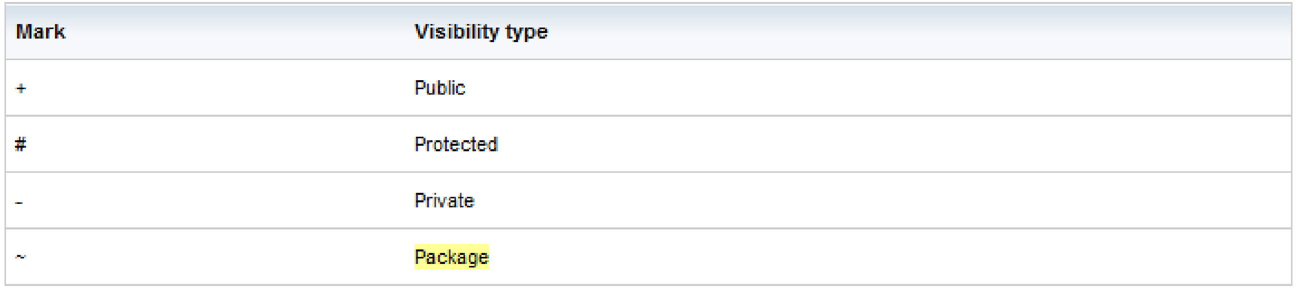
Sequence diagrams in UML (Unified Modeling Language)

http://www.paxcel.net

By:-
Tanjot Singh Sandhu

Open questions

1. How to represent 'internal' and 'protected internal' members inside a CD?

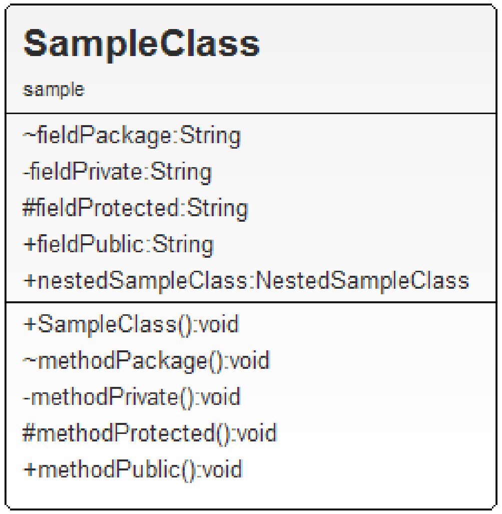


Mark	Visibility type
+	Public
#	Protected
-	Private
~	Package

For "protected internal", we can either use a stereotype or some other special symbol/ image.

2. How to represent a nested class?

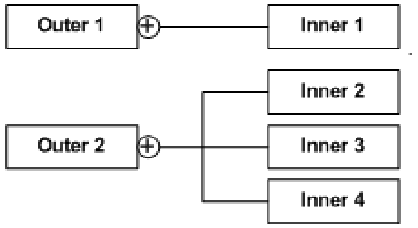
(a) It can be represented as a normal class member.



```
SampleClass
sample
~fieldPackage:String
-fieldPrivate:String
#fieldProtected:String
+fieldPublic:String
+nestedSampleClass:NestedSampleClass
+SampleClass():void
~methodPackage():void
-methodPrivate():void
#methodProtected():void
+methodPublic():void
```

Open questions (Continued)

(b) We can use containment arrow, if we need to define the structure of nested class.



3. Is aggregation a strong form or week form of plain association?

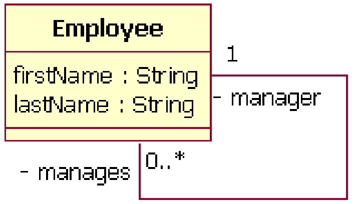
Aggregation is a specialized form of Association. It is relatively weak form of "whole-part", because no statement is made about lifecycle dependency or creation/ destruction responsibility.

4. What is the difference between aggregation and containment?

Aggregation shows the relation between two separate classes whereas containment is for outer-inner class representation or package-class representation.

5. What is a reflexive relationship?

A class can also be associated with itself, using a reflexive association.



What is a Sequence Diagram?

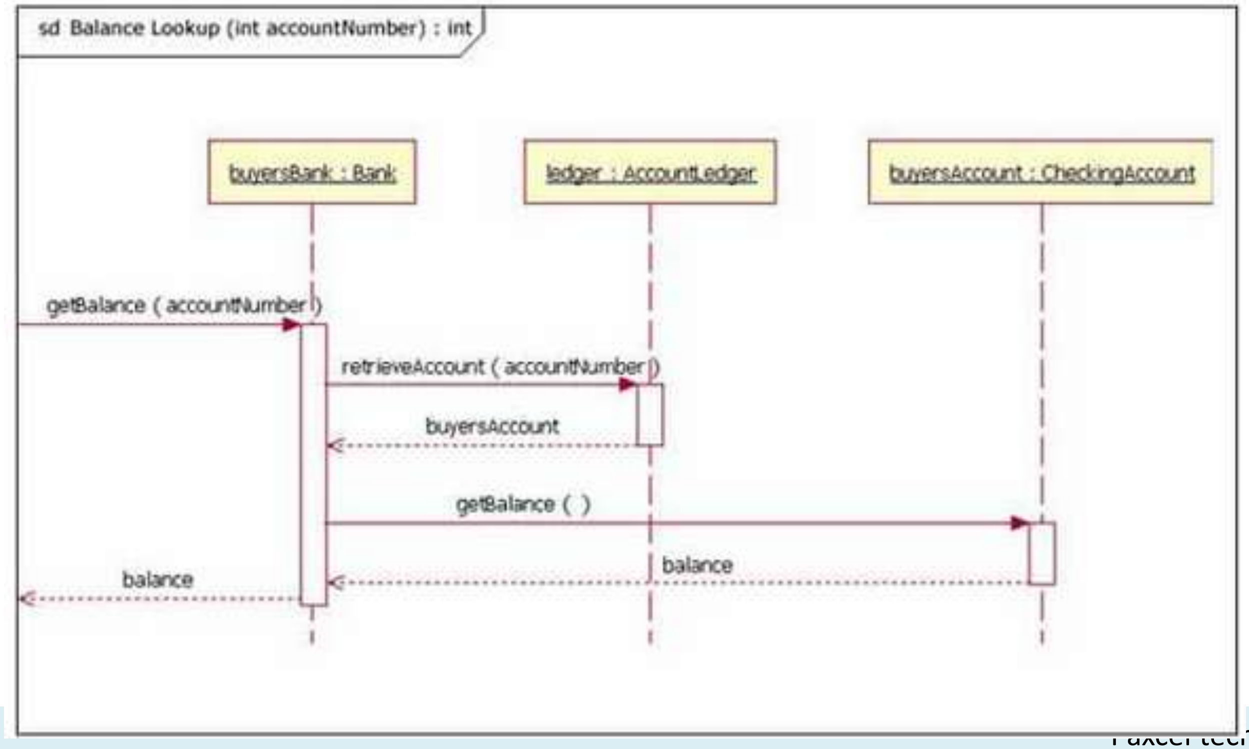
- ✓ UML sequence diagrams are a dynamic modeling technique.
- ✓ Used primarily to show the interactions between objects in the sequential order.
- ✓ Use cases are often refined into one or more sequence diagrams.
- ✓ A business-level sequence diagram can be used as a requirement document to communicate requirements for a future system implementation.
- ✓ Can be used to document how objects in an existing (call it "legacy") system currently interact. This documentation is very useful when transitioning a system to another person or organization.

Drawing a Sequence Diagram

A visual border (called frame element) is used to visualize a sequence diagram.

Incoming and outgoing messages (a.k.a. interactions) for a sequence can be modeled by connecting the messages to the border of the frame element.

Fig: Sequence Diagram


sd Balance Lookup (int accountNumber) : int
buyersBank : Bank
ledger : AccountLedger
buyersAccount : CheckingAccount
getBalance (accountNumber)
retrieveAccount (accountNumber)
buyersAccount
getBalance ( )
balance
balance

Drawing a Sequence Diagram (Continued)

- ✓ Diagram's label begins with the letters "sd," for Sequence Diagram.
- ✓ When using a frame element to enclose a diagram, the diagram's label needs to follow the format of:

  Diagram Type Diagram Name

- ✓ The UML specification provides specific text values for diagram types (e.g., sd = Sequence Diagram, activity = Activity Diagram, and use case = Use Case Diagram).

Constituents of a Sequence Diagram

- ✓ The main purpose of a sequence diagram is to define **event sequences** that result in some desired outcome.
- ✓ The focus is on the **order** in which messages occur.
- ✓ Horizontal and vertical dimensions are used to draw an SD.
- ✓ Vertical dimension shows the time sequence of messages/ calls as they occur.
- ✓ Horizontal dimension shows, left to right, the object instances that the messages are sent to.
- ✓ The basic constitutes of an SD include :
  - Lifelines
  - Messages
  - Guards
  - Combined fragments (alternatives, options, loops etc.)

Lifeline

A lifeline represent either roles or object instances that participate in the sequence being modeled.

Lifeline notation elements are placed at the top of the diagram.

Lifelines are drawn as a box with a dashed line descending from the center of the bottom edge.

The lifeline's name is placed inside the box.

The UML standard for naming a lifeline follows the format of:

Instance Name : Class Name

An example of lifeline diagram is shown in the following figure:

Lifeline (Continued)

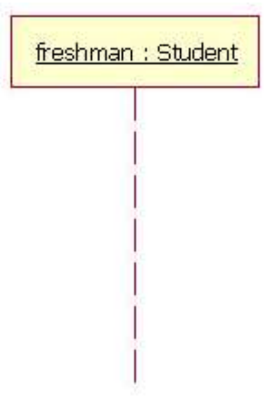


Figure: Student class used in a lifeline whose instance name is freshman

- ✓ An underline defines that the lifeline represents a specific instance of a class in a sequence diagram, and not a particular kind of instance (i.e., a role).
- ✓ Instance names in sequence diagrams are underlined; roles names are not.
- ✓ A lifeline can be used to represent an anonymous or unnamed instance.

Message

- ✓ The first message of a sequence diagram always starts at the top and is typically located on the left side of the diagram for readability.
- ✓ Subsequent messages are then added to the diagram slightly lower then the previous message.
- ✓ To show an object (i.e. lifeline) sending a message to another object, draw a line to the receiving object with a solid arrowhead for a synchronous call.
- ✓ Use a stick arrowhead for an asynchronous call.
- ✓ The message/ method name is placed above the arrowed line.
- ✓ Response/ Return message is optional and is denoted by a dotted line with an open arrowhead back to the originating lifeline.

Message Example

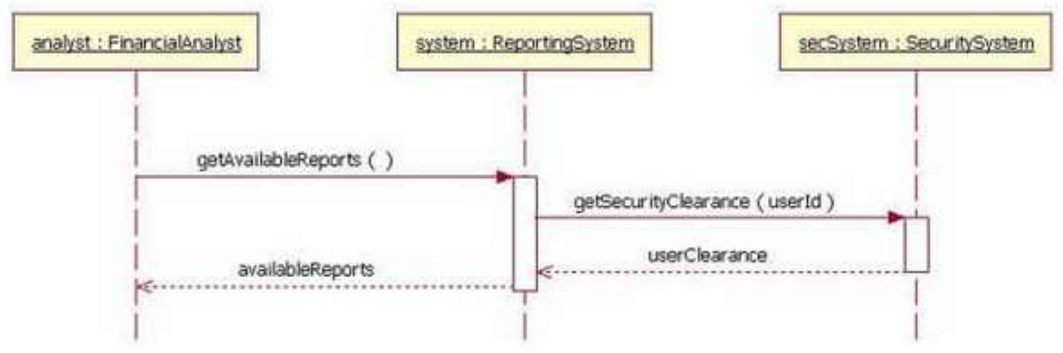

analyst : FinancialAnalyst
system : ReportingSystem
secSystem : SecuritySystem
getAvailableReports ( )
getSecurityClearance (userId)
userClearance
availableReports

Figure: An example of messages being sent between objects

- ✓ Besides just showing message calls above Figure includes return messages (optional).
- ✓ Use of return messages depends on the level of detail/abstraction that is being modeled.
- ✓ Sometimes an object will need to send a message to itself.
- ✓ To draw an object calling itself, connect the message back to the object itself.

Message Example (Continued)

- ✓ The following Figure shows the system object calling its determineAvailableReports method.
- ✓ The model draws attention to the fact that this processing takes place in the system object.

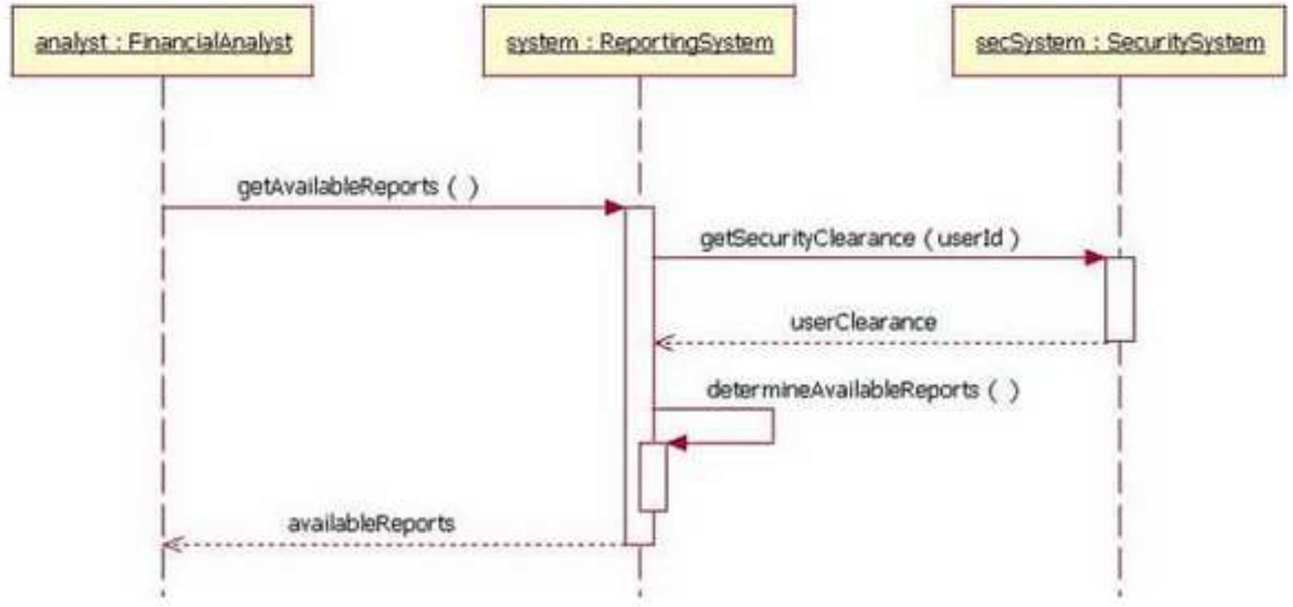
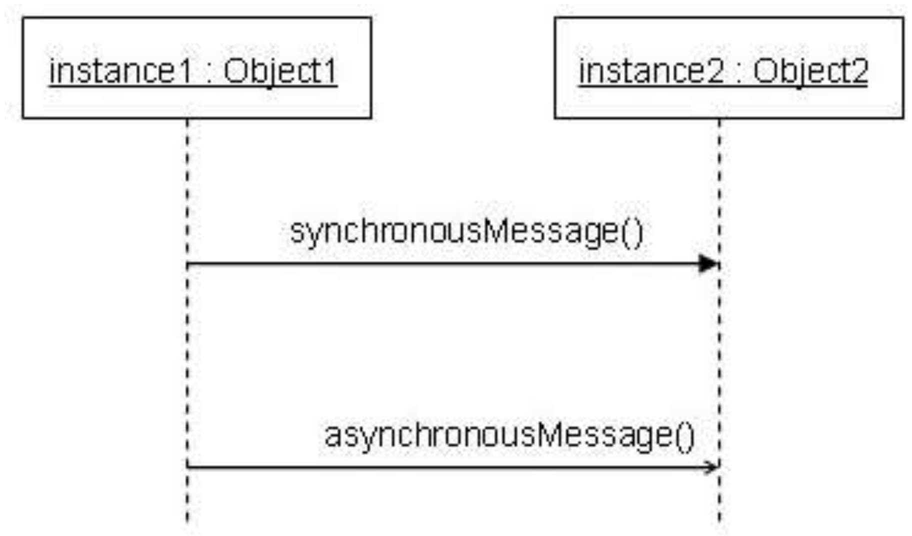

analyst : FinancialAnalyst
system : ReportingSystem
secSystem : SecuritySystem
getAvailableReports ( )
getSecurityClearance (userId)
userClearance
determineAvailableReports ( )
availableReports

Figure: The system object calling its determineAvailableReports method.

Synchronous VS Asynchronous Messages



• Figure: A synchronous and an asynchronous message being sent to instance2.

Guards

- ✓ Guard is a condition, that must be met for a message to be sent to the object.
- ✓ Guards are used throughout UML diagrams to control flow.
- ✓ Place the guard element above the message line being guarded and in front of the message name.
- ✓ The notation of a guard is very simple; the format is:

  [Boolean Test]

- ✓ For example,

  [pastDueBalance = 0]

Guards - Example

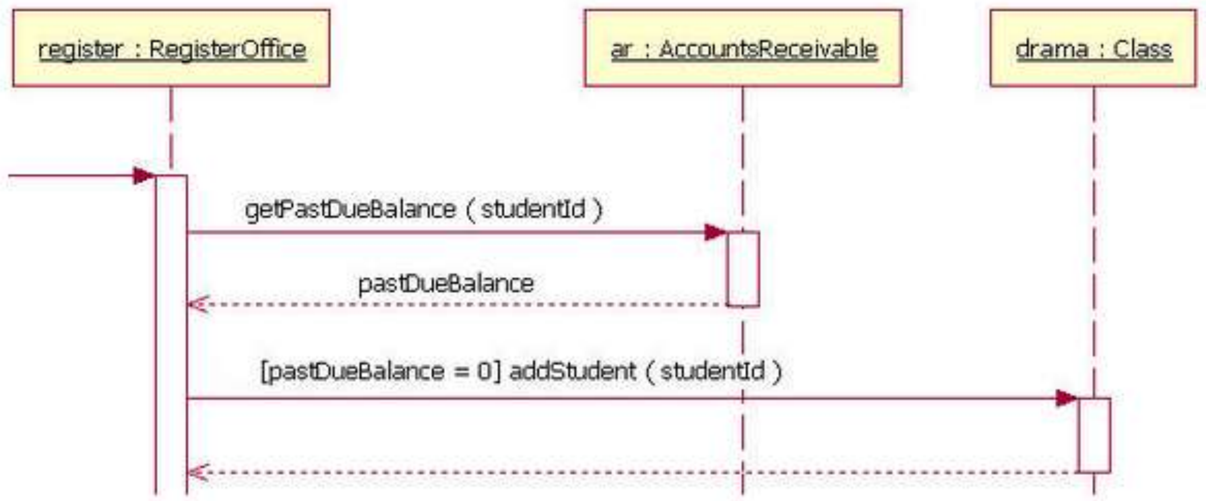

register : RegisterOffice
ar : AccountsReceivable
drama : Class
getPastDueBalance (studentId)
pastDueBalance
[pastDueBalance = 0] addStudent (studentId)

Figure: A segment sequence diagram in which the addStudent message has a guard

- ✓ In Figure above, the guard is the text "[pastDueBalance = 0]."
- ✓ addStudent message will only be sent if the accounts receivable system returns a past due balance of zero.

Combined fragments - alternatives

- ✓ UML 1.x "in-line" guard is not sufficient to handle the logic required for a sequence being modeled.
- ✓ UML 2 has addressed this problem by removing the "in-line" guard and adding a notation element called a Combined Fragment.
- ✓ A combined fragment is used to group sets of messages together to show conditional flow in a sequence diagram.
- ✓ The UML 2 specification identifies 11 interaction types for combined fragments.
- ✓ Alternatives are used to designate a mutually exclusive choice between two or more message sequences.
- ✓ Alternatives allow the modeling of the classic "if then else" logic (e.g., if I buy three items, then I get 20% off my purchase; else I get 10% off my purchase).

Alternatives - Example

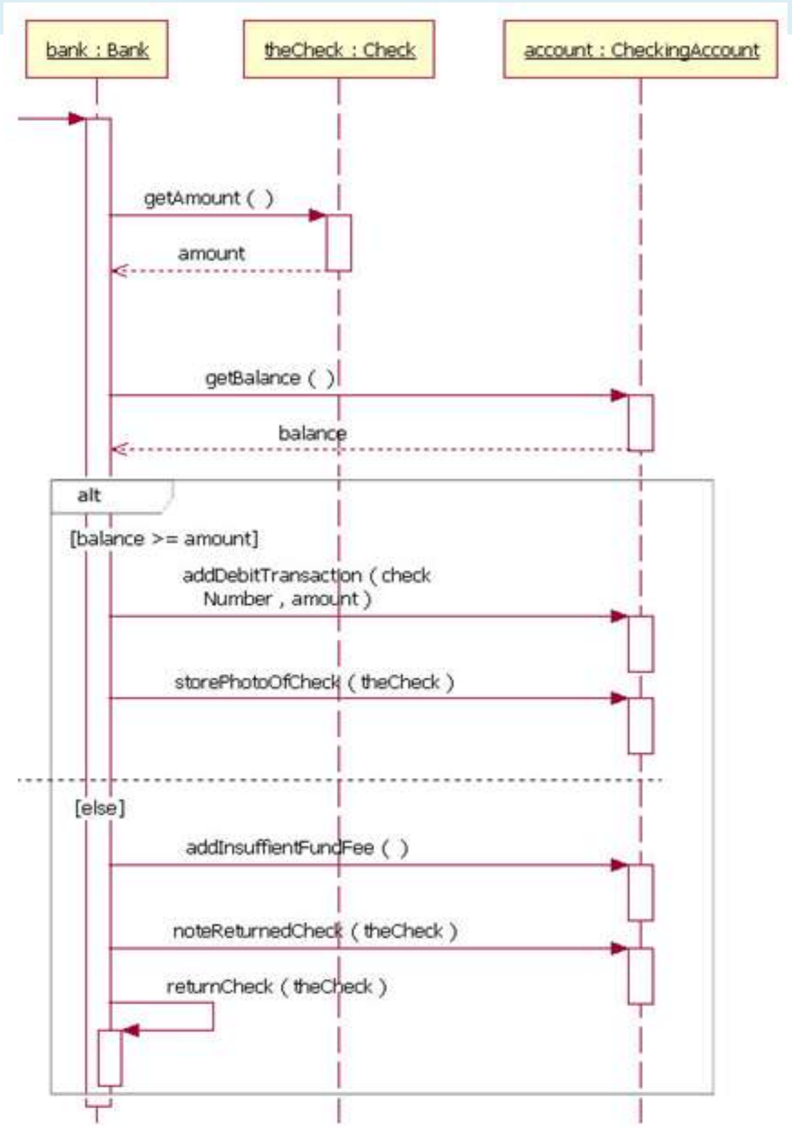


Figure: A sequence diagram fragment that contains an alternative combination fragment

- In last Figure, an alternative combination fragment element is drawn using a frame.
- The word "alt" is placed inside the frame's namebox.
- The larger rectangle is then divided into what UML 2 calls operands.
- Operands 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 on top of a lifeline. [Note: Usually, the lifeline to which the guard is attached is the lifeline that owns the variable that is included in the guard expression.]
- If an operand's guard equates to "true," then that operand is the operand to follow.

- ✓ Bank object is getting the check's amount and the account's balance.
- ✓ At this point in the sequence the alternative combination fragment takes over. Because of the guard "[balance >= amount]," if the account's balance is greater than or equal to the amount, then the sequence continues with the bank object sending the addDebitTransaction and storePhotoOfCheck messages to the account object.
- ✓ However, if the balance is not greater than or equal to the amount, then the sequence proceeds with the bank object sending the addInsuffientFundFee and noteReturnedCheck message to the account object and the returnCheck message to itself.

- ✓ The second sequence is called when the balance is not greater than or equal to the amount because of the "[else]" guard.
- ✓ In alternative combination fragments, the "[else]" guard is not required; and if an operand does not have an explicit guard on it, then the "[else]" guard is to be assumed.
- ✓ Alternative combination fragments are not limited to simple "if then else" tests.
- ✓ There can be as many alternative paths as are needed.
- ✓ If more alternatives are needed, all you must do is add an operand to the rectangle with that sequence's guard and messages.

Option

- ✓ The option combination fragment is used to model a sequence that will either occur or will not occur.
- ✓ An option is used to model a simple "if then" statement (i.e., if there are fewer than five donuts on the shelf, then make two dozen more donuts).
- ✓ It only has one operand and there never can be an "else" guard.
- ✓ To draw an option combination you draw a frame - text "opt" is placed inside the frame's namebox, and in the frame's content area the option's guard is placed towards the top left corner on top of a lifeline.
- ✓ Then the option's sequence of messages is placed in the remainder of the frame's content area.
- ✓ These elements are illustrated in following Figure.

Option - Example

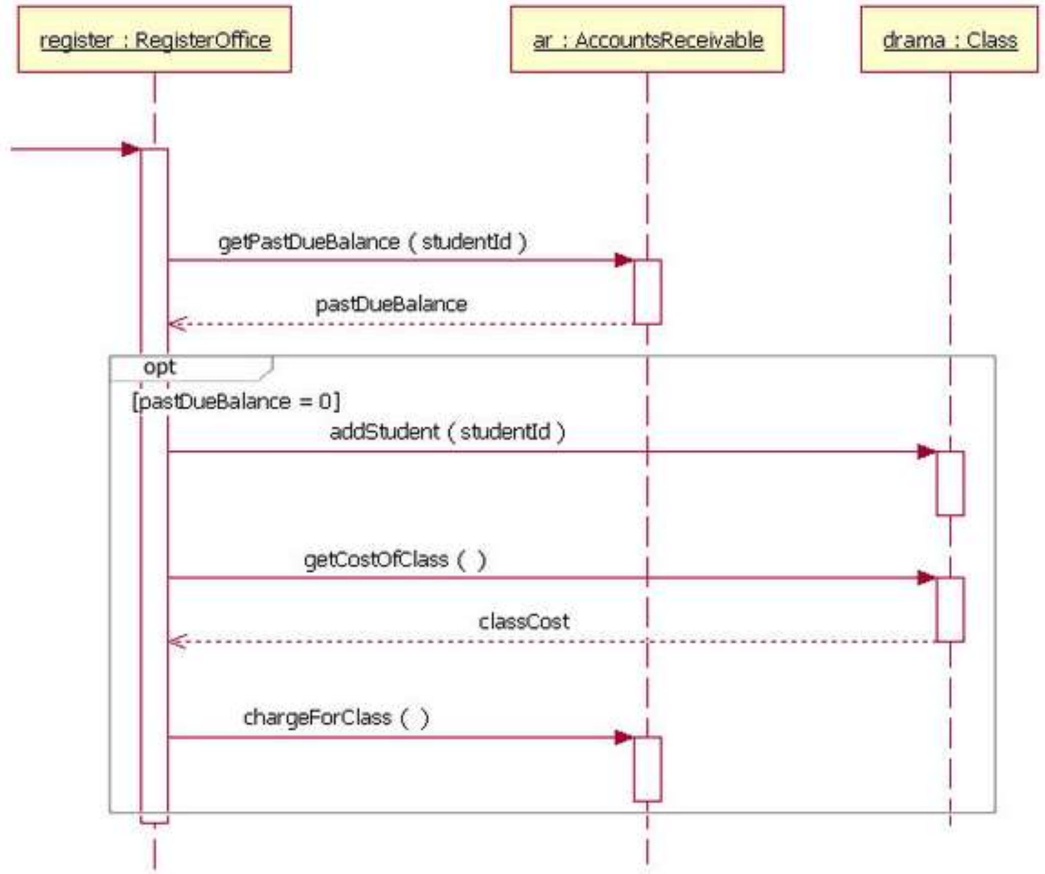


Figure: A sequence diagram fragment that includes an option combination fragment

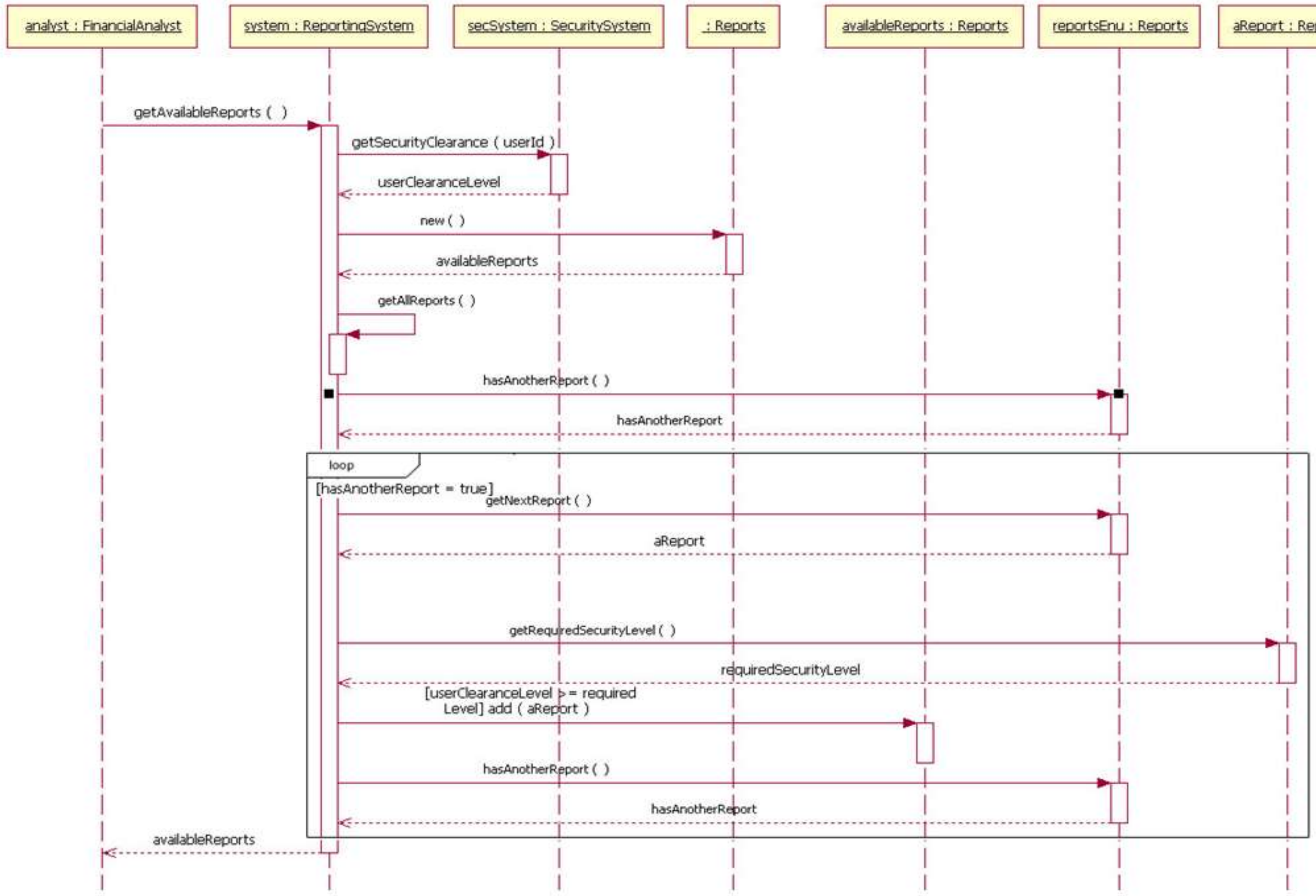
Option – Example

- ✓ If a student's past due balance equals zero, then the addStudent, getCostOfClass, and chargeForClass messages are sent.
- ✓ If the student's past due balance does not equal zero, then the sequence skips sending any of the messages in the option combination fragment.
- ✓ The example in last figure sequence diagram fragment includes a guard for the option; however, the guard is not a required element.
- ✓ In high-level, abstract sequence diagrams you might not want to specify the condition of the option. You may simply want to indicate that the fragment is optional.

Loops

- ✓ To model a repetitive sequence.
- ✓ In frame's namebox the text "loop" is placed.
- ✓ Inside the frame's content area the loop's guard is placed.
- ✓ Then the loop's sequence of messages is placed in the remainder of the frame's content area.
- ✓ 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]" (e.g., "minint = 1") 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.

Loops - Example



Loops – Example

- ✓ The loop shown in last figure executes until the reportsEnu object's hasAnotherReport message returns false.
- ✓ The loop in this sequence diagram uses a Boolean test to verify if the loop sequence should be run.
- ✓ To read this diagram, you start at the top, as normal. When you get to the loop combination fragment a test is done to see if the value hasAnotherReport equals true.
- ✓ If the hasAnotherReport value equals true, then the sequence goes into the loop fragment.
- ✓ You can then follow the messages in the loop as you would normally do in a sequence diagram .

Referencing another sequence diagram

- ✓ In UML 2, the "Interaction Occurrence" element was introduced.
- ✓ Interaction occurrences add the ability to compose primitive sequence diagrams into complex sequence diagrams.
- ✓ An interaction occurrence element is drawn using a frame, text "ref" is placed inside the frame's namebox.
- ✓ Name of the sequence diagram being referenced is placed inside the frame's content area along with any parameters to the sequence diagram.
- ✓ The notation of the referenced sequence diagram's name follows the pattern of:

sequence diagram name[(arguments)] [: return value]

Referencing another sequence diagram - Example

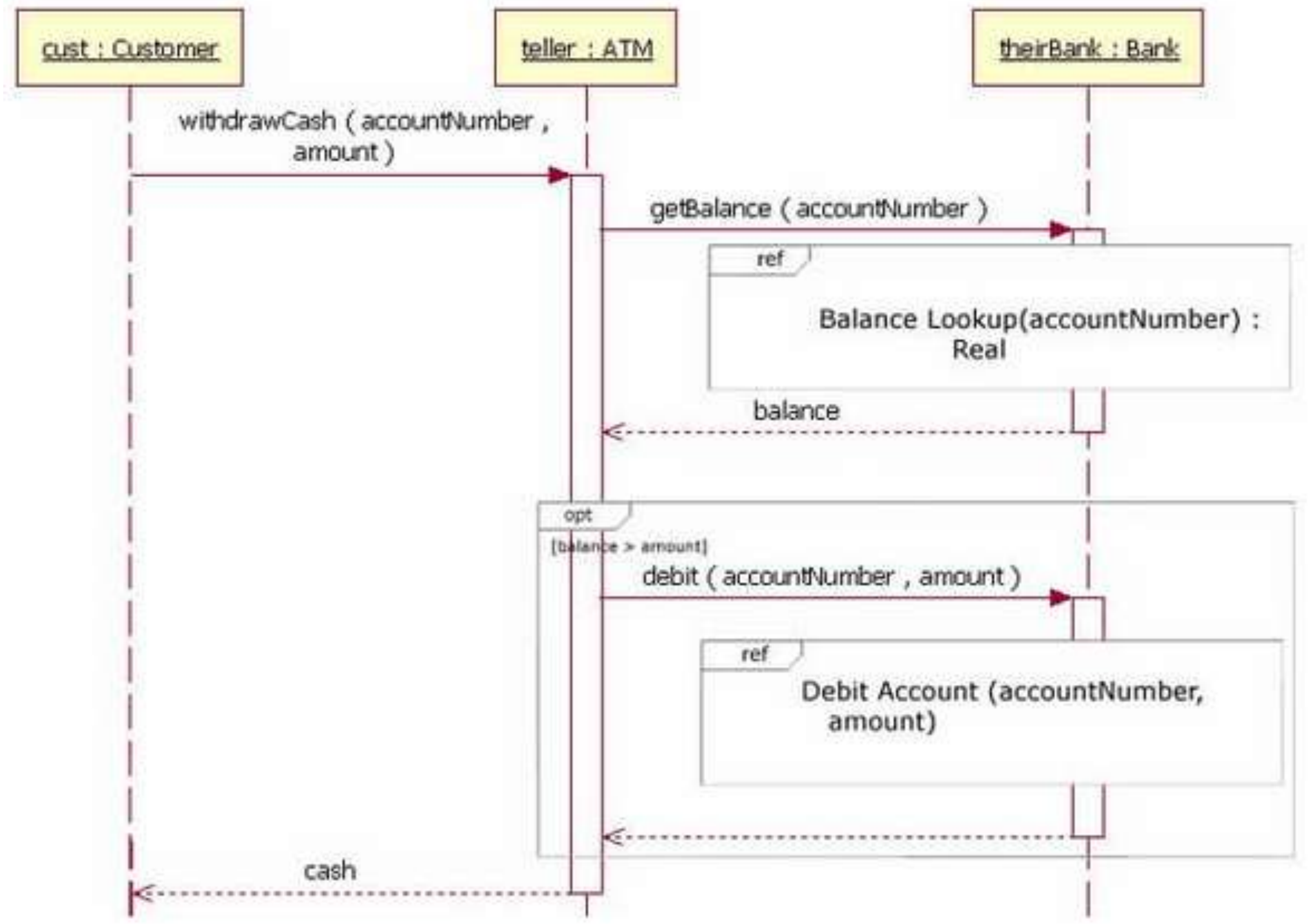
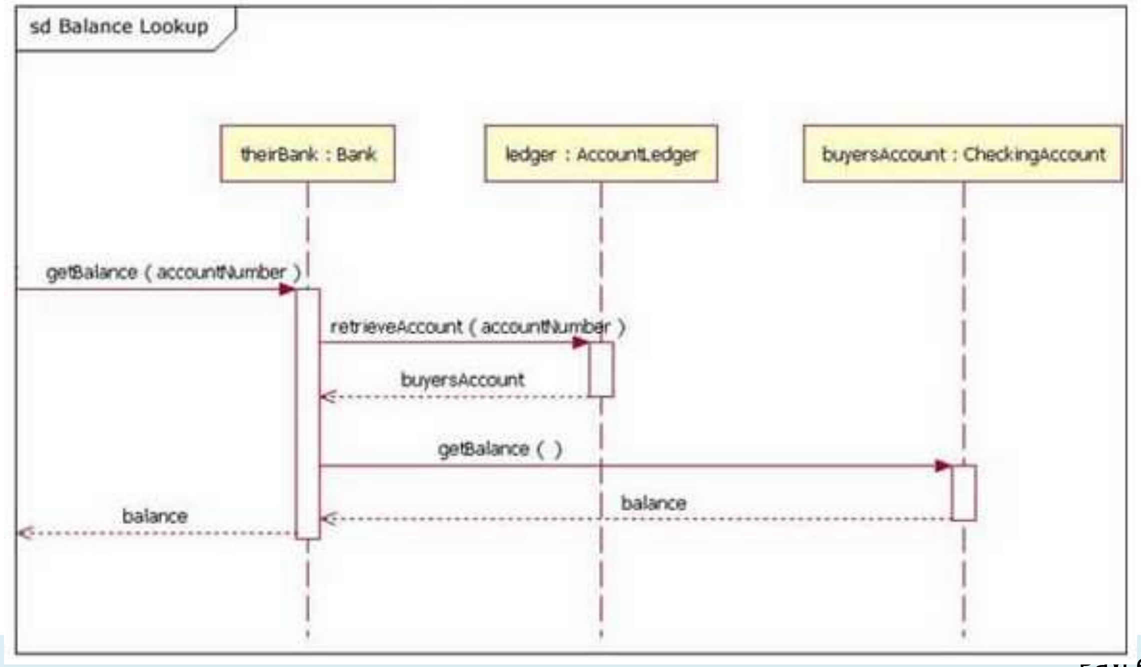

cust : Customer
teller : ATM
theirBank : Bank
withdrawCash (accountNumber , amount)
getBalance (accountNumber)
ref
Balance Lookup(accountNumber) : Real
balance
opt
[balance > amount]
debit (accountNumber , amount)
ref
Debit Account (accountNumber, amount)
cash

Figure: A sequence diagram that references two different sequence diagrams

Gates

- 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.
- Gates - entry gate (getBalance) and exit gate (balance).



Break

- ✓ Identical in every way to the option combined fragment, with two exceptions.
- ✓ First, frame has a namebox with the text "break" instead of "option”.
- ✓ Second, when a break combined fragment's message is to be executed, the enclosing interaction's remainder messages will not be executed because the sequence breaks out of the enclosing interaction.
- ✓ Thus break combined fragment is much like the break keyword in a programming language like C++ or Java.

•Figure: A sequence diagram with gates.

Break - Example

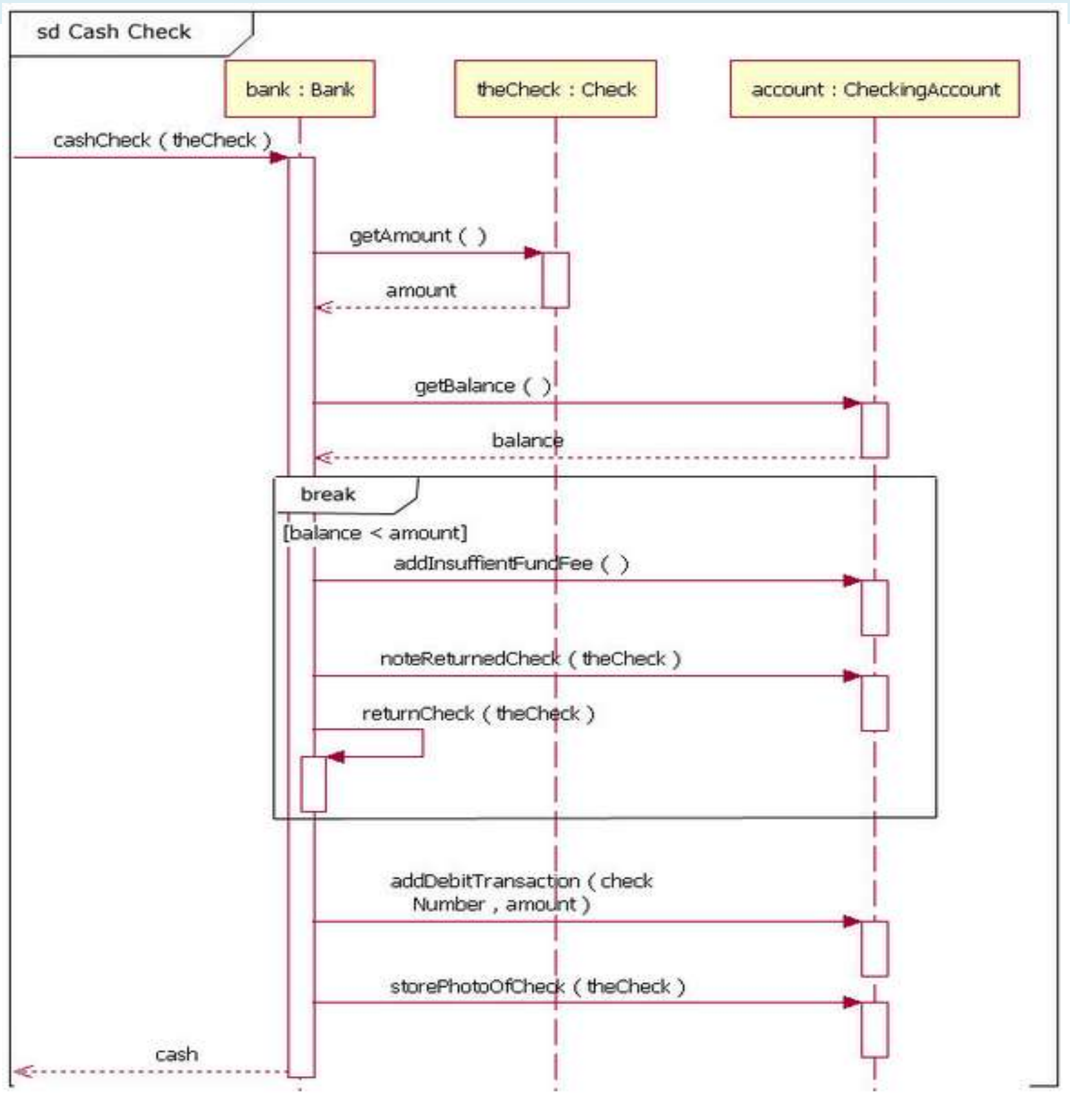
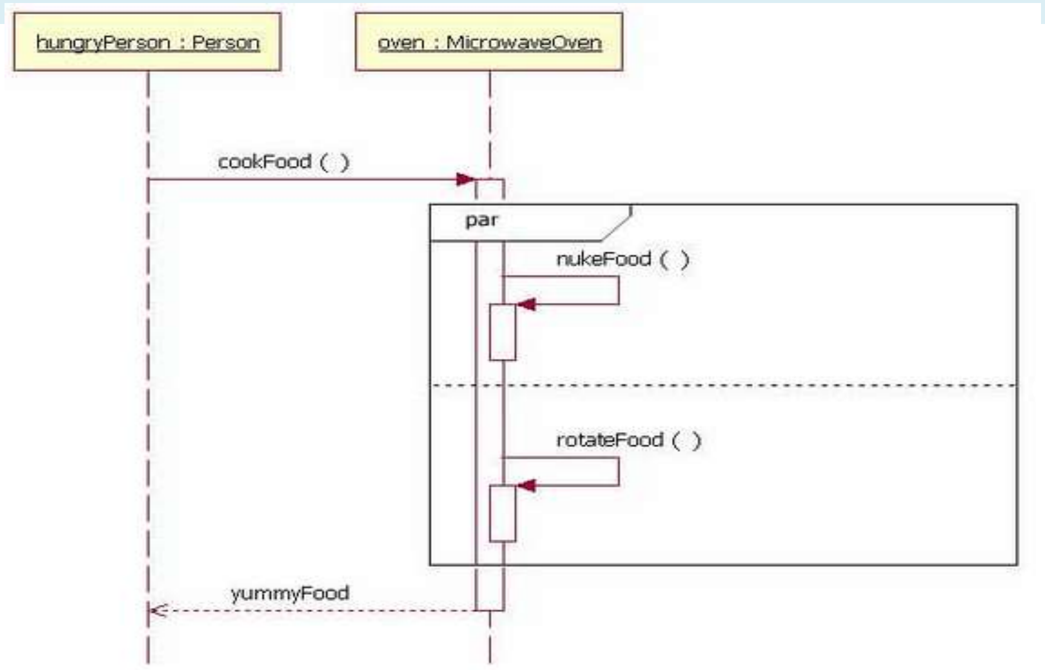


Figure: A sequence diagram with the fragment using a break.

Parallel

- ✓ Modern computer systems are advancing in complexity and at times perform concurrent tasks.
- ✓ Parallel element is used for creating a sequence diagram that shows parallel processing activities.
- ✓ Drawn using a frame and place the text "par" in the frame's namebox.
- ✓ Break up the frame's content section into horizontal operands separated by a dashed line.
- ✓ Each operand in the frame represents a thread of execution done in parallel.

Parallel - Example



- Figure: A microwave is an example of an object that does two tasks in parallel

✓ A hungryPerson sends the cookFood message to the oven object. Oven object sends two messages to itself at the same time (nukeFood and rotateFood). After both of these messages are done, the hungryPerson object is returned yummyFood from the oven object.

References

- http://www.ibm.com/developerworks/rational/library/3101.html
- http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/
- http://stackoverflow.com/questions/3434034/how-to-show-protected-internal-in-uml
- http://www.holub.com/goodies/uml/
- http://code.google.com/a/eclipselabs.org/p/javadocasumlview/wiki/JavaToUMLDescription
- http://books.google.co.in/books?id=M856gzs7L3gC&pg=PA13&lpg=PA13&dq=is+aggregation+strong+or+weak+form+of+association&source=bl&ots=pm4zv9-wIe&sig=EOi8f0v9o5r4syaoHlOg_Wr8RNM&hl=en&sa=X&ei=ww2VT_mILMHWrQfGtumkBQ&ved=0CEMQ6AEwBQ#v=snippet&q=is%20aggregation%20strong%20or%20weak%20form%20of%20association&f=false