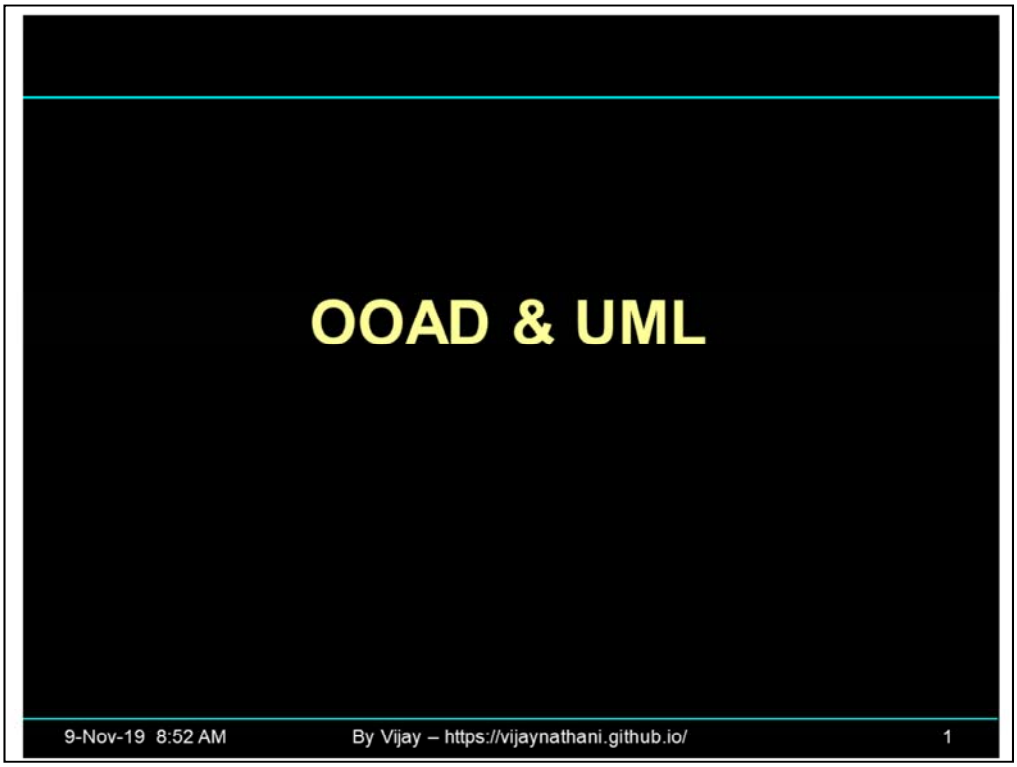
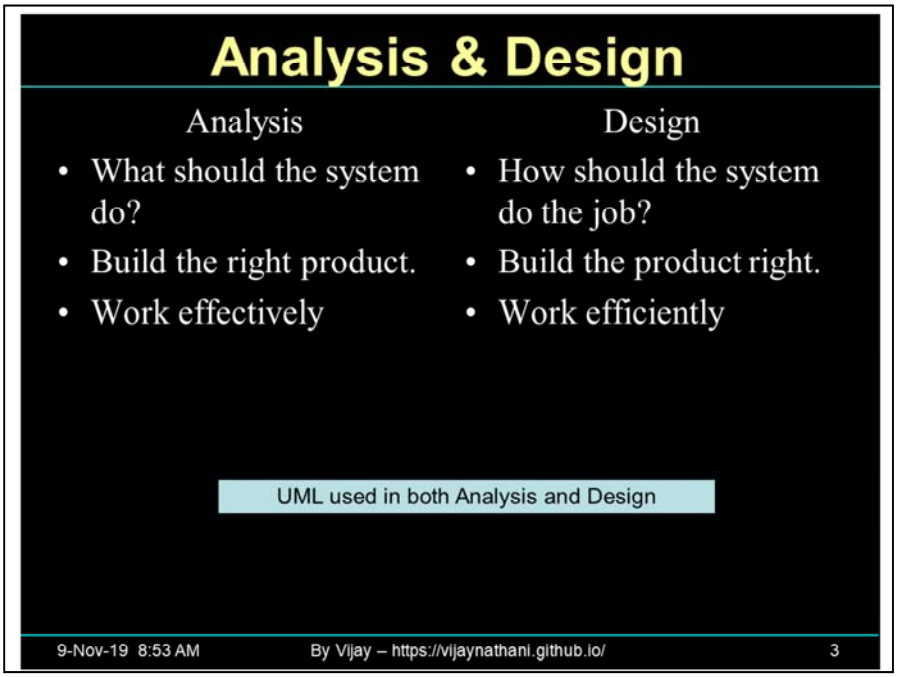
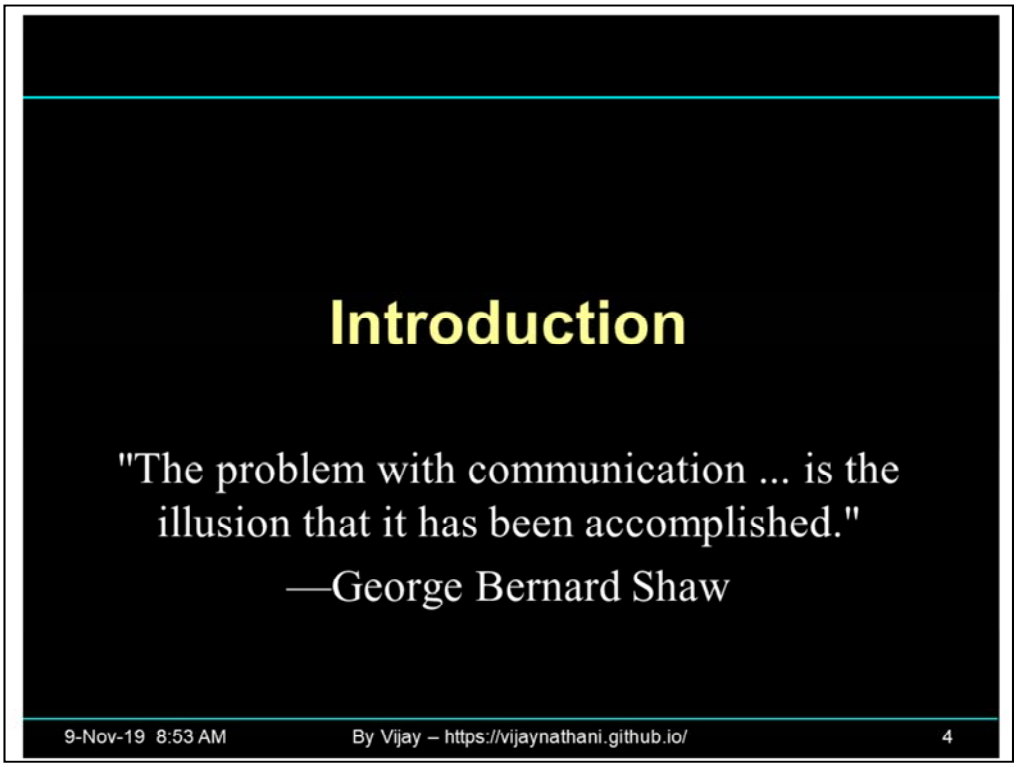
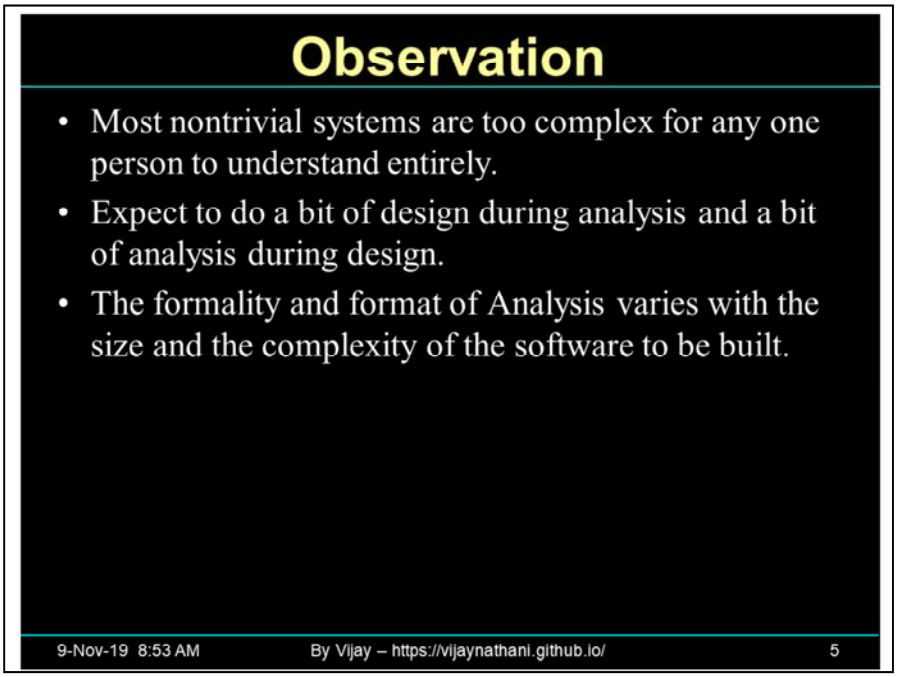

OOAD & UML
9-Nov-19 8:52 AM
By Vijay – https://vijaynathani.github.io/
1

Contents


Analysis & Design
Analysis
What should the system do?
Build the right product.
Work effectively
Design
How should the system do the job?
Build the product right.
Work efficiently
UML used in both Analysis and Design
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
3


Introduction
"The problem with communication ... is the illusion that it has been accomplished."
—George Bernard Shaw
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
4

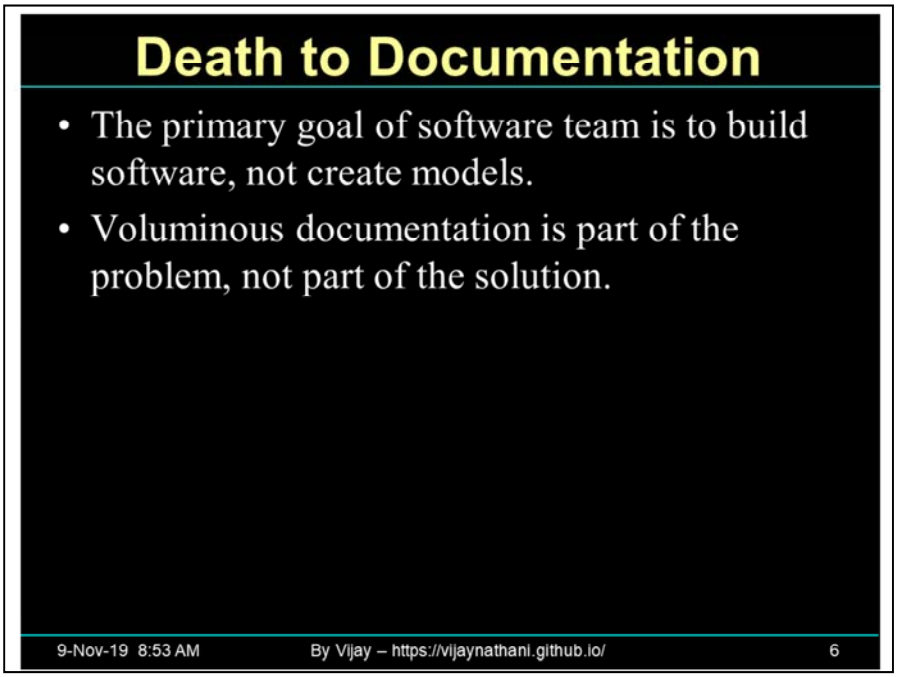


Question till you understand.

Analysis is a good thing, but we have to know when to stop.

Not all customers can give you correct data. Get feedback as soon as possible.

> We need to have a high-level understanding of most of the parts in order to understand what pieces of the system interact with each other, in addition to a deeper understanding of the particular parts on which we're working now.

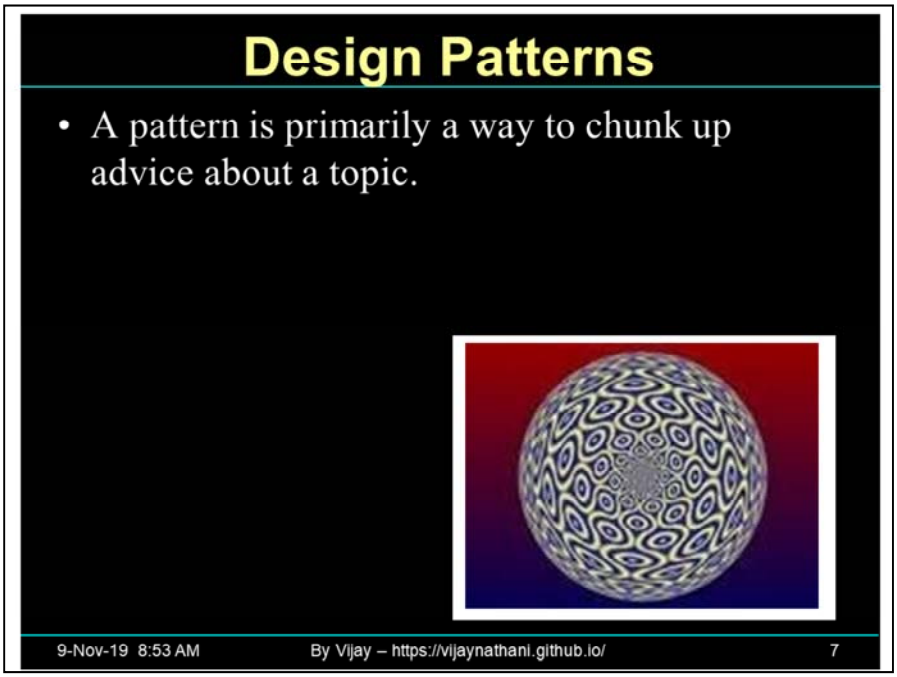


Software without documentation is a disaster.

Code is not the ideal medium for communicating the overall rationale and structure of a system.

Travel light – Have the minimum documentation.

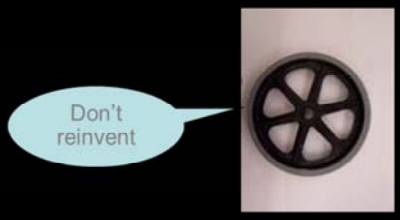
Rule: Produce no document unless its need is immediate and significant.

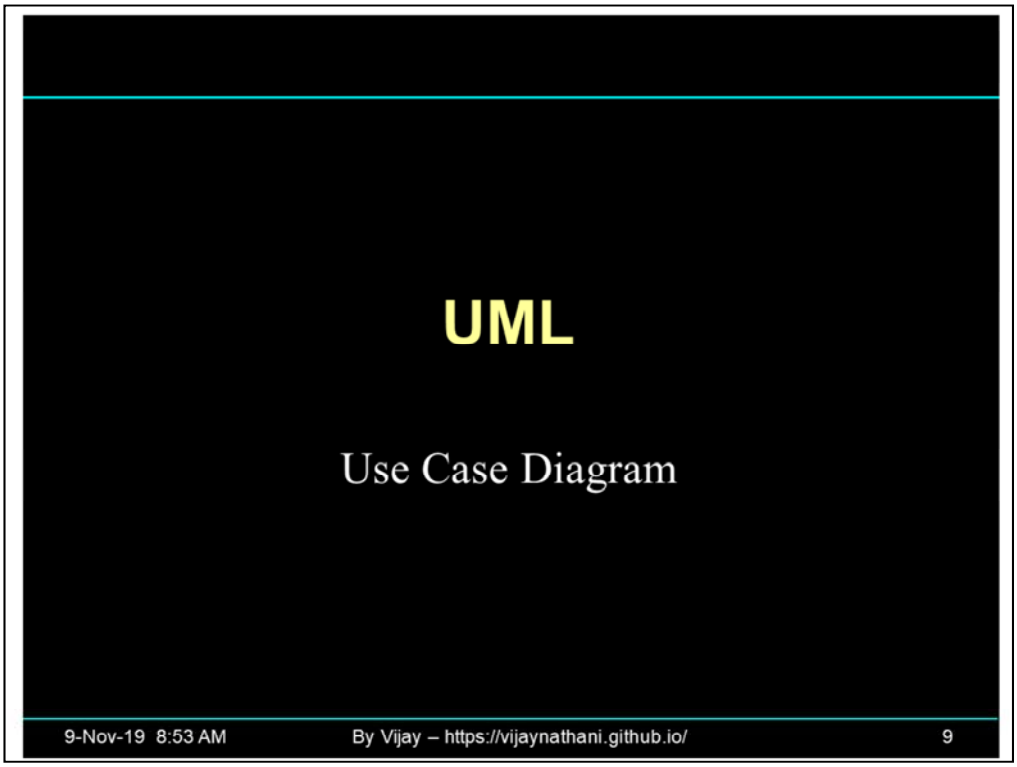


The best designers in any field have an uncanny ability to see patterns that characterize a problem and corresponding patterns that can be combined to create a solution

Observations

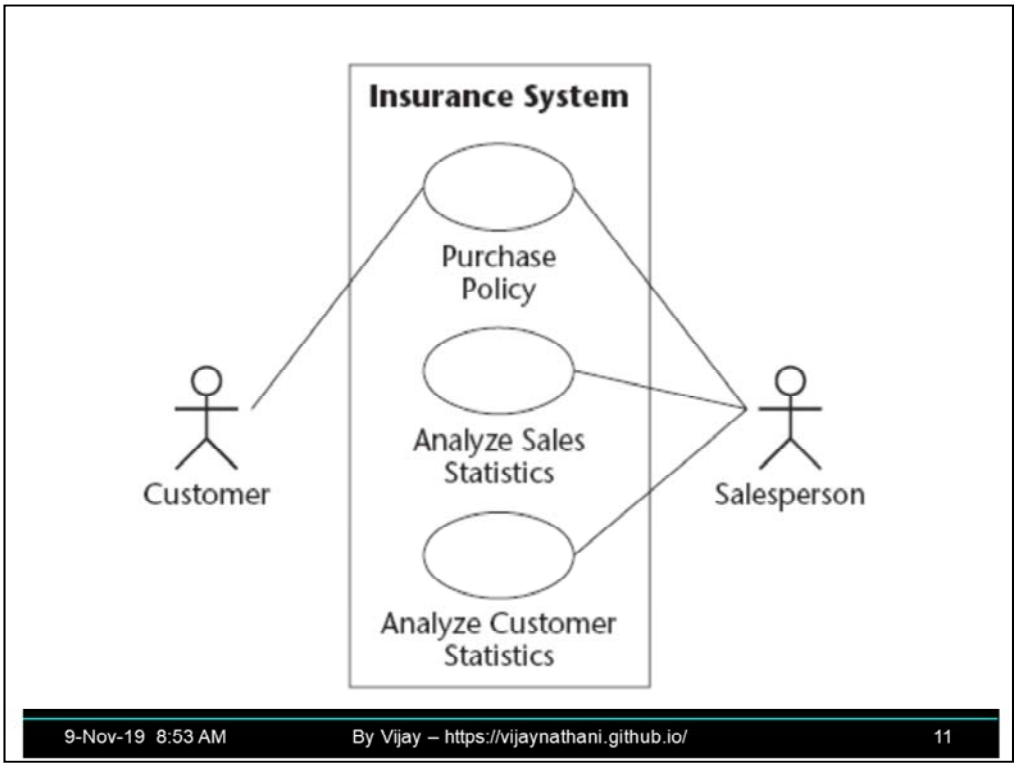
- The beginning of wisdom for a software engineer is to recognize the difference between getting the program to work and getting it right.
- Quality isn't something you lay on top of classes and objects like tinsel on a Christmas tree.
- Analysis and design are both iterative actions

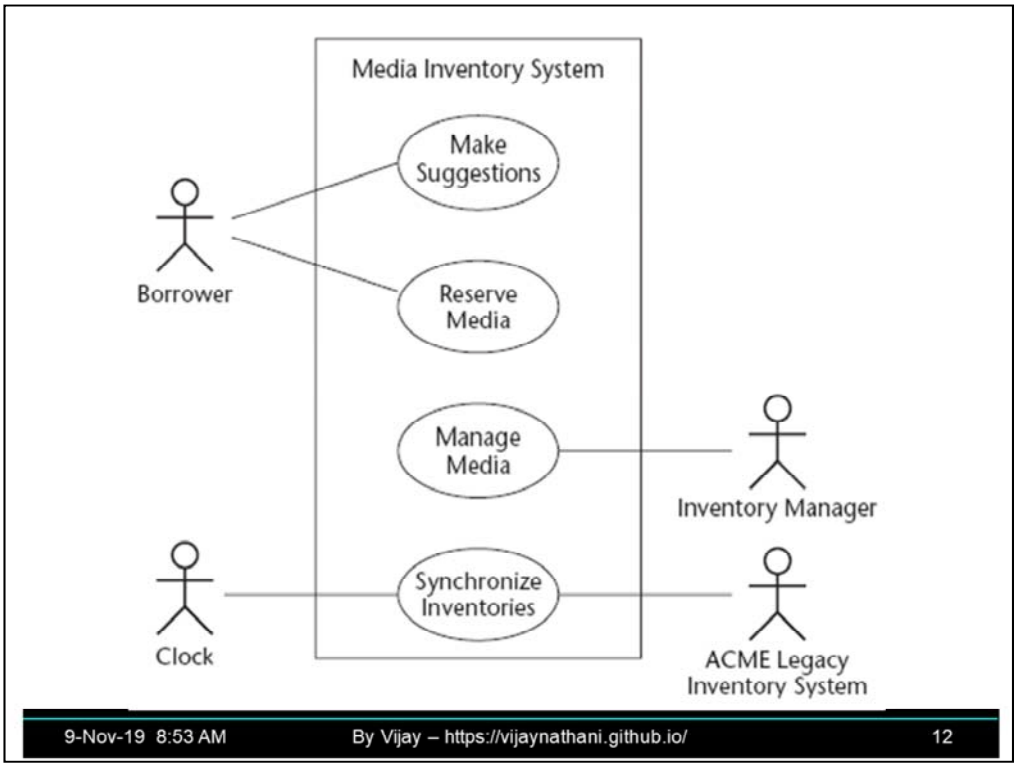


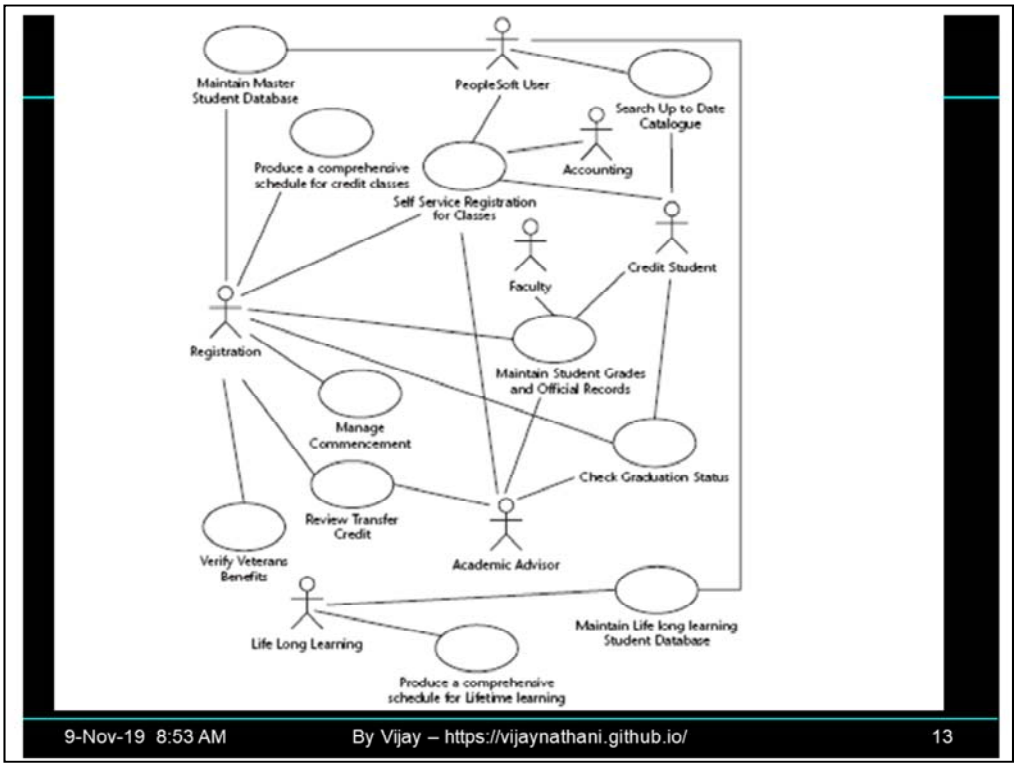

UML
Use Case Diagram
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
9

Use case

- *Buy something*
  - The Requestor initiates a request and sends it to her or his Approver, who completes the request for submission and sends it to the Buyer. The Buyer finds the best vendor, initiates PO with Vendor.
  - At any time prior to receiving goods, Requestor can change or cancel the request. Canceling it removes it from any active processing.


Insurance System
Purchase Policy
Analyze Sales Statistics
Analyze Customer Statistics
Customer
Salesperson
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
11


Media Inventory System
Make Suggestions
Reserve Media
Manage Media
Synchronize Inventories
Borrower
Clock
Inventory Manager
ACME Legacy Inventory System
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
12


Maintain Master Student Database
PeopleSoft User
Search Up to Date Catalogue
Produce a comprehensive schedule for credit classes
Accounting
Self Service Registration for Classes
Credit Student
Faculty
Registration
Maintain Student Grades and Official Records
Manage Commencement
Check Graduation Status
Review Transfer Credit
Verify Veterans Benefits
Academic Advisor
Maintain Life long learning Student Database
Life Long Learning
Produce a comprehensive schedule for Lifetime learning
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
13

Recognize a Good Use Case

- Independent
  - Each use case can be scheduled in a different iteration, irrespective of other use cases.
- Negotiable
  - Details are negotiated during design
- Valuable to users or customers
- Estimatable
- Small
- Testable

Good Use Cases

Are	Aren't
• Text	• UML use case diagrams
• No GUI	• describing the GUI
• No data formats	• describing data formats
• Few lines	• multiple-page
• Easy to read	• complicated to read
• At user's goal level	• at program-feature level
• Record of decisions made	• tutorial on the domain

Use cases can be written just-in-time & in increments

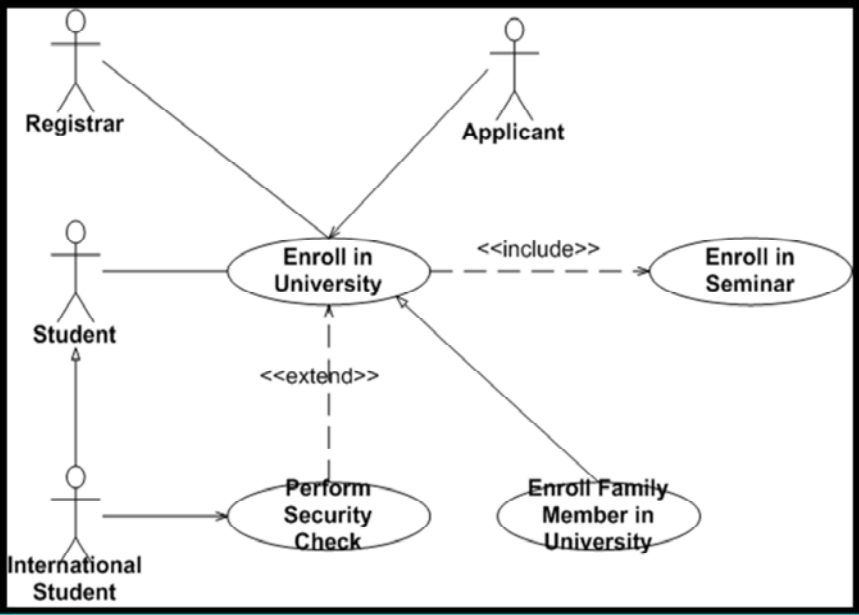
Notes

- Use cases are not read by a compiler but by a human...--> so,
  - Don't be rule-bound, but adapt the form to your needs.
- The hard part of use cases in not typing or drawing, but thinking and agreeing
- Don't try to teach a tutorial on the subject domain within the use cases!
  - To teach a domain, you need a textbook, not use cases.
- Get feedback from users/sponsors as use cases are being written

Smells

- Small use cases
- Interdependent use cases
- Gold plating
- Too much details
- Including a UI
- Thinking too far ahead
- No Use case descriptions
- Customer will not write/prioritize the Use cases

Features

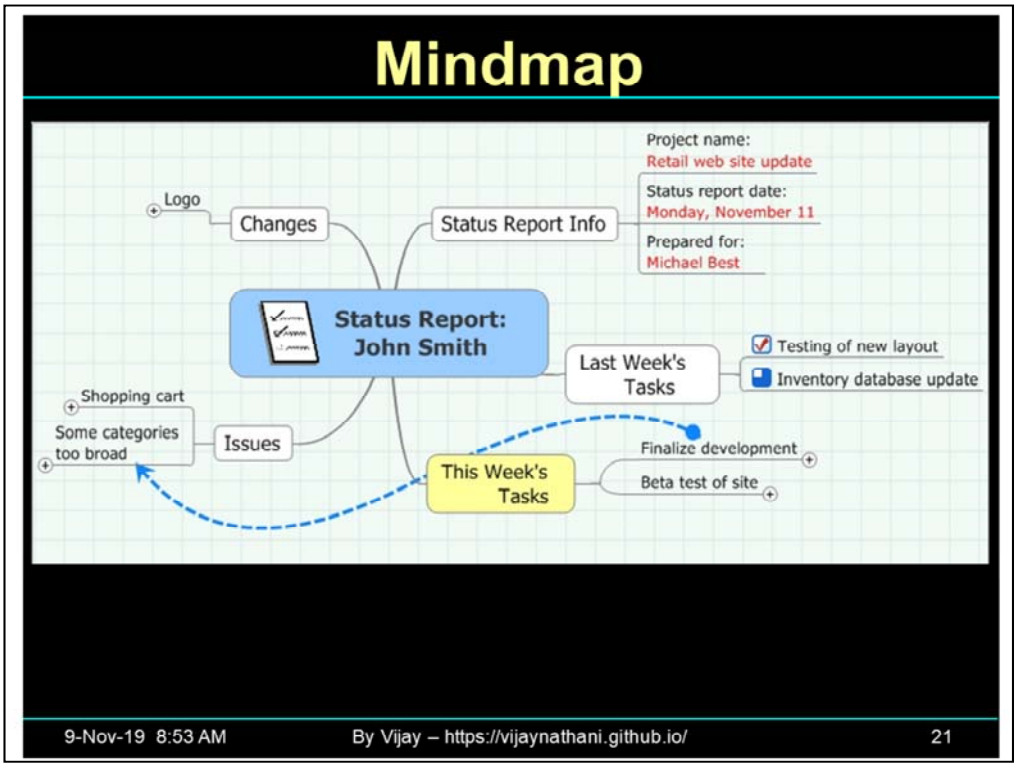

Registrar
Applicant
Student
Enroll in University
<<include>>
Enroll in Seminar
<<extend>>
International Student
Perform Security Check
Enroll Family Member in University

Use Case Narrative

Element	Description
Actor	Refers to a human or another system interacting with the system being modeled. Primary or active actors are those that the system uses to achieve its goals. Secondary or passive actors are those that provide some service to the system in order to help the system realize the goals of the primary actors.
Trigger	Refers to an event that causes the use case to start.
Preconditions	Refer to assumptions that can be made for a use case before it begins.
Basic Scenarios	Refers to the sequence of steps describing the normal case scenario.

Use Case Narratives

Element	Description
Alternate Scenarios	Refer to alternate sequence of steps to achieve the sub-goal or goal.
Exception Scenarios	Refer to the sequence of steps executed when something unexpected to exceptional happens in any basic or alternate scenario.
Post condition	Refer to the state of the system when the use case has been executed.
Includes	Refers to the names of use cases that are called or included in the current use case.



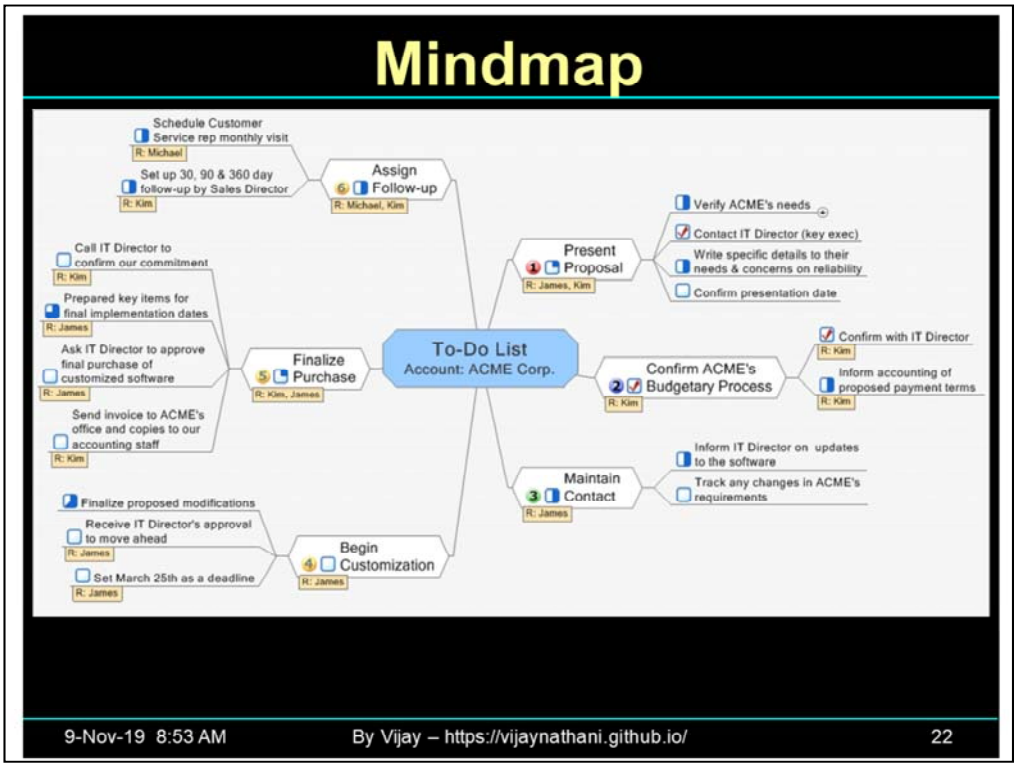
A **mind map** is a diagram used to represent words, ideas, tasks, or other items linked to and arranged radially around a central key word or idea. It is used to generate, visualize, structure, and classify ideas, and as an aid in study, organization, problem solving, decision making, and writing.

A mind map is also a map that has one big word or text in the middle, and you can place things around the text in the middle.

Rough mindmap notes taken during a course session

Mind maps have many applications in personal, family, educational, and business situations, including notetaking, brainstorming (wherein ideas are inserted into the map radially around the center node, without the implicit prioritization that comes from hierarchy or sequential arrangements, and wherein grouping and organizing is reserved for later stages), summarizing, revising, and general clarifying of thoughts. One could listen to a lecture, for example, and take down notes using mind maps for the most important points or keywords. One can also use mind maps as a mnemonic technique or to sort out a complicated idea. Mind maps are also promoted as a way to collaborate in color pen creativity sessions.

Software and technique research have concluded that managers and students find the techniques of mind mapping to be useful, being better able to retain information and ideas than by using traditional 'linear' note taking methods.[



Tony Buzan suggests using the following foundation structures for Mind Mapping:

Start in the center with an image of the topic, using at least 3 colors.

Use images, symbols, codes, and dimensions throughout your Mind Map.

Select key words and print using upper or lower case letters.

Each word/image must be alone and sitting on its own line.

The lines must be connected, starting from the central image. The central lines are thicker, organic and flowing, becoming thinner as they radiate out from the center.

Make the lines the same length as the word/image.

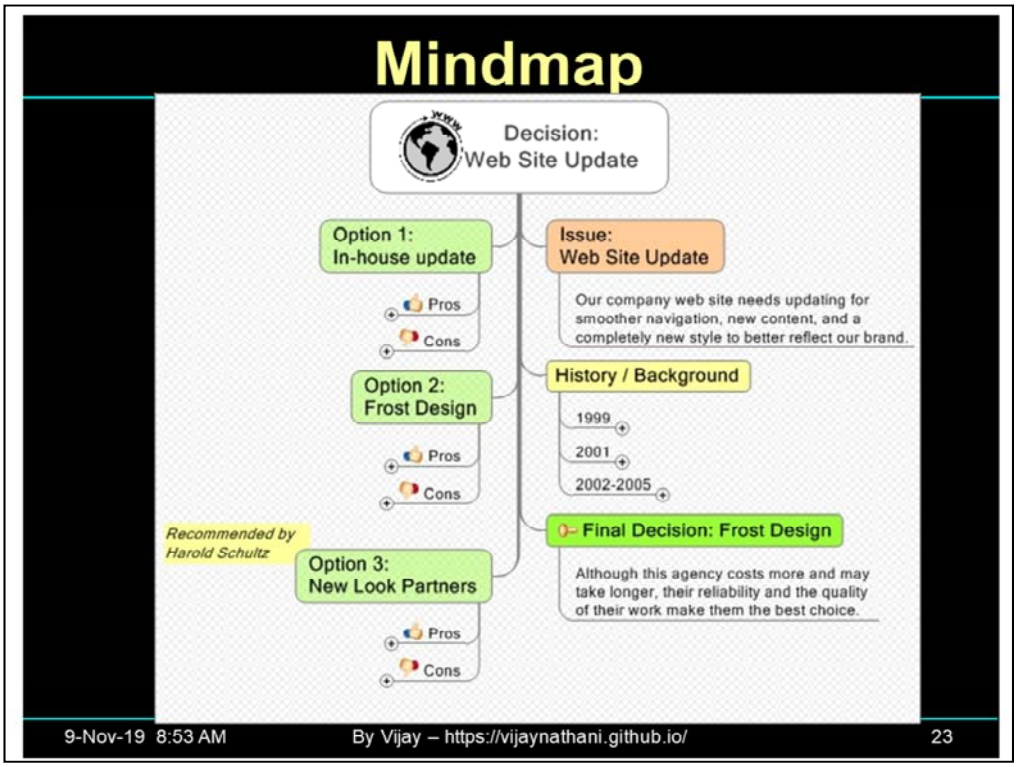
Use colors – your own code – throughout the Mind Map.

Develop your own personal style of Mind Mapping.

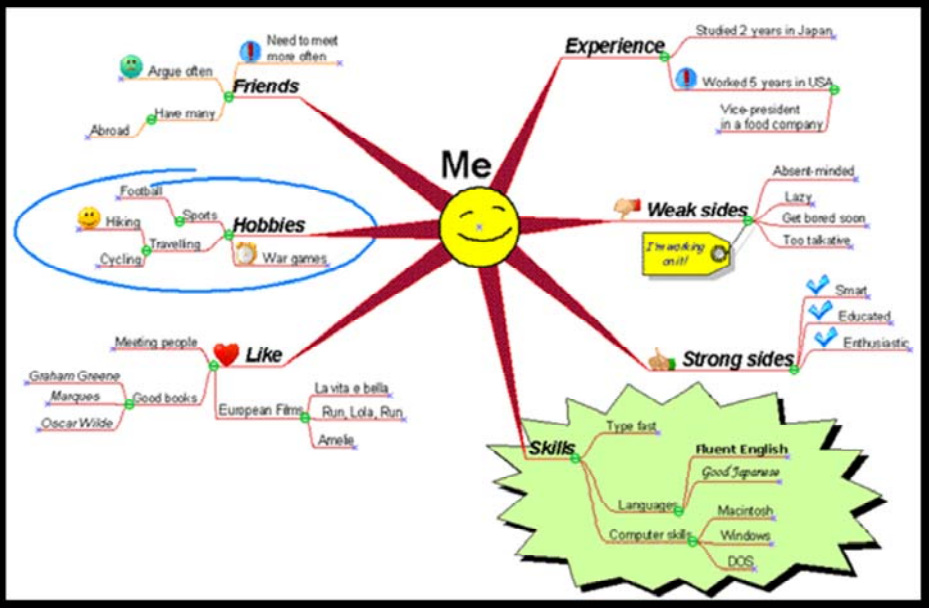
Use emphasis and show associations in your Mind Map.

Keep the Mind Map clear by using radial hierarchy, numerical order or outlines to embrace your branches.[1]

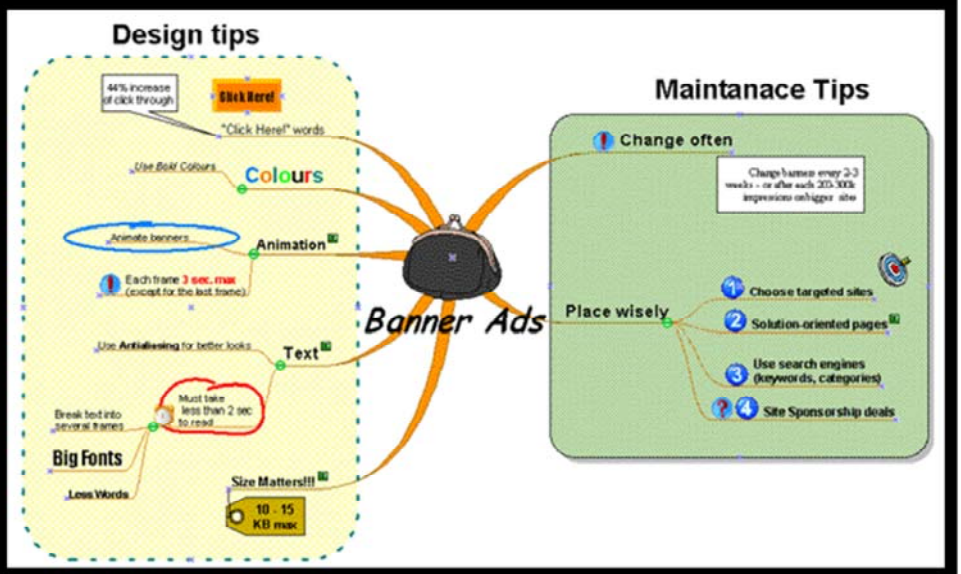
An idea map is similar to a mind map but does not adhere to the above guidelines. Rules are constantly broken based on the purpose and application of the Map.


Mindmap
Decision:
Web Site Update
Option 1:
In-house update
Pros
Cons
Option 2:
Frost Design
Pros
Cons
Recommended by
Harold Schultz
Option 3:
New Look Partners
Pros
Cons
Issue:
Web Site Update
Our company web site needs updating for smoother navigation, new content, and a completely new style to better reflect our brand.
History / Background
1999
2001
2002-2005
Final Decision: Frost Design
Although this agency costs more and may take longer, their reliability and the quality of their work make them the best choice.
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
23

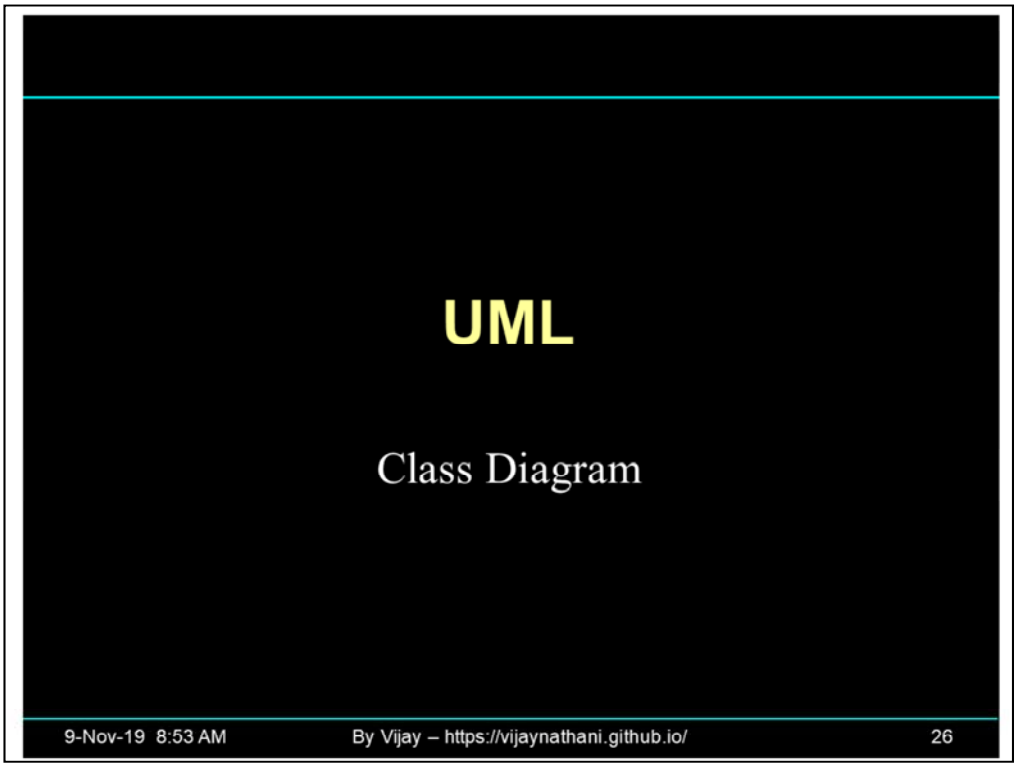
Mindmap


Me
Friends
Need to meet more often
Argue often
Have many
Abroad
Hobbies
Football
Sports
Hiking
Travelling
Cycling
War games
Like
Meeting people
Graham Greene
Marques
Good books
Oscar Wilde
European Films
La vita e bella
Run, Lola, Run
Amelie
Experience
Studied 2 years in Japan
Worked 5 years in USA
Vice-president in a food company
Weak sides
Absent-minded
Lazy
Get bored soon
Too talkative
I'm working on it!
Strong sides
Smart
Educated
Enthusiastic
Skills
Type fast
Fluent English
Good Japanese
Languages
Computer skills
Macintosh
Windows
DOS

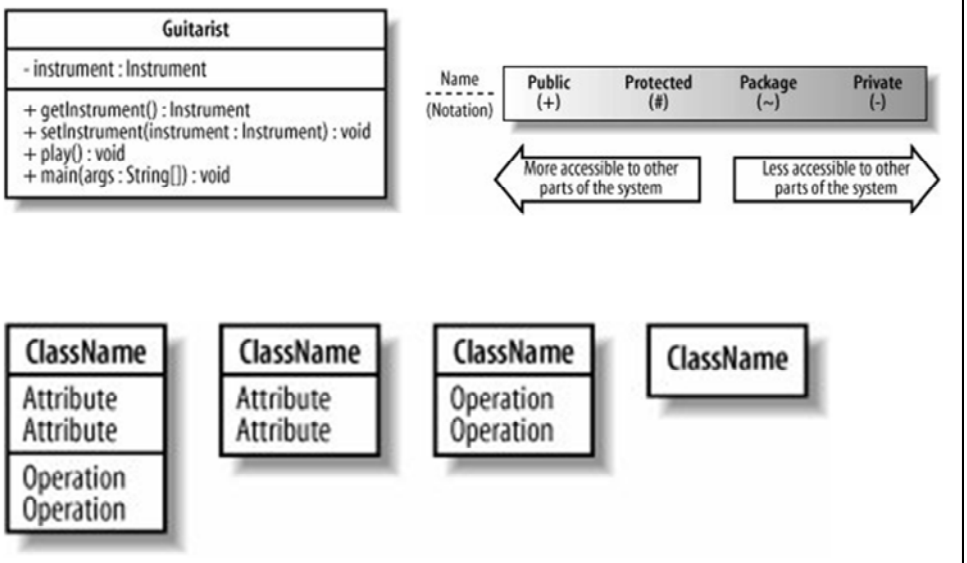
Mindmap



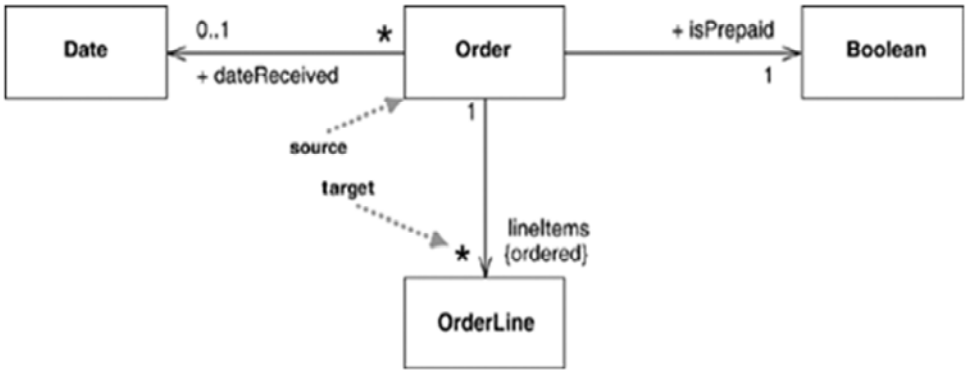
9-Nov-19 8:53 AM By Vijay – https://vijaynathani.github.io/ 25


UML
Class Diagram
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
26

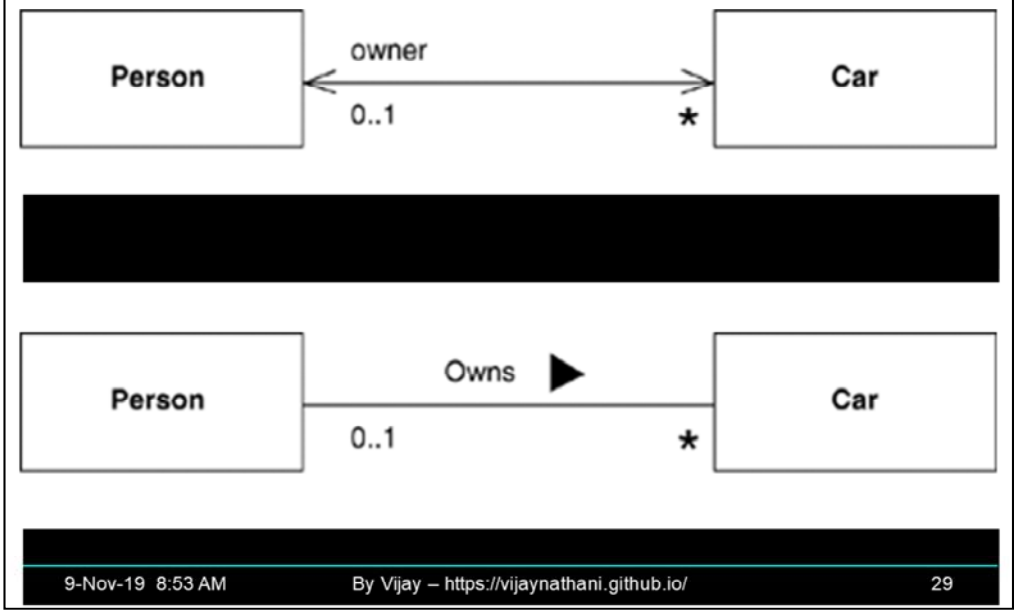
Class in UML


Guitarist
- instrument : Instrument
+ getInstrument() : Instrument
+ setInstrument(instrument : Instrument) : void
+ play() : void
+ main(args : String[]) : void
Name
(Notation)
Public
(+)
Protected
(#)
Package
(~)
Private
(-)
More accessible to other parts of the system
Less accessible to other parts of the system
ClassName
Attribute
Attribute
Operation
Operation
ClassName
Attribute
Attribute
ClassName
Operation
Operation
ClassName

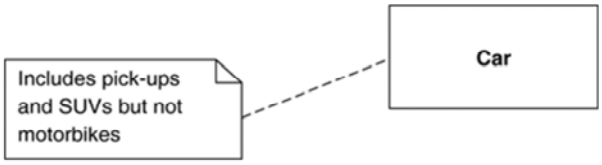
Class Diagram

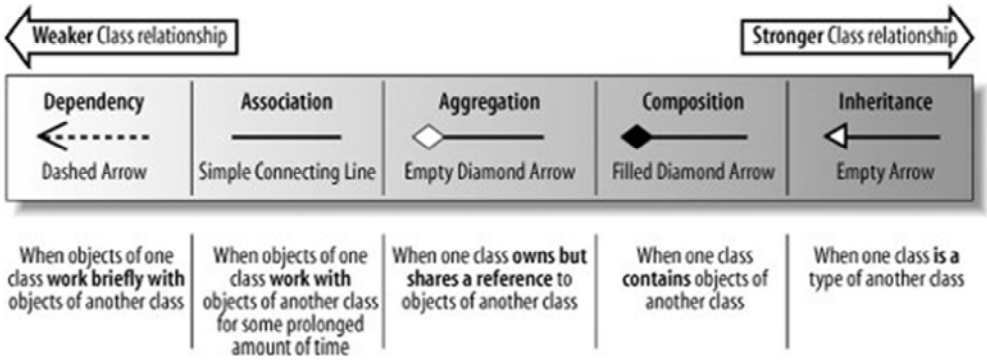


Class Diagram

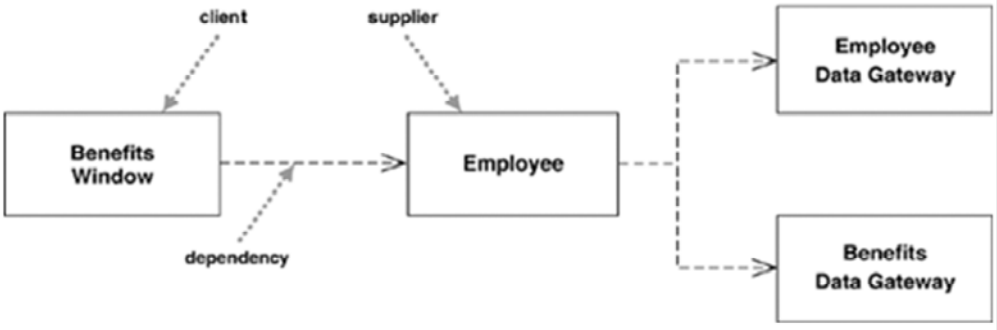


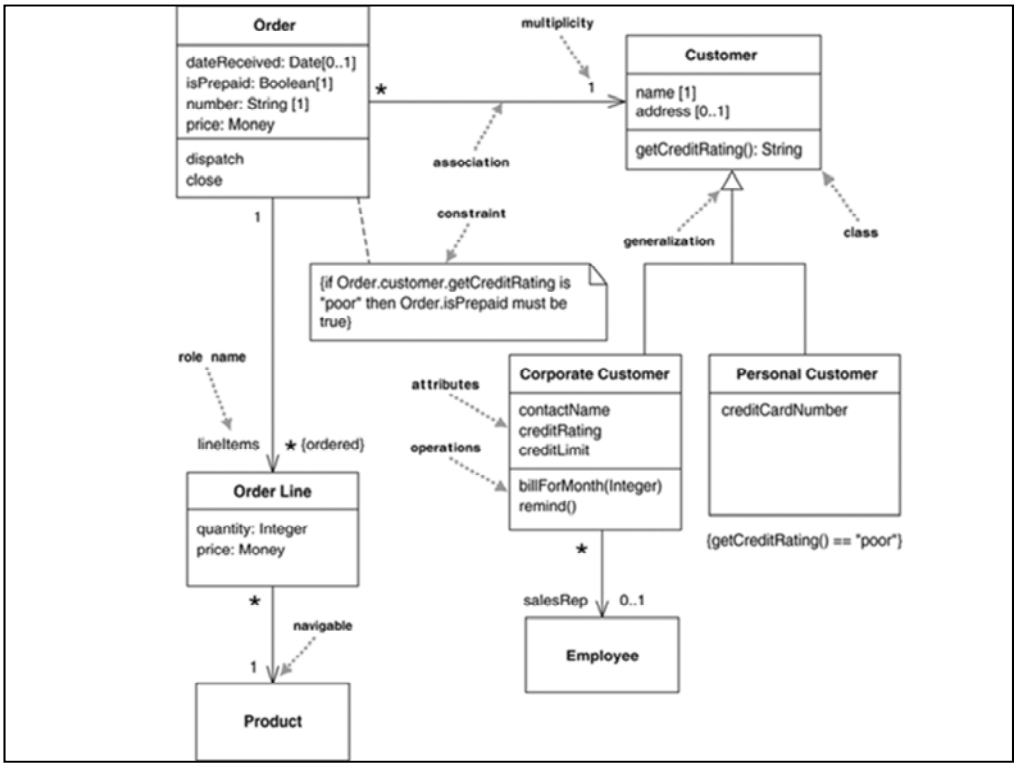
Comments and Relationships

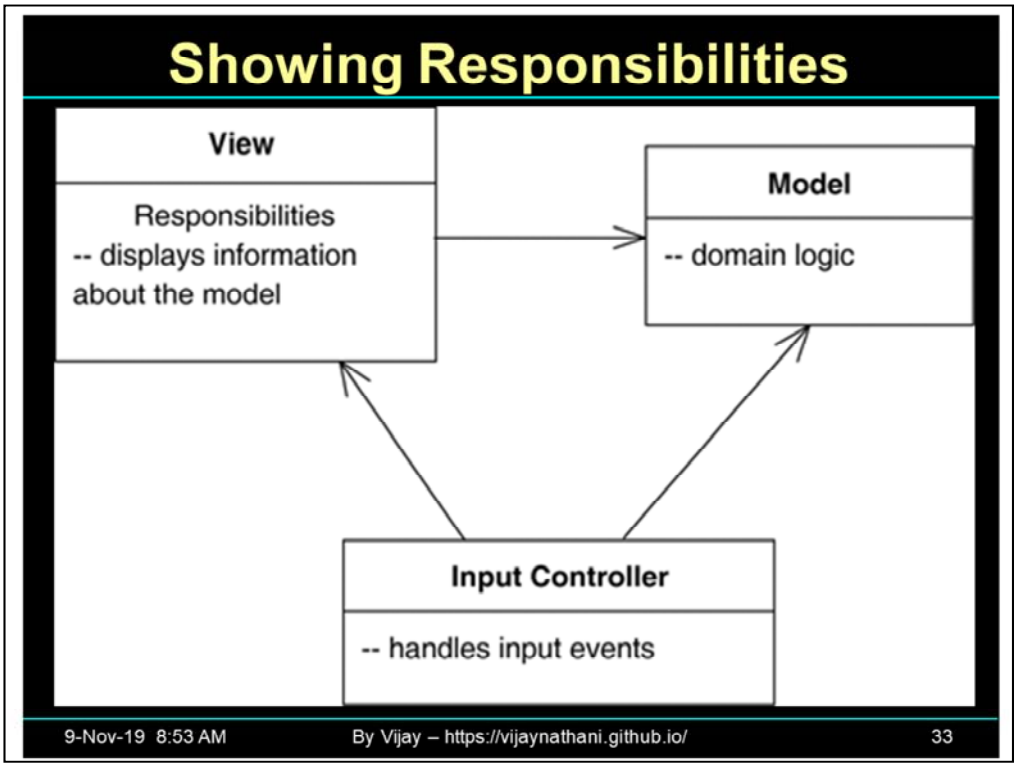




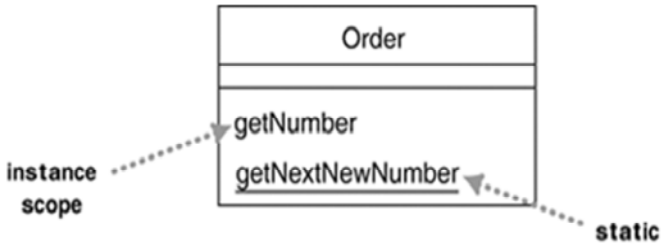
Dependency


client
supplier
Benefits Window
Employee
dependency
Employee Data Gateway
Benefits Data Gateway

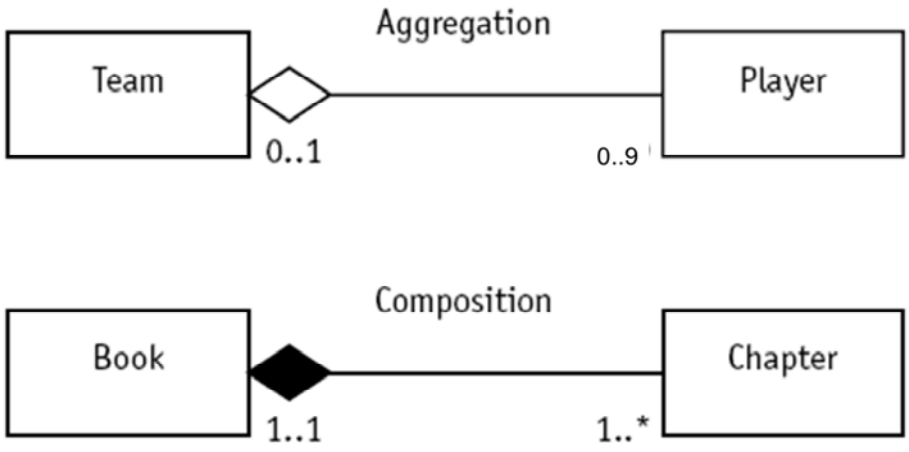

Order
dateReceived: Date[0..1]
isPrepaid: Boolean[1]
number: String [1]
price: Money
dispatch
close
multiplicity
*
1
Customer
name [1]
address [0..1]
getCreditRating(): String
association
constraint
generalization
class
1
{if Order.customer.getCreditRating is "poor" then Order.isPrepaid must be true}
role name
lineItems
* {ordered}
attributes
operations
Corporate Customer
contactName
creditRating
creditLimit
billForMonth(Integer)
remind()
Personal Customer
creditCardNumber
{getCreditRating() == "poor"}
Order Line
quantity: Integer
price: Money
*
salesRep
0..1
navigable
1
Employee
Product


Showing Responsibilities
View
Responsibilities
-- displays information about the model
Model
-- domain logic
Input Controller
-- handles input events
9-Nov-19 8:53 AM
By Vijay – https://vijaynathani.github.io/
33

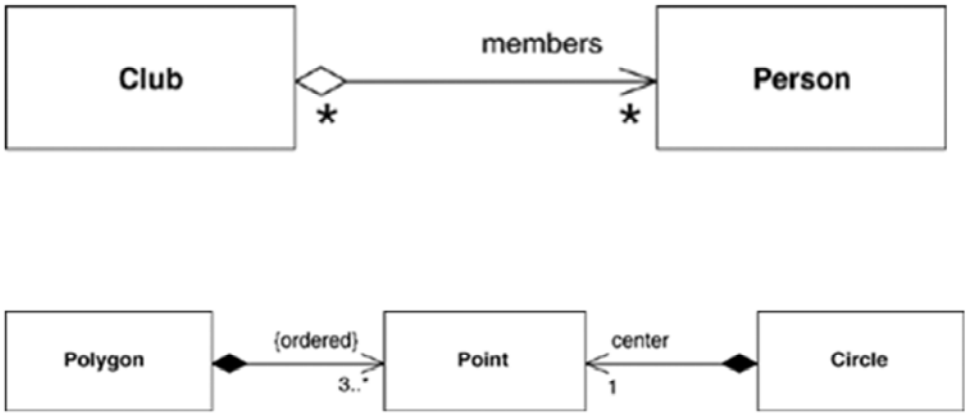
Notation



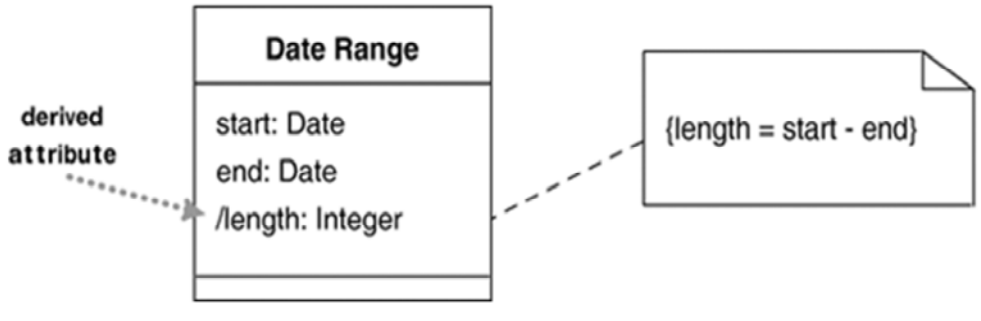
Aggregation & Composition



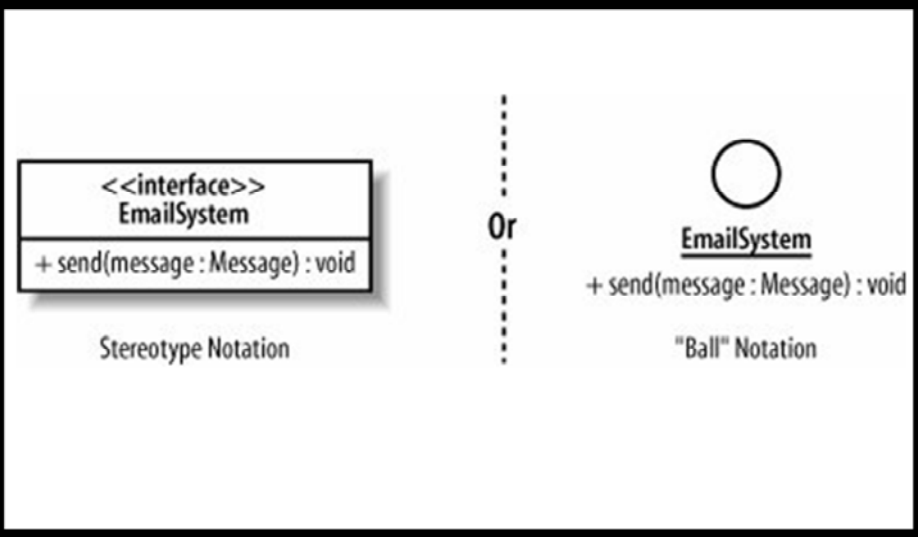
Aggregation & Composition



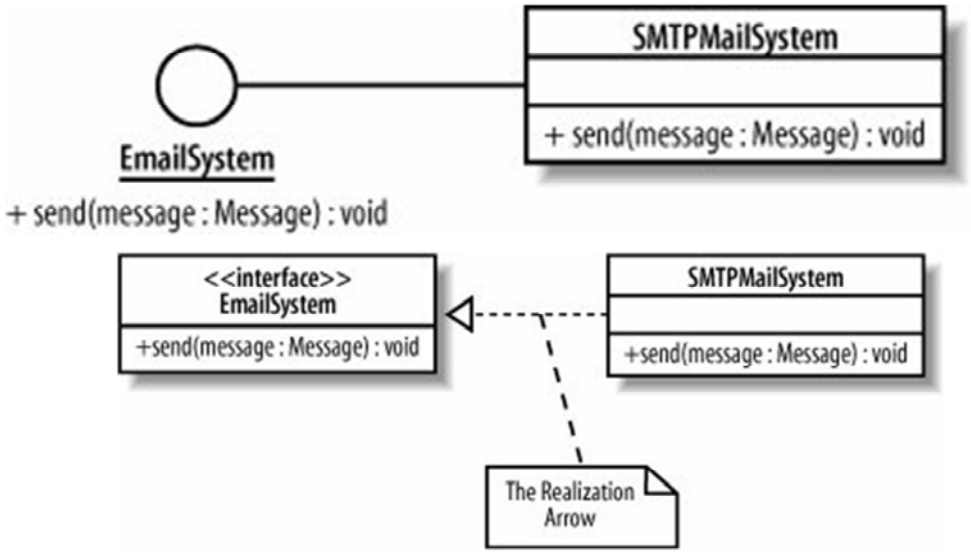
Derived


Date Range
start: Date
end: Date
/length: Integer
derived
attribute
{length = start - end}

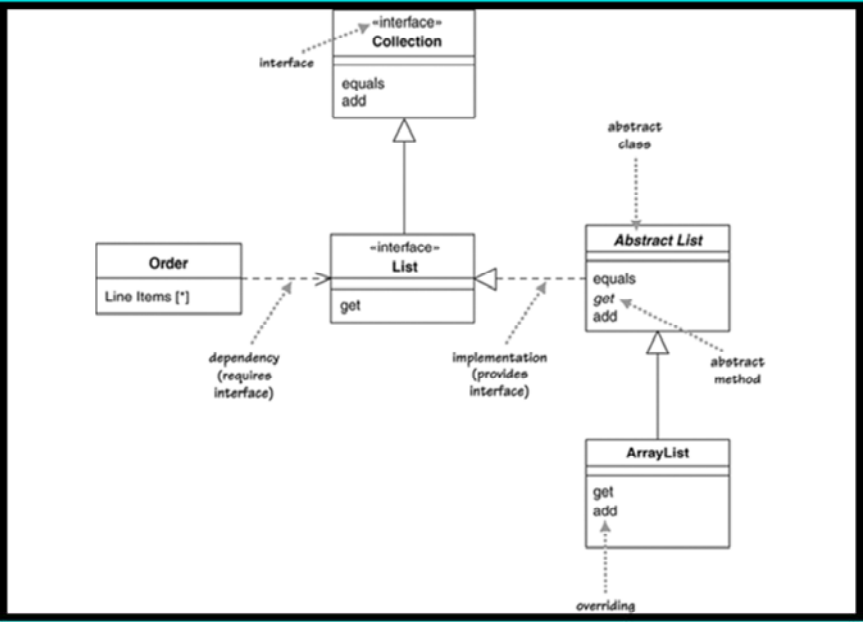
Interface notation



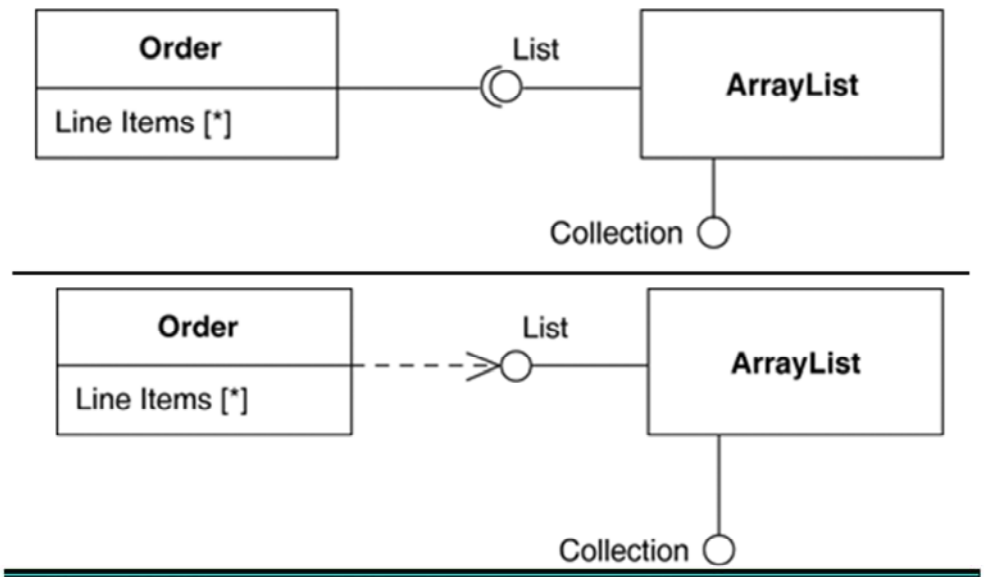
Class implementing Interface



Interface & Abstract classes

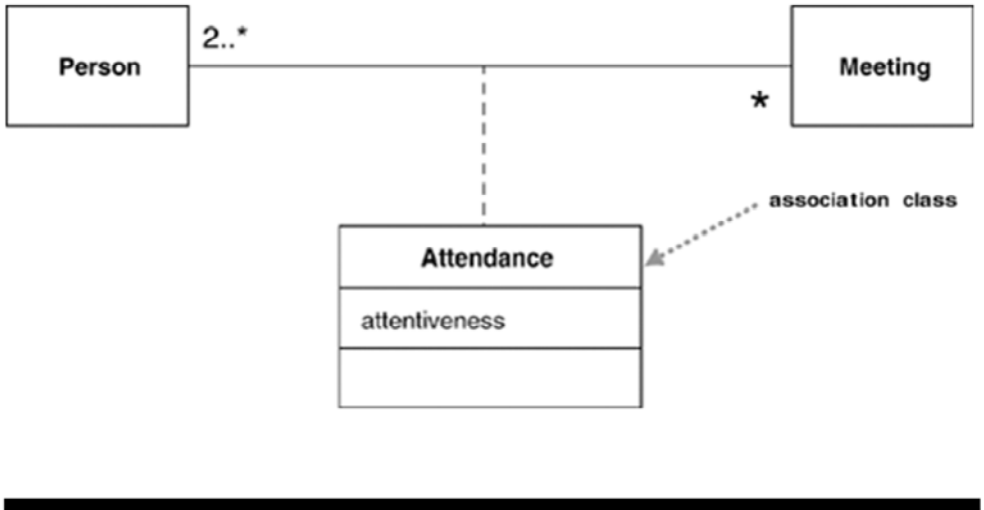


Interface


Order
Line Items [*]
List
ArrayList
Collection
Order
Line Items [*]
List
ArrayList
Collection

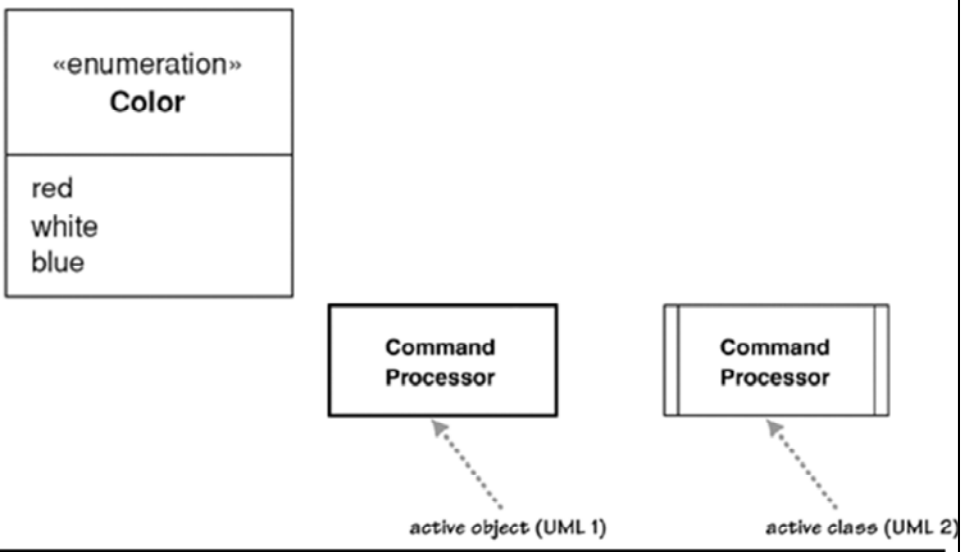
Qualified Association

Association class

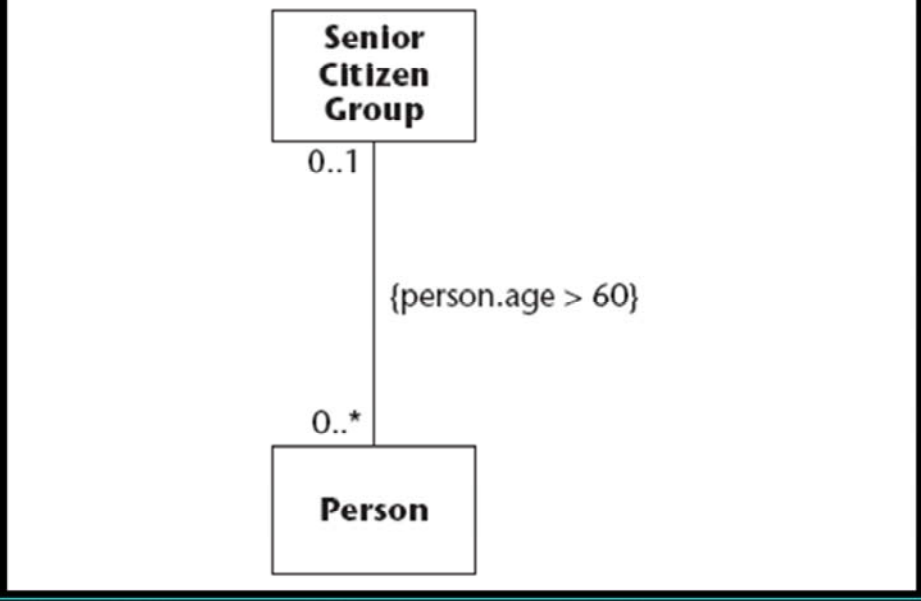


Temporal Relationship

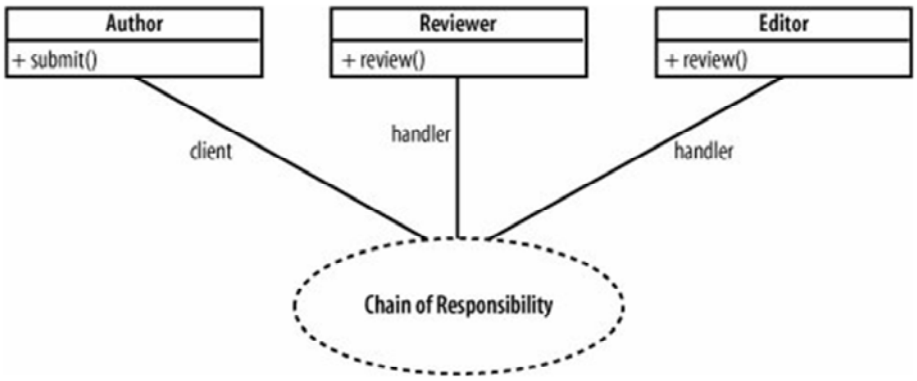
Enumerations & Active Objects



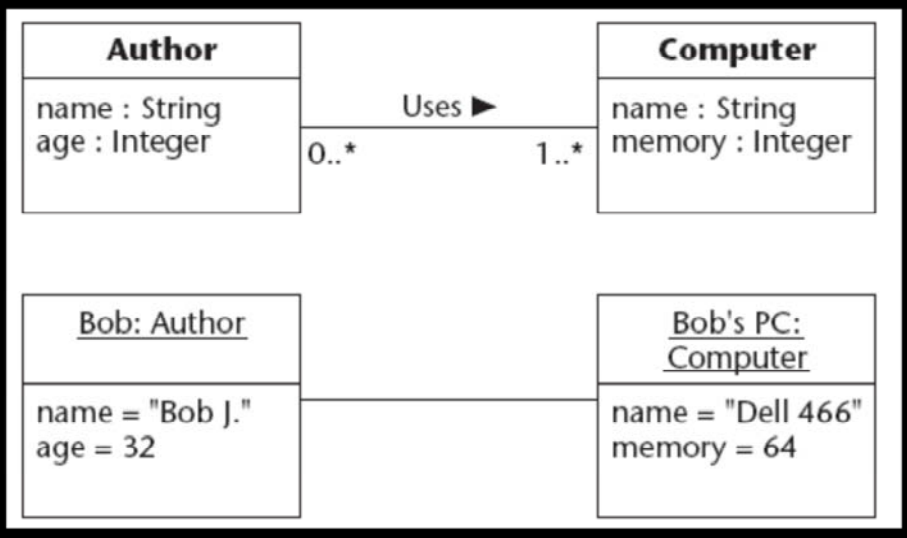
Constraints



Design Patterns



Object Diagram



Object Diagram