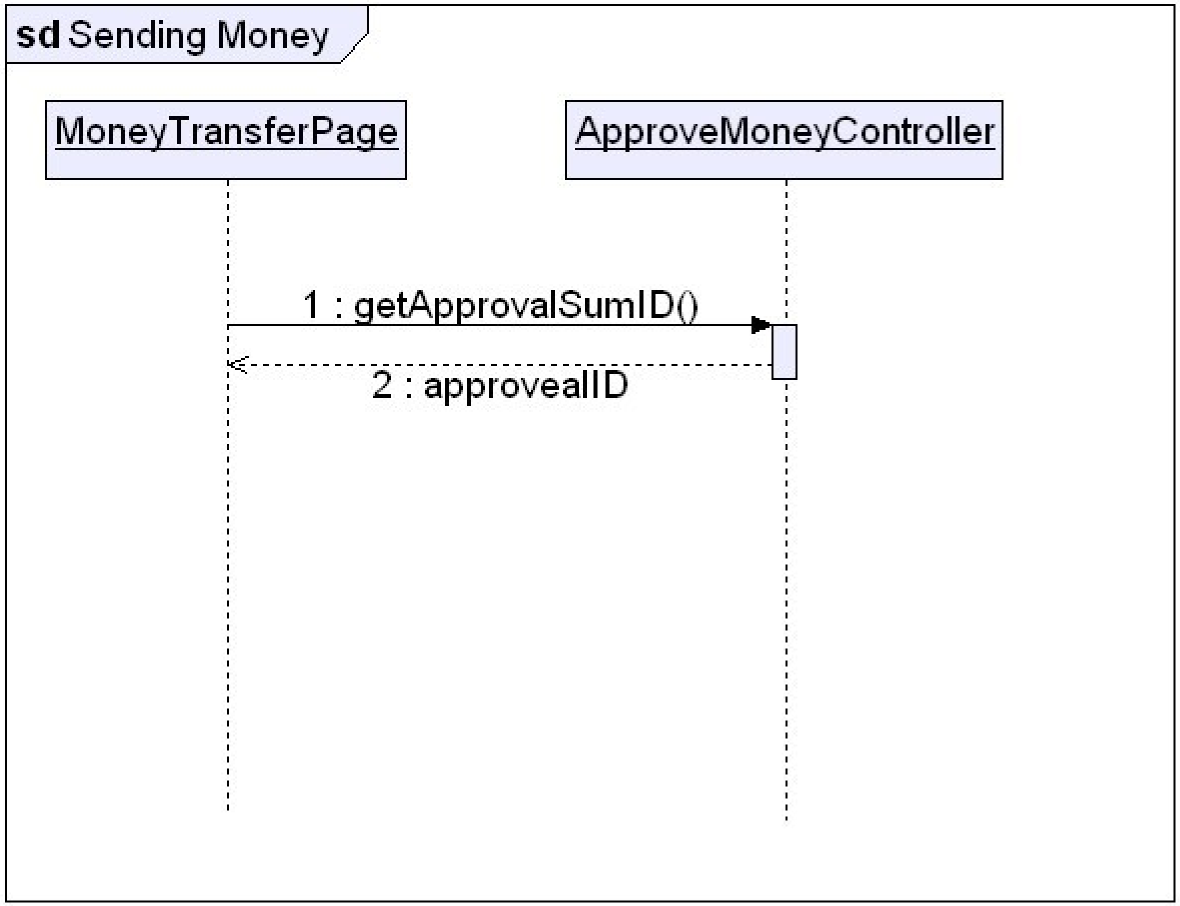
UML Sequence Diagrams

© 2008 Haim Michael. All Rights Reserved.

Introduction

- The UML sequence diagrams (part of a bigger group of diagrams called interaction diagrams) capture the communications between the objects.
- The UML interaction diagrams focuses on the messages the objects send to each other.

Introduction



The Life Line

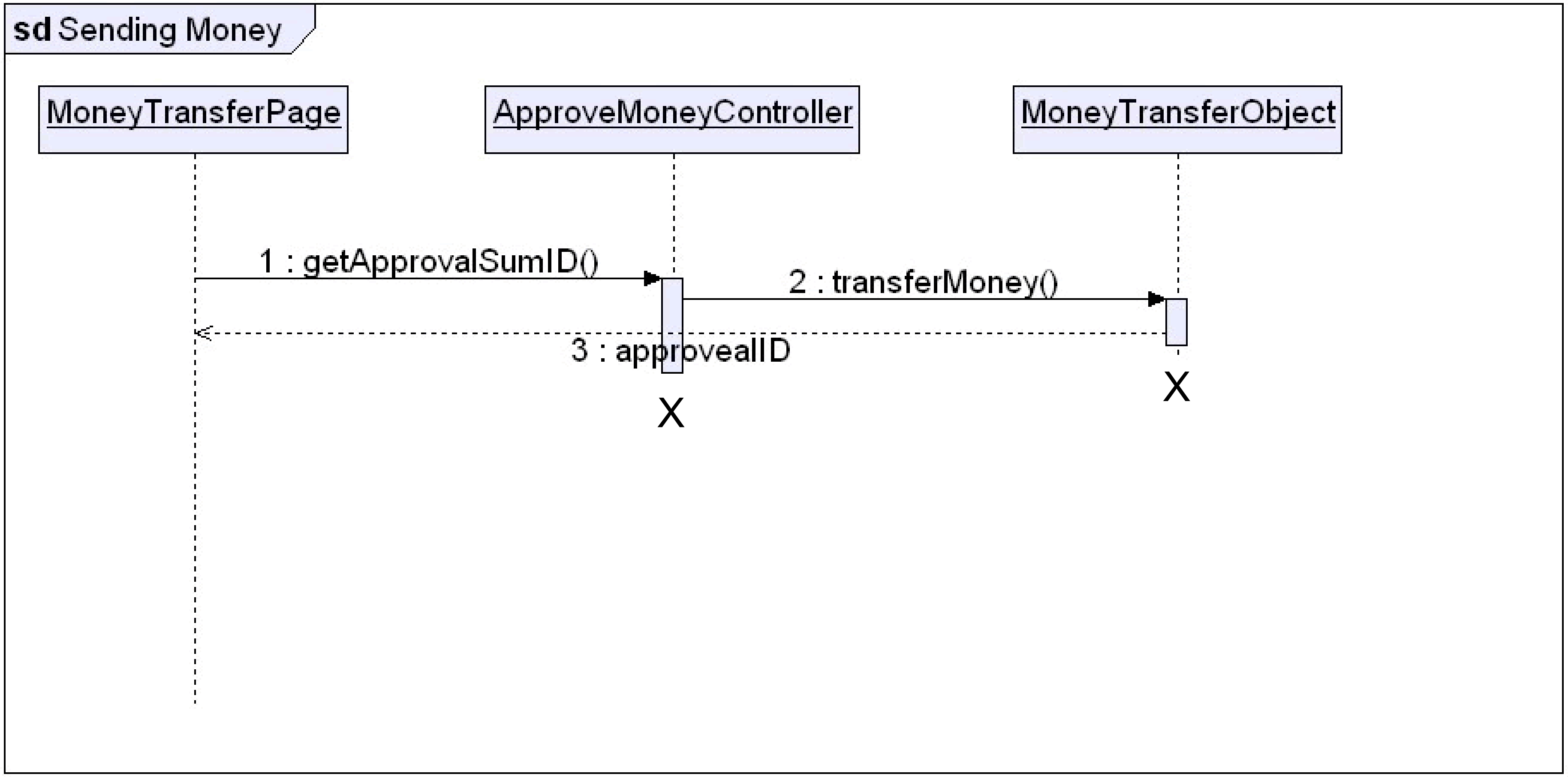
- The life line is the dashed line coming down from each object showing for how long the object exists.
- The object notation is the rectangle from which the life line goes down.
- The sequence diagram is presented within a frame that on its top left part we write the diagram name preceding with “sd” that stands for “sequence diagram”.

© 2008 Haim Michael. All Rights Reserved.

The Stop Symbol

- The destruction of a participant during the interaction can be depicted using the stop symbol.
- The stop symbol is a bold X.

The Stop Symbol

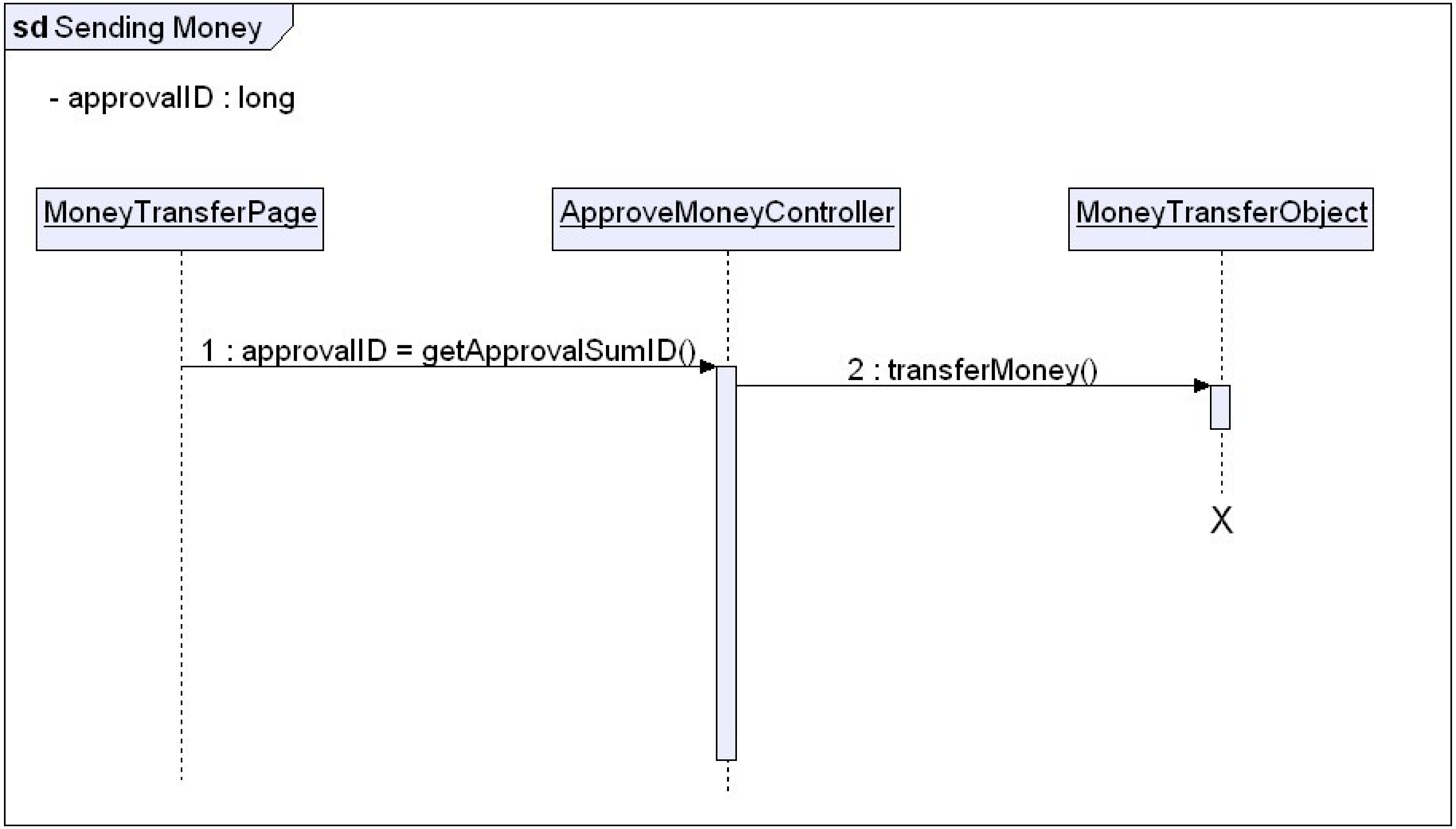


Local Variables

- In order to improve the readability of the diagram it is possible to add local variables.
- The local variables should be listed on top left part of the diagram using the same notation used in class diagrams.

© 2008 Haim Michael. All Rights Reserved.

Local Variables



Messages

- The communication between the objects shown in a sequence diagram takes place via messages.
- A messages can be one of the following types:
  - calling a specific method on the other object
  - sending a signal to the other object
  - creating an instance (the other object)
  - destroying the other object
- A message is defined by its specific type, the sender and the receiver.

Messages

- When a message represents a method call we can show the arguments been sent to that method.
- Arguments been sent to a method can be any of the following:

  Attributes of The Sending Object

  Constants

  Parameters of the Enclosing Interaction

  Attributes of The Class Owning the Enclosing Interaction

  Expressions

Messages

- The message syntax is:

  attribute = signal_or_operation_name(arguments)

  attribute

  This element is optional. If using this element please note the attribute name must be one of the following:

  + an attribute of the lifeline sending the message

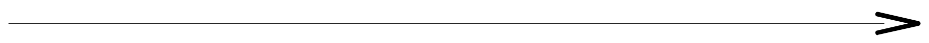
  + global attribute

  + attribute of the class that owns this interaction

  + local variable you declared

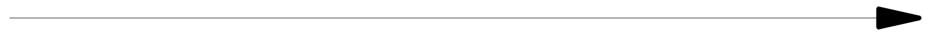
Asynchronous Messages

- The asynchronous message is depicted using a solid line with an open arrowhead pointing at the receiver's end.
- Sending an asynchronous message means that the sender doesn't need to wait for the reply.

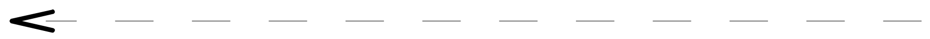


Synchronous Messages

- The synchronous message is depicted using a solid line with a filled arrowhead pointing at the receiver's end.
- Sending a synchronous message means that the sender needs to wait for the reply.
- Calling a method is usually a synchronous message.



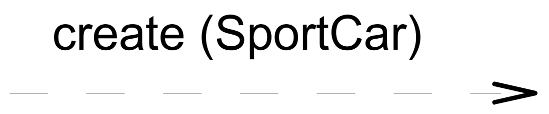
- You can show a return value using a dashed line with open arrow pointing at the sender.



© 2008 Haim Michael. All Rights Reserved.

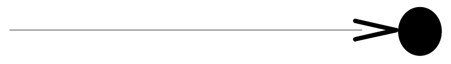
Object Creation

- If the message represents an object creation we should use a dashed line with an open arrow pointing a the new created object. It is common to write above the dashed line the word create together with the name of the instantiated class within brackets.

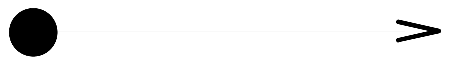


Lost & Found Messages

- A lost message is a message that was sent and never arrived its destination. We depict such message by connecting the outgoing line that represents its sending with a black circle.



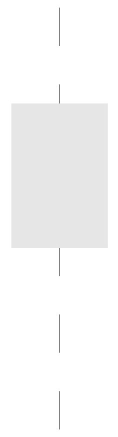
- A found message is a message that was received out of no where without knowing who sent it. We depict such message by connecting the incoming line that represents it with a black circle.



© 2008 Haim Michael. All Rights Reserved.

The Execution Occurrences

- The execution occurrences are shown as small gray rectangles above the life line.
- They represent the fact that the object is involved with executing the action.



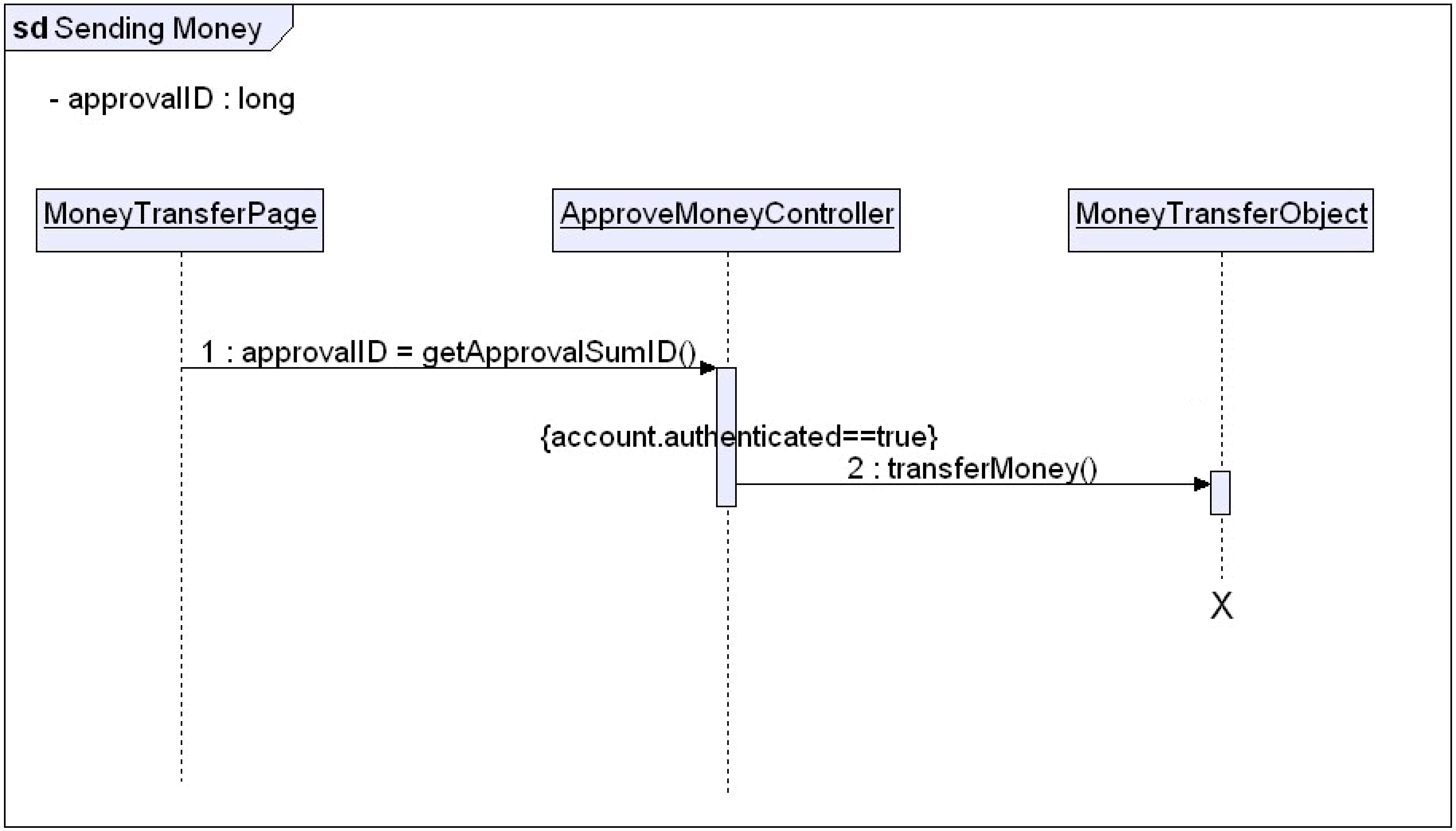
The State Invariants

- The state invariants are conditions (usually boolean expressions) we write inside curly braces { } as labels along the life line of the object we want to check.
- The state invariant describ a condition that must be true through the reminder of the interaction.

  The reminder of the interaction is evaluated from left to right top to bottom.
- Each time a message reaches a life line the state invariant is evaluated. If the state invariant is false the execution doesn't continue.

© 2008 Haim Michael. All Rights Reserved.

The State Invariants

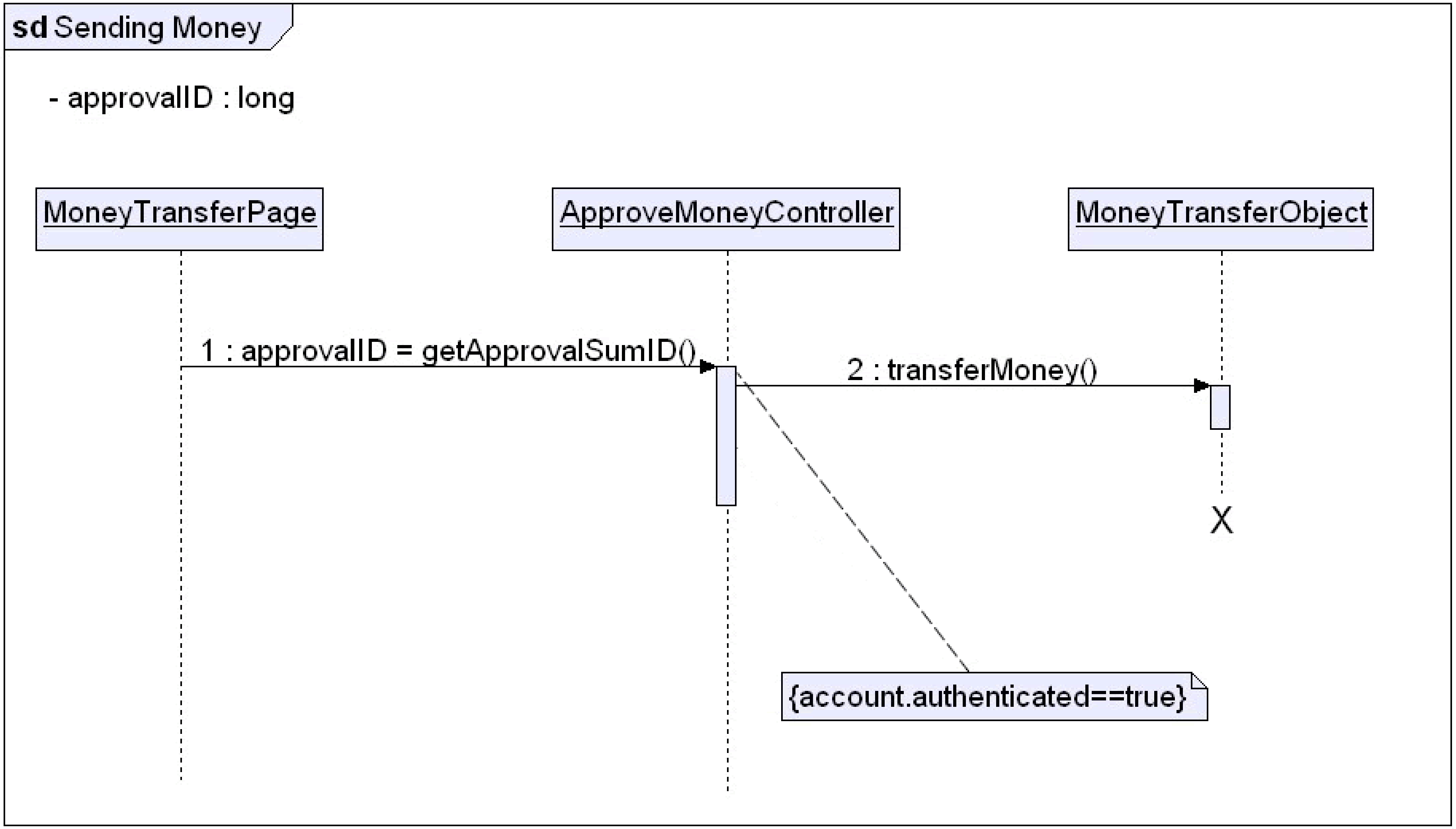


© 2008 Haim Michael. All Rights Reserved.

The State Invariants

- The state invariant can also be written within a note connected with the relevant life line.

The State Invariants



© 2008 Haim Michael. All Rights Reserved.

The Events Occurrences

- The event occurrences, the smallest part of the sequence diagram, represent specific moments in time.
- Each time that something happens it is a specific moment we consider as an event occurrence.
- We track the event occurrences from left to right... top to bottom.
- In most cases, the events occurrences are messages being sent and received.

Interaction Fragment

- Interaction Fragment is a sequence of events occurrences.

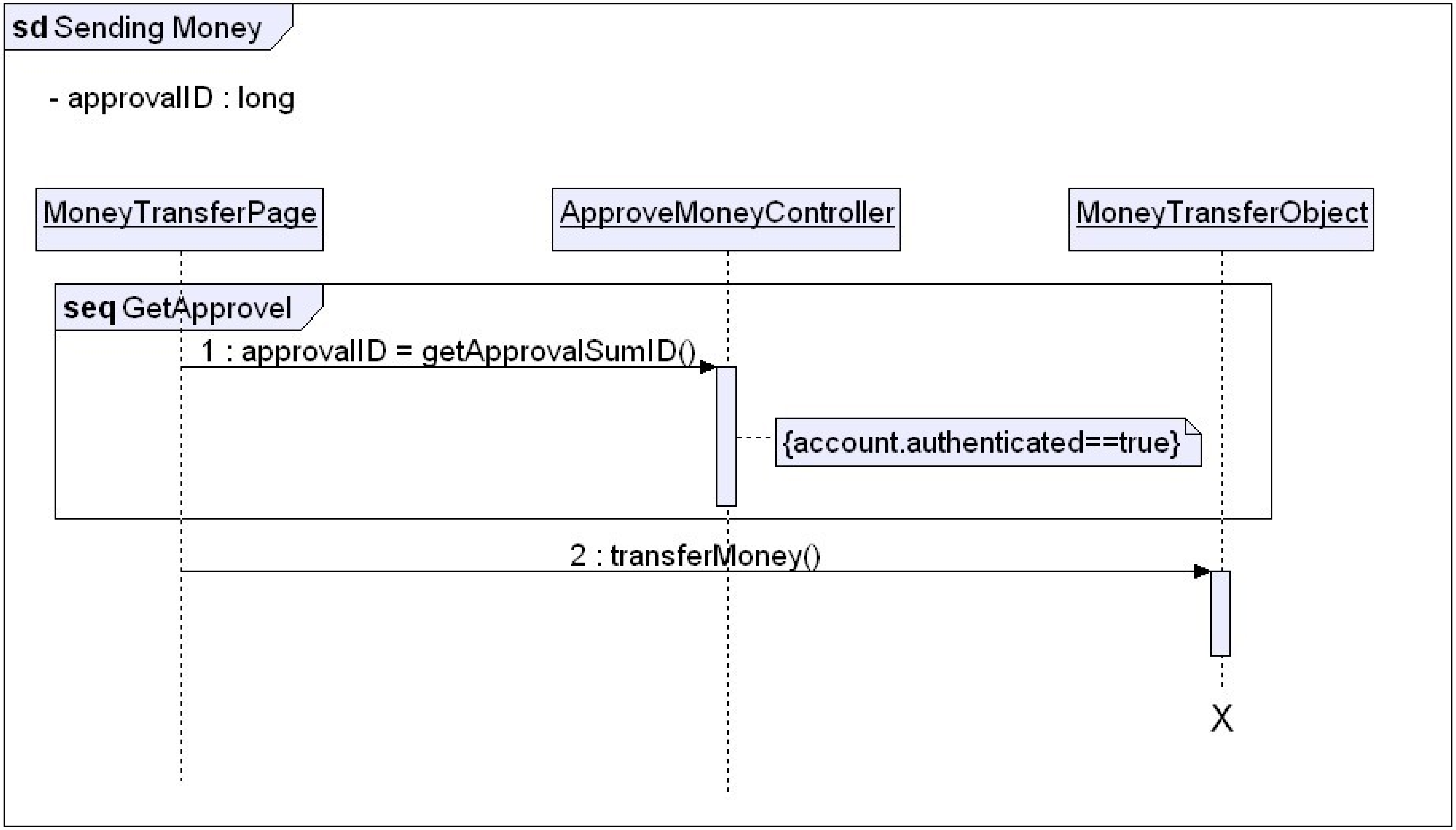
  Trace is another name for a interaction fragment.

© 2008 Haim Michael. All Rights Reserved.

Combined Fragments

- Combined fragment is a group of one (or more) separated interaction fragments.
- Each combined fragment includes an operator and one or more operand(s) it works on. Each operand is a separated interaction fragment.
- The operator specifies how to interpret the operands.
- We place the combined fragment within a frame similar to the one in use when presenting a sequence diagram.

Combined Fragments

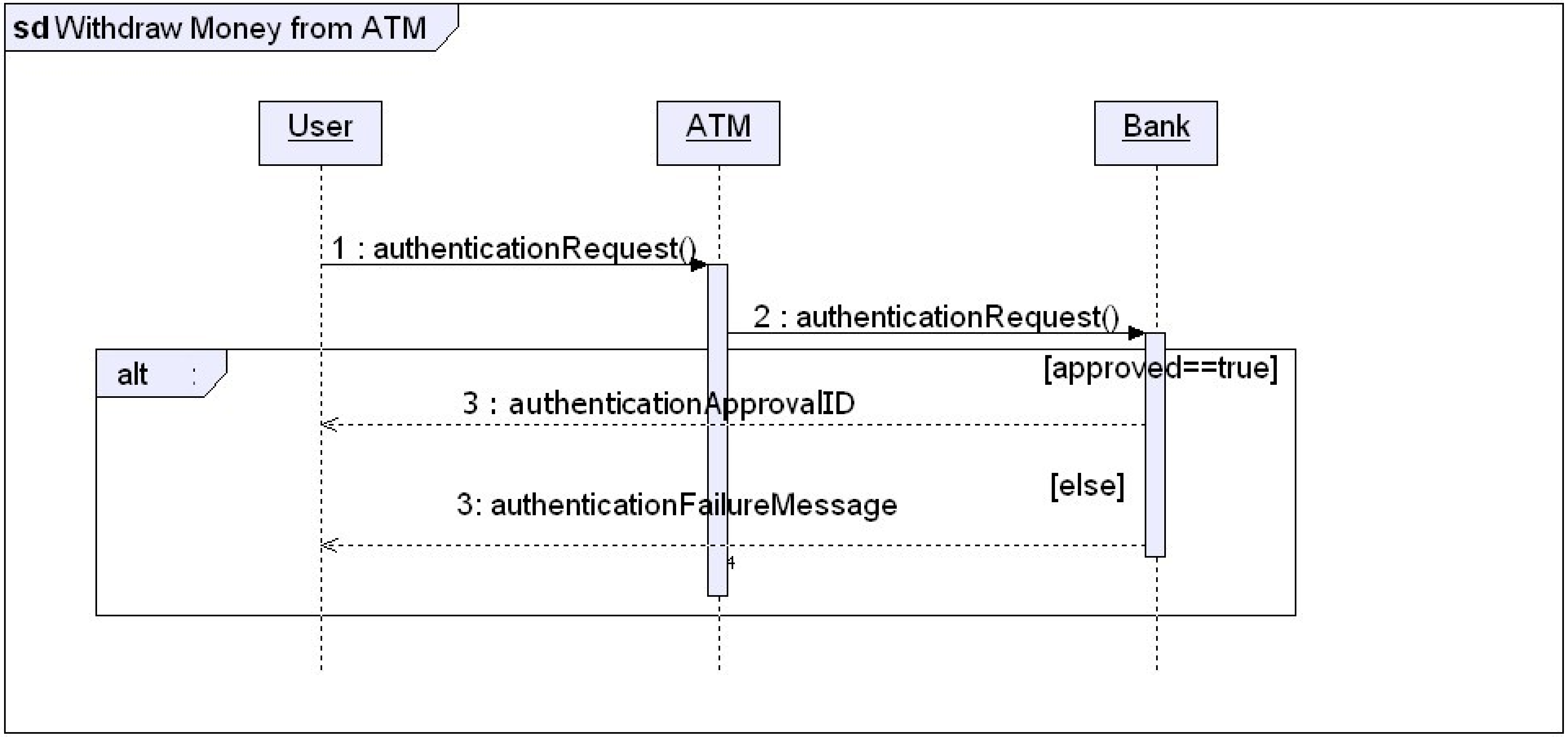


© 2008 Haim Michael. All Rights Reserved.

Combined Fragments

- It is possible to add a guard condition to an interaction fragment.
- The condition will be written within brackets and be placed directly above the first event occurrence life line within the interaction fragment it refers.

Combined Fragments


sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
alt
[approved==true]
3 : authenticationApprovalID
[else]
3: authenticationFailureMessage

© 2008 Haim Michael. All Rights Reserved.

Combined Fragments Operators

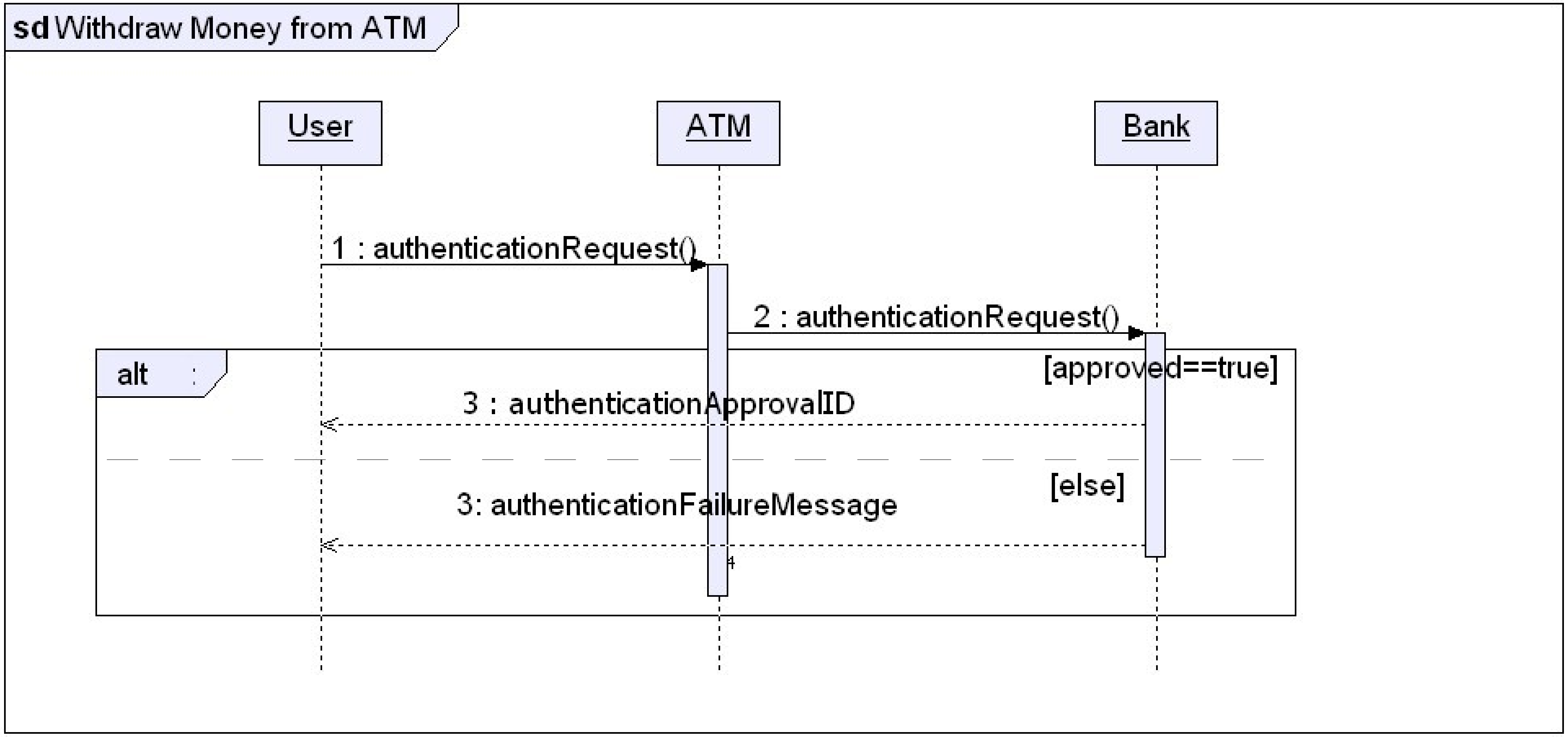
- UML specifications defines the following combined fragments operators:

alternatives	ignore / consider
option	assertion
break	loop
parallel	
weak sequence	
strict sequence	
negative	
critical region	

Alternatives Interaction Fragment

- An alternative interaction fragment is a choice of behavior executes based on a guard conditions evaluation with true/false.
- Top left pentagon should include the text “alt”.

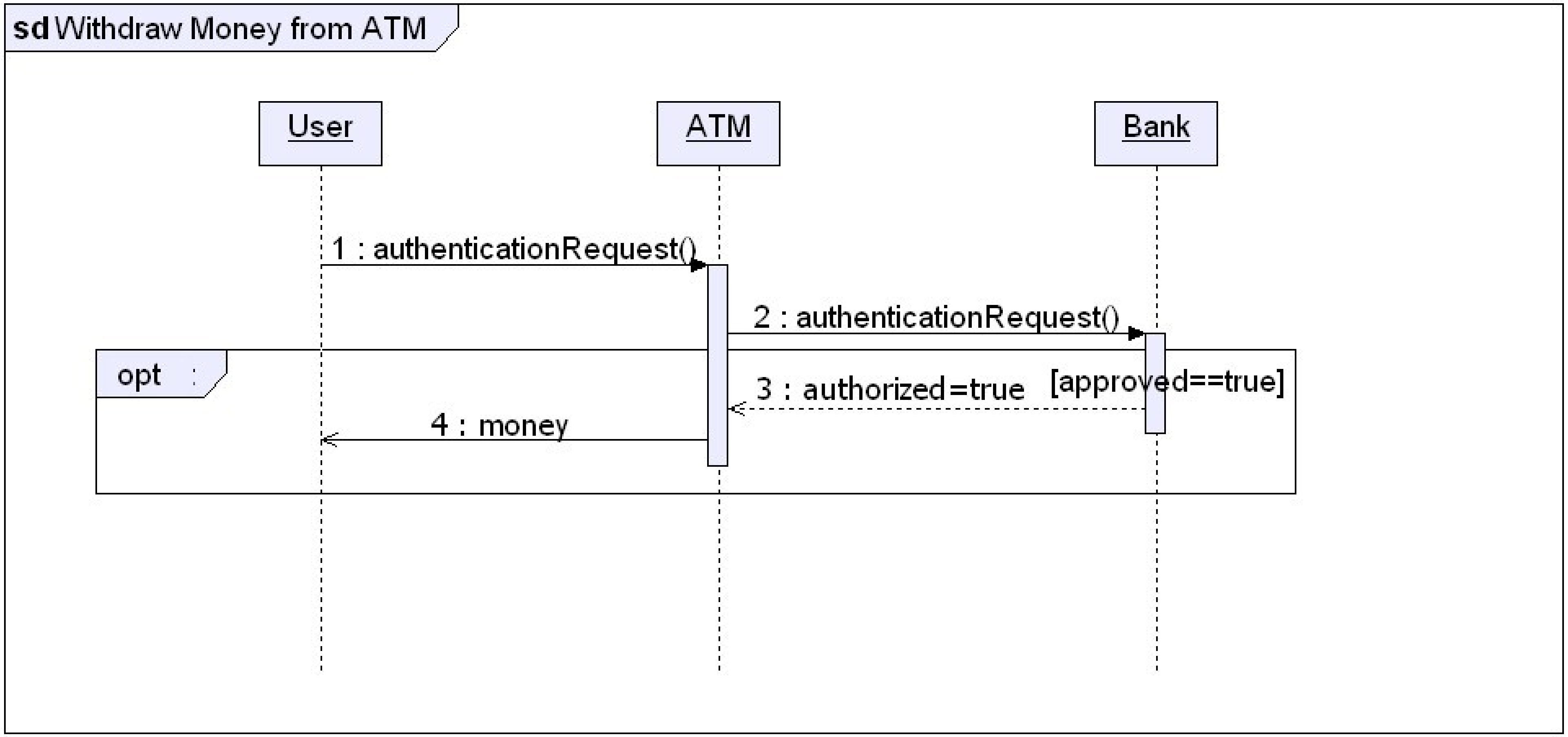
Alternatives Interaction Fragment


sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
alt
[approved==true]
3 : authenticationApprovalID
[else]
3: authenticationFailureMessage

Options Interaction Fragment

- Options are interaction fragments that executes if (and only if) the guard condition is true.
- Top left pentagon should include the text “opt”.

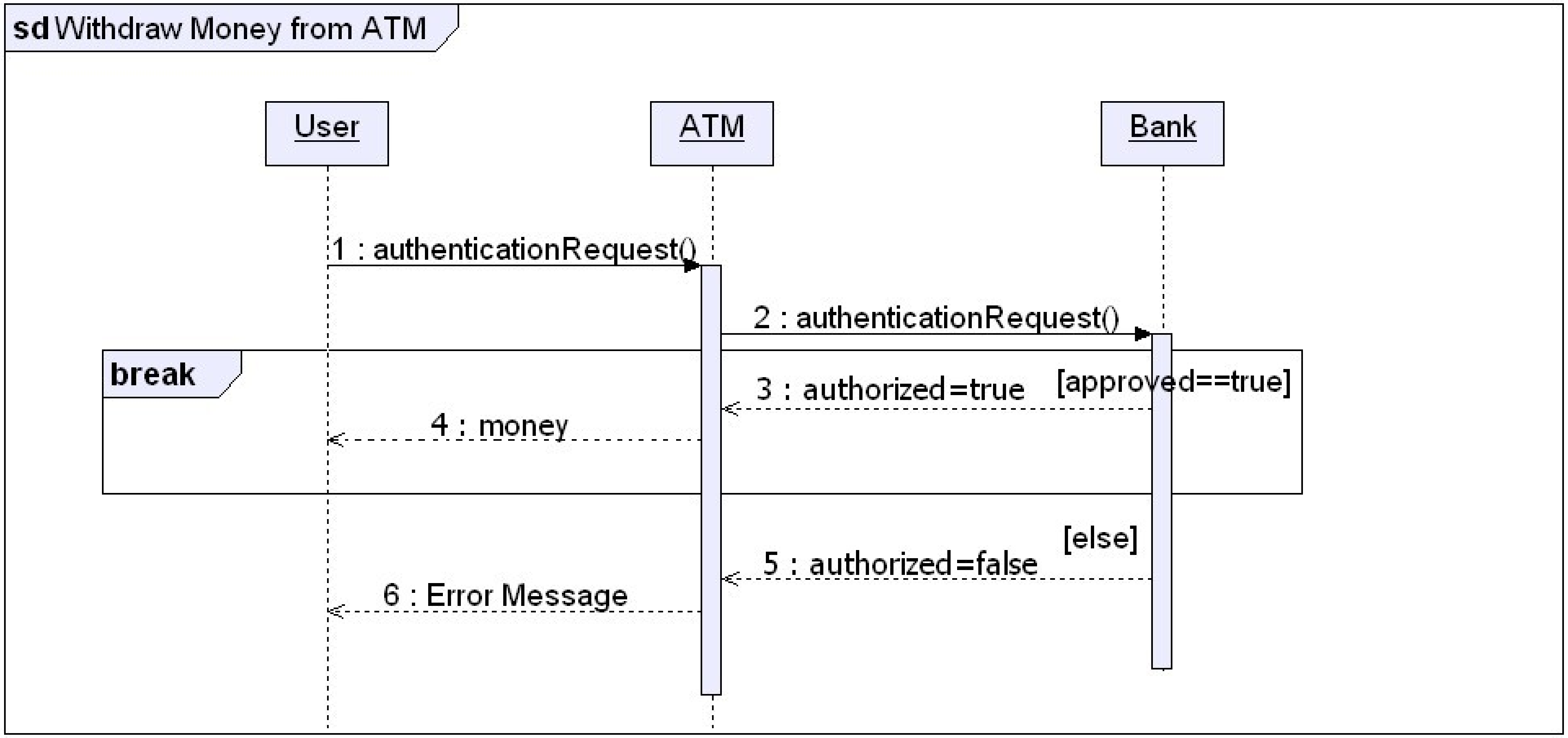
Options Interaction Fragment


sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
opt :
3 : authorized=true
[approved==true]
4 : money

Break Interaction Fragment

- Break is an interaction fragments that once executes it terminates the enclosing interaction.
- Top left pentagon should include the text “break”.

Break Interaction Fragment



© 2008 Haim Michael. All Rights Reserved.

Loop Interaction Fragment

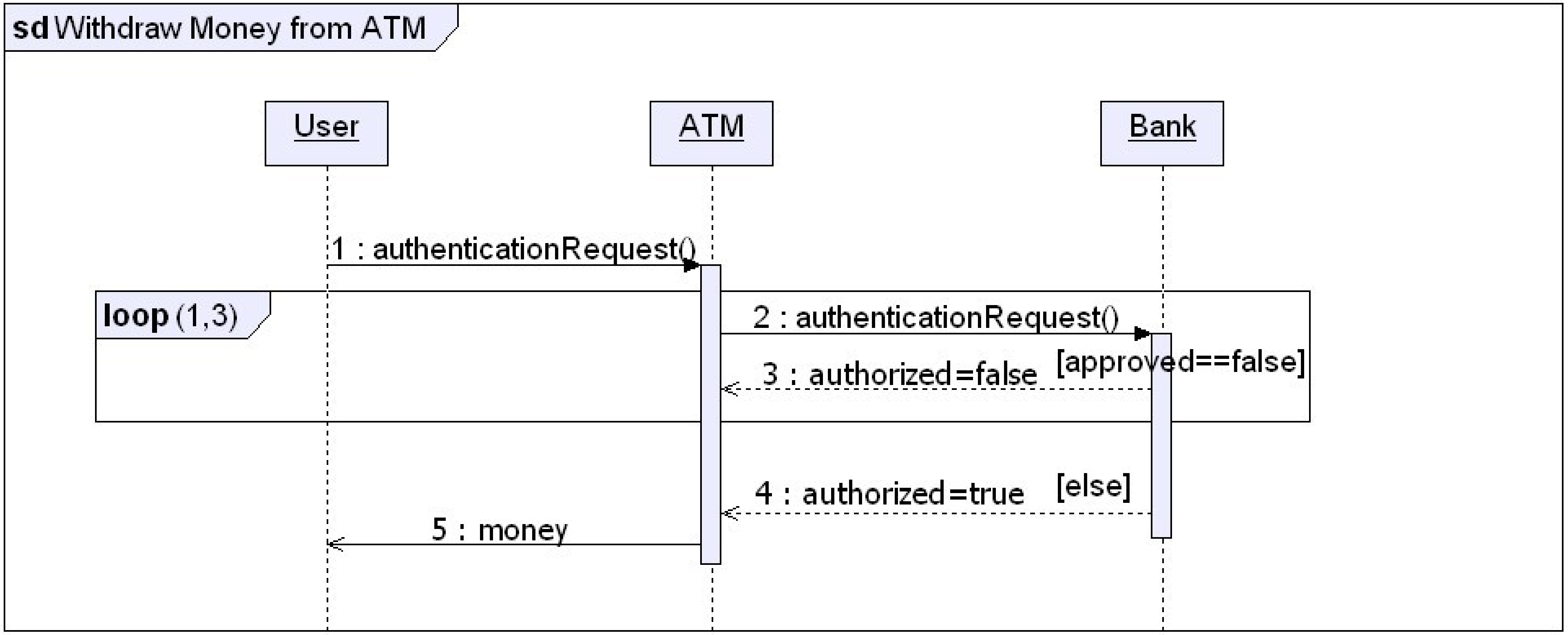
- Indicating that an interaction fragment, a contained event occurrence, should be executed several times is depicted by writing “loop” on the top left part of the combined fragment, and by writing either a guard condition or the word loop with the minimum and maximum number of steps written within braces.
  loop (min,max)
- If max is excluded then max equals min. The same apply for min in case it is excluded.

Loop Interaction Fragment

- The max value can be asterisk (*) to indicate an infinite loop.
- If both max and min are excluded then min equals 0 and max equals infinity.
- You can add a guard condition.

© 2008 Haim Michael. All Rights Reserved.

Loop Interaction Fragment



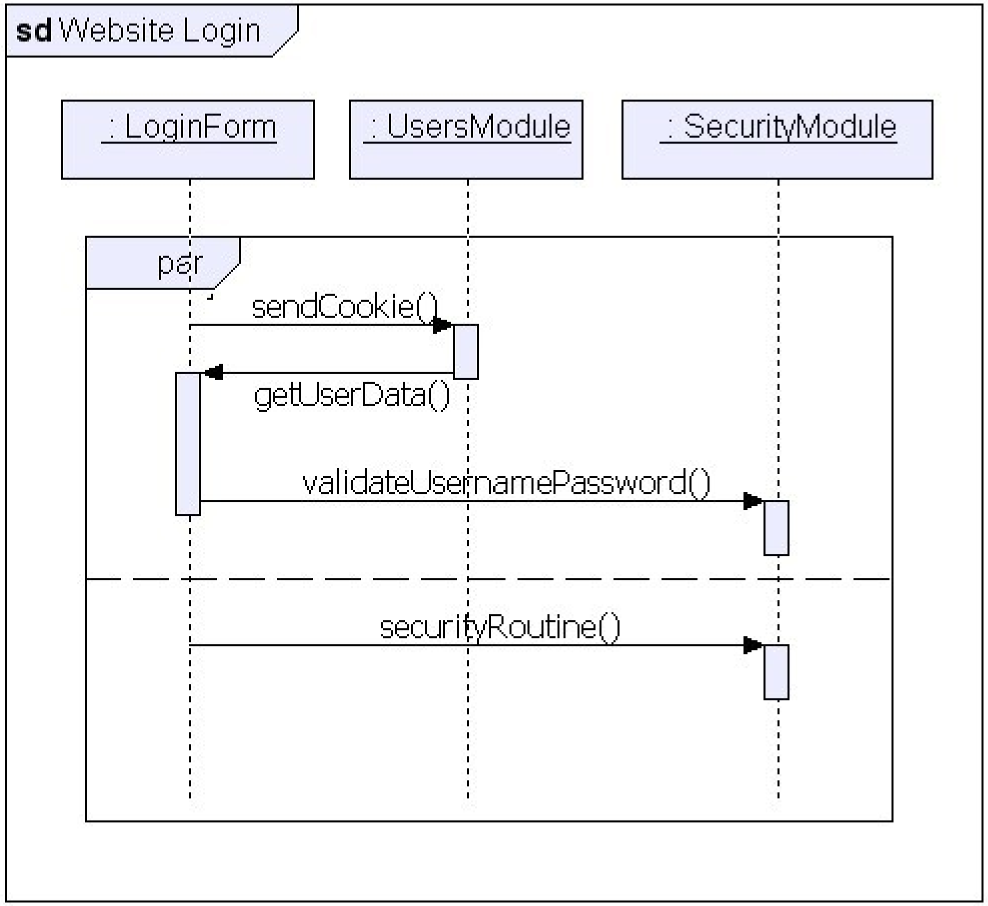
© 2008 Haim Michael. All Rights Reserved.

Parallel Interaction Fragment

- A parallel fragment means that each one of the associated interaction fragments can be executed concurrently.
- Top left pentagon should include the text “par”.
- The ordering of each one of the original fragments is maintained.

© 2008 Haim Michael. All Rights Reserved.

Parallel Interaction Fragment



© 2008 Haim Michael. All Rights Reserved.

Weak Sequencing Interaction Fragment

- A weak sequencing interaction fragment (top left pentagon should include the text “seq”) maintains the following rules regarding the event occurrences in each one of the operands:

  Event Occurrences Ordering

  If the first operand has <do_1, do_2, do_3> and the second one has <operate_1, operate_2, operate_3> the event occurrences within each operand must always be maintained. Having <do_1, operate_1, do_3, operate_2, operate_3, do_2> won't be legal as do_3 comes before do_2.

Weak Sequencing Interaction Fragment

Event Occurrences Ordering (Different Lifelines)

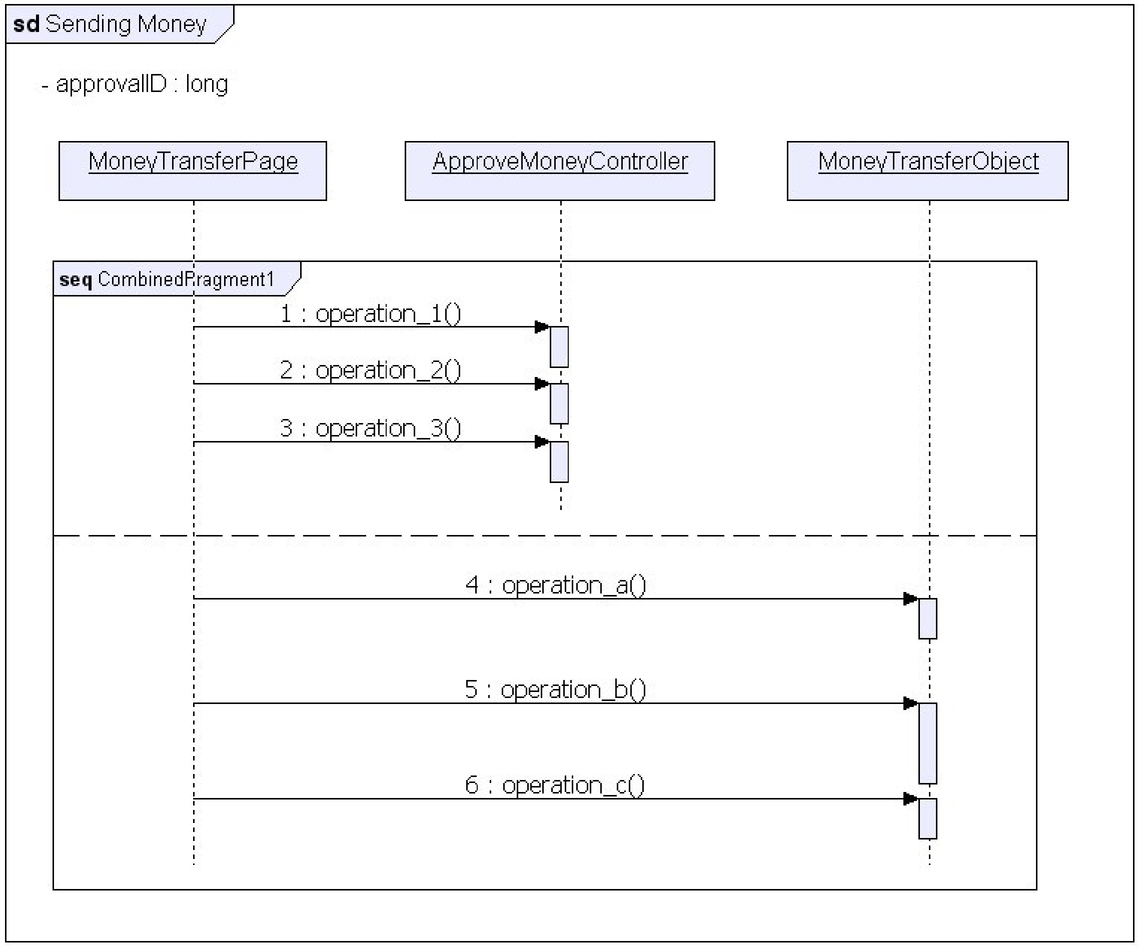
If the event occurrences are in different operands on different lifelines they can be interleaved in any order.

Event Occurrences Ordering (The Same Lifeline)

If the even occurrences are on the same lifeline they can be interleaved as long as all the event occurrences of the first operand execute before the occurrences of the second operand.

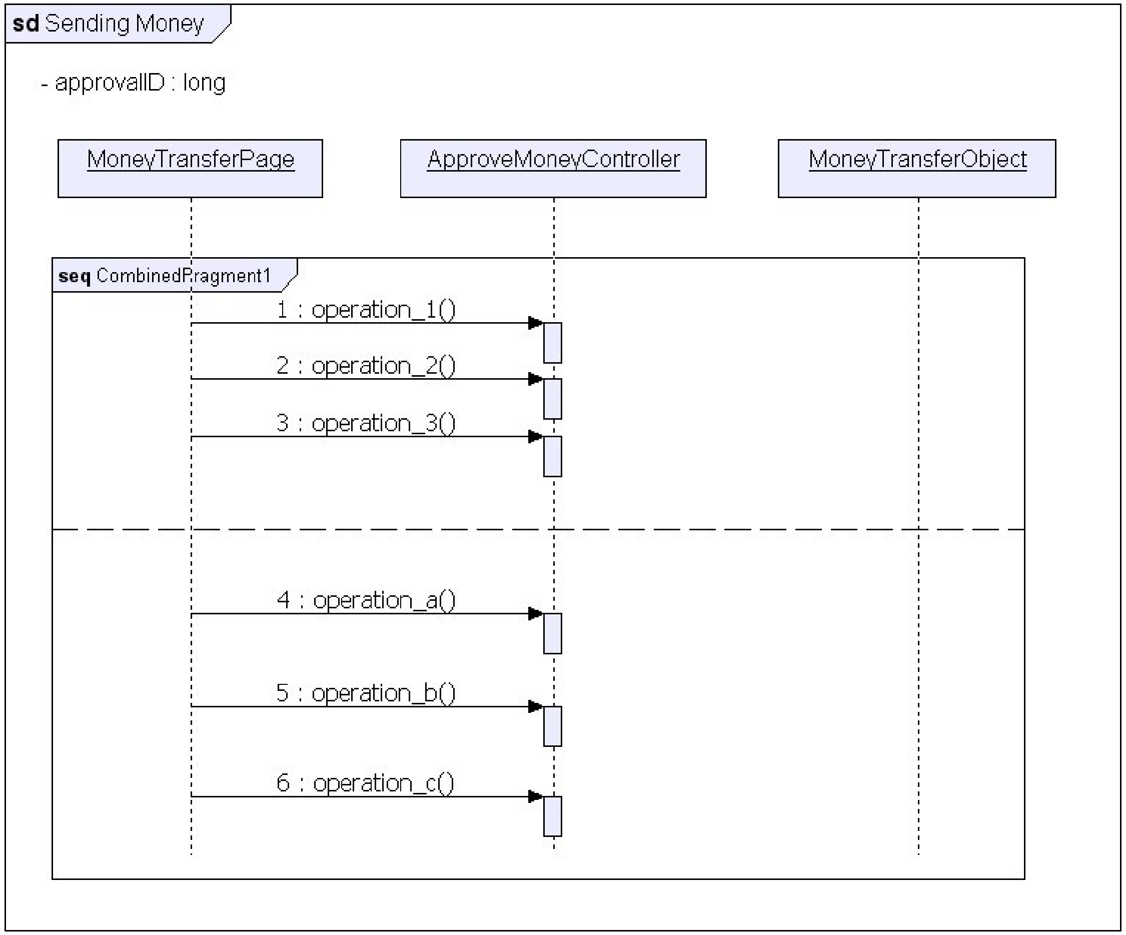
© 2008 Haim Michael. All Rights Reserved.

Weak Sequencing Interaction Fragment



© 2008 Haim Michael. All Rights Reserved.

Weak Sequencing Interaction Fragment



© 2008 Haim Michael. All Rights Reserved.

Strict Sequencing Interaction Fragment

- The ordering of the event occurrences must be kept in all cases, even if they happen within different separated life lines.
- Top left pentagon should include the text “strict”.
- The operands must be executed in order from top to bottom.

Negative Interaction Fragment

- The event occurrences are considered invalid and the interaction can never execute.
- Top left pentagon should include the text “neg”.

© 2008 Haim Michael. All Rights Reserved.

Critical Region Interaction Fragment

- The event occurrences must be treated as an atomic block.
- Top left pentagon should include the text “critical”.

© 2008 Haim Michael. All Rights Reserved.

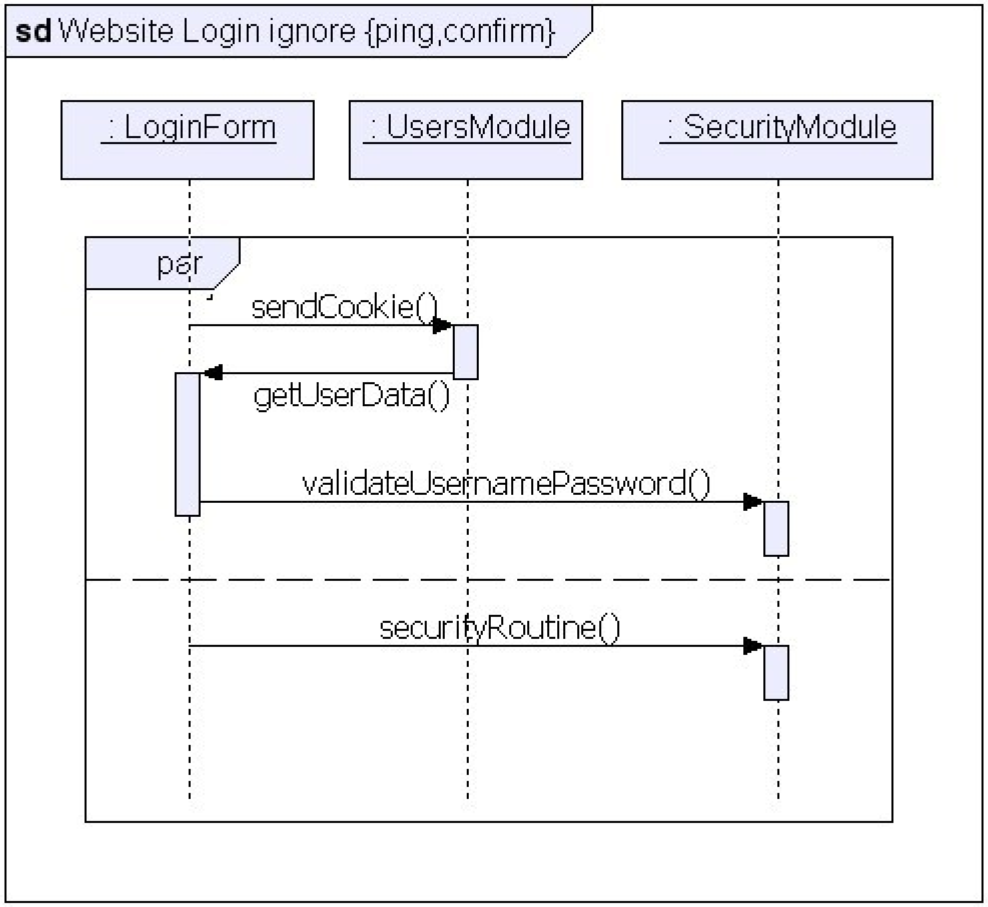
Ignore Interaction Fragment

- Using “ignore” we can specify set of messages that can be safely ignored. We should add the “ignore” operand to the pentagon of our fragment in the following format.

  ignore {messagename1, messagename2...}

© 2008 Haim Michael. All Rights Reserved.

Ignore Interaction Fragment



© 2008 Haim Michael. All Rights Reserved.

Consider Interaction Fragment

- Using “consider” we can specify the messages that are explicitly relevant to the diagram from which we cannot ignore, while from all others we can. If a message is shown in the consider list and yet it doesn't show on the diagram it won't occur.

  consider {messagename1, messagename2...}

Assertion Interaction Fragment

- Using “assert” we can indicate the contained event occurrences are the only valid execution path.
- Usually, a state invariant is added.

© 2008 Haim Michael. All Rights Reserved.

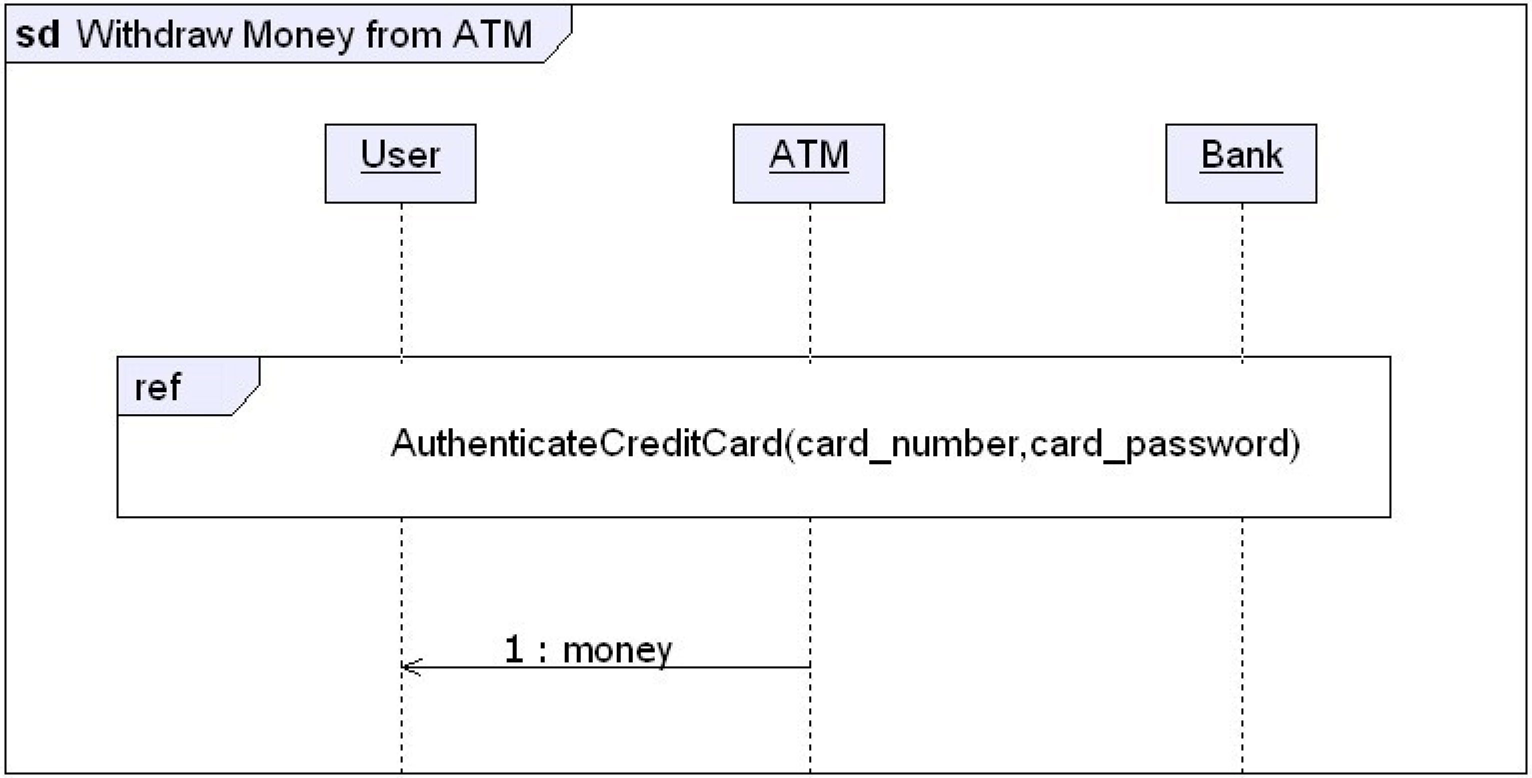
Interaction Occurrences

- When dealing with large interactions that include various interactions within each other you can simplify the diagram by drawing an interaction occurrence instead of a detailed interaction.
- The syntax for interaction occurrence includes a combined fragment rectangle with the “ref” text on its top left and the interaction name within the rectangle.

Interaction Occurrences

- The interaction occurrence references a detailed interaction you can draw in a separated diagram.
- UML specifications allow sending arguments to the referenced interaction using the following syntax:
  attribute_name = interaction_name(arguments) : returned_value

Interaction Occurrences



sd Withdraw Money from ATM
User
ATM
Bank
ref
AuthenticateCreditCard(card_number,card_password)
1 : money

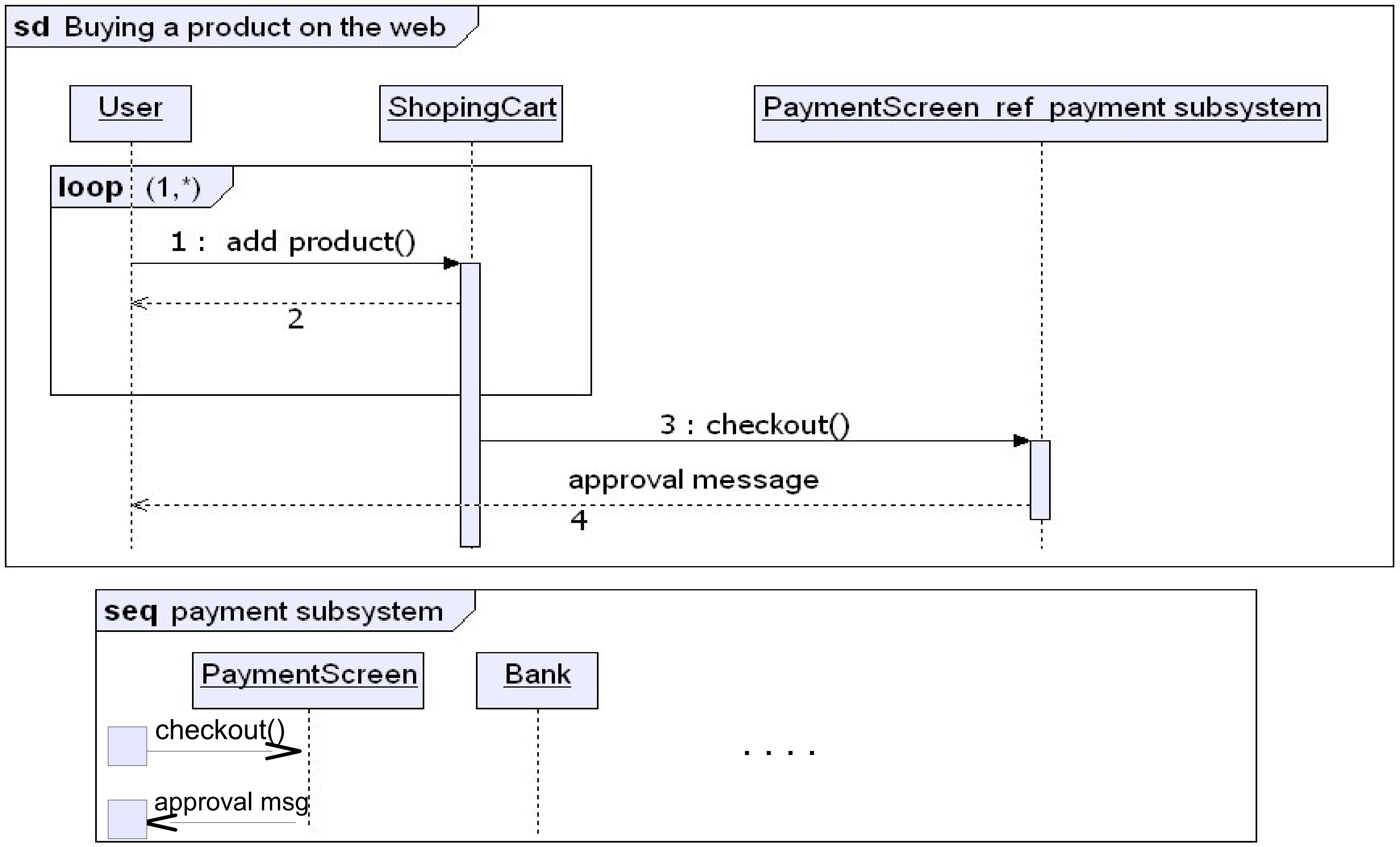
© 2008 Haim Michael. All Rights Reserved.

Decomposition Interaction Diagram

- UML allows linking separated interaction diagrams by adding a decomposition reference from a specific participant to a separated interaction diagram.
- We can include within the secondary diagram input/output gates that match the message the specific participant get and its reply.
- The name of the secondary diagram will be added to the primary diagram specific participant name prefix with “ref”.

© 2008 Haim Michael. All Rights Reserved.

Decomposition Interaction Diagram



© 2008 Haim Michael. All Rights Reserved.

Continuations

- Using continuations we can define different alternative branches.
- The notation is a rectangle with rounded sides.
- Placing a continuation in the beginning of the interaction is done to define the behavior of that continuation.
- Placing a continuation at the end of the interaction is to indicate how the interaction should continue.

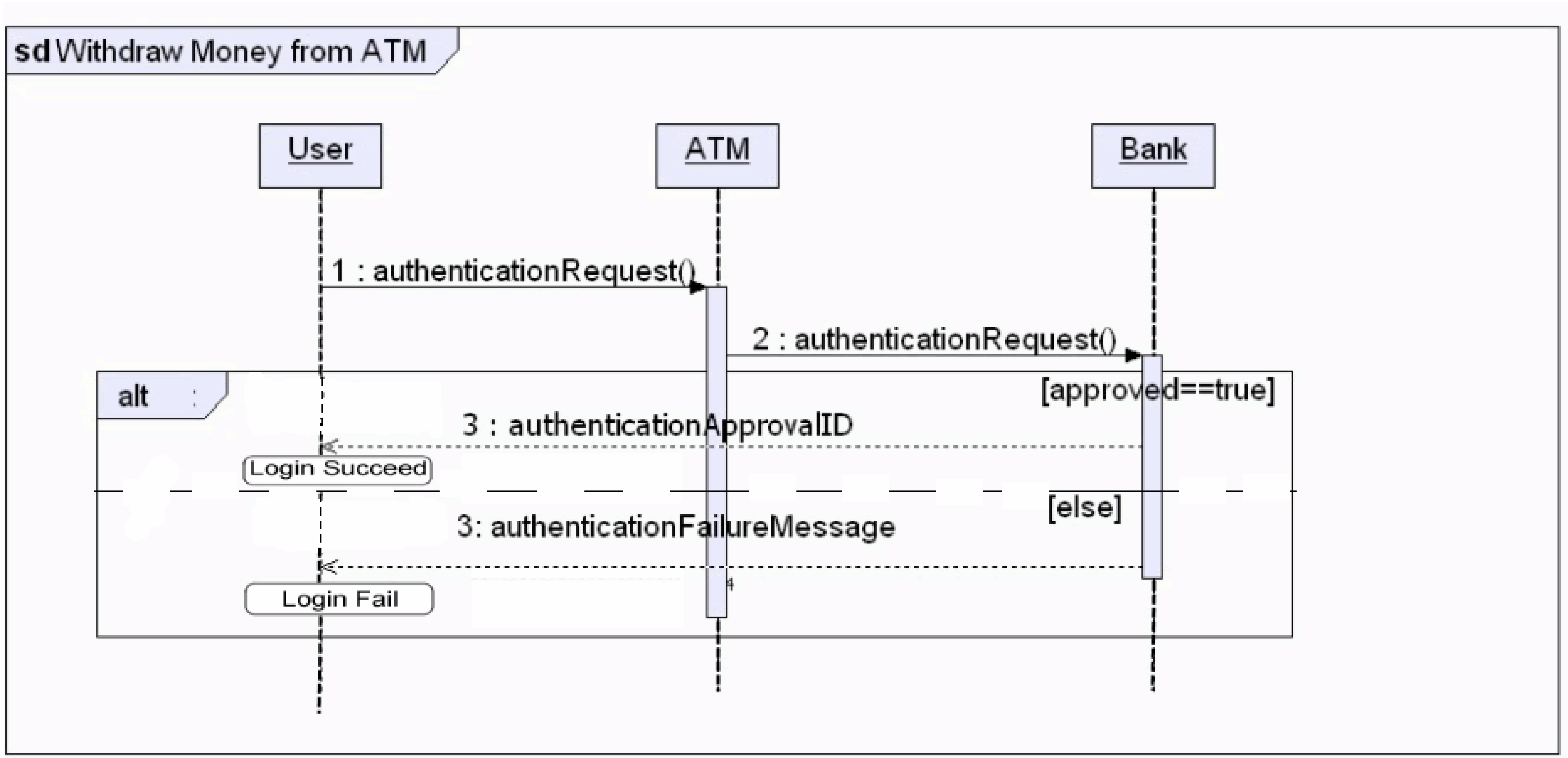
© 2008 Haim Michael. All Rights Reserved.

Continuations

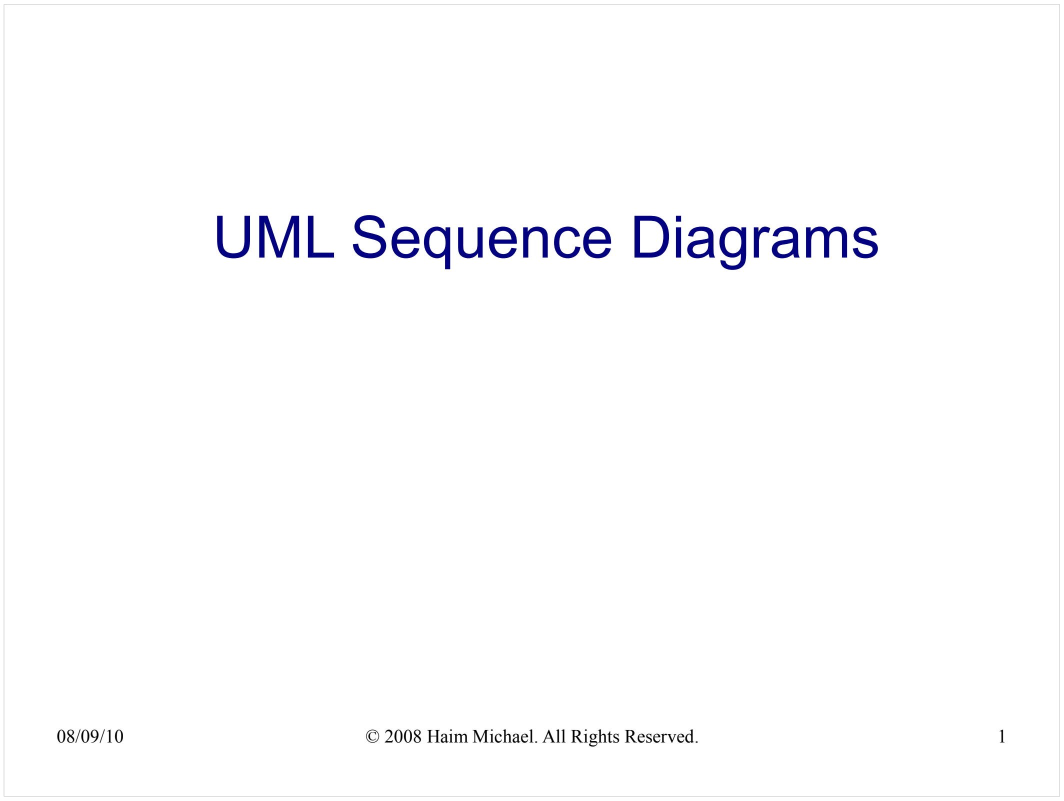
- When drawing the continuations they must cover the exact same life lines, both in the diagram that defines them and in the diagram that uses them.

© 2008 Haim Michael. All Rights Reserved.

Continuations

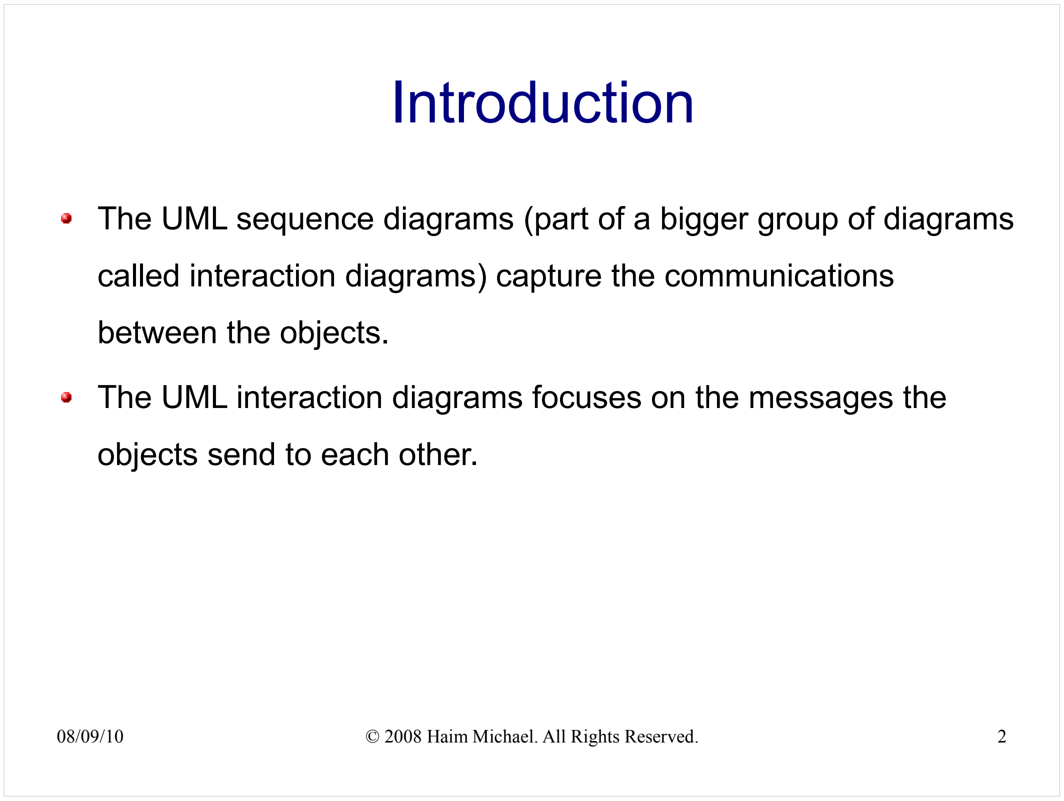

sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
alt
[approved==true]
3 : authenticationApprovalID
Login Succeed
[else]
3: authenticationFailureMessage
Login Fail

© 2008 Haim Michael. All Rights Reserved.



UML Sequence Diagrams

08/09/10 © 2008 Haim Michael. All Rights Reserved. 1

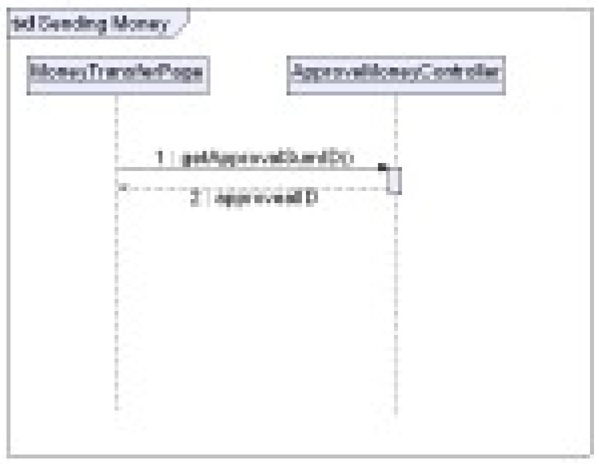


Introduction

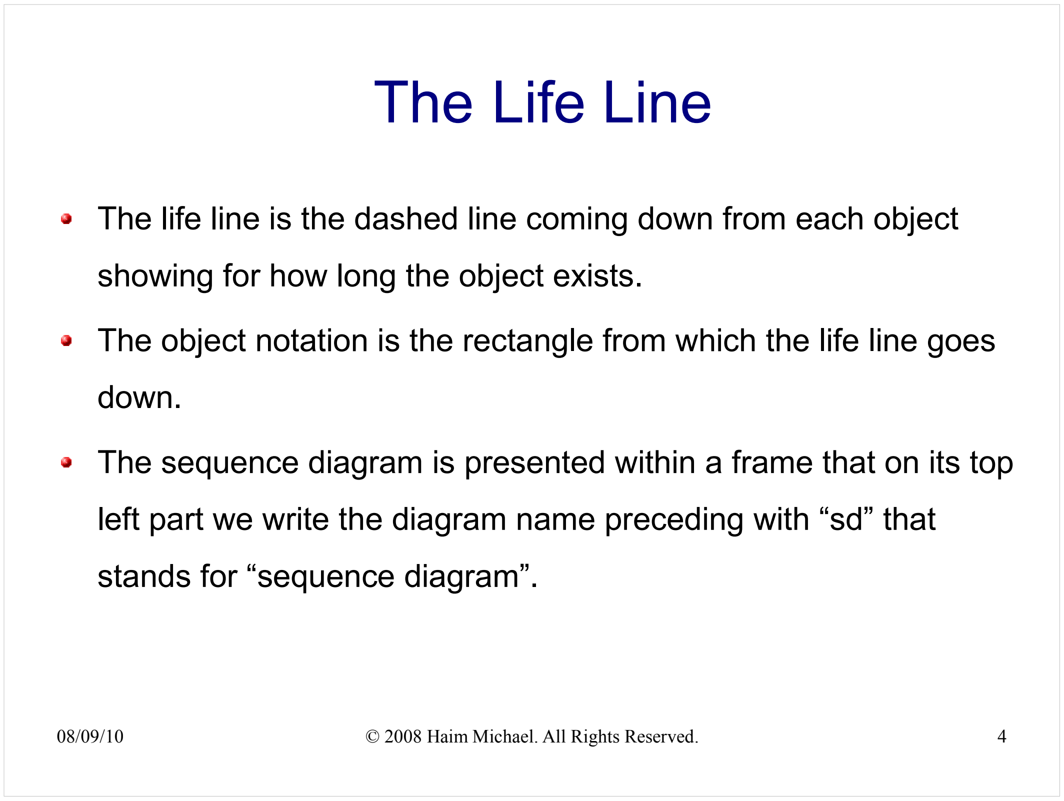
- The UML sequence diagrams (part of a bigger group of diagrams called interaction diagrams) capture the communications between the objects.
- The UML interaction diagrams focuses on the messages the objects send to each other.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 2

Introduction



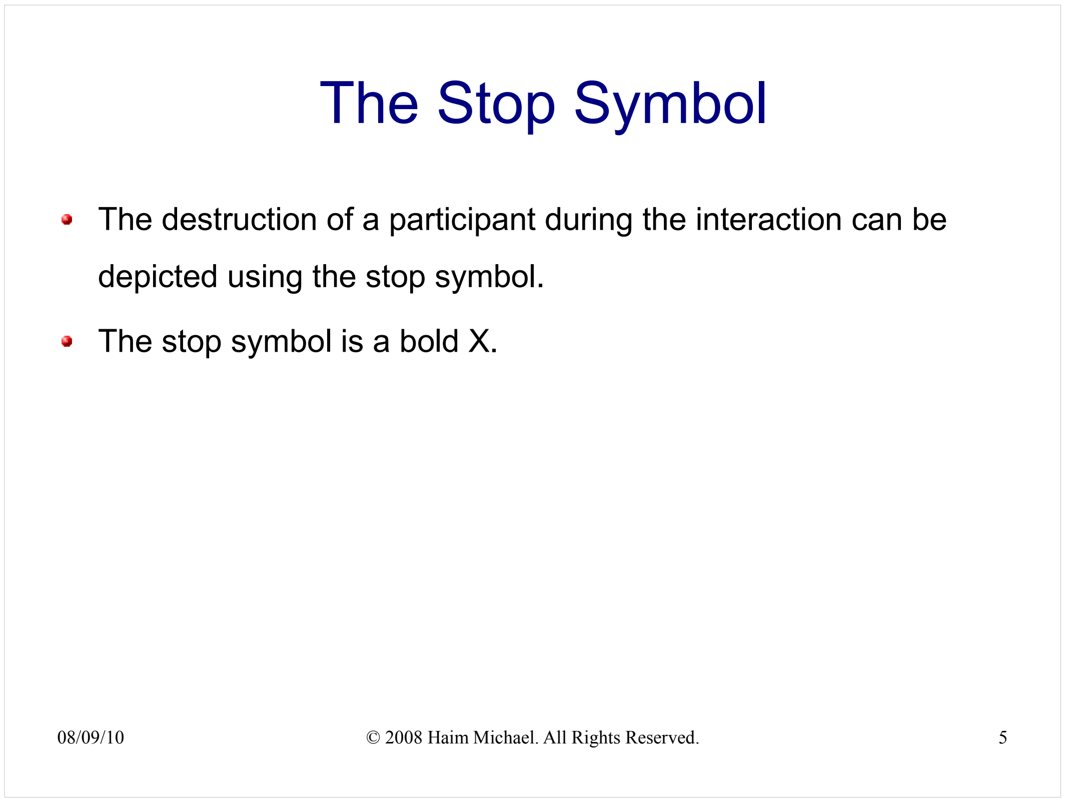
You can note that on top left of the frame we write the sequence diagram name starting with sd (stands for Sequence Diagram).



The Life Line

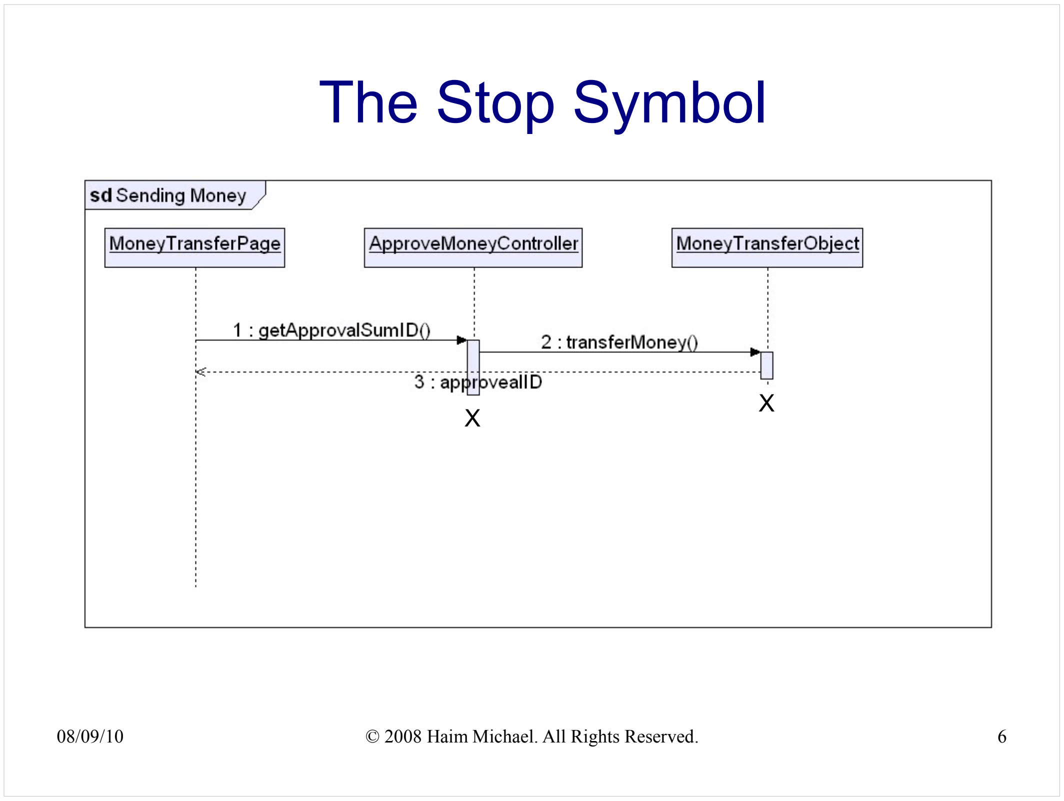
- The life line is the dashed line coming down from each object showing for how long the object exists.
- The object notation is the rectangle from which the life line goes down.
- The sequence diagram is presented within a frame that on its top left part we write the diagram name preceding with “sd” that stands for “sequence diagram”.

Please note the notation for life line might be different when dealing with other diagrams (e.g. communication diagram).

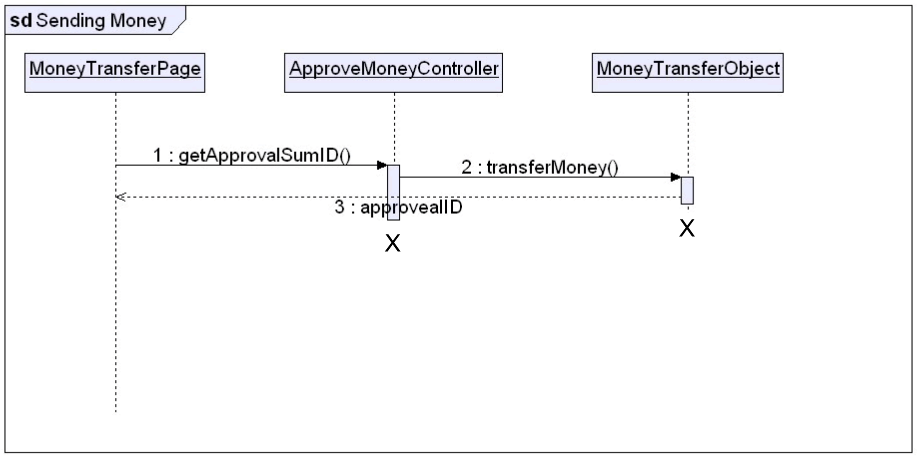


The Stop Symbol

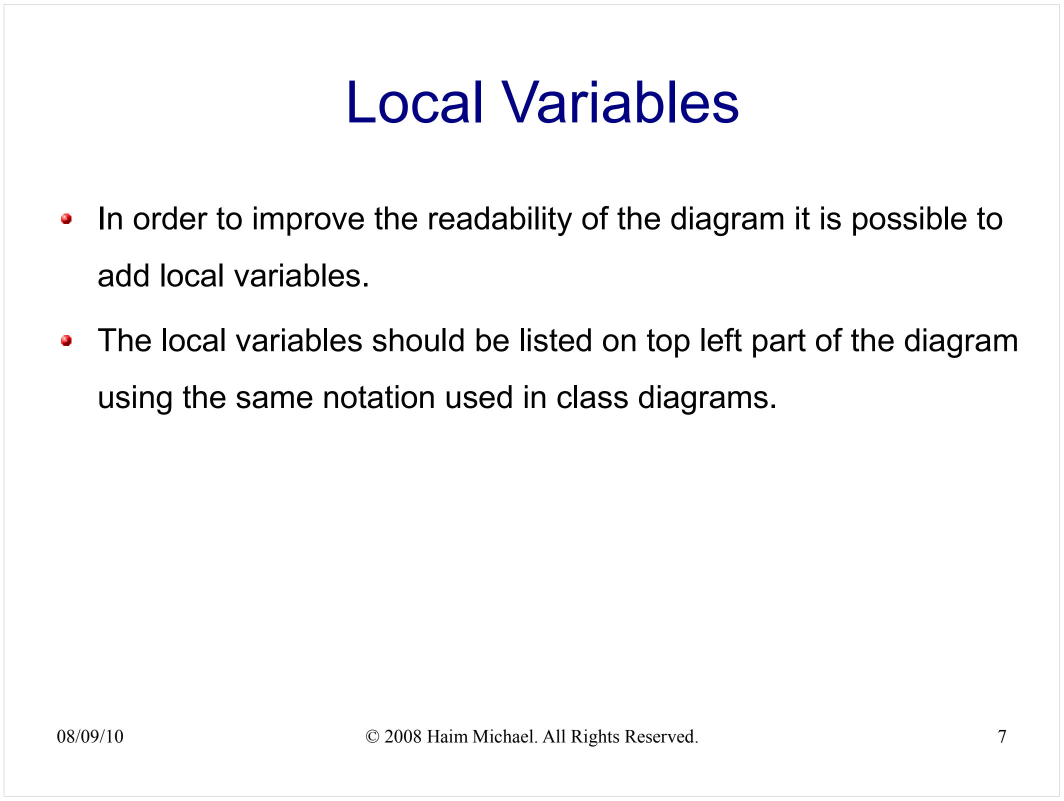
- The destruction of a participant during the interaction can be depicted using the stop symbol.
- The stop symbol is a bold X.



The Stop Symbol



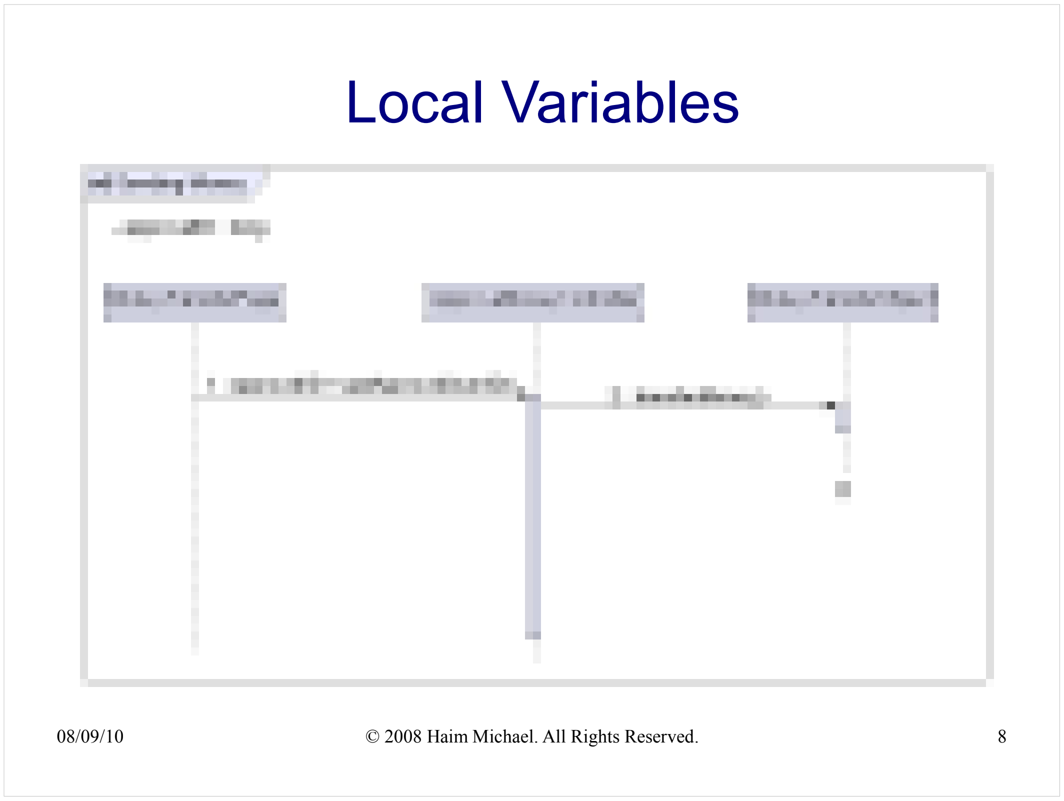
08/09/10 © 2008 Haim Michael. All Rights Reserved. 6



Local Variables

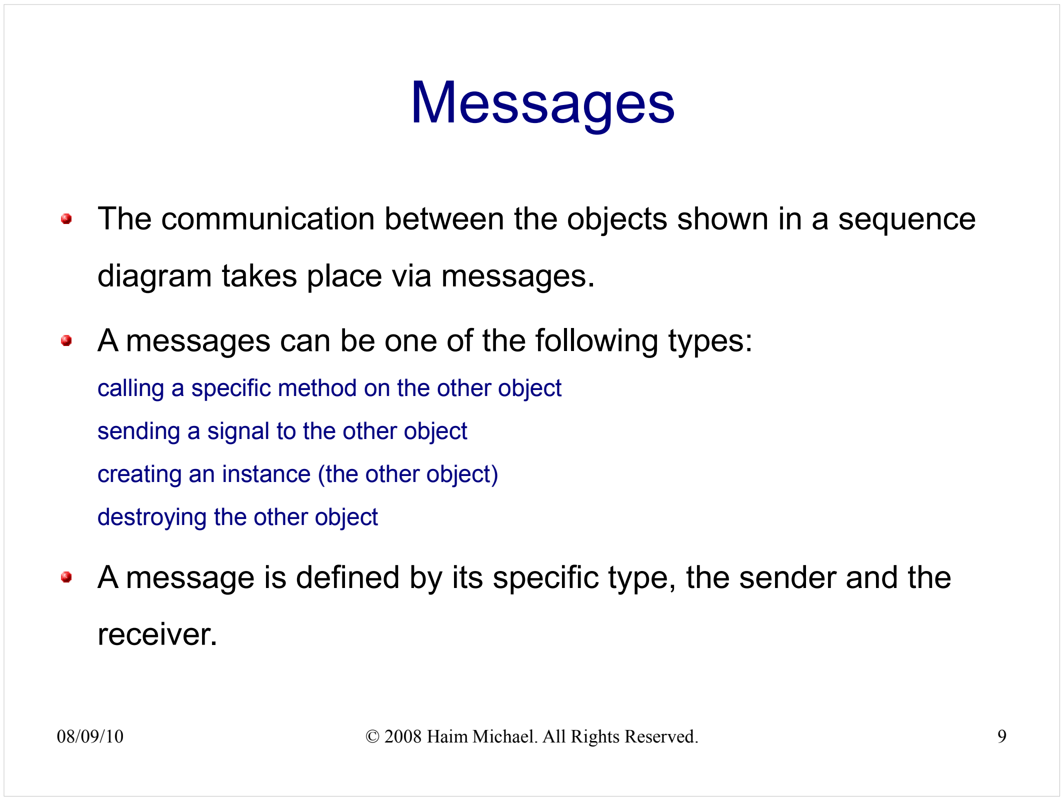
- In order to improve the readability of the diagram it is possible to add local variables.
- The local variables should be listed on top left part of the diagram using the same notation used in class diagrams.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 7



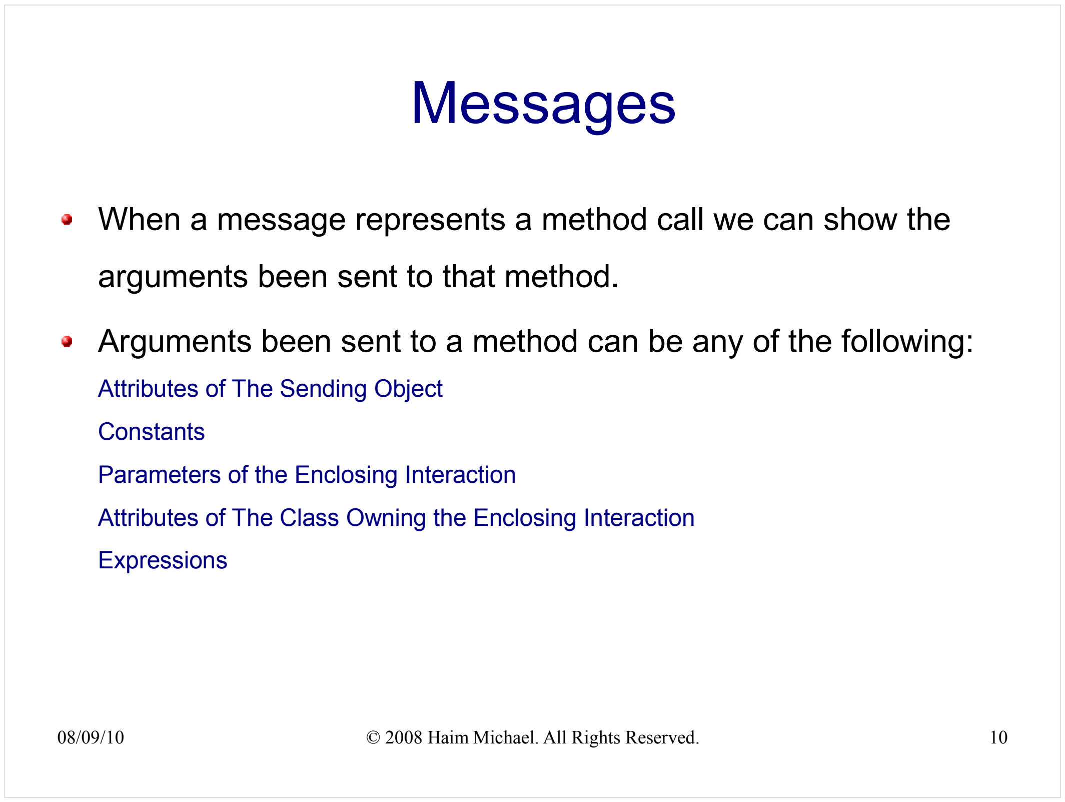
Local Variables

08/09/10 © 2008 Haim Michael. All Rights Reserved. 8



Messages

- The communication between the objects shown in a sequence diagram takes place via messages.
- A messages can be one of the following types:
  - calling a specific method on the other object
  - sending a signal to the other object
  - creating an instance (the other object)
  - destroying the other object
- A message is defined by its specific type, the sender and the receiver.



Messages

- When a message represents a method call we can show the arguments been sent to that method.
- Arguments been sent to a method can be any of the following:

  Attributes of The Sending Object

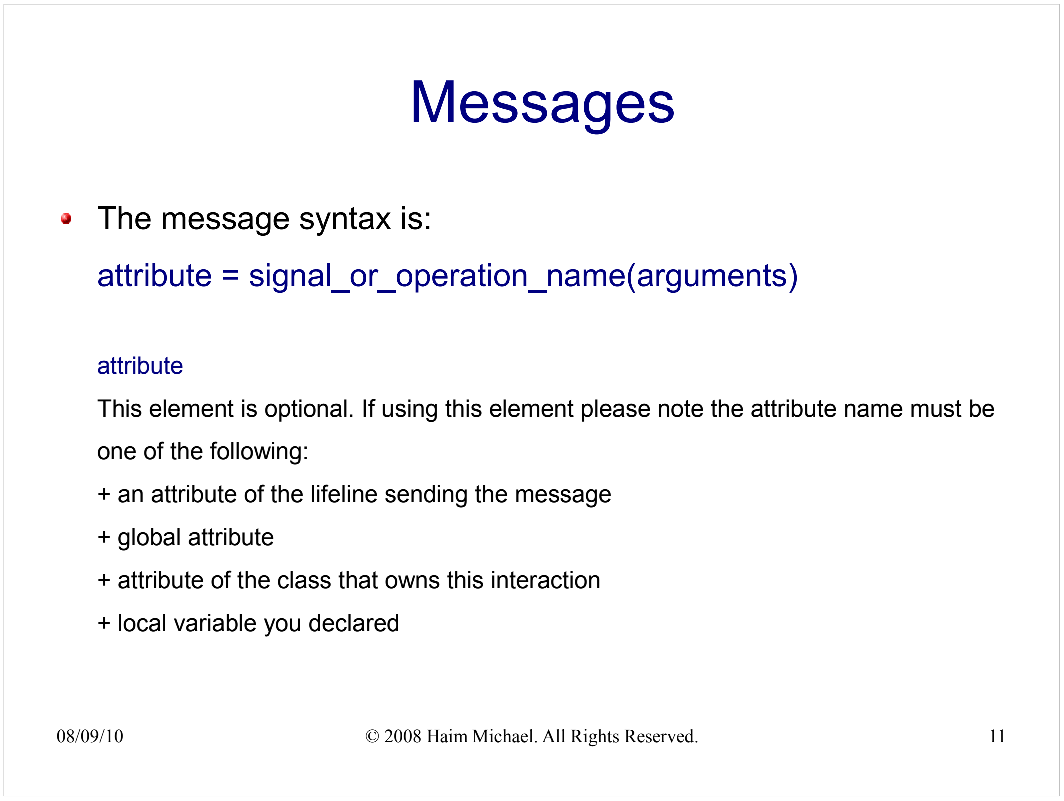
  Constants

  Parameters of the Enclosing Interaction

  Attributes of The Class Owning the Enclosing Interaction

  Expressions

08/09/10 © 2008 Haim Michael. All Rights Reserved. 10



Messages

- The message syntax is:

  attribute = signal_or_operation_name(arguments)

  attribute

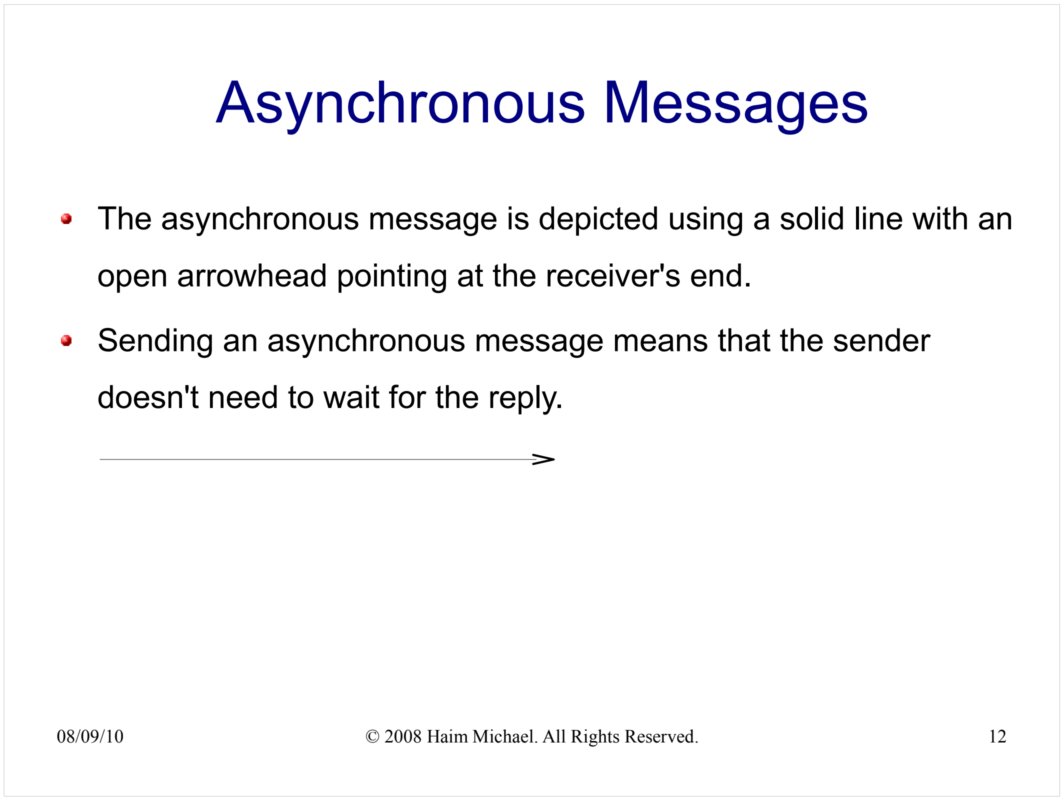
  This element is optional. If using this element please note the attribute name must be one of the following:

  + an attribute of the lifeline sending the message

  + global attribute

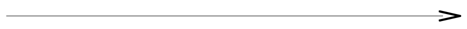
  + attribute of the class that owns this interaction

  + local variable you declared



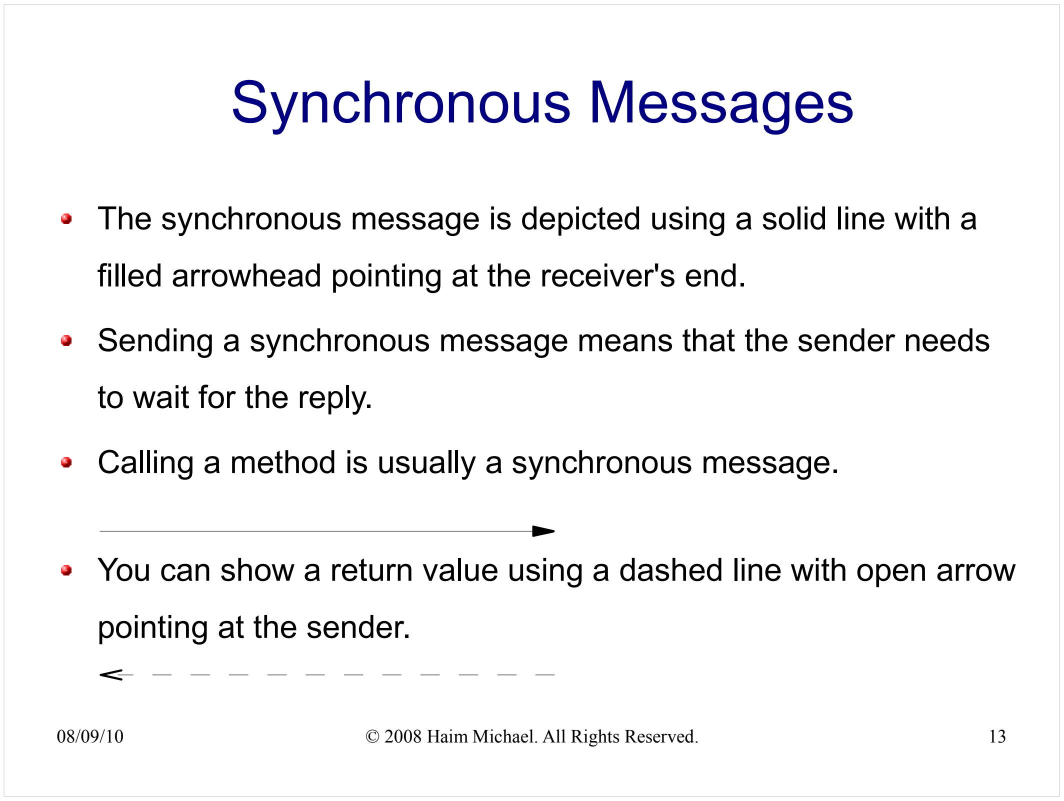
Asynchronous Messages

- The asynchronous message is depicted using a solid line with an open arrowhead pointing at the receiver's end.
- Sending an asynchronous message means that the sender doesn't need to wait for the reply.



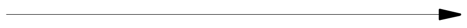
© 2008 Haim Michael. All Rights Reserved.

© 2008 Haim Michael. All Rights Reserved.

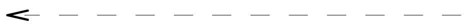


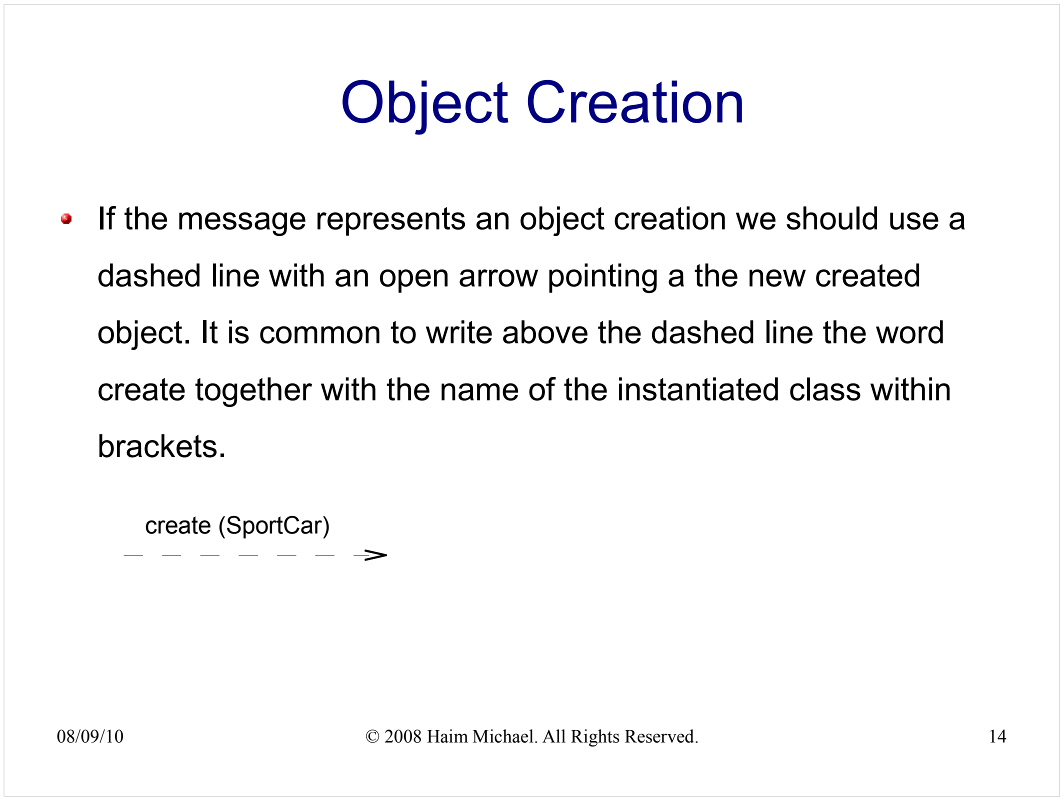
Synchronous Messages

- The synchronous message is depicted using a solid line with a filled arrowhead pointing at the receiver's end.
- Sending a synchronous message means that the sender needs to wait for the reply.
- Calling a method is usually a synchronous message.



- You can show a return value using a dashed line with open arrow pointing at the sender.



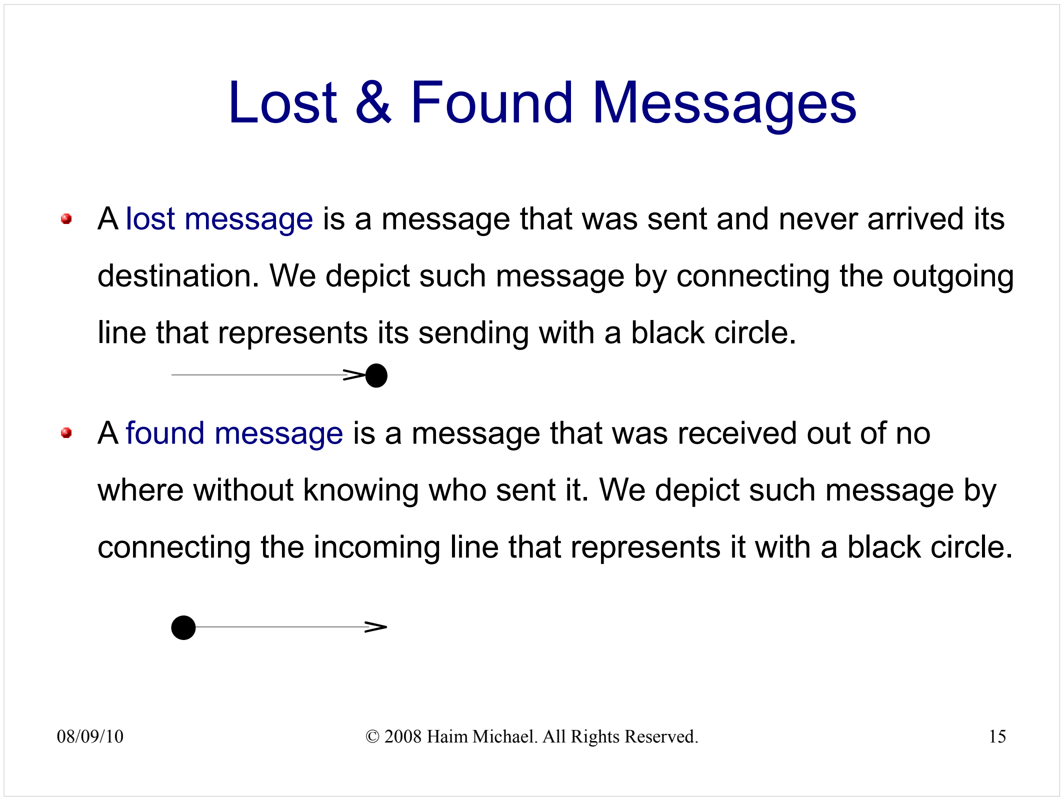


Object Creation

- If the message represents an object creation we should use a dashed line with an open arrow pointing a the new created object. It is common to write above the dashed line the word create together with the name of the instantiated class within brackets.

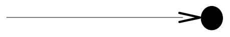
$$\overset{\text{create (SportCar)}}{- - - - - - - \!\!>}$$

08/09/10 © 2008 Haim Michael. All Rights Reserved. 14

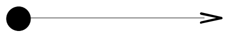


Lost & Found Messages

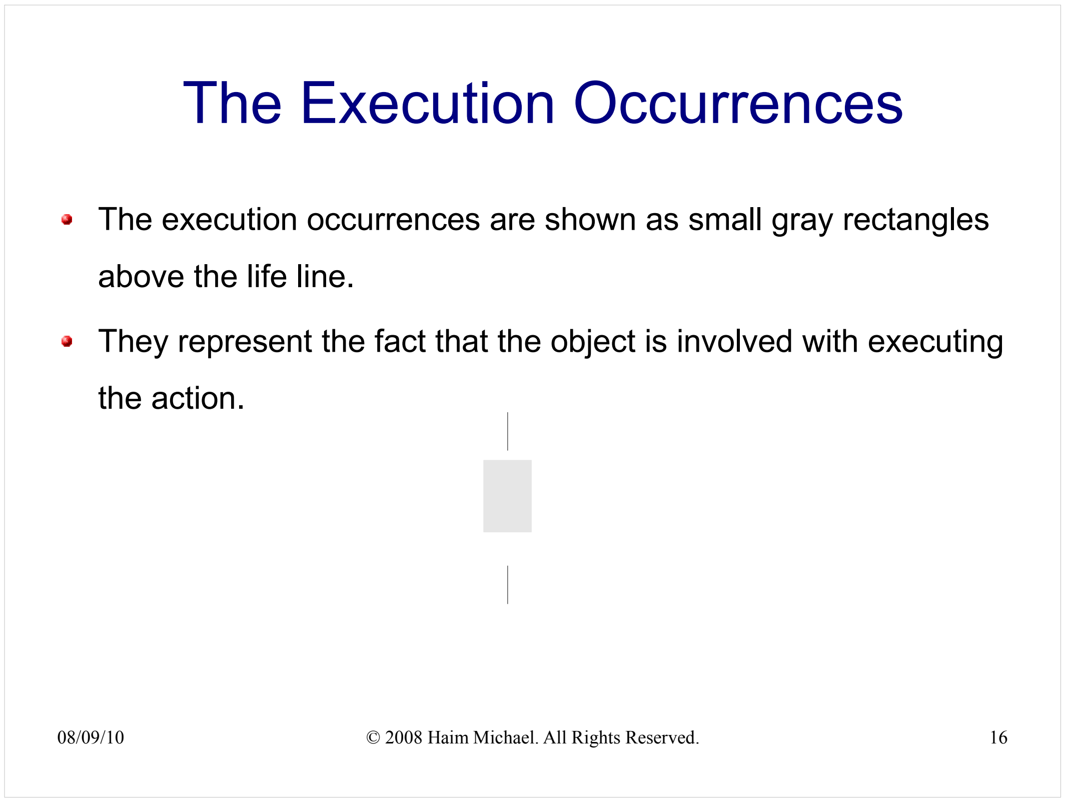
- A lost message is a message that was sent and never arrived its destination. We depict such message by connecting the outgoing line that represents its sending with a black circle.



- A found message is a message that was received out of no where without knowing who sent it. We depict such message by connecting the incoming line that represents it with a black circle.



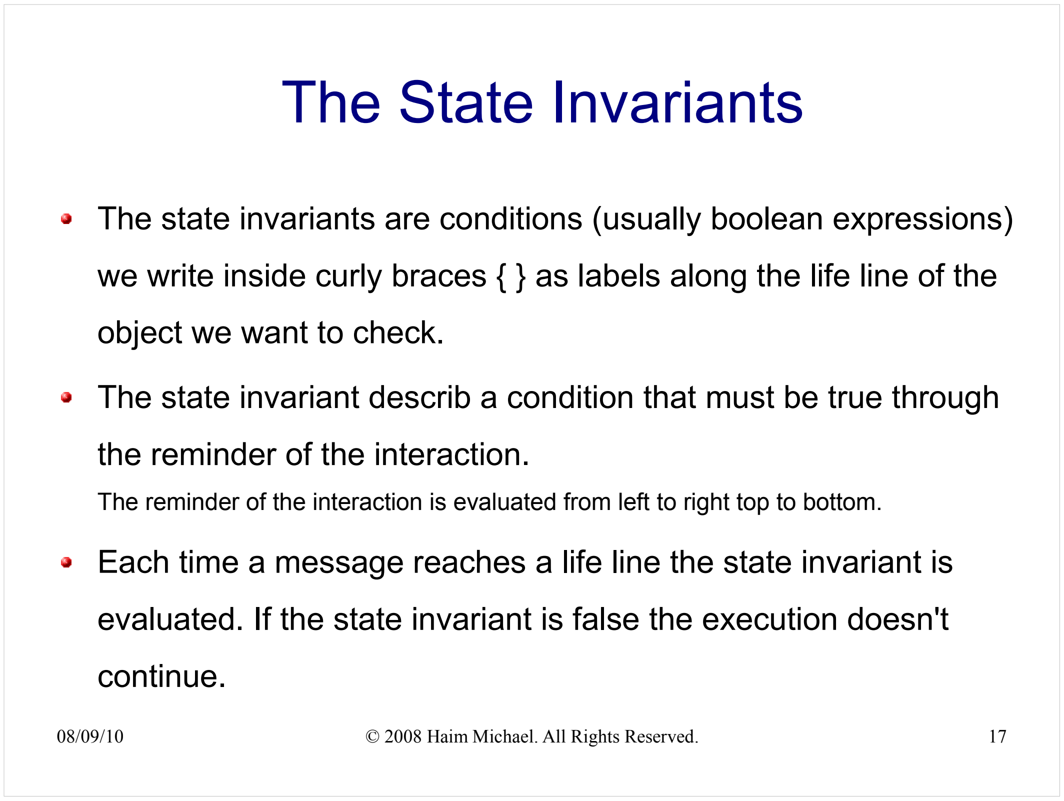
08/09/10 © 2008 Haim Michael. All Rights Reserved. 15



The Execution Occurrences

- The execution occurrences are shown as small gray rectangles above the life line.
- They represent the fact that the object is involved with executing the action.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 16

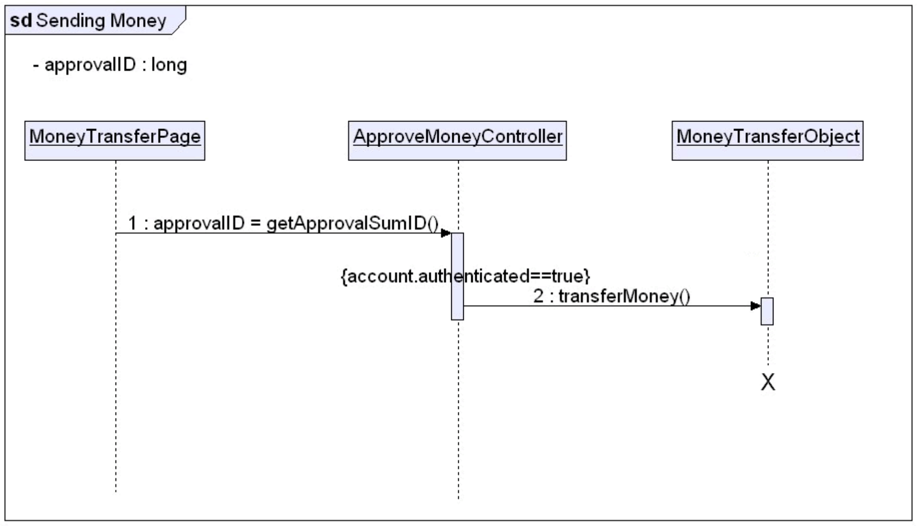


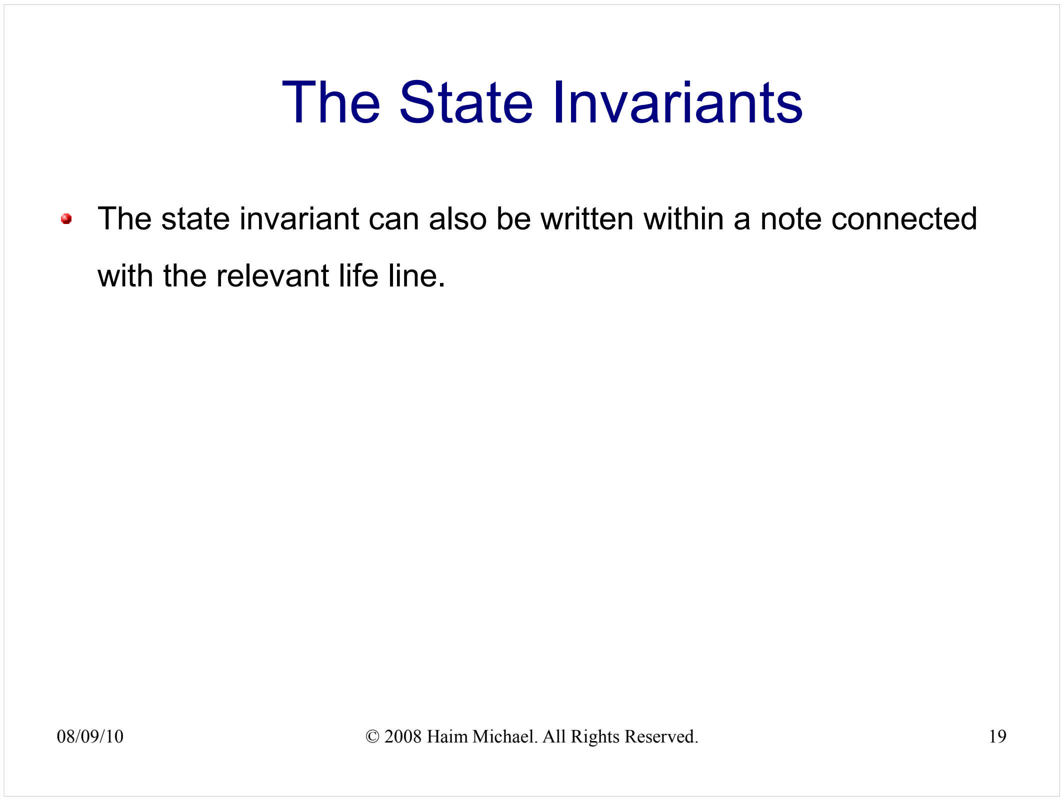
The State Invariants

- The state invariants are conditions (usually boolean expressions) we write inside curly braces { } as labels along the life line of the object we want to check.
- The state invariant describ a condition that must be true through the reminder of the interaction.
  The reminder of the interaction is evaluated from left to right top to bottom.
- Each time a message reaches a life line the state invariant is evaluated. If the state invariant is false the execution doesn't continue.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 17

The State Invariants





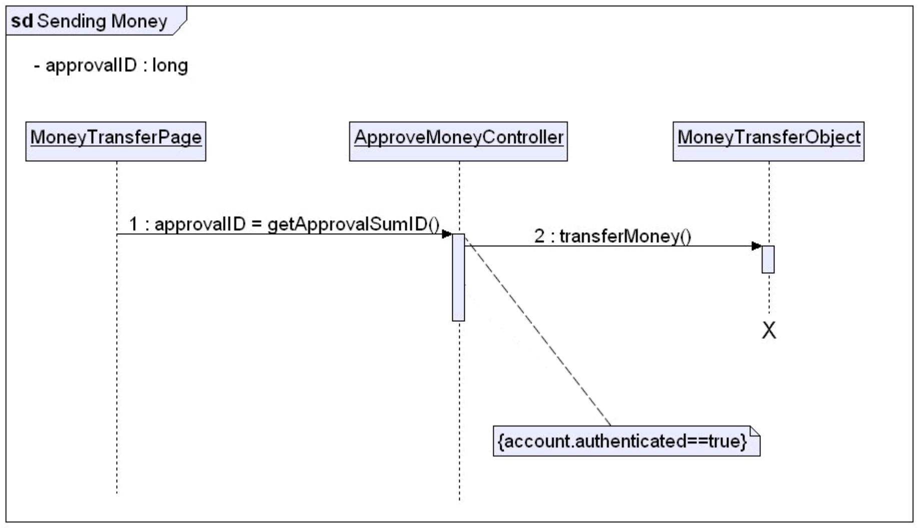
The State Invariants

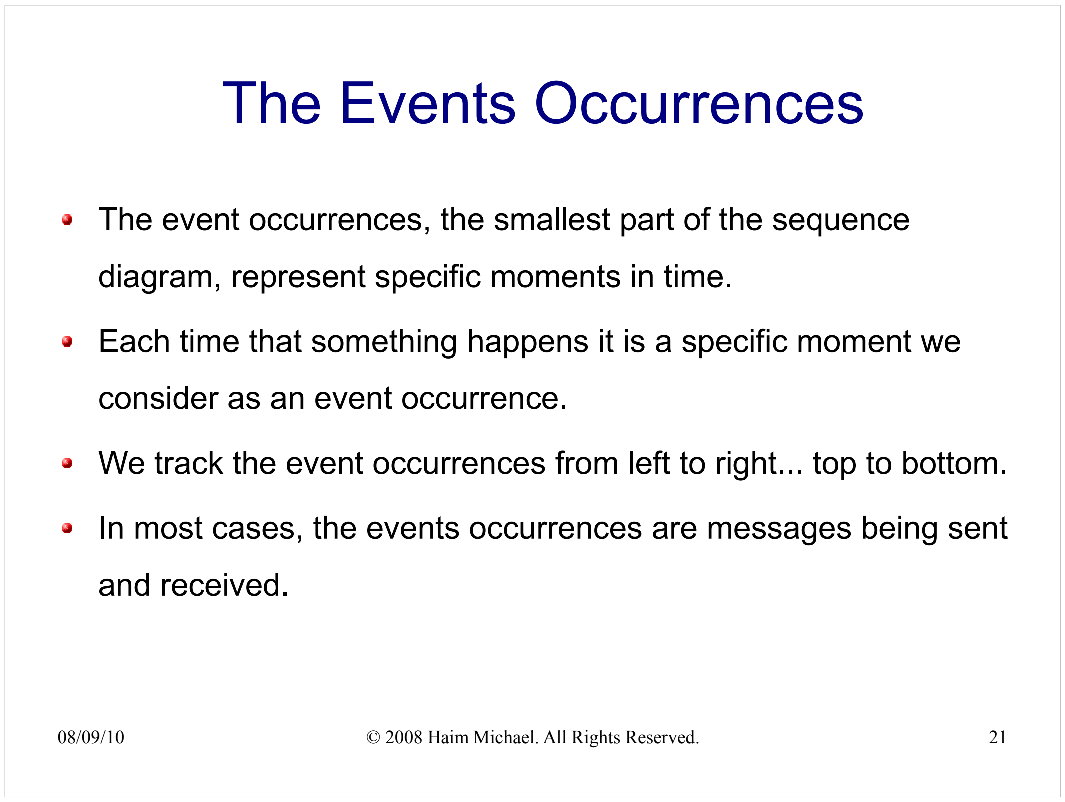
- The state invariant can also be written within a note connected with the relevant life line.

© 2008 Haim Michael. All Rights Reserved.

© 2008 Haim Michael. All Rights Reserved.

The State Invariants



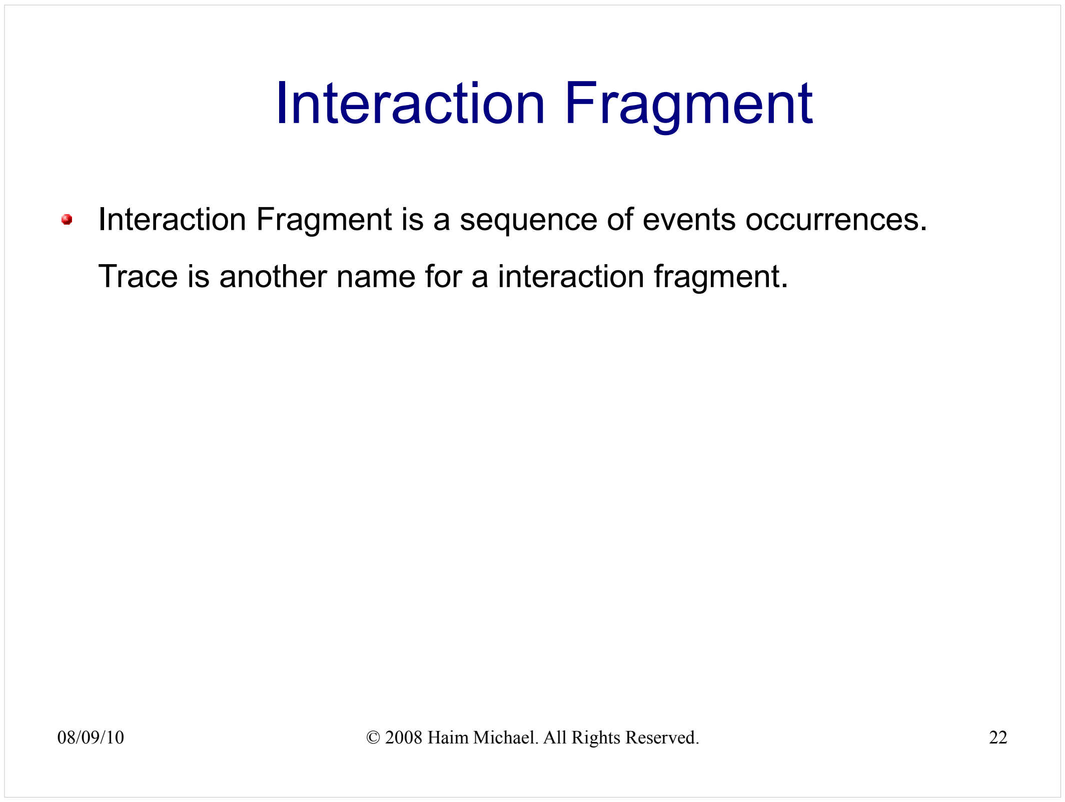


The Events Occurrences

- The event occurrences, the smallest part of the sequence diagram, represent specific moments in time.
- Each time that something happens it is a specific moment we consider as an event occurrence.
- We track the event occurrences from left to right... top to bottom.
- In most cases, the events occurrences are messages being sent and received.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 21

When a message is being sent and being received by another object we can identify two events occurrences. The message being sent is the first and the message being received is the second.



Interaction Fragment

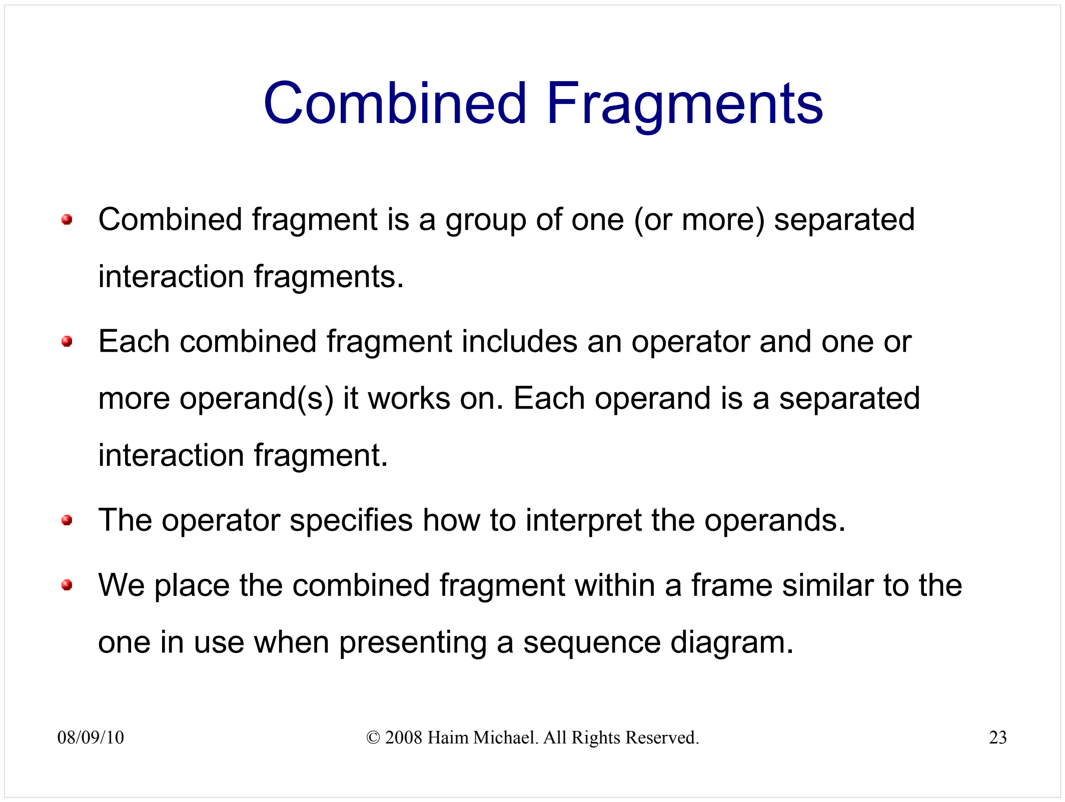
- Interaction Fragment is a sequence of events occurrences. Trace is another name for a interaction fragment.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 22

Interaction fragments can be combined into combined fragments. Each combined fragment has an operator that works on the combined fragment operand\s. The available operators are: alternatives, option, break, parallel, weak sequence, strict, negative, critical region, ignore/consider, assertion and loop.

We write the operand within a pentagon in the upper left of the combined fragment rectangle.

When having more than one operand a dashed horizontal line should cross the rectangle horizontally.



Combined Fragments

- Combined fragment is a group of one (or more) separated interaction fragments.
- Each combined fragment includes an operator and one or more operand(s) it works on. Each operand is a separated interaction fragment.
- The operator specifies how to interpret the operands.
- We place the combined fragment within a frame similar to the one in use when presenting a sequence diagram.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 23

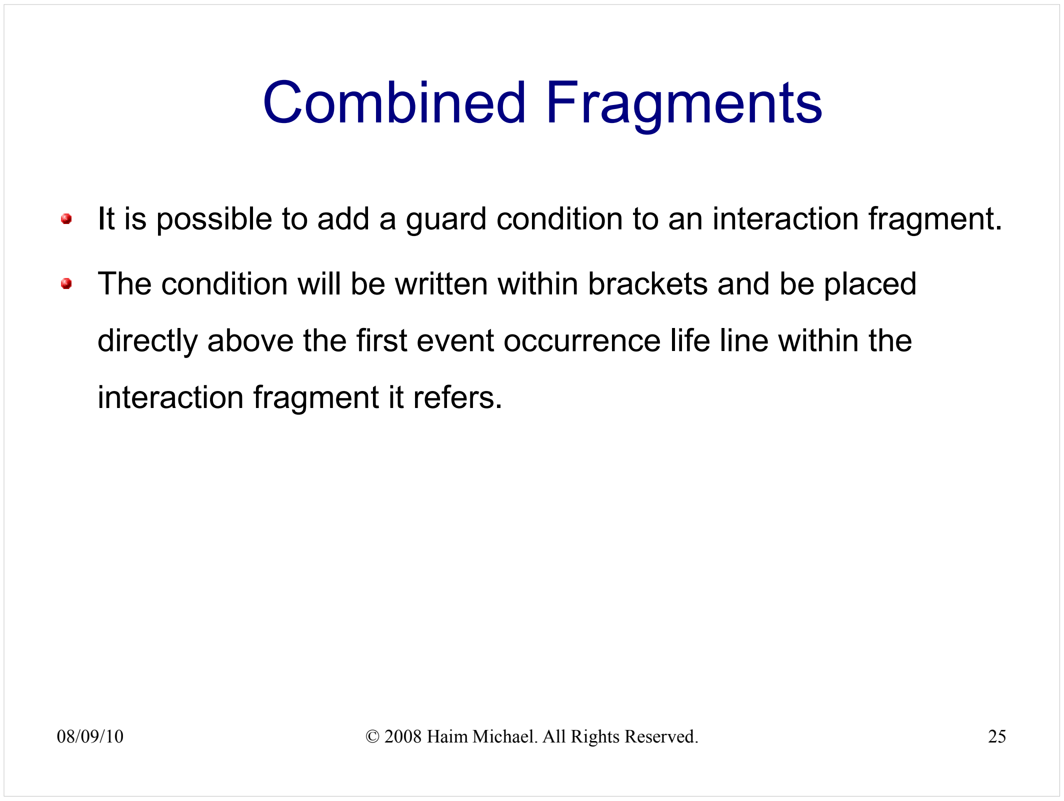
Interaction fragments can be combined into combined fragments. Each combined fragment has an operator that works on the combined fragment operand\s. The available operators are: alternatives, option, break, parallel, weak sequence, strict, negative, critical region, ignore/consider, assertion and loop.

We write the operand within a pentagon in the upper left of the combined fragment rectangle.

When having more than one operand a dashed horizontal line should cross the rectangle horizontally.

Combined Fragments

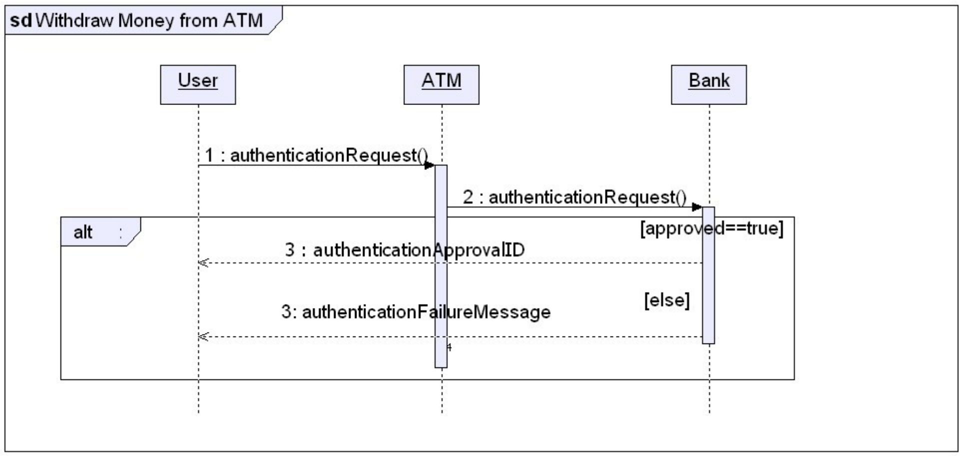
08/09/10 © 2008 Haim Michael. All Rights Reserved. 24



Combined Fragments

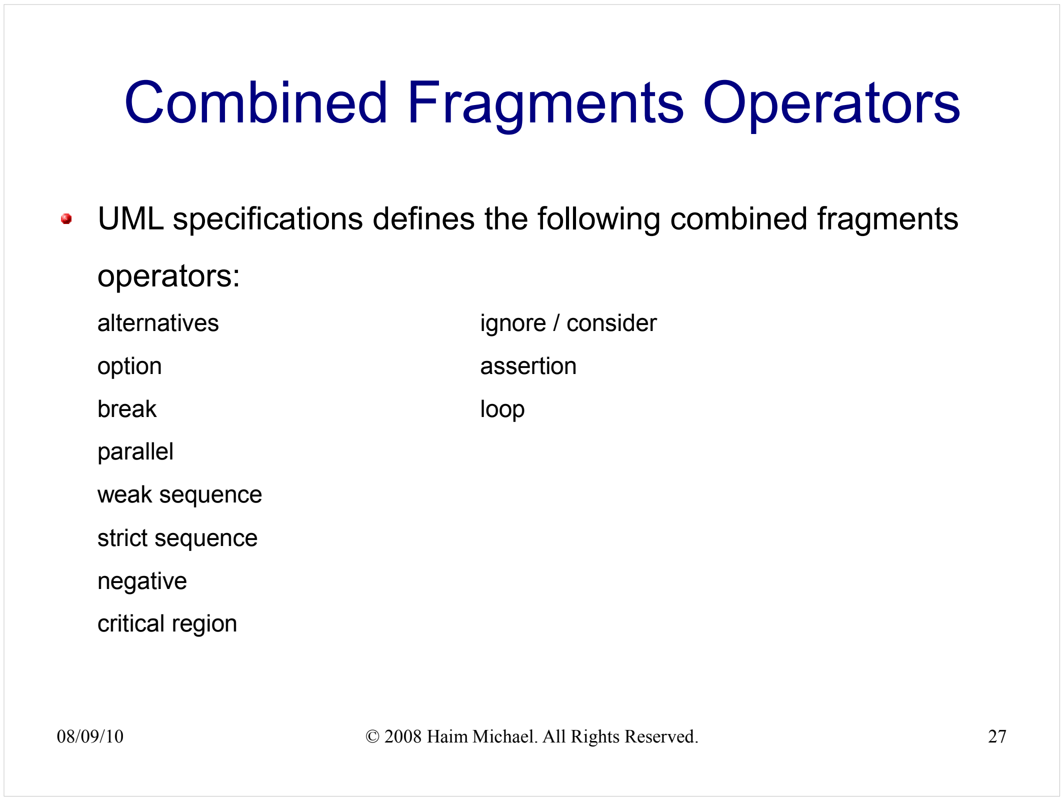
- It is possible to add a guard condition to an interaction fragment.
- The condition will be written within brackets and be placed directly above the first event occurrence life line within the interaction fragment it refers.

Combined Fragments


sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
alt
[approved==true]
3 : authenticationApprovalID
[else]
3: authenticationFailureMessage

© 2008 Haim Michael. All Rights Reserved.

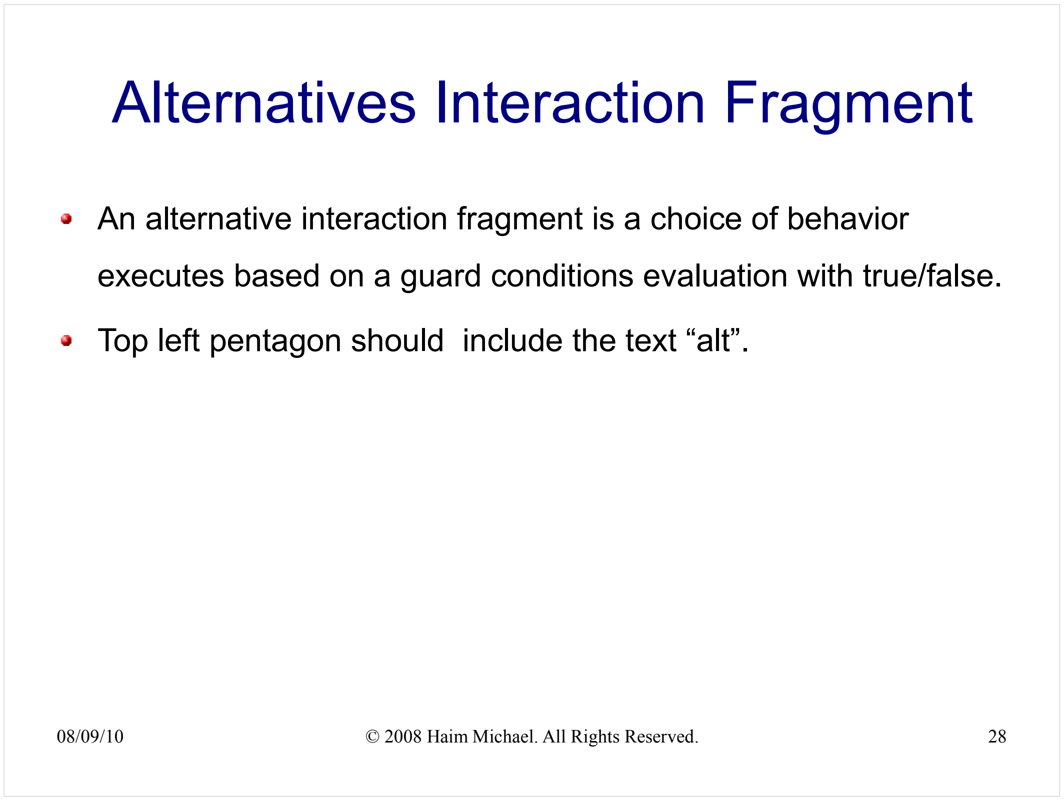
© 2008 Haim Michael. All Rights Reserved.



Combined Fragments Operators

- UML specifications defines the following combined fragments operators:

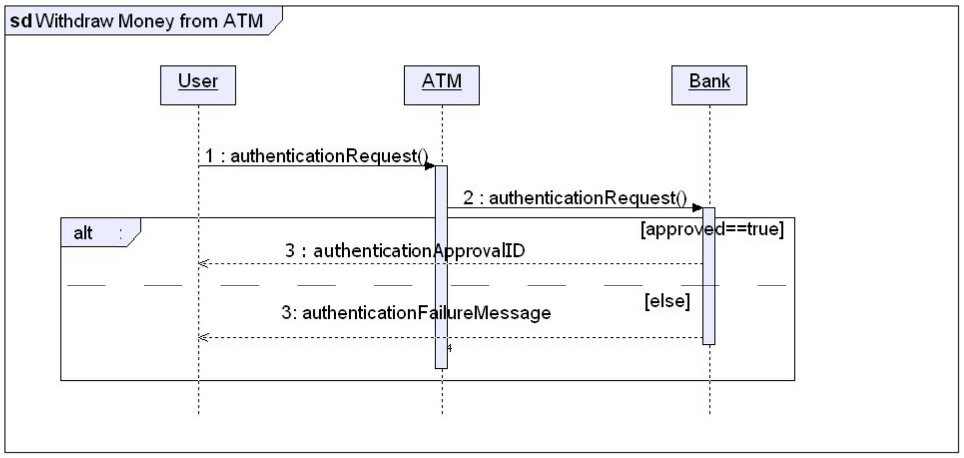
alternatives
option
break
parallel
weak sequence
strict sequence
negative
critical region
ignore / consider
assertion
loop



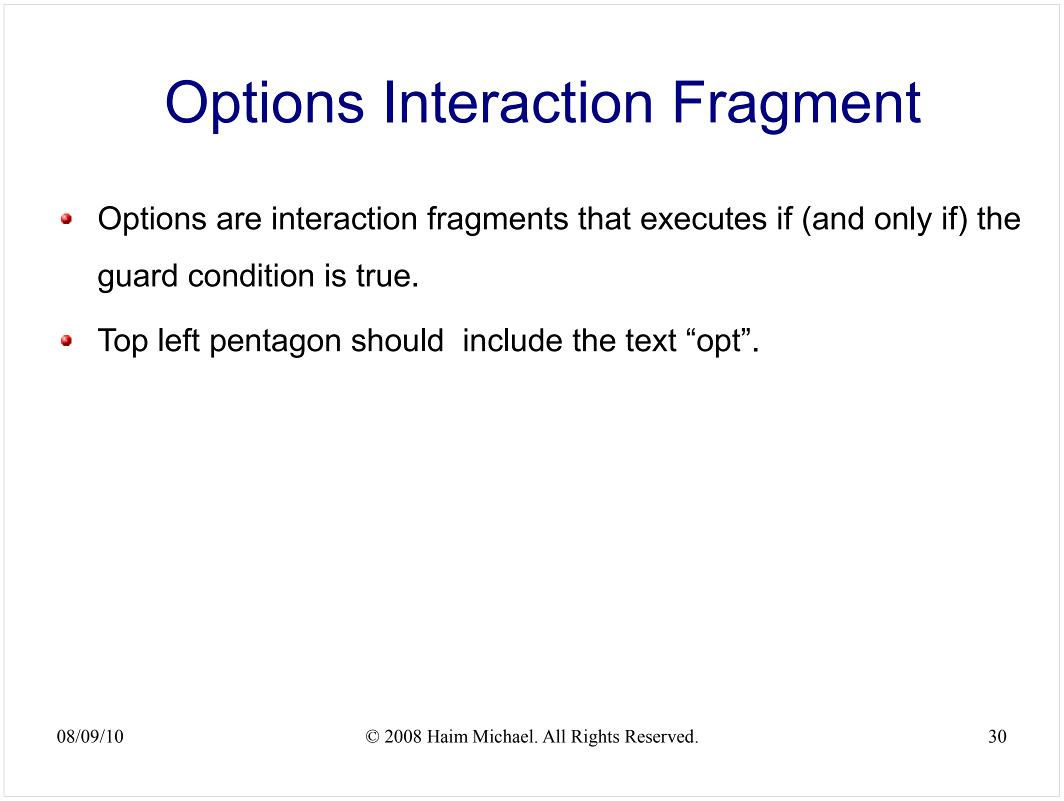
Alternatives Interaction Fragment

- An alternative interaction fragment is a choice of behavior executes based on a guard conditions evaluation with true/false.
- Top left pentagon should include the text “alt”.

Alternatives Interaction Fragment

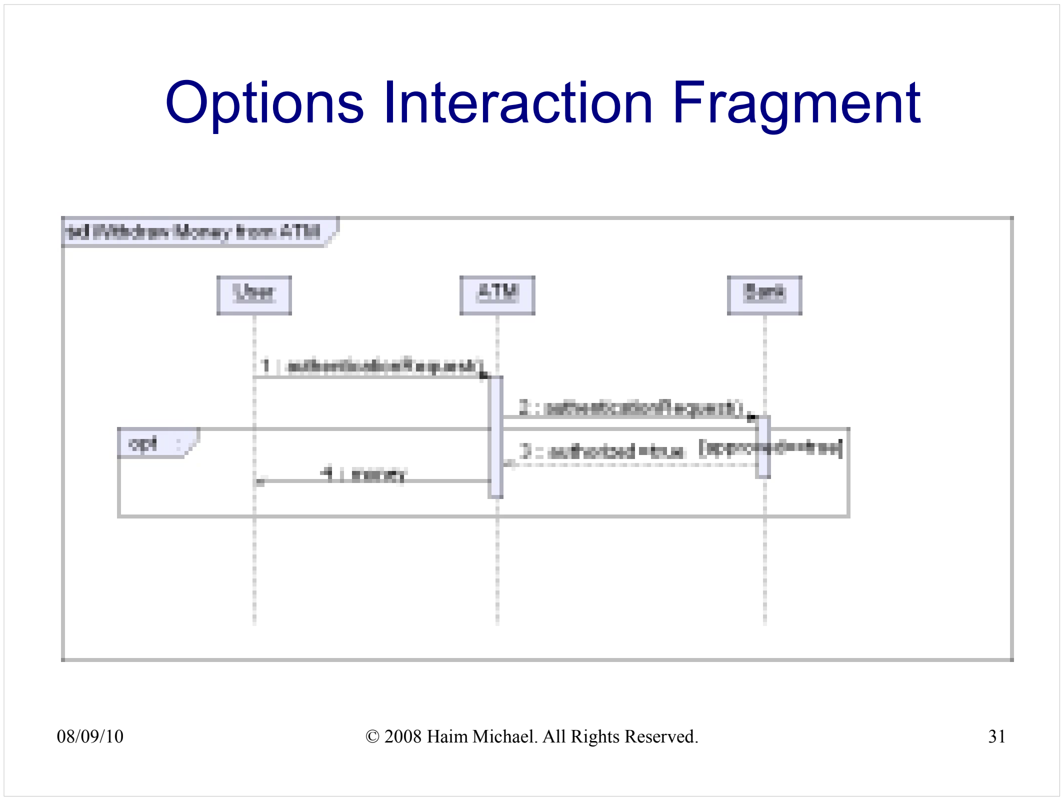

sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
alt
[approved==true]
3 : authenticationApprovalID
[else]
3: authenticationFailureMessage

08/09/10 © 2008 Haim Michael. All Rights Reserved. 29

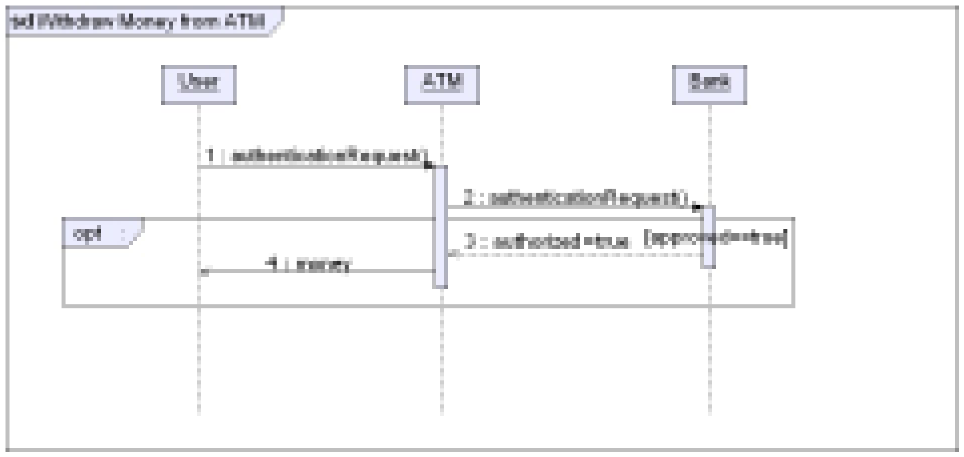


Options Interaction Fragment

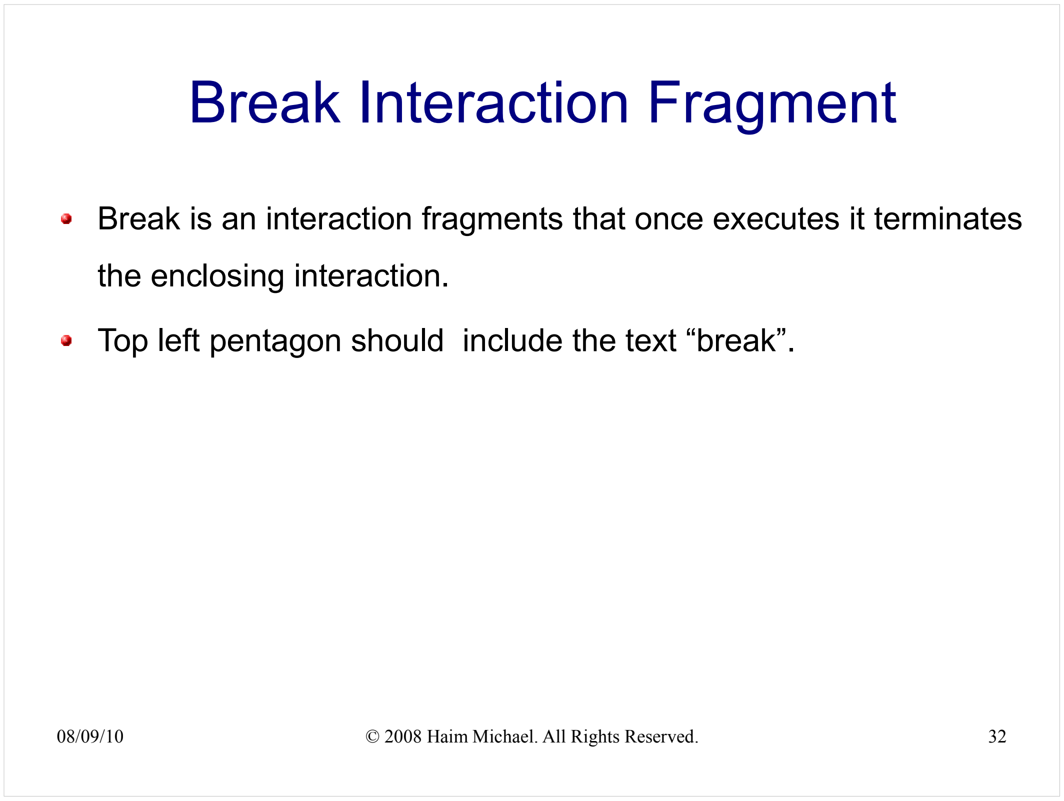
- Options are interaction fragments that executes if (and only if) the guard condition is true.
- Top left pentagon should include the text “opt”.



Options Interaction Fragment


sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
opt
3 : authorized =true
[approved==true]
4 : money

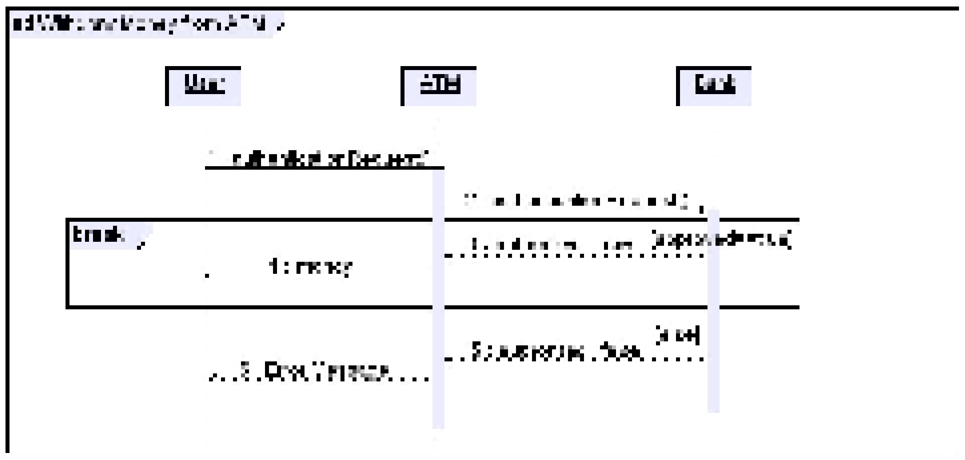
08/09/10 © 2008 Haim Michael. All Rights Reserved. 31



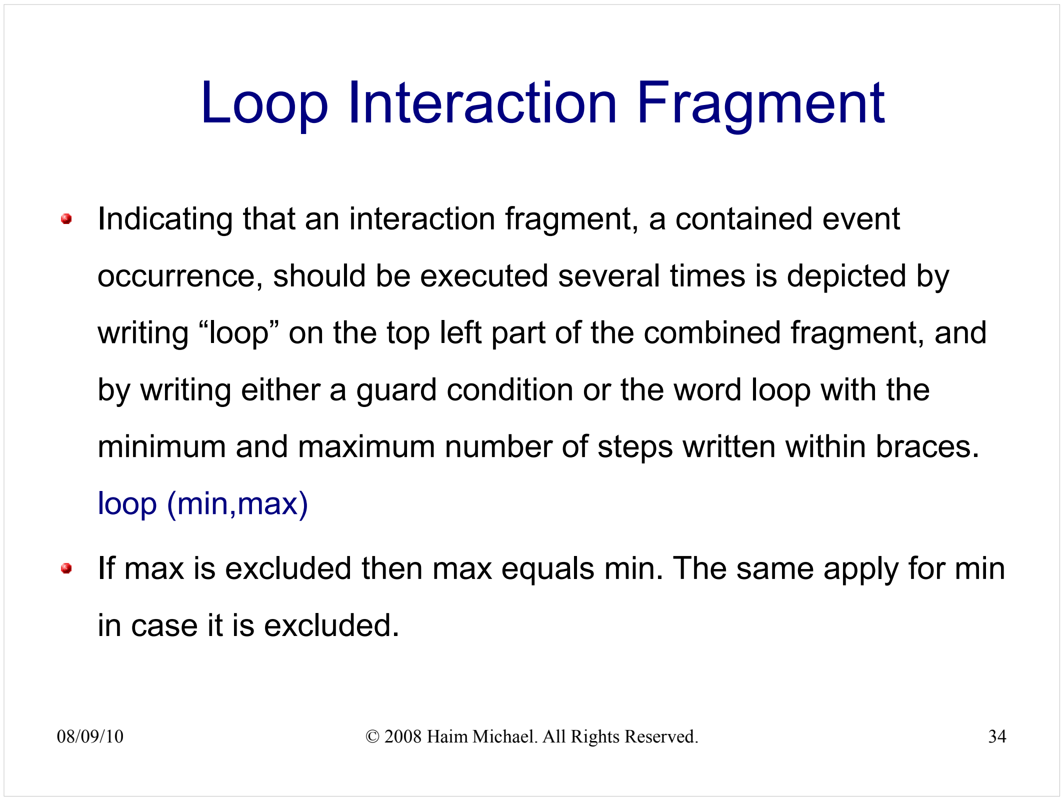
Break Interaction Fragment

- Break is an interaction fragments that once executes it terminates the enclosing interaction.
- Top left pentagon should include the text “break”.

Break Interaction Fragment

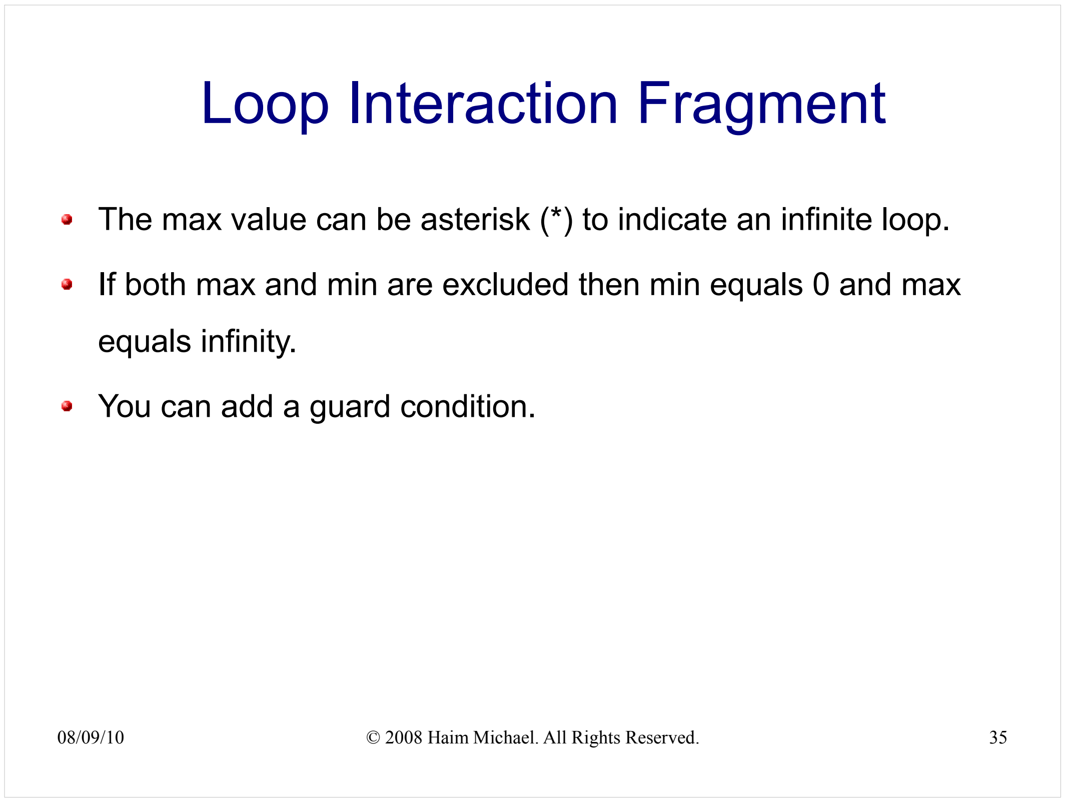


08/09/10 © 2008 Haim Michael. All Rights Reserved. 33



Loop Interaction Fragment

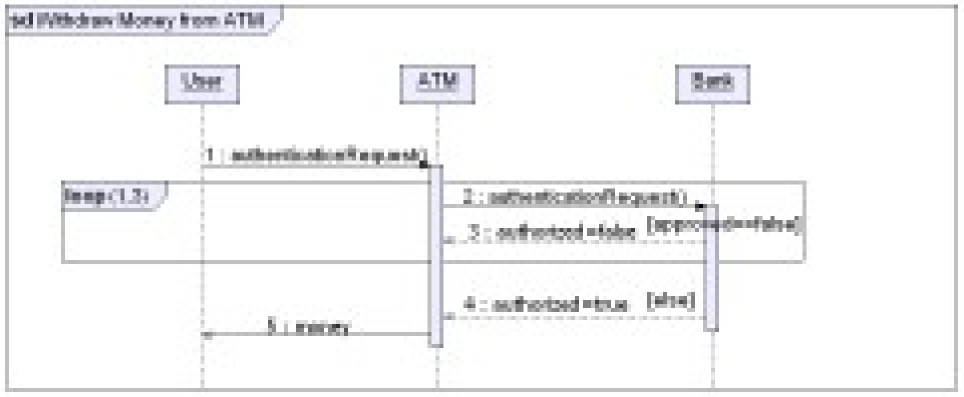
- Indicating that an interaction fragment, a contained event occurrence, should be executed several times is depicted by writing “loop” on the top left part of the combined fragment, and by writing either a guard condition or the word loop with the minimum and maximum number of steps written within braces.
  loop (min,max)
- If max is excluded then max equals min. The same apply for min in case it is excluded.



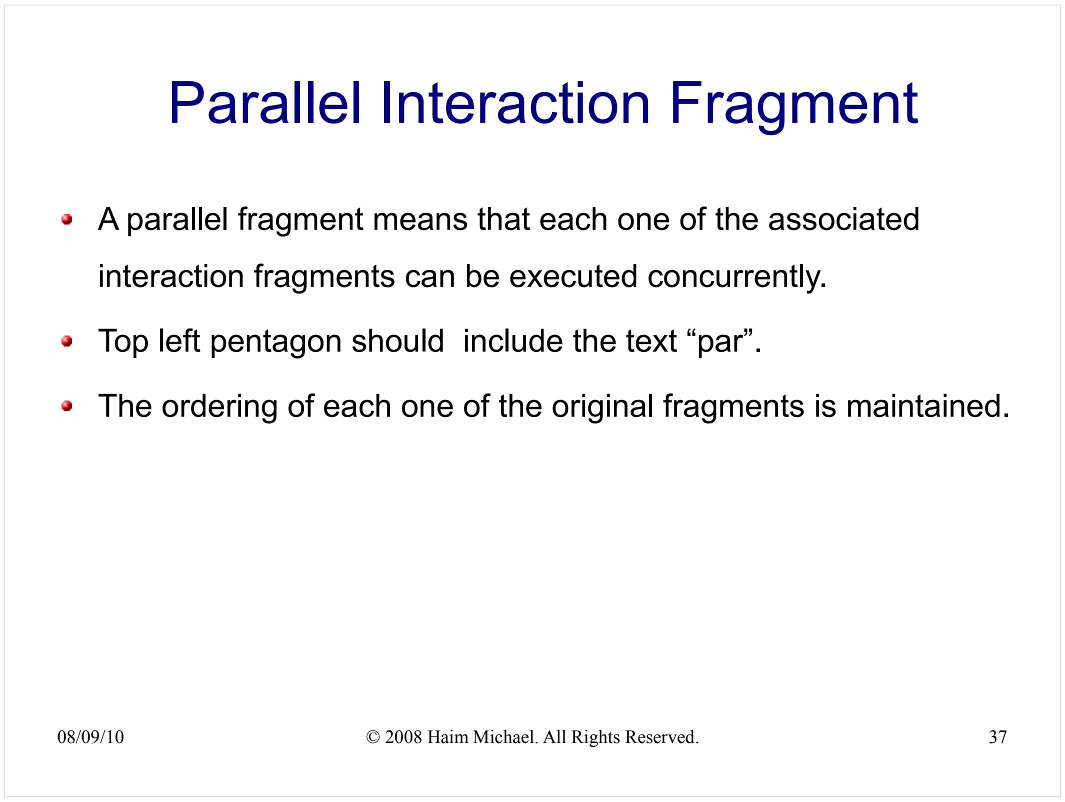
Loop Interaction Fragment

- The max value can be asterisk (*) to indicate an infinite loop.
- If both max and min are excluded then min equals 0 and max equals infinity.
- You can add a guard condition.

Loop Interaction Fragment



08/09/10 © 2008 Haim Michael. All Rights Reserved. 36

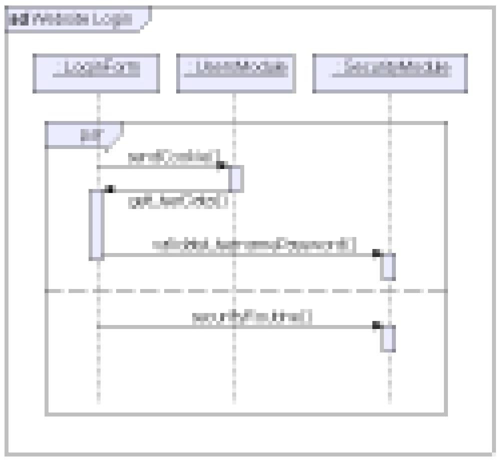


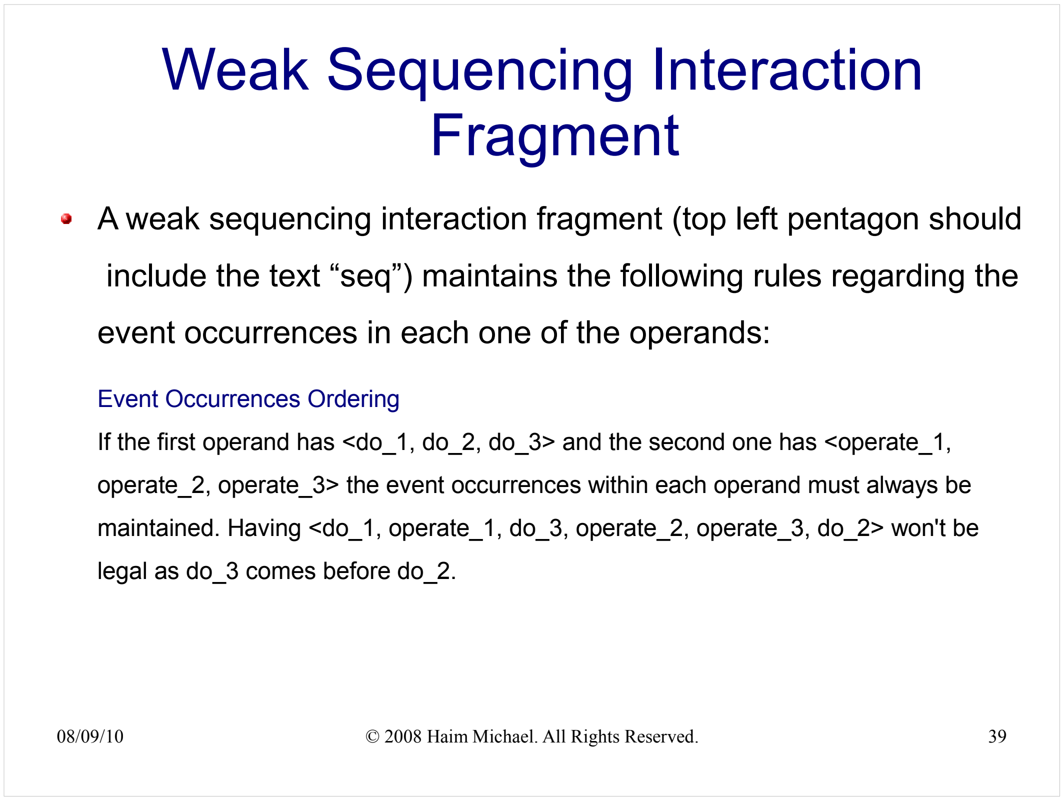
Parallel Interaction Fragment

- A parallel fragment means that each one of the associated interaction fragments can be executed concurrently.
- Top left pentagon should include the text “par”.
- The ordering of each one of the original fragments is maintained.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 37

Parallel Interaction Fragment





Weak Sequencing Interaction Fragment

- A weak sequencing interaction fragment (top left pentagon should include the text “seq”) maintains the following rules regarding the event occurrences in each one of the operands:

Event Occurrences Ordering

If the first operand has <do_1, do_2, do_3> and the second one has <operate_1, operate_2, operate_3> the event occurrences within each operand must always be maintained. Having <do_1, operate_1, do_3, operate_2, operate_3, do_2> won't be legal as do_3 comes before do_2.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 39

Weak Sequencing Interaction Fragment

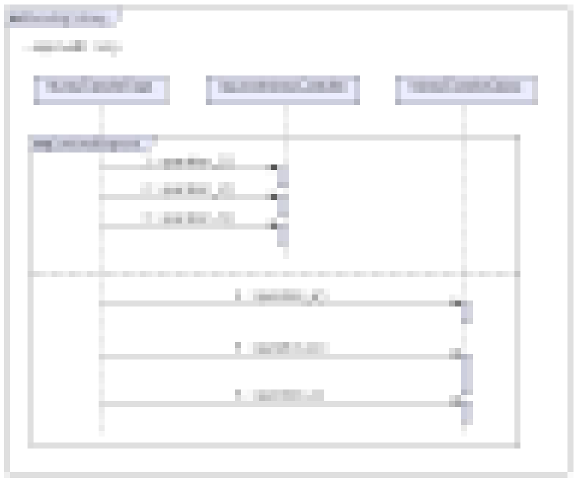
Event Occurrences Ordering (Different Lifelines)

If the event occurrences are in different operands on different lifelines they can be interleaved in any order.

Event Occurrences Ordering (The Same Lifeline)

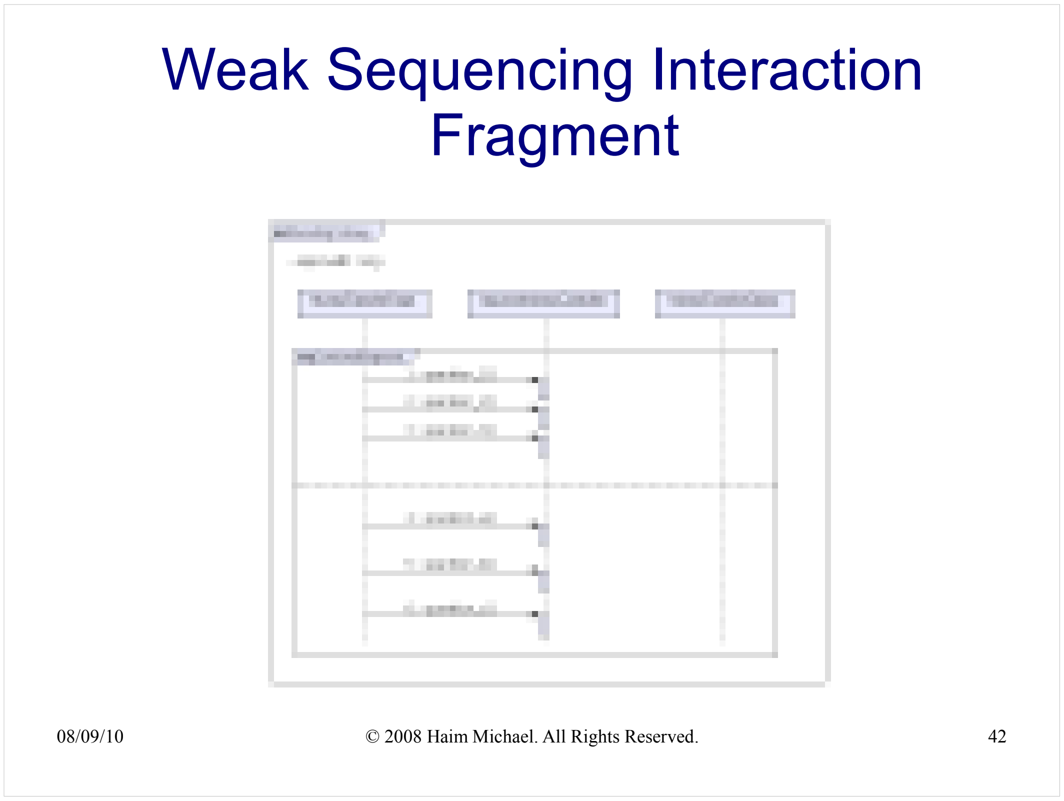
If the even occurrences are on the same lifeline they can be interleaved as long as all the event occurrences of the first operand execute before the occurrences of the second operand.

Weak Sequencing Interaction Fragment

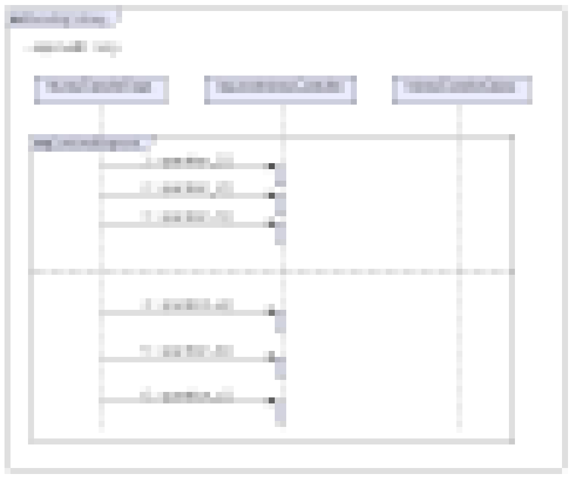


08/09/10 © 2008 Haim Michael. All Rights Reserved. 41

This diagram presents a unique weak sequencing case in which the events occurrences in each one of the two operands occur on different lifelines. In this case they can be interleaved in any order without any limitation.



Weak Sequencing Interaction Fragment

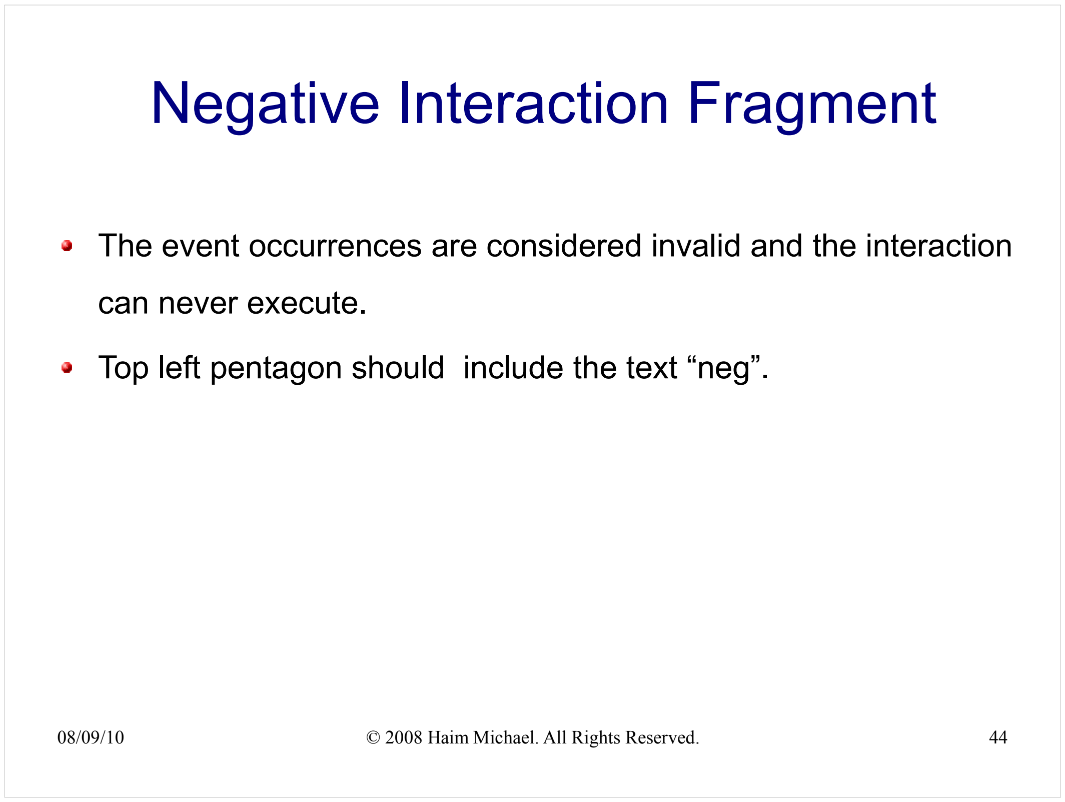


08/09/10 © 2008 Haim Michael. All Rights Reserved. 42

This diagram presents a unique weak sequencing case in which the events occurrences in each one of the two operands occur on the same lifeline. In this case the events occurrences of the first operand must complete before the events occurrences of the second one are executed.

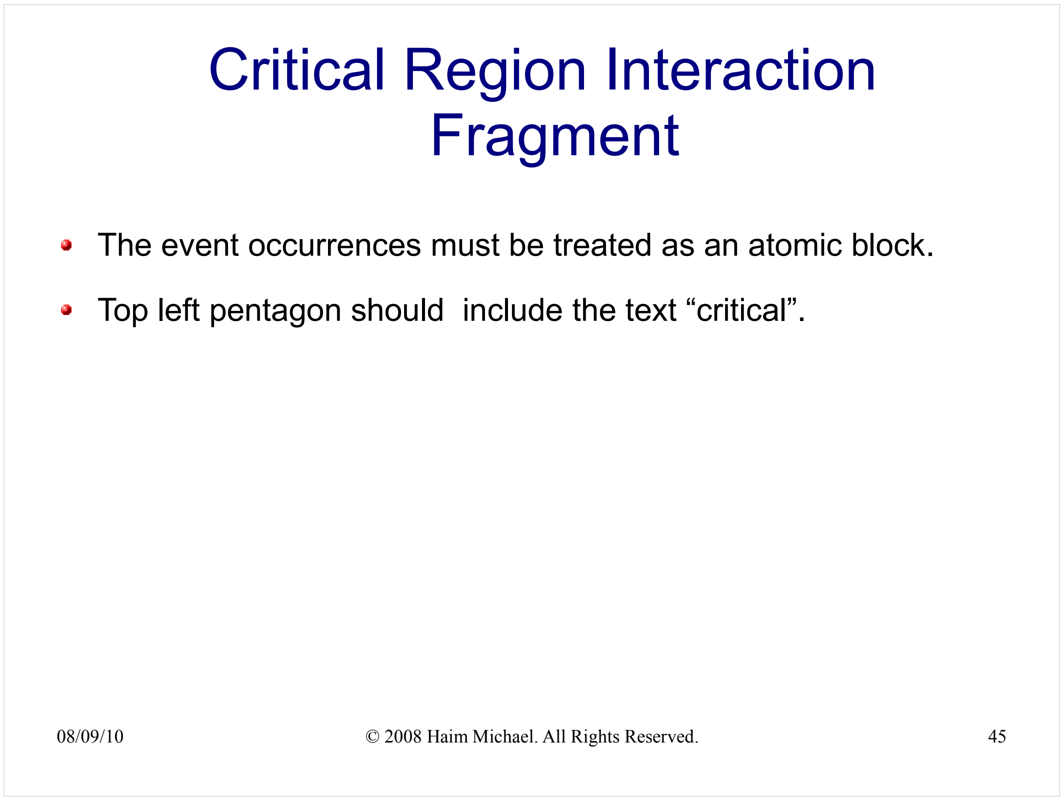
Strict Sequencing Interaction Fragment

- The ordering of the event occurrences must be kept in all cases, even if they happen within different separated life lines.
- Top left pentagon should include the text “strict”.
- The operands must be executed in order from top to bottom.



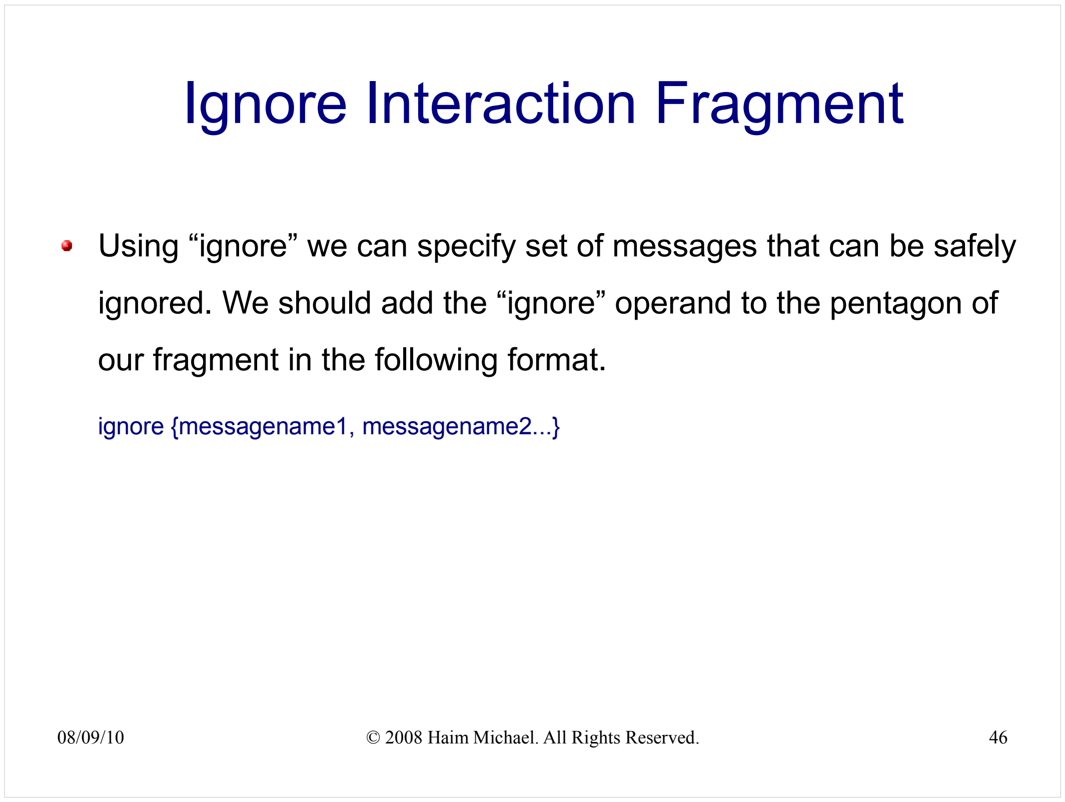
Negative Interaction Fragment

- The event occurrences are considered invalid and the interaction can never execute.
- Top left pentagon should include the text “neg”.



Critical Region Interaction Fragment

- The event occurrences must be treated as an atomic block.
- Top left pentagon should include the text “critical”.



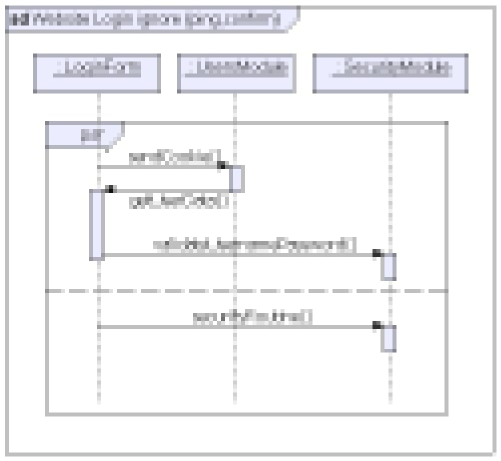
Ignore Interaction Fragment

- Using “ignore” we can specify set of messages that can be safely ignored. We should add the “ignore” operand to the pentagon of our fragment in the following format.

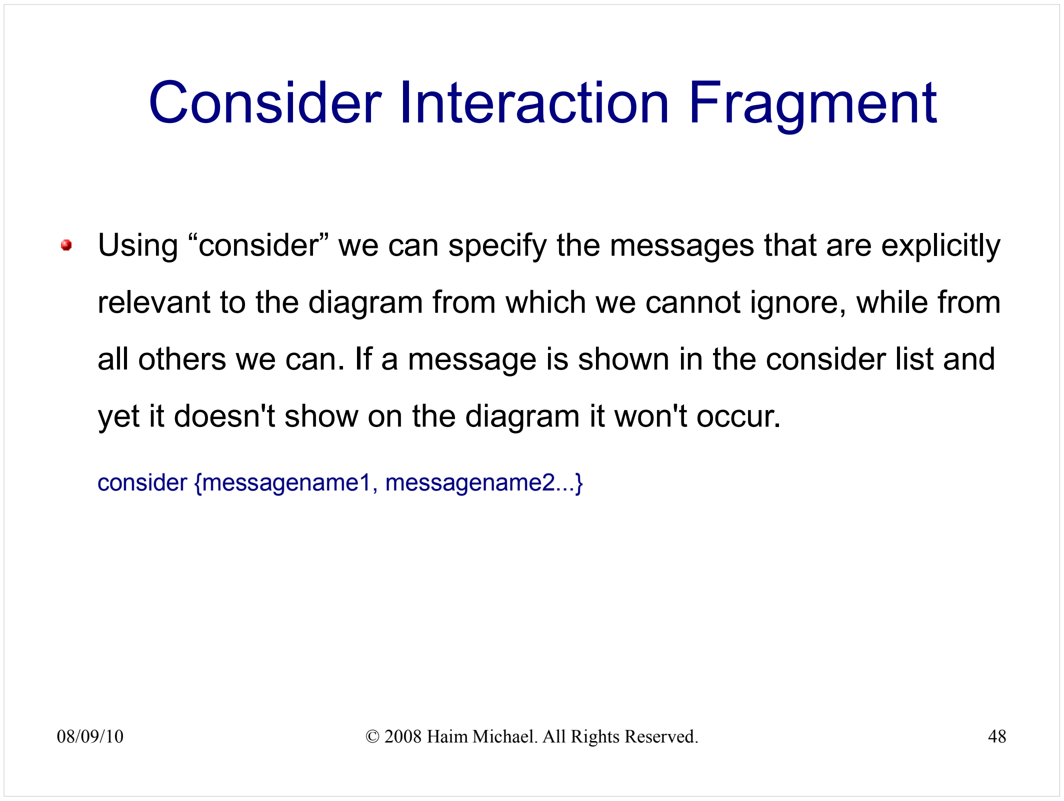
  ignore {messagename1, messagename2...}

08/09/10 © 2008 Haim Michael. All Rights Reserved. 46

Ignore Interaction Fragment



08/09/10 © 2008 Haim Michael. All Rights Reserved. 47



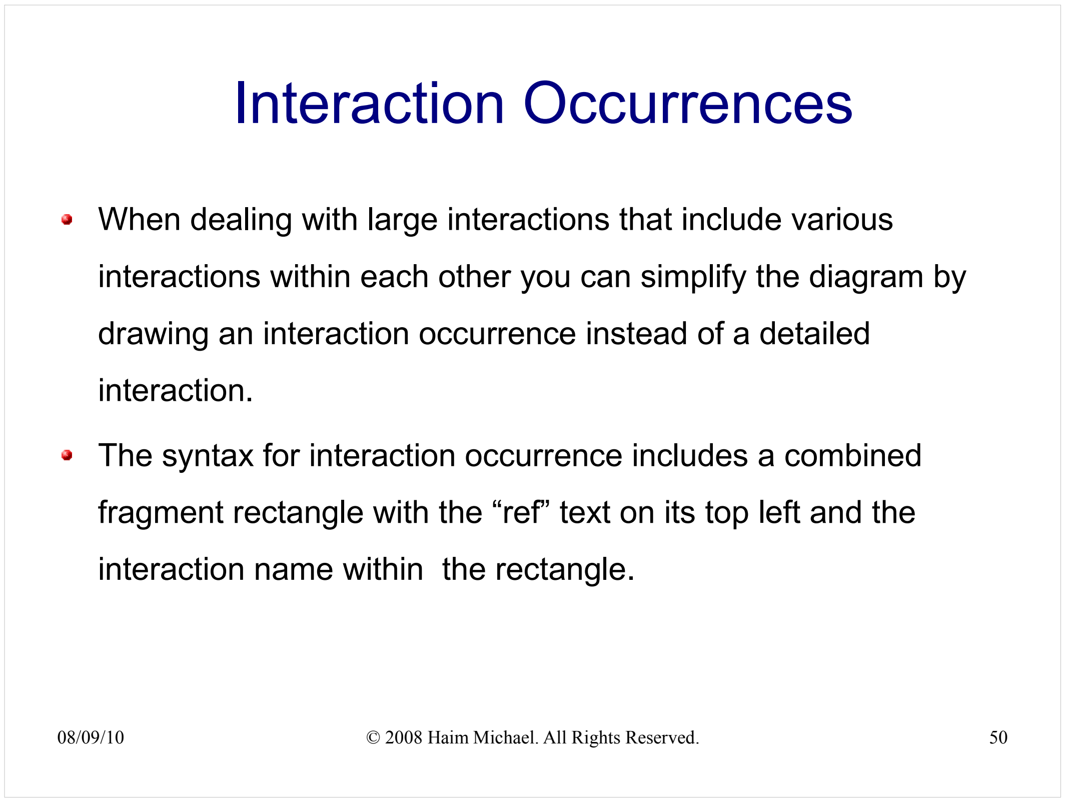
Consider Interaction Fragment

- Using “consider” we can specify the messages that are explicitly relevant to the diagram from which we cannot ignore, while from all others we can. If a message is shown in the consider list and yet it doesn't show on the diagram it won't occur.

  consider {messagename1, messagename2...}

Assertion Interaction Fragment

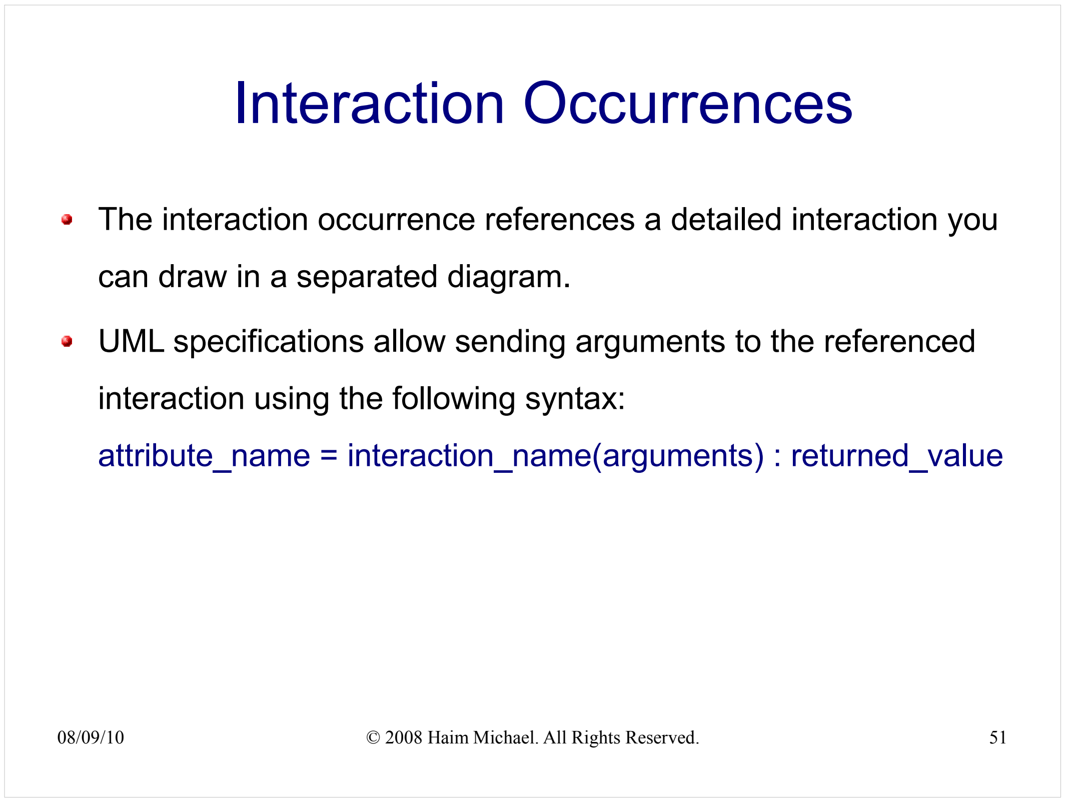
- Using “assert” we can indicate the contained event occurrences are the only valid execution path.
- Usually, a state invariant is added.



Interaction Occurrences

- When dealing with large interactions that include various interactions within each other you can simplify the diagram by drawing an interaction occurrence instead of a detailed interaction.
- The syntax for interaction occurrence includes a combined fragment rectangle with the "ref" text on its top left and the interaction name within the rectangle.

08/09/10 © 2008 Haim Michael. All Rights Reserved. 50



Interaction Occurrences

- The interaction occurrence references a detailed interaction you can draw in a separated diagram.
- UML specifications allow sending arguments to the referenced interaction using the following syntax:
  attribute_name = interaction_name(arguments) : returned_value

08/09/10 © 2008 Haim Michael. All Rights Reserved. 51

attribute_name
The attribute_name is an attribute in which we want to enter the returned value. This is an optional syntax.

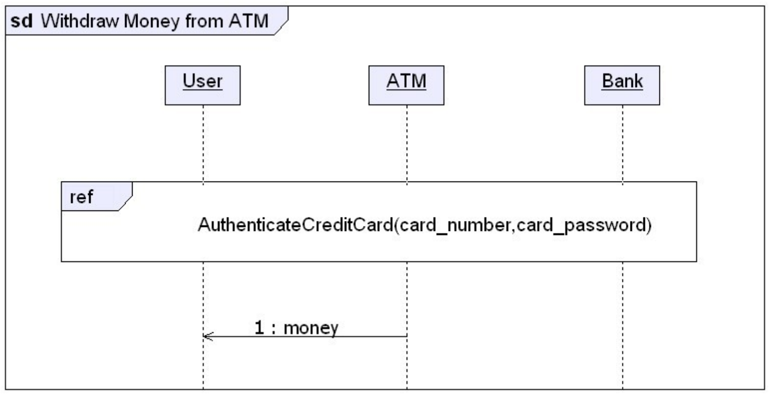
occurrence
When dealing with very big diagrams in which the same interaction occurrence might be shown more than once it can be a good idea to reference each occurrence with a difference occurrence name. This is an optional syntax.

interation_name
This is the name of the interaction the interaction occurrence represents.

arguments
This is a comma-separated list of arguments passed to the referenced interaction. You can prefix each argument with in/out/inout keyword in order to indicate whether the argument is used to get a value returned from the reference interaction.

returned_value
This is an optional part that indicates about the value returned from the referenced interaction.

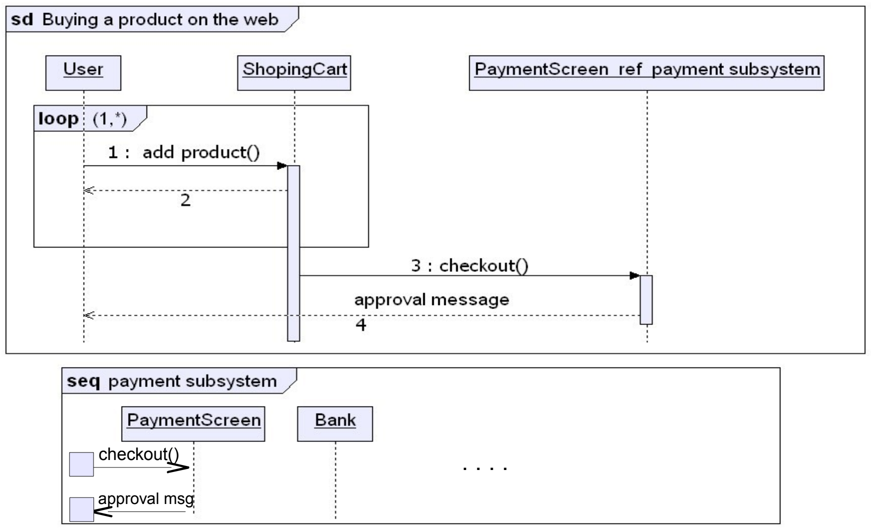
Interaction Occurrences


sd Withdraw Money from ATM
User
ATM
Bank
ref
AuthenticateCreditCard(card_number,card_password)
1 : money

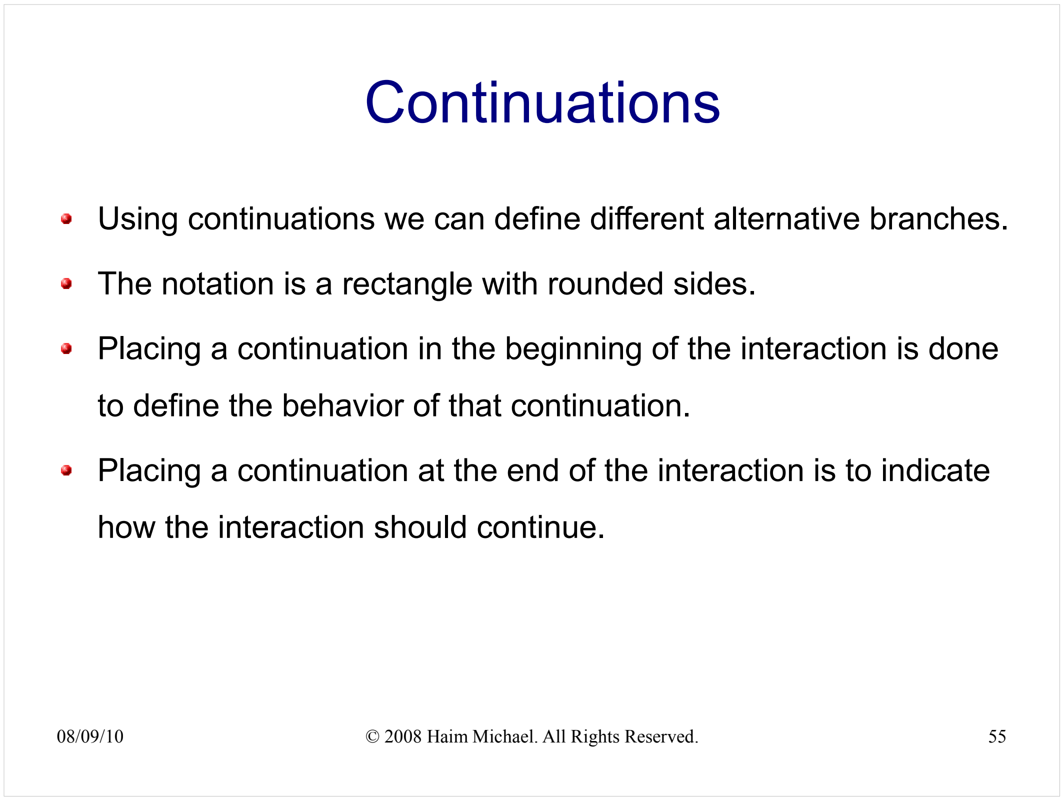
Decomposition Interaction Diagram

- UML allows linking separated interaction diagrams by adding a decomposition reference from a specific participant to a separated interaction diagram.
- We can include within the secondary diagram input/output gates that match the message the specific participant get and its reply.
- The name of the secondary diagram will be added to the primary diagram specific participant name prefix with “ref”.

Decomposition Interaction Diagram

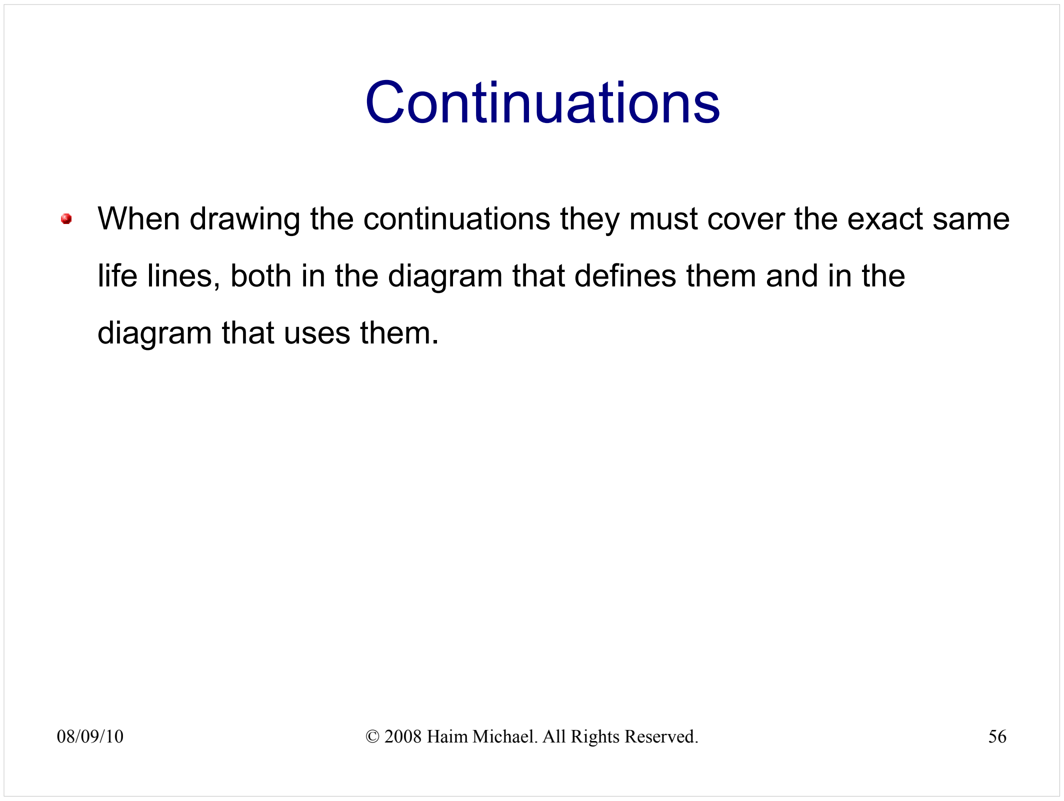


08/09/10 © 2008 Haim Michael. All Rights Reserved. 54



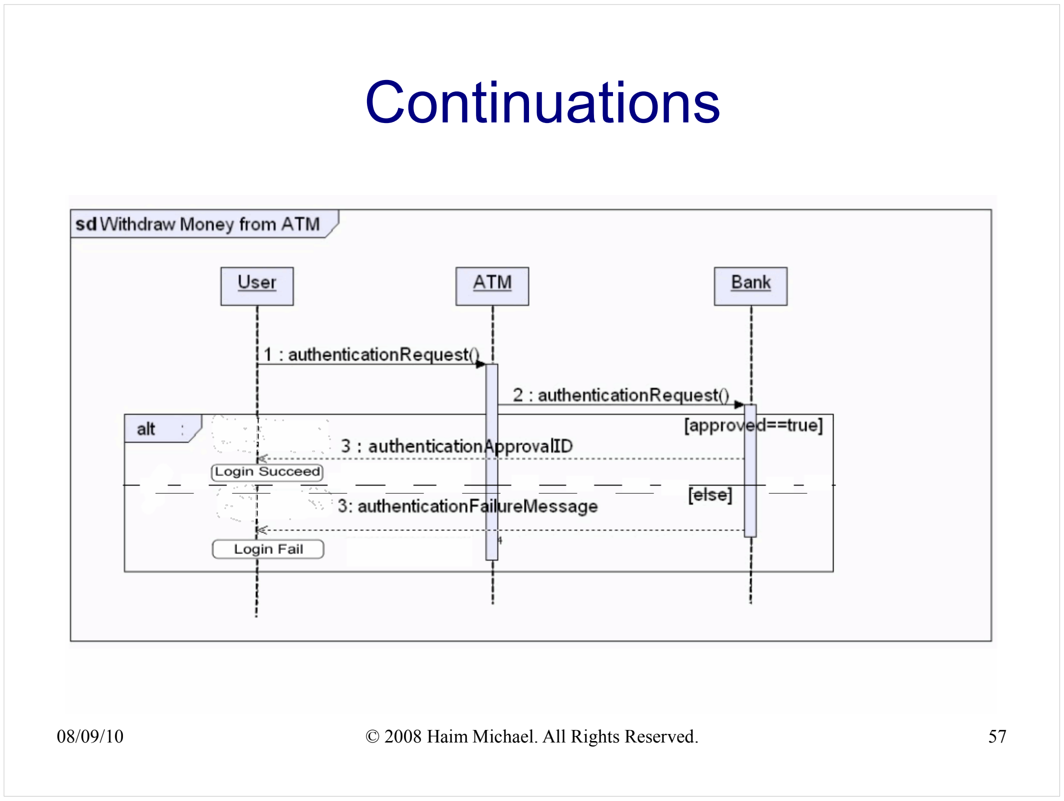
Continuations

- Using continuations we can define different alternative branches.
- The notation is a rectangle with rounded sides.
- Placing a continuation in the beginning of the interaction is done to define the behavior of that continuation.
- Placing a continuation at the end of the interaction is to indicate how the interaction should continue.

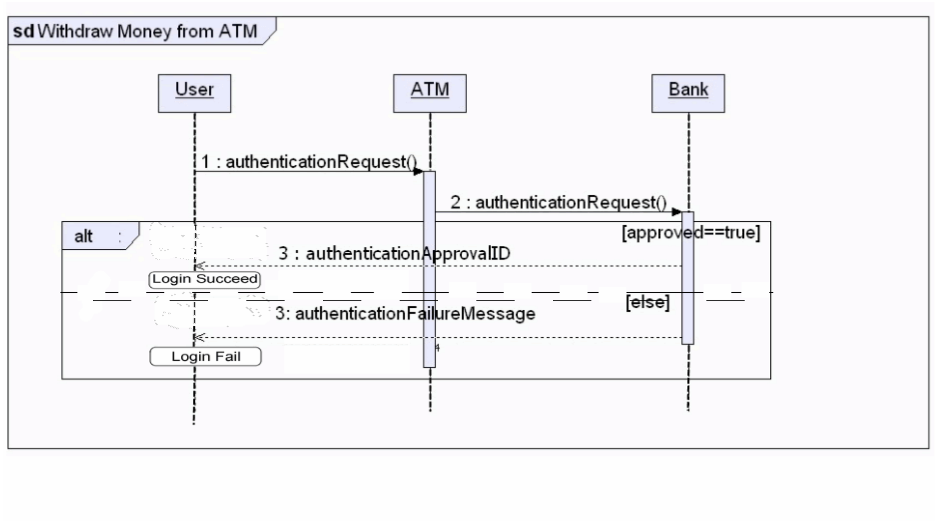


Continuations

- When drawing the continuations they must cover the exact same life lines, both in the diagram that defines them and in the diagram that uses them.



Continuations


sd Withdraw Money from ATM
User
ATM
Bank
1 : authenticationRequest()
2 : authenticationRequest()
alt
[approved==true]
3 : authenticationApprovalID
Login Succeed
[else]
3: authenticationFailureMessage
Login Fail

08/09/10 © 2008 Haim Michael. All Rights Reserved. 57

In this diagram the two continuations “Login Succeed” and “Login Fail” are used. They represent the events occurrences that should happen in accordance with each one of the two cases (the two operands).