BHARATH INSTITUTE OF SCIENCE AND TECHNOLOGY

(Department of Computer Science and Engineering)

CONTINUOUS LEARNING ASSESSMENT-2

U18PCCS602 – Objected Oriented Software Engineering

Date : 18.04.2022 Duration: 1.5 HOURS

Academic Year / Semester : 2021-2022/VI EVEN Marks : 50

Instructions : Descriptive Type Questions

Q.No	Question	Weightage	CO	Bloom'sLevel
	PART-A [Answer all Questions – 6 X 2 = 12 marks]			
1	How to Name an Association in UML?	2	CO2	R
2	What are the various implementation diagram in UML?	2	CO2	R
3	List down the guidelines for adding a description class	2	CO3	R
4	Distinguish between cohesion and coupling.	2	CO3	A
5	Illustrate any three purpose of a view layer interface.	2	CO3	U
6	Classify the relationships used in class diagram.	2	CO3	U
	PART-B [Answer any three questions=6X3=18 marks]			
7a [OR]	Explain Object Constraint language for designing classes, methods and attributes.	6	CO2	U
7b	Design State chart Diagram and Activity Diagram using E-Ticketing Example.	6	CO2	C
8a [OR]	Explain about Designing for visibility?	6	CO3	U
8b	Design the Model and Creating Design Class Diagrams.	6	CO3	C
9a [OR]	Determine all attribute specified in the Domain model with example.	6	CO3	E
9b	Elaborate operation contracts in detail	6	CO3	C
	PART-C [Answer any Two questions=10X2=20 marks]			
10a [OR]	Elaborate the Sequence and Collaboration diagram for Library Management System in detail.	10	CO2	C
b	Categorize are the various implementation diagram in UML and explain in detail with examples?	10	CO2	AN
11a [OR]	Write briefly about elaboration and distinguish between Elaboration and Inception with examples.	10	CO3	AN
b	Explain the logical architecture refinement with diagram.	10	CO3	U

CO	Weightage
CO1	00
CO2	20
CO3	30
CO4	00
CO5	00
CO6	00
Total	50

Prepared by	**Staff Name** Mrs.A.Martin, AP/CSE	Signature AMartin
Verified by	**HoD** Dr.B.Persis Urbana Ivy	Signature

Naming Association in UML →

⊗ Format : class name - Verb Phrase - class Name.

Basically → Has Association
Uses Association.

1

eg: Sale Paid-by CashPayment - good.
Sale Uses CashPayment

Association name → should start with capital letters.

eg: Paid-by / PaidBy.

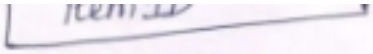
IMPLEMENTATION DIAGRAMS 2

→ Shows the implementation phase of systems development,

→ shows source code structure and run time implementation structure.

→ Two types of implementation diagrams

* Component diagram → shows structure of code
* Deployment diagram → shows the structure of runtime system.



GUIDELINES FOR ADDING A DESCRIPTION CLASS

Add a description class, when: 3

* There needs to be a description about an item/service.
* Deleting instances of things results in loss of information
* It reduces redundant or duplicated information.

COUPLING is a measure of the strength of association established by a connection from one object or software component to another.

→ Coupling is a binary relationship: A coupled with B.

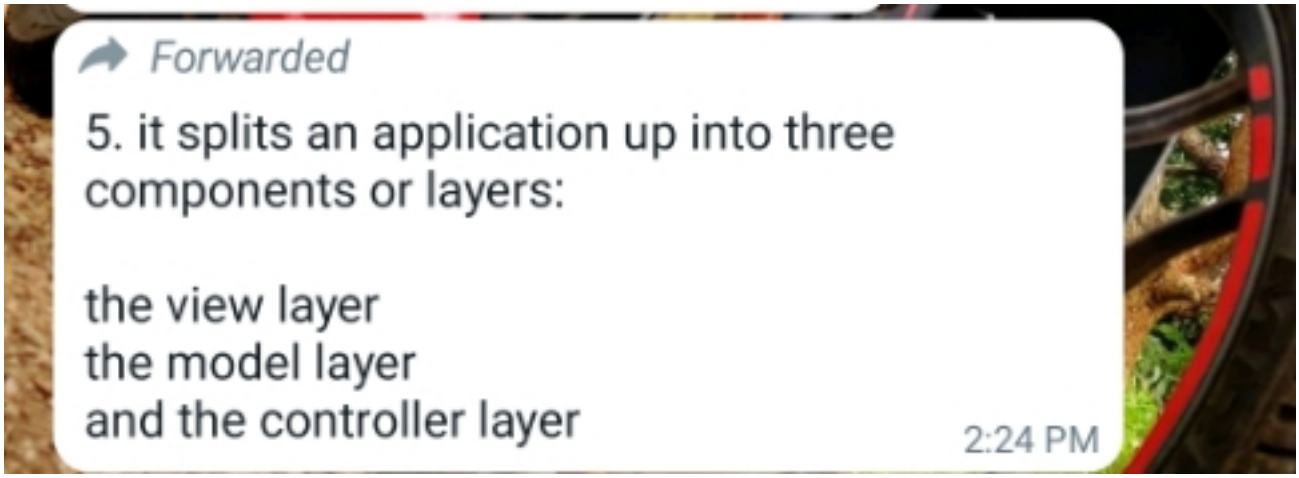
→ A change in one component of a system should have a minimal impact on other components.

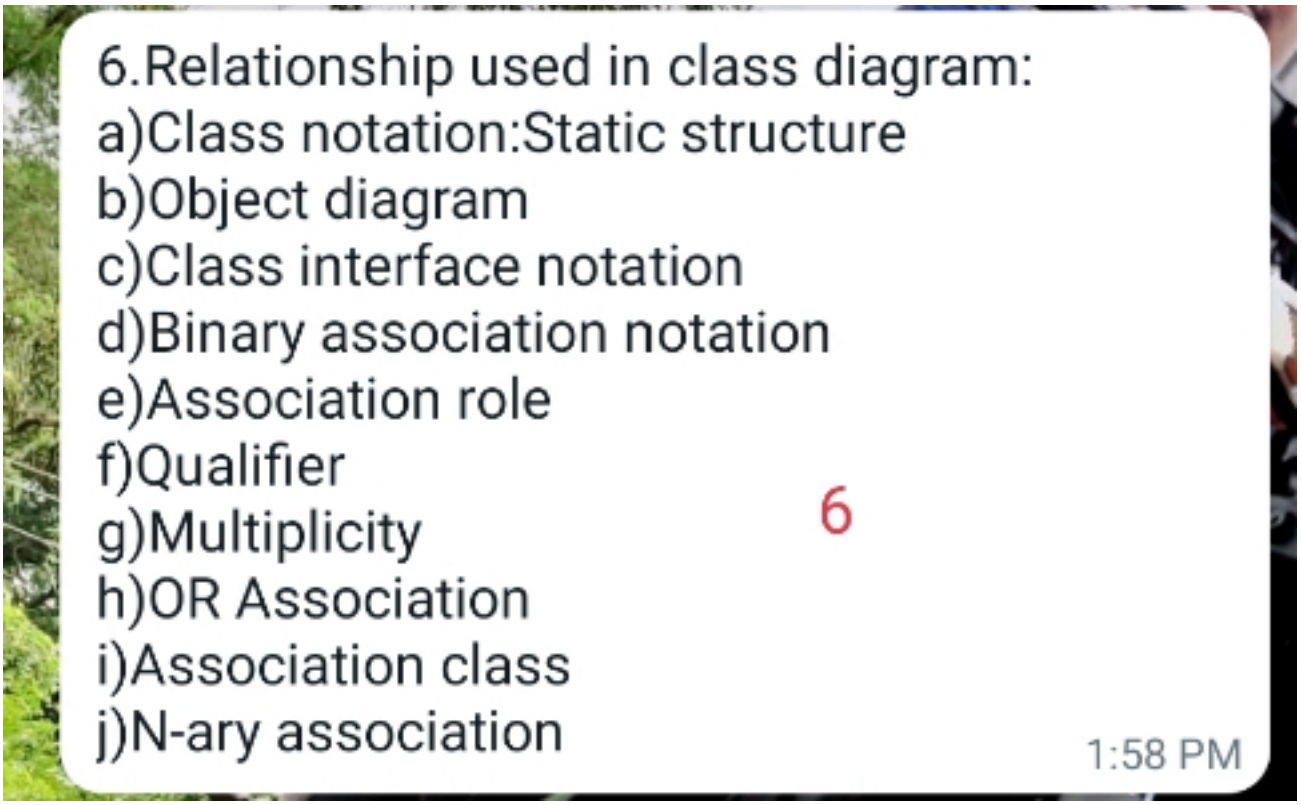
(14)

COHESION → reflects "single-purposeness" of an object.

→ Coupling deals with interactions between objects or software components but cohesion consider interactions within a single object or software component.

→ Highly cohesive components can lower coupling because only a minimum of essential information


Forwarded
5. it splits an application up into three components or layers:
the view layer
the model layer
and the controller layer
2:24 PM


6.Relationship used in class diagram:
a)Class notation:Static structure
b)Object diagram
c)Class interface notation
d)Binary association notation
e)Association role
f)Qualifier
g)Multiplicity
h)OR Association
i)Association class
j)N-ary association
6
1:58 PM

→ OCL (Object Constraint Language) is a specification language that uses simple logic for specifying the properties of a system.

→ Constraints are shown as text in braces, { }.

→ A constraint may be a "comment", in which case it is written in text. Constraint may be written in natural language.

→ UML modeling constructs require expressions for types boolean values and numbers. Expressions are stated as strings in OCL.

→ Common navigational expressions:

1) Item.Selector → Selector is the name of the attribute in the item. The result is the value of the attribute.

7.a.1

eg: John.age (age is an attribute of the object John and John.age represents the value of the attribute).

2) Item.Selector[qualifier-value] → Selector indicated a qualified association that qualifies the item. The result is the related object selected by the qualifier.

eg: Array indexing as a form of qualification. John.Phone[2], assuming John has several Phones.

3) set → select (boolean-expression)

The boolean expression is written in terms of objects within the set. The result is the subset of objects in the set for which the boolean expression is true.

eg: company.employee → select(salary > 30000)

This represents employees with salaries over $30,000.

7.a.2

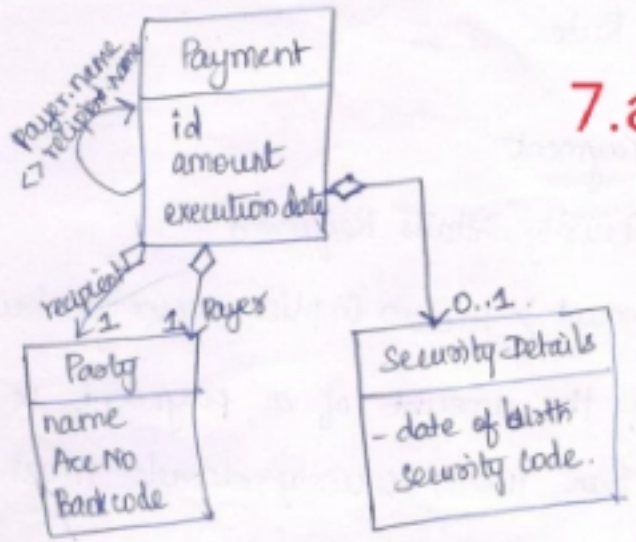
OCL Rule Basics:

→ Define four things for a rule:

1. context → The classifier / class, which the rule is associated.
2. Name → The name of the rule.
3. OCL → The rule expression.
4. Message/Error Message → A textual description or saying error msg.

* A class model can define the structure of data.
* But OCL is needed to define interdependencies between the data.

7.a.3



→ The payer & the recipient cannot be the same.

→ Payer.name <> recipient.name.

DESIGNING FOR VISIBILITY

8.a.1

Visibility → is the ability of an object to "see" or have a reference to another object.

* There are four common ways - visibility can be achieved from object A to object B.

1. Attribute Visibility → B is an attribute of A.
2. Parameter Visibility → B is a parameter of a method of A.
3. Local Visibility → B is a (non-parameter) local object in a method of A.
4. Global Visibility → B is in some way globally visible.

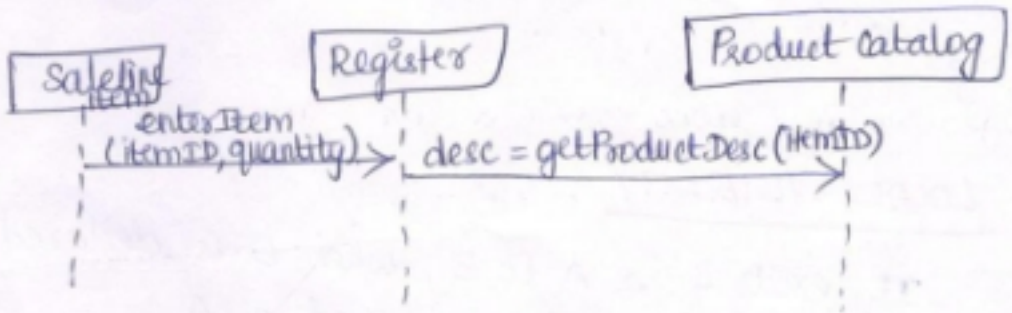
Motivation to consider Visibility

→ For an object A to send a message to an object B, B must be visible to A.

1. ATTRIBUTE VISIBILITY

8.a.2

Public class Register
{ ...
Private ProductCatalog catalog;
...
}.



class Register
{
...
Private ProductCatalog catalog;
...
}

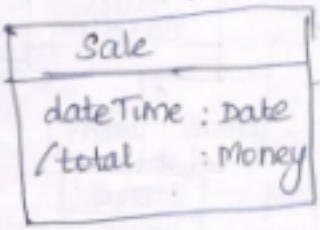
Public Void enterItem(ItemID, qty)
{
....
desc = catalog.getProduct
Desc(ItemID)
.....
}.

Attributes

→ An attribute is a logical data value of an object.

→ Include attributes that the requirements suggest or imply a need to remember information.

9.a.1



→ The total attribute in the sale can be calculated from the information in "saleslineitems" class → called as derived attribute, should start with "/" symbol.

→ SYNTAX: visibility name : type multiplicity = default {property-string}.

eg→ + Pi : Real

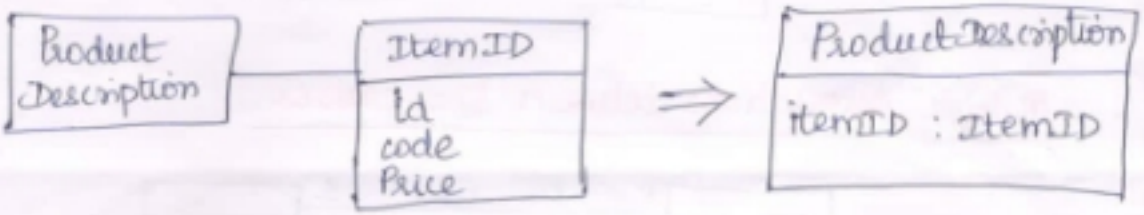
eg: - dateTime : Date

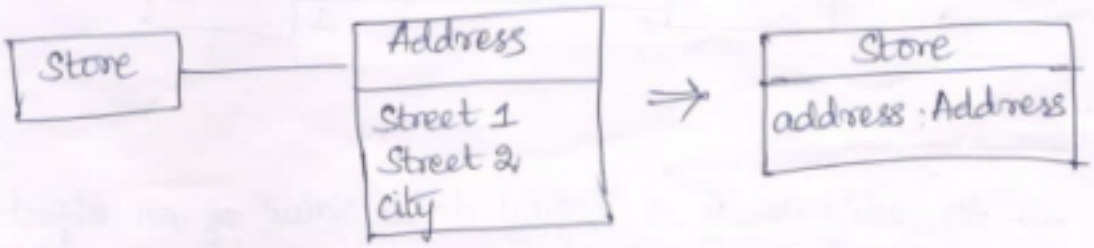
\+ Pi : Real = 3.14 {read only}

Common datatypes

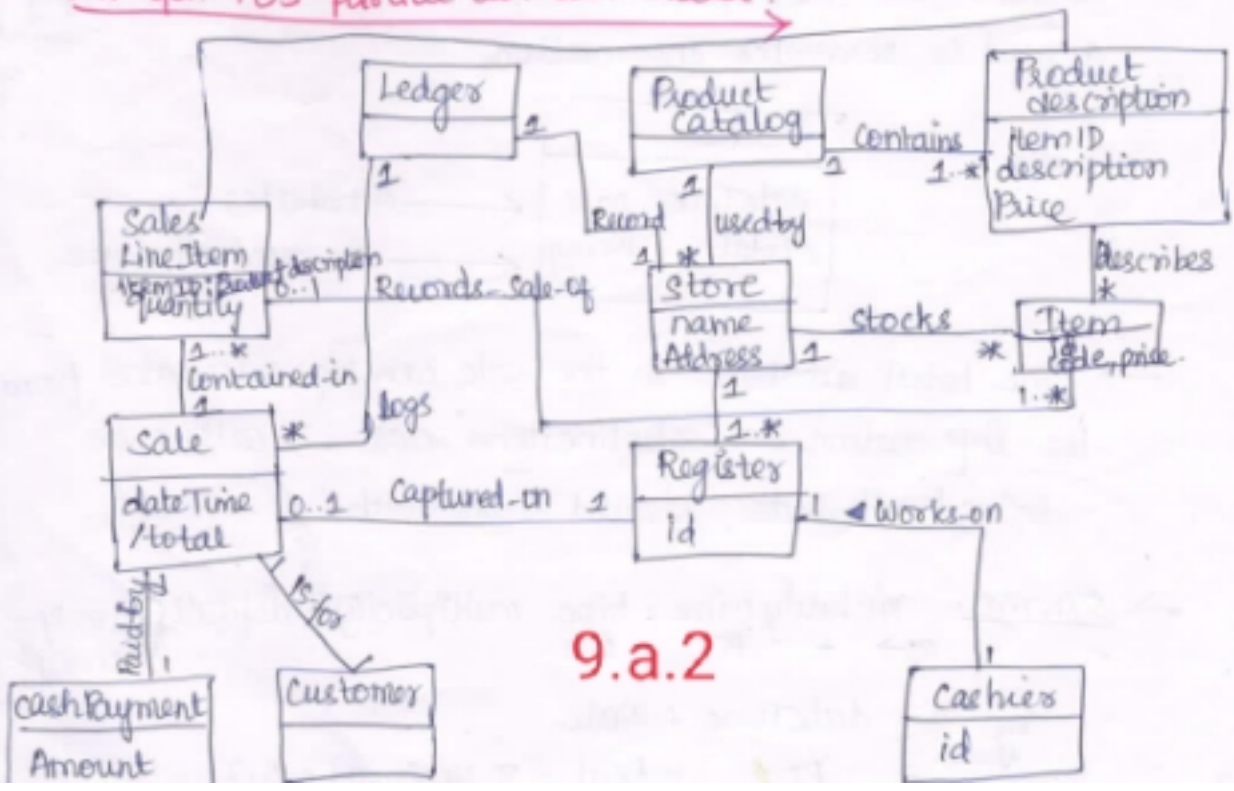
Boolean, Date, Time, DateTime, Number, character, String.

Other types → address, color, Ph.no, SSN, ZIP ..





NextGen POS partial domain Model



9.a.2

UML Sequence Diagram

10.a.1

→ gives behavior of a system

→ This diagram shows an interaction between the system and its environment in a time sequence.

→ objects / classes participating in the interaction by their life lines and the messages they exchange, arranged in time sequence.

→ A sequence diagram has two dimensions: the vertical dimension represents time, the horizontal dimension represents different objects.

→ The vertical lines is called the object's lifeline. The lifeline represents the objects existence during the interaction. This was popularized by Jacobson.

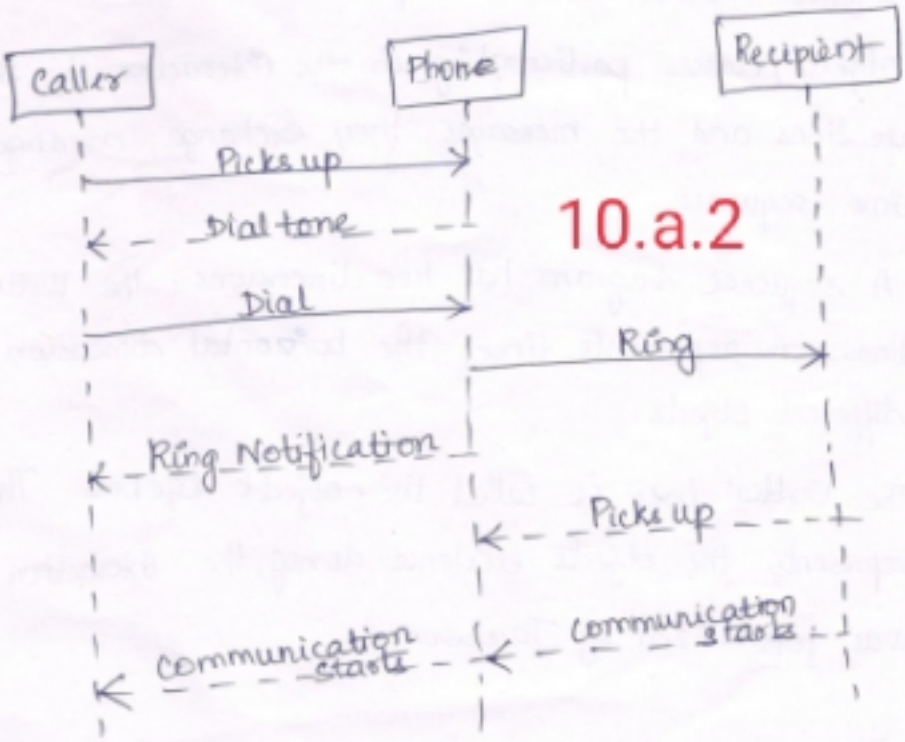
→ An object / class is shown as a box at the top of a dashed vertical line.

→ Each message is represented by an arrow between the lifelines of two objects.

→ The order in which these messages occur is shown top to bottom on the page. Each message is labeled with the message name.

→ The label also can include the argument, some control information, or a message that an object sends to itself, by sending the message arrow back to the same lifeline.

→ A sequence diagram is an alternative way to understand the overall flow of the control of a program.



UML Collaboration Diagram

→ Another type of ~~colla~~ Interaction diagram.

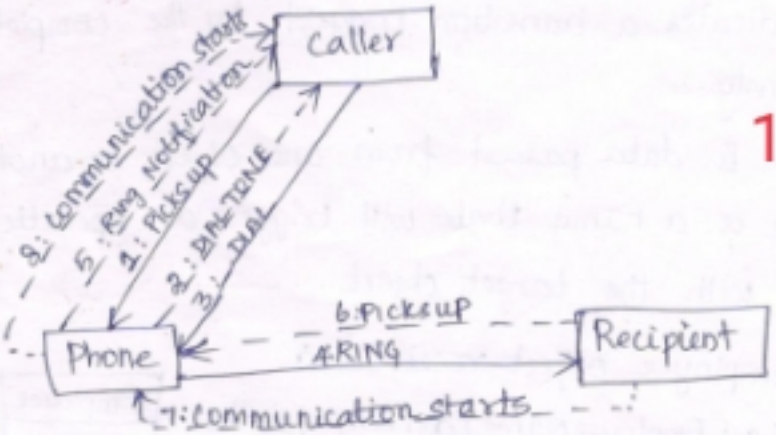
→ A collaboration diagram represents a collaboration, which is a set of objects related in particular context and interaction, which is a set of messages exchanged among the objects within the collaboration to achieve a desired outcome.

→ Sequence of collaboration is indicated by numbering the messages.

→ Numbering the messages make it more difficult to see the sequence than drawing the lines on the page.

→ A collaboration diagram provides several numbering schemes.

→ The simplest numbering is Integer values.



10.a.2

→ Decimal Numbering scheme can also be used.

LOGICAL ARCHITECTURE (Pg till 18)

→ logical architecture is the large-scale organization of the software classes into packages, subsystems and layers.

→ A layer is grouping of classes, packages or subsystems that has cohesive responsibility for a major aspect of the system. Layers are organized such that "higher" layers call upon services of "lower" layers.

→ The layers in an OO system include: 11.b.1

* User Interface
* Application Logic and Domain Objects → s/w objects representing domain concepts that fulfill application requirements.
* Technical Services → such as interfacing with a database or error logging. These services are usually application independent and reusable across several systems.

Guideline to design with layers →

→ organize the structure of the system into discrete layers of distinct and related responsibilities such that lower layers are low-level and higher layers are application specific.

→ Collaboration and Coupling is from higher to lower layers

→ Lower to higher layer coupling is avoided.

LOGICAL ARCHITECTURE – LAYERS

11.b.2

- GUI Windows
- Reports
- Speech Interface
- HTML, XML, XSLT, JSP, Javascript...

UI
(AKA Presentation, View)

More app specific

- Handles presentation layer request
- Workflow
- Session state
- Wind/Page transitions
- Consolidated/transformation of disparate data for presentation

Application
(AKA Workflow, Process, Mediation, App Controller)

dependency

- Handles application layer request
- Implementation of domain rules
- Domain services (POS, Inventory)
- Services may be used by just one application, but there is also the possibility of multi-appln. services.

Domain
(AKA Business, Application Logic, Model)

→ software classes for domains POS, Tax, inventory

- Very general low-level business services used in many domains
- Currency Converter

Business Infrastructure
(AKA low-level Business Services)

- High-level technical services and frameworks
- Persistence, security.

Technical Services
(AKA Technical Infrastructure, high level technical services)

- Low-level technical services, utilities and frameworks
- Data structures, threads, math, file, DB & n/w I/O.

Foundation
(AKA core services, Base Services, Low-level Technical Services/ Infrastructure)

Width implies range of applicability.

Benefits of using layers

* A separation of high from low level services and of application specific from general services. This reduces coupling and dependencies, improves cohesion, increase reuse potential and increases clarity.
* Related complexity is encapsulated and decomposable.
* Some layers can be replaced with new implementations. eg. UI, Application & Domain layers.
* Lower layers contain reusable functions.
* Development by teams is aided because of the logical segmentation.

11.b.3

Definition: tier, layers and Partitions →

→ A tier is a row or layer in a series of similarly arranged objects.

→ The layers of an architecture are said to represent the vertical slices while partitions represent a horizontal division of layer.