PURPOSE

One of the primary uses of sequence diagrams is in **the transition from requirements expressed as use cases to the next and more formal level of refinement**

The main purpose of a sequence diagram **is to define event sequences that result in some desired outcome**.

The **focus is less on messages themselves** and more on the order in **which messages** occur

USEFUL TO

communicate how the business currently works by **showing how various business objects interact**

can be used as a **requirements document** to communicate requirements for a future system implementation

documenting **how a future system should behave**.

design phase, architects and developers can use the diagram to force out the system's object interactions

is very useful when **transitioning a system to another person** or organization

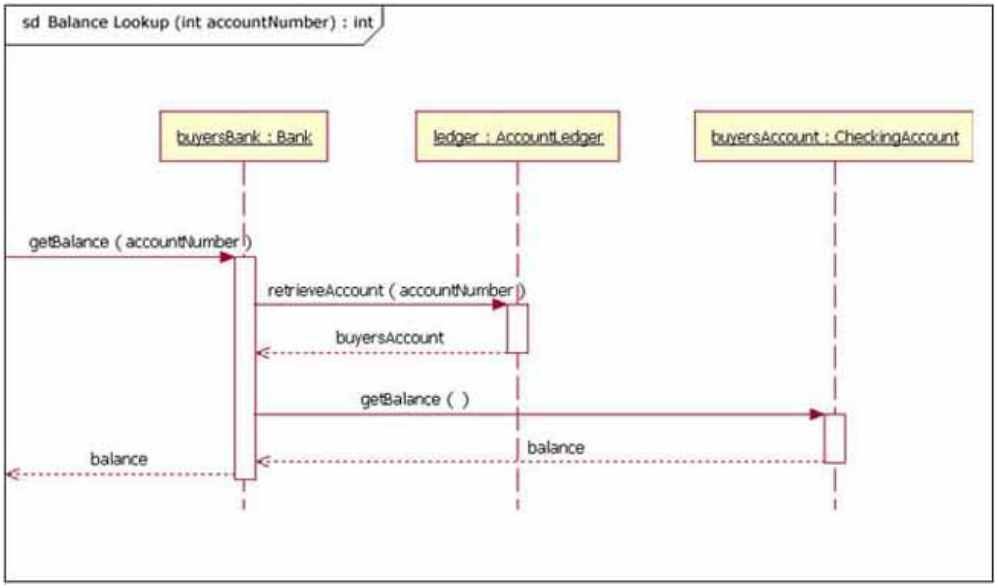
HOW TO READ

the **vertical dimension** shows, **top down**, the time sequence of messages/calls as they occur, and the **horizontal dimension** shows, **left to right,** the object instances that the messages are sent to.

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.

NOTATION

Frame element:
(optional) A frame element provides a **consistent place for a diagram's label**
On sequence diagrams 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 (as seen in Figure 2)



Lifelines

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

Lifelines represent either **roles or object instances** that participate in the sequence being modeled

When an **underline is used**, it means 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 like student:Student)

When modeling an **unnamed instance** on a sequence diagram, the lifeline's name follows the same pattern as a named instance; but instead of providing an instance name, **that portion of the lifeline's name is left blank**

MESSAGES

To show an object (i.e., lifeline) sending a message to another object, you **draw a line to the receiving object**.

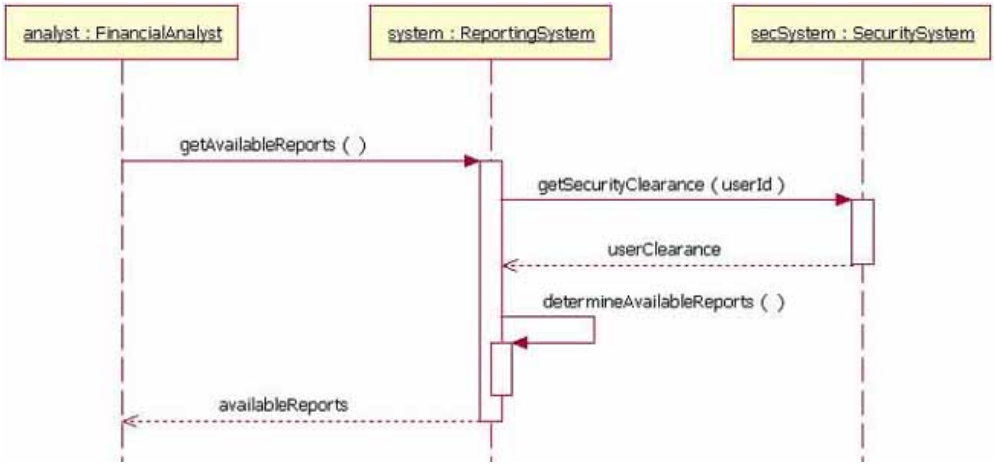
with a **solid** arrowhead (if a **synchronous** call operation) with a stick arrowhead (if an **asynchronous** signal).

The message/method name is placed above the arrowed line

The message that is being sent to the receiving object **represents an operation/method** that the receiving object's class implements.

a return message(optional) is drawn as a dotted line with an open arrowhead back to the originating lifeline above you place the **return value** from the operation. They are useful if finer detail is required

To draw an **object calling itself**, you connect the message back to the object itself.



GUARDS

a guard could only be assigned to a **single message**
To draw a guard on a sequence diagram in UML 1.x, you placed the guard element above the message line being guarded and **in front of the message name**

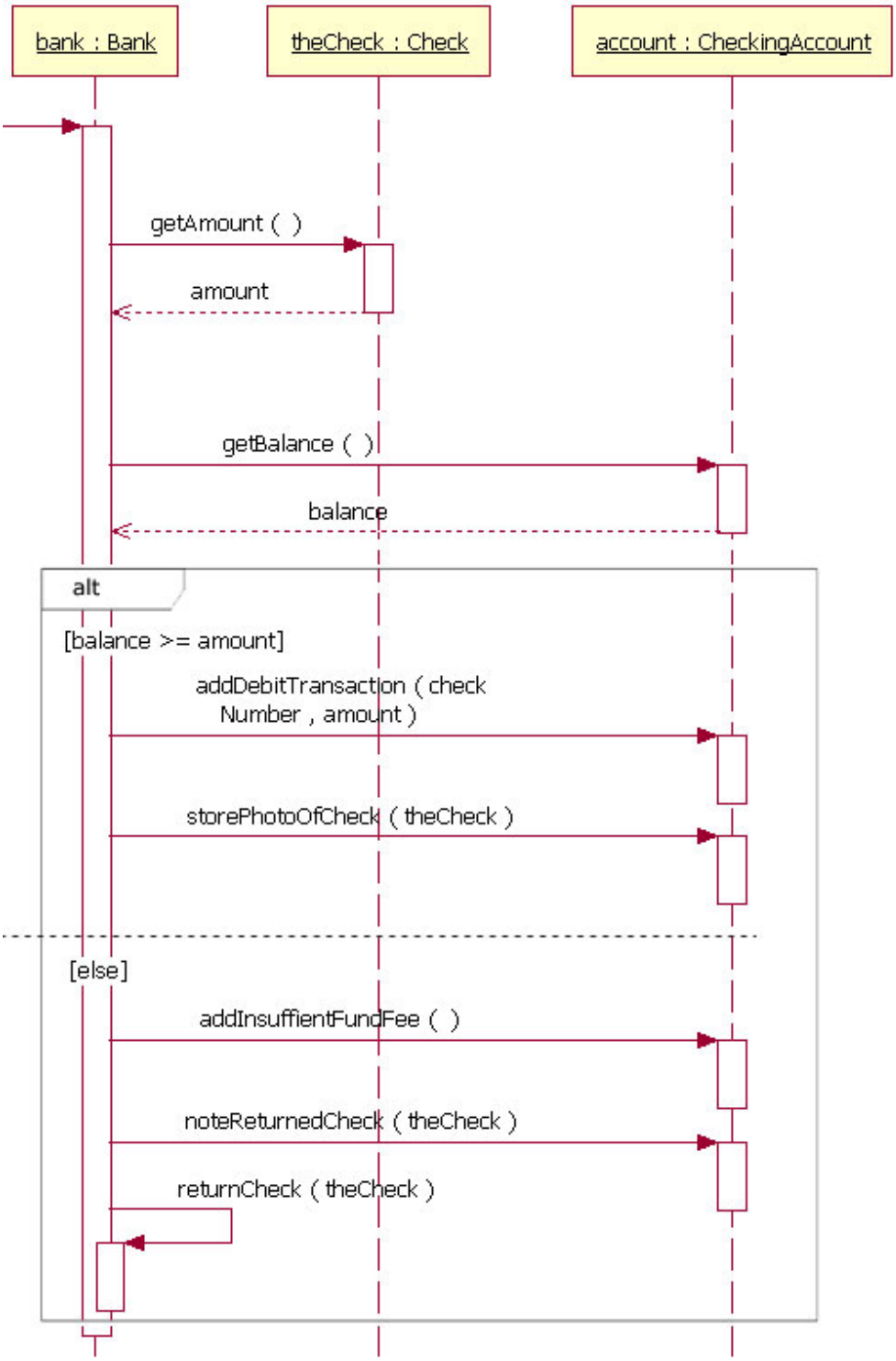
the format is: [Boolean Test]
e.g. [pastDueBalance = 0]

COMBINES FRAGMENTS

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

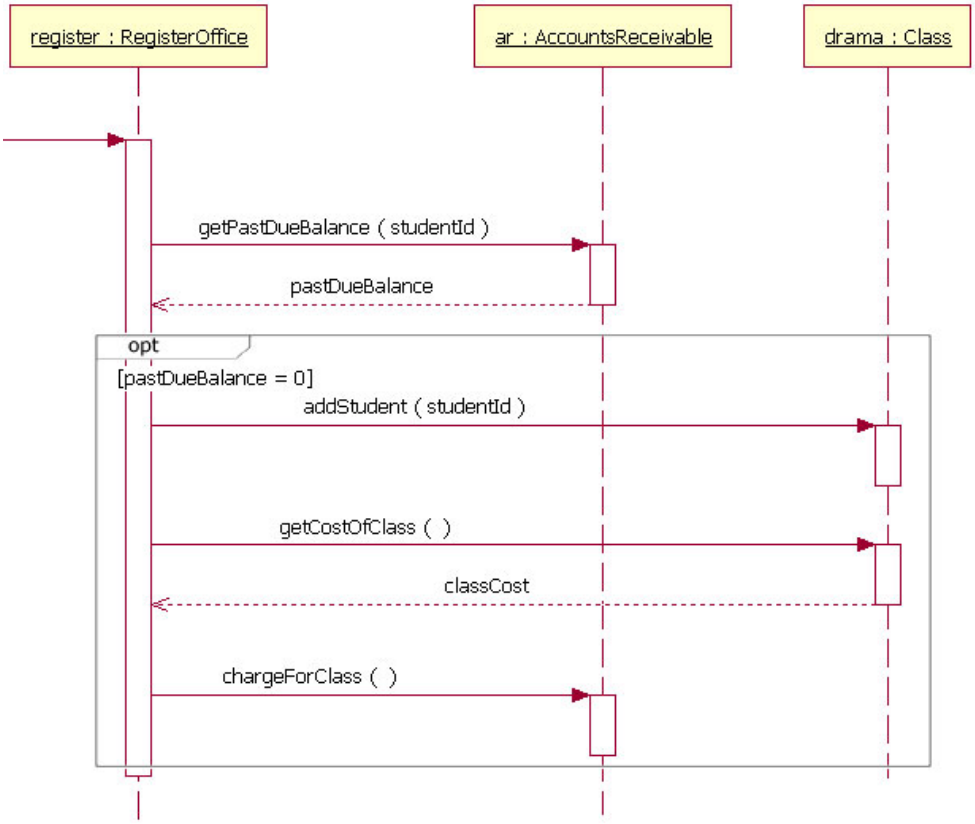
Alternatives are used to designate a mutually exclusive choice between two or more message sequences
Alternatives allow the modeling of the classic "if then

else"



The **option** combination fragment is used to model a sequence that, given a certain condition, will occur;

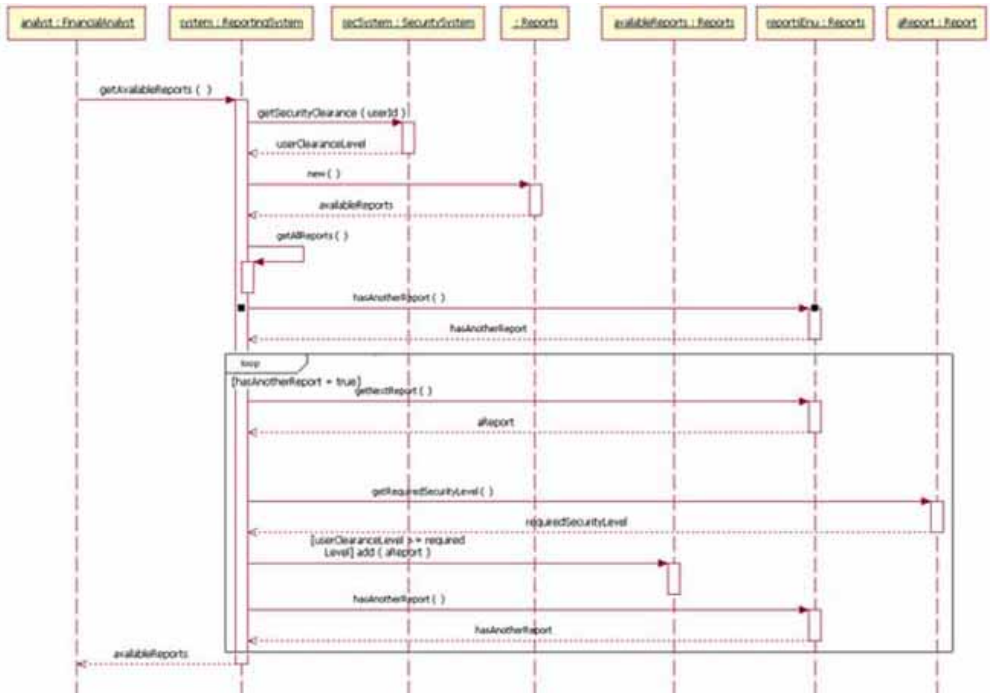
otherwise, the sequence does not occur. An option is used to model a simple **"if then" statement**



The **loop** combination fragment is very similar in appearance to the option combination fragment. You draw a frame, and in the frame's namebox the text "loop" is placed

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]"

(e.g., "maxint = 5")


analyst : FinancialAnalyst
system : ReportingSystem
secSystem : SecuritySystem
: Reports
availableReports : Reports
reportsEnu : Reports
aReport : Report
getAvailableReports ( )
getSecurityClearance (userId)
userClearanceLevel
new ( )
availableReports
getAllReports ( )
hasAnotherReport ( )
hasAnotherReport
loop
[hasAnotherReport = true]
getNextReport ( )
aReport
getRequiredSecurityLevel ( )
requiredSecurityLevel
[userClearanceLevel >= required Level] add (aReport)
hasAnotherReport ( )
hasAnotherReport
availableReports