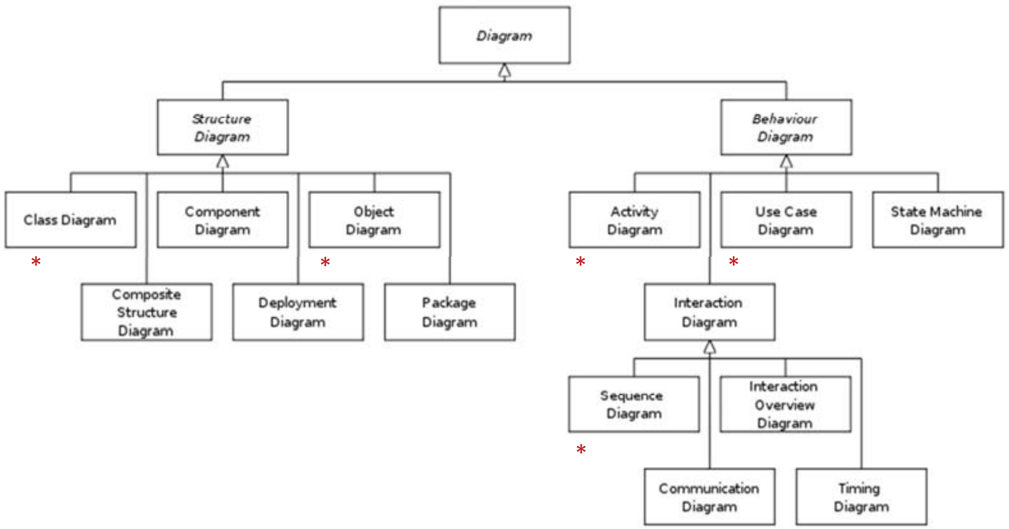
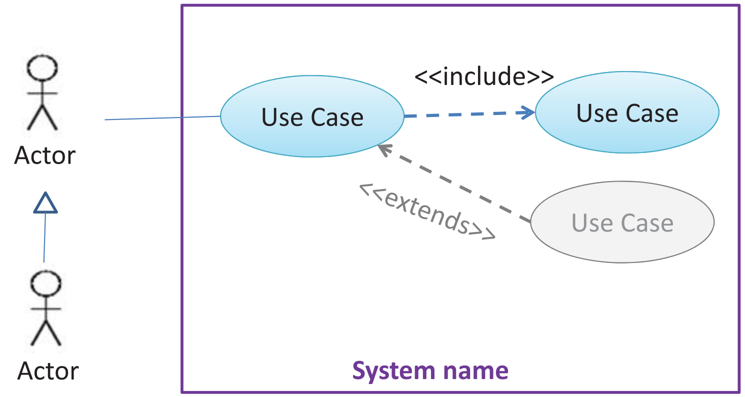
UML Reference Sheet



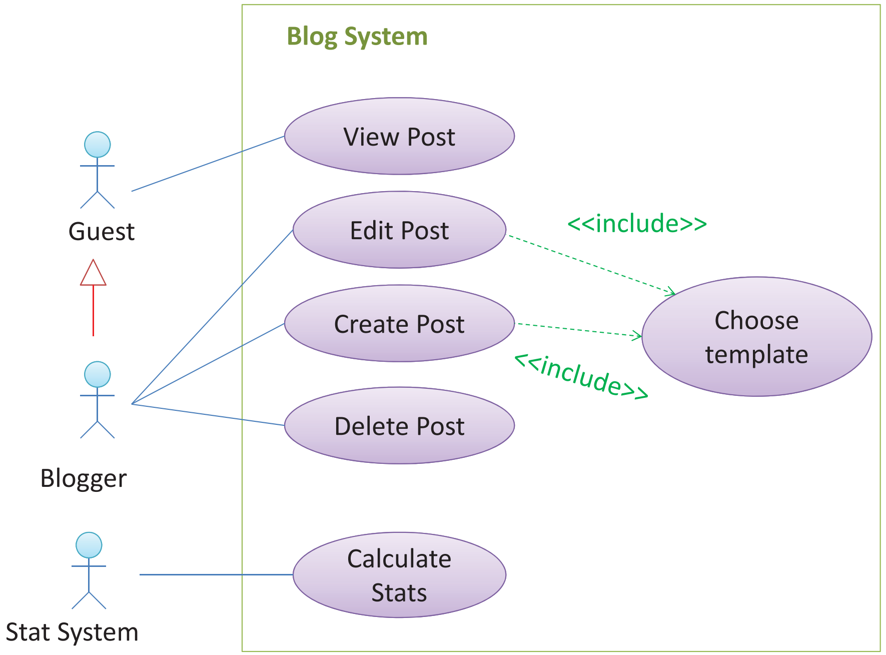
* Examinable

[This UML reference sheet was inspired by Martin Fowler's *UML Distilled*.]

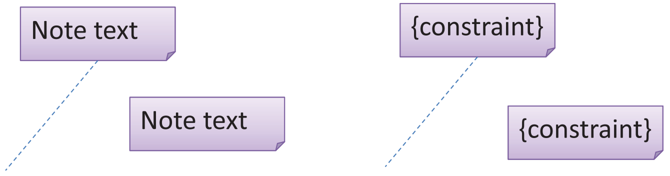
Use case diagrams



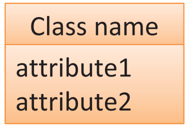
Use case diagrams [example]



Notes and constraints



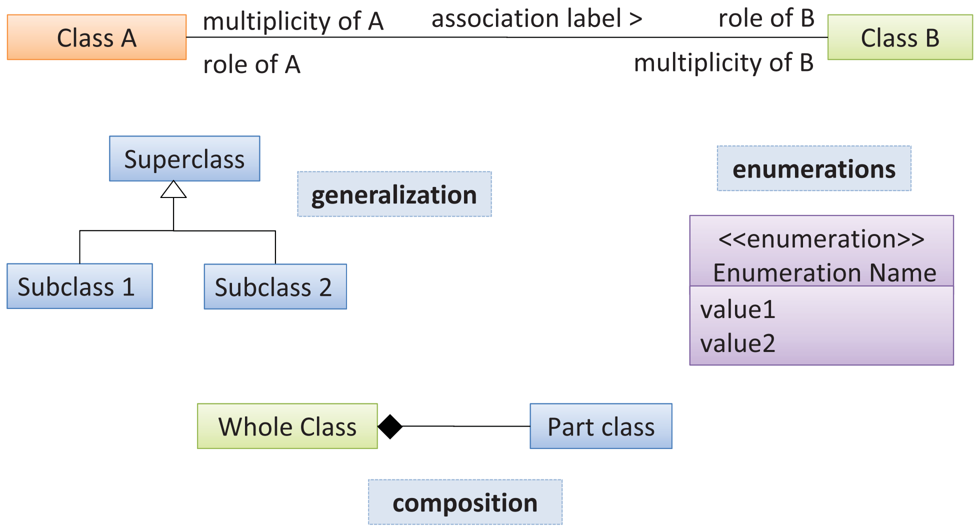
Class diagrams



Class name

attribute1
attribute2

association



Class A — multiplicity of A, role of A — association label > — role of B, multiplicity of B — Class B

Superclass — Subclass 1, Subclass 2

generalization

enumerations

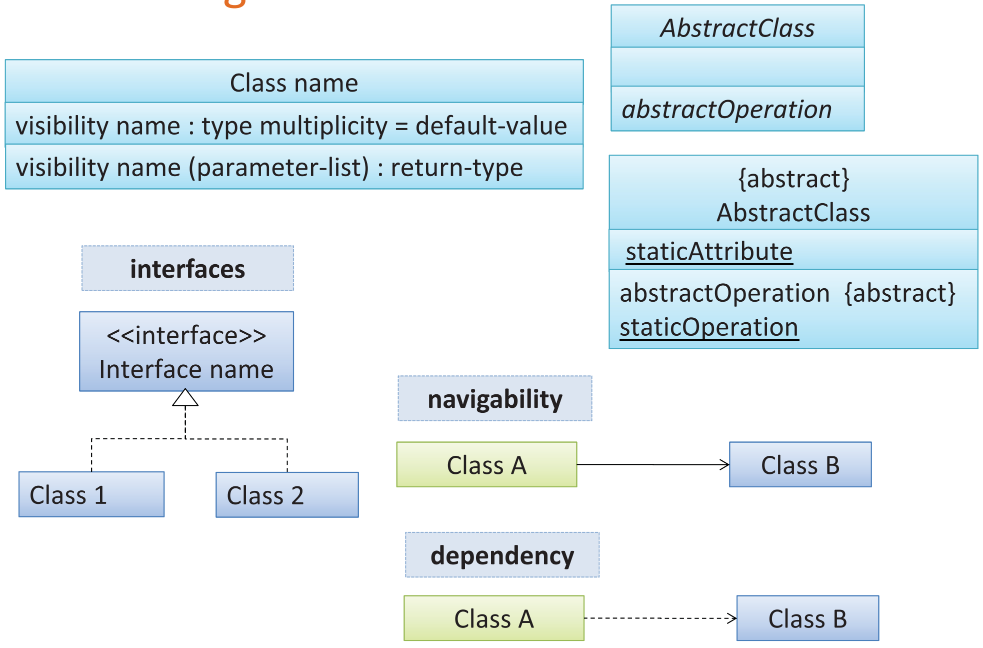
<<enumeration>>
Enumeration Name
value1
value2

Whole Class — Part class

composition

Class diagrams

abstract/ static



AbstractClass

abstractOperation

Class name

visibility name : type multiplicity = default-value

visibility name (parameter-list) : return-type

{abstract}
AbstractClass

staticAttribute

abstractOperation {abstract}
staticOperation

interfaces

<<interface>>
Interface name

Class 1, Class 2

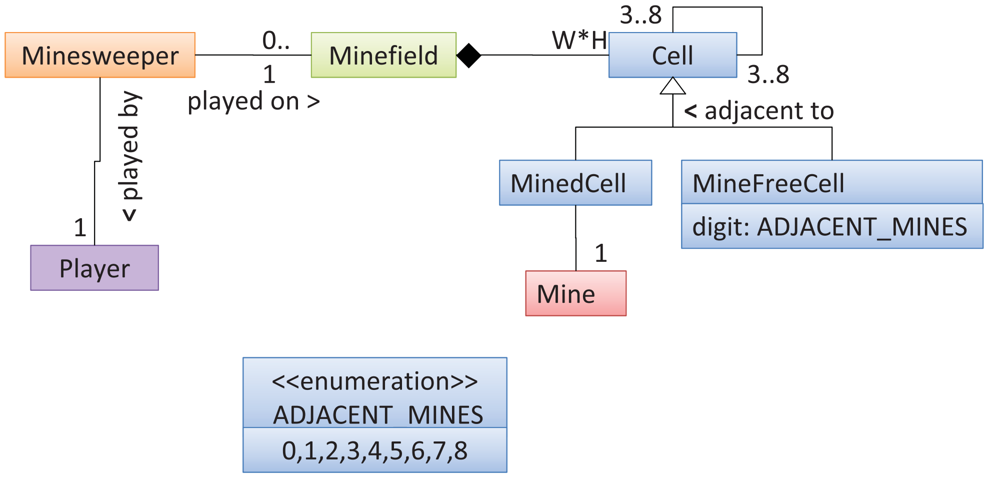
navigability

Class A → Class B

dependency

Class A → Class B

Class diagrams [example]



Minesweeper — 0.. — 1 — played on > — Minefield

Minesweeper — < played by — 1 — Player

Minefield — W*H — Cell

Cell — 3..8 — 3..8 — < adjacent to

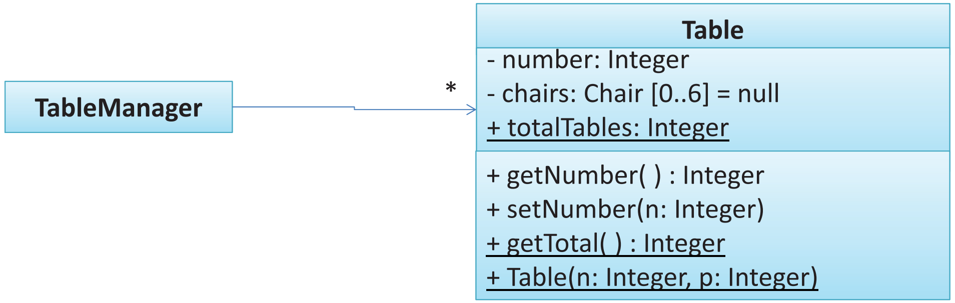
Cell — MinedCell, MineFreeCell

MineFreeCell: digit: ADJACENT_MINES

MinedCell — 1 — Mine

<<enumeration>>
ADJACENT_MINES
0,1,2,3,4,5,6,7,8

class diagrams [example]



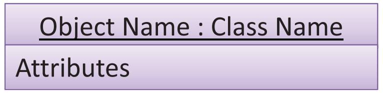
TableManager — * — **Table**

Table

- number: Integer
- chairs: Chair [0..6] = null
+ totalTables: Integer

+ getNumber( ) : Integer
+ setNumber(n: Integer)
+ getTotal( ) : Integer
+ Table(n: Integer, p: Integer)

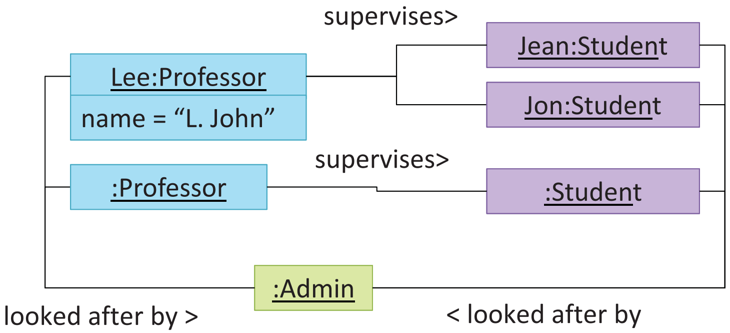
Object diagrams



Object Name : Class Name

Attributes

Object diagrams [example]



supervises>

Jean:Student

Lee:Professor

name = “L. John”

Jon:Student

supervises>

:Professor

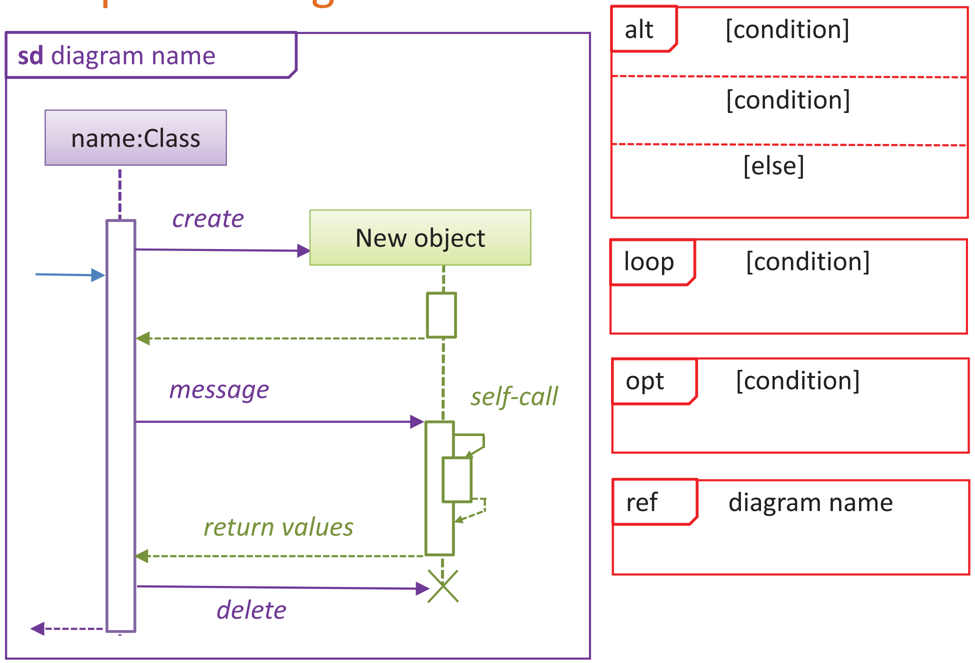
:Student

:Admin

looked after by >

< looked after by

Sequence diagrams



sd diagram name

name:Class

create

New object

message

self-call

return values

delete

alt [condition]

[condition]

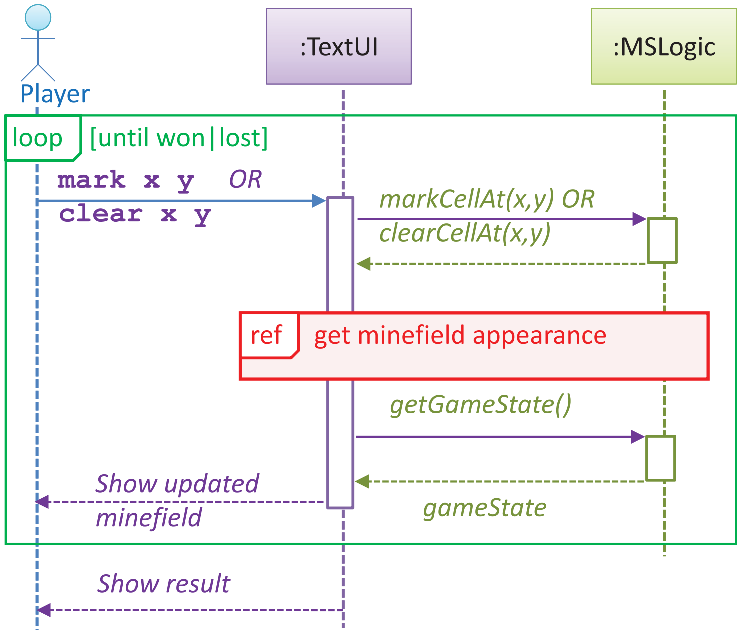
[else]

loop [condition]

opt [condition]

ref diagram name

Sequence diagrams [example]



Player

:TextUI

:MSLogic

loop [until won|lost]

`mark x y` *OR*

`clear x y`

markCellAt(x,y) OR

clearCellAt(x,y)

ref get minefield appearance

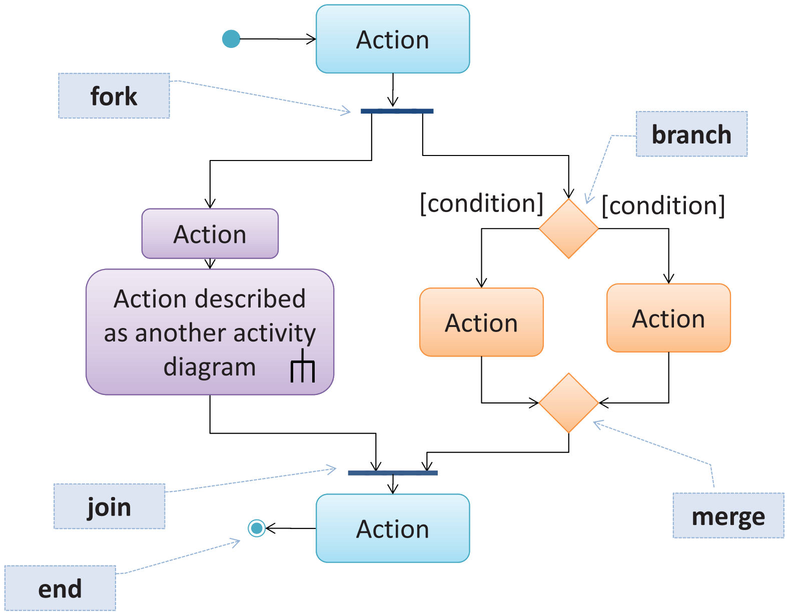
getGameState()

Show updated minefield

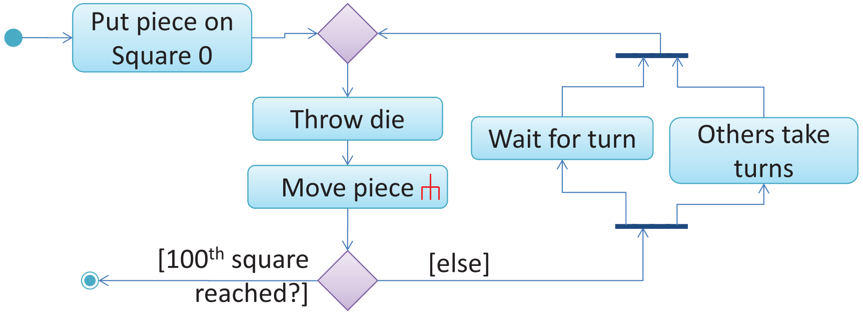
gameState

Show result

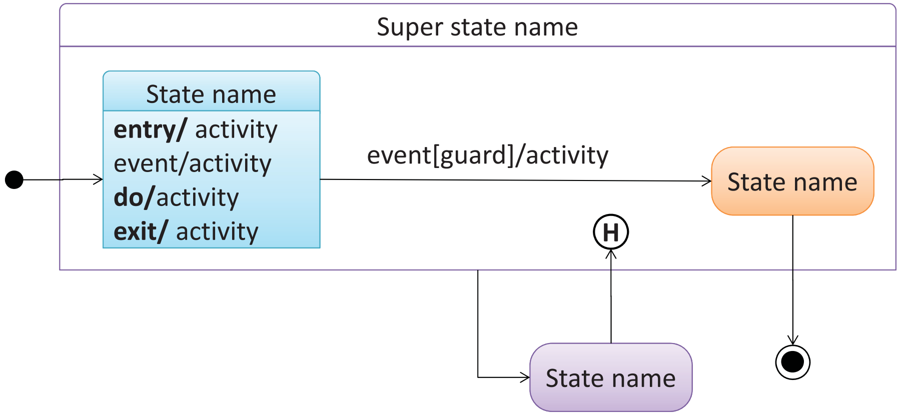
Activity diagrams



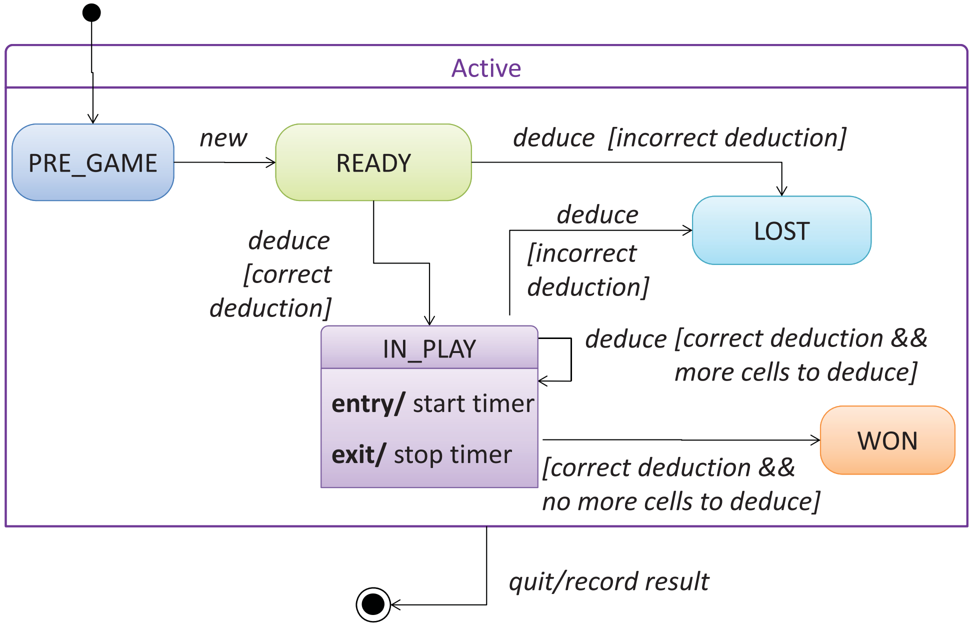
Activity diagrams [example]



State machine diagrams

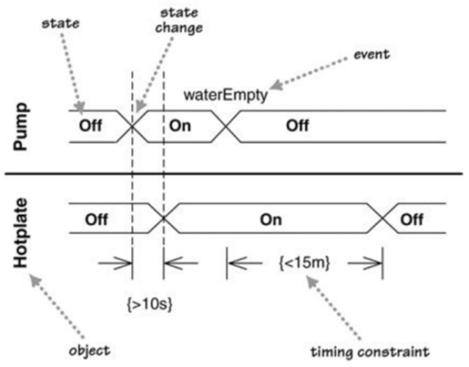


State machine diagrams [example]

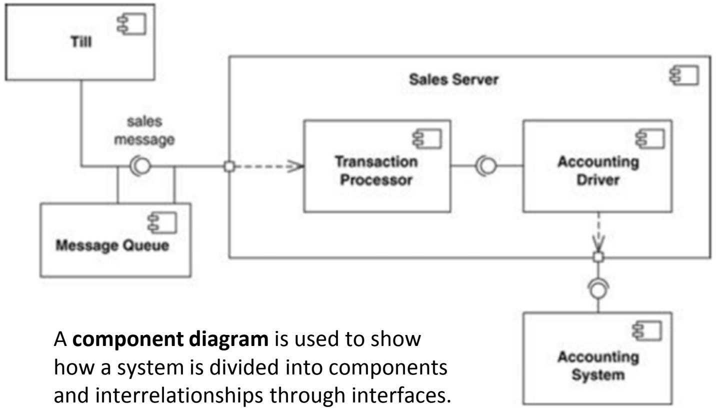

Active
PRE_GAME
new
READY
deduce [incorrect deduction]
deduce [incorrect deduction]
LOST
deduce [correct deduction]
IN_PLAY
entry/ start timer
exit/ stop timer
deduce [correct deduction && more cells to deduce]
WON
[correct deduction && no more cells to deduce]
quit/record result

Other UML diagrams

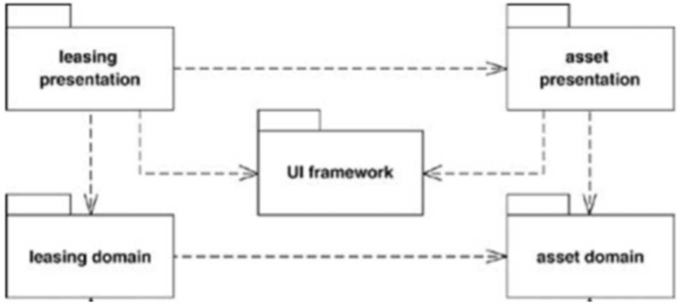
This page contains a peek of UML diagrams not covered in lectures. These are not examinable.



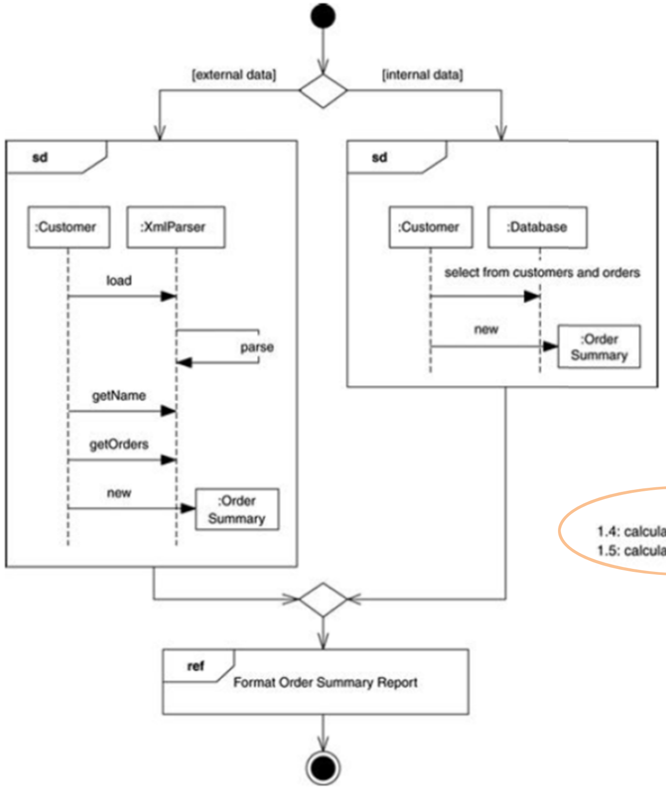
Timing diagrams focus is on timing constraints.



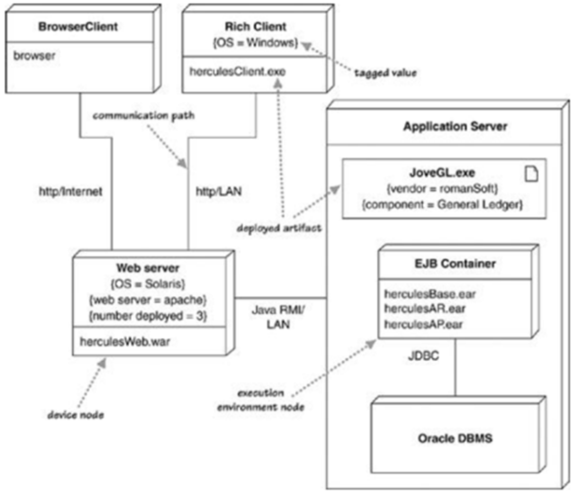
A **component diagram** is used to show how a system is divided into components and interrelationships through interfaces.



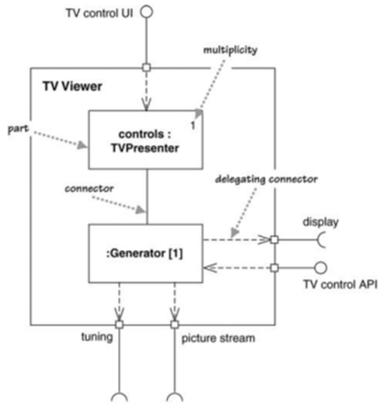
A package diagram shows packages and their dependencies. A package is a grouping construct for grouping UML elements (classes, use cases, etc.).



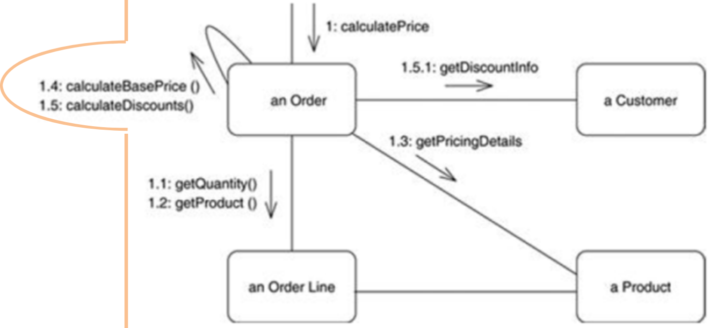
Interaction overview diagrams are a combination of activity diagrams and sequence diagrams.



Deployment diagrams show a system's physical layout, revealing which pieces of software run on what pieces of hardware.



A composite structure diagram hierarchically decomposes a class into its internal structure.



Communication diagrams are like sequence diagrams but emphasize the data links between the various participants in the interaction rather than the sequence of interactions.