ASSIGNMENT-I

Object Oriented Analysis & Design

III year - II Semester

Submitted by: P.V. Aishwarya

Name : P.V. Aishwarya

Reg No : 18B01A1231

Branch : IT

Section : A

Sign of faculty

Marks Awarded

SHRI VISHNU ENGINEERING FOR WOMEN

1. List the principles of modelling

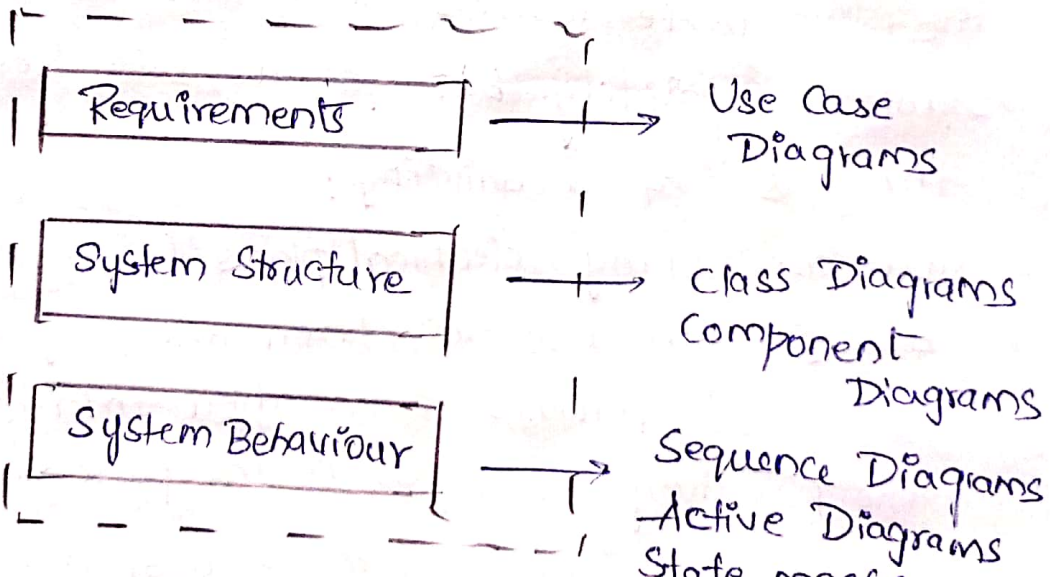
Ans Principles of UML modelling

1. The choice of model is important

The choice of what models to create has a profund influence on how a problem is attacked and how a solution is shaped. We need to choose your models well.

* The right models will mislead you the most critical development problems.
* Wrong models will mislead you, causing you to focous on irrevalent issues

For example we can use different types of diagrams for different phases in software development



2. Every model may be expersed of different levels of pecision

For example,

* If you are building a high rise, sometimes you need a 30,000 - foot view - for instance, to help your investors visualize its look & feel
* Other times, you need to down to the level of the studs for instance, when there's a tricky pipe run (or) an unusual structural element.

3. The best models are connected to reality.

All models simplify reality & good model reflects important key characteristics.

4 No Single model is Sufficient.

Every non-trivial System is best approached through a small set of nearly independent models. Create models that can be built and studied separately. but you are still interrelated. In the case of a building:

* you can study electrical plans in isolation
* But you can also see their mapping to the floor plan and perhaps even their interaction with the routing of pipes in the plumbing plan.
* Every non trivial system is best approached through a small set of nearly independent models.

2) Discuss Software Development life cycle?

Software Development life cycle

1. UML is involved in each phase of the software development life cycle.
2. The UML Development process is

→ Use case driver

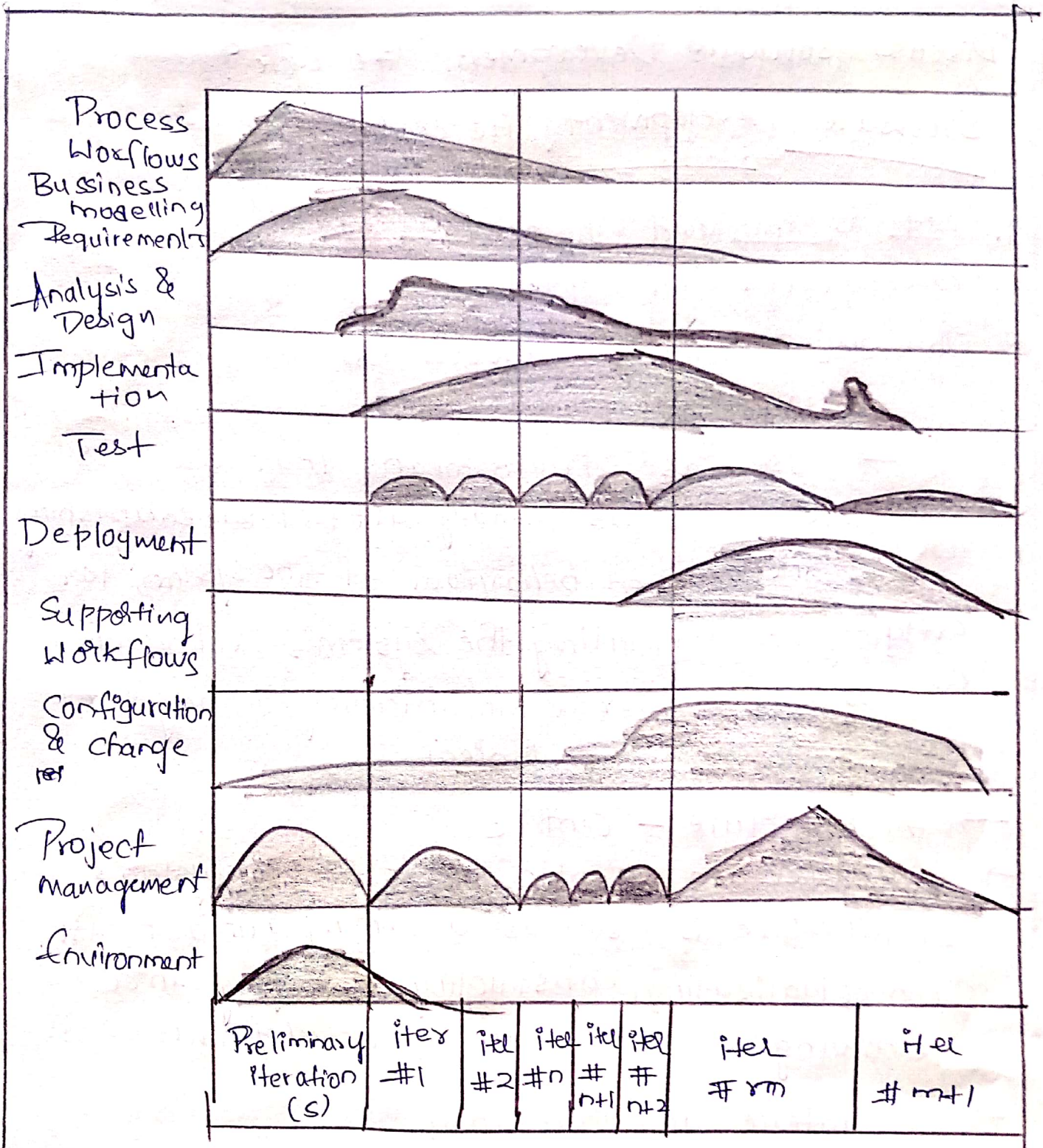
→ Use case driven means that use cases are used as primary artifact for establishing the desired behaviour of the system, for verifying and validating the system's architecture, for testing, and for communicating among the stakeholders of the project

→ Architecture - centric

→ Architecture - Centric means that a system's architecture is used as a primary artifact for conceptualizating, constructing, managing and evolving the system under development.

→ Iterative and incremental

→ An iterative process is one that involves managing a stream of executable releases. An is one that involves the continous integration of the system's architecture to produce these releases. With each new release embodying incremental improvements over the other



Inception: Inception is the first phase of the process when the seed idea for the development is brought up to the point of being atleast internally sufficiently well-founded to warrant entering into the elaboration phase.

- Sufficiently well-founded to warrant entering into the elaboration phase

Elaboration: is the second phase of the process, when the product vision and its architecture are defined. In prioritized, and baselined. A systems requirment against the buisness needs of the project may range from each specifying particular functional (or) Non-functional behaviour and each providing a basic for testing.

Construction:

Construction is the third phase of process, when the software is brought from an executable architectual community. Here also, the systems requirements and especially its evolution criteria are constantly reexamined are allocated as appropriate to actively attack risks to the project

Transition:

Transition is the forth phase of the process, when the software is turned into the hands of the user community. Rarely does the software development process end improved, bugs are eradicated and features that didn't make an earlier release are added.

9) Explain the diagrams in UML?

Ans

UML Diagrams:

1. A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and paths (relationships)
2. A diagram represents an elided view of the elements that make up a system
3. In theory, a diagram may contain any combination of things and relationships.
4. In practice, a small number of common combinations arise, which are consistent with the five most useful views that comprise the architecture of a software intensive system.

The UML includes nine kinds of diagrams:

1. Class diagram
2. Object diagram
3. Use Case diagram
4. Sequence diagram
5. Collaboration diagram
6. State Chart diagram
7. Activity diagram
8. Component diagram
9. Deployment diagram

1. Class diagram: shows a set of classes, interfaces and collaborations and their relationships. These diagrams are the most common diagram found in modeling object oriented systems. Class diagrams address the static design view of system. Class diagrams that include active classes address the static process view of a system.

2. Object diagram: shows a set of objects and their relationships object diagrams represent static snapshots of instances of the things found in class diagrams. These diagrams address the static design view or static process view of a system as do class diagrams.

3. Use case diagram: shows a set of use cases and actors and their relationships. Use case diagrams address the static use case view of a system.

4. Sequence diagram: is an interaction diagram that emphasizes the time-ordering of messages.

5. Collaboration diagram:

a communication diagram is an integration diagram that emphasizes the structural organization of the objects or roles that send and receive messages

6 Statechart diagram:

Shows a state machine, consisting of states, transitions, events and activities. A state diagram shows the dynamic view of an object

7. Activity diagram:

Shows the structure of a process or other computations as the flow of control and data from step to step within the computation. Actively diagram address the dynamic view of a system.

8. Component diagram:

Shows an encapsulation class and its interfaces, ports and internal structure consisting of nested components and connectors. Component diagrams address the static design implementation view of a system.

9. Deployment diagram:

Shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams address the static deployment view of an architecture

4 Explain Relations

Ans Relationships:

1. A relationship is a connection among things.

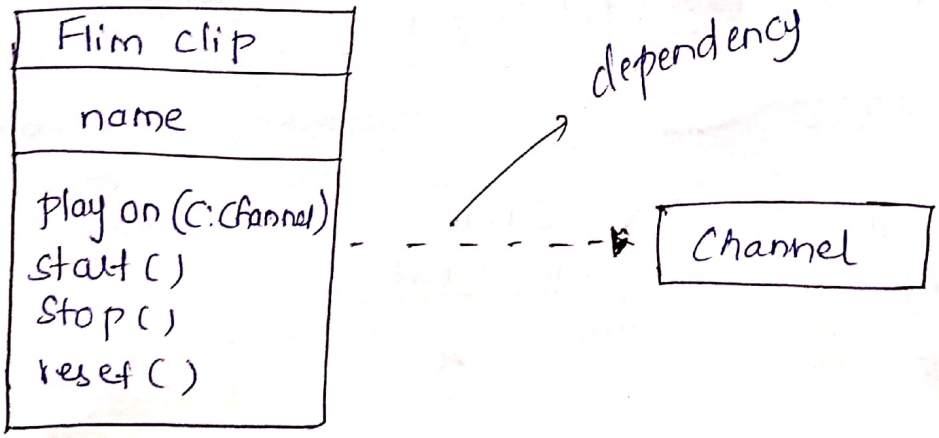
2. The 3 most important relationships are dependencies, generalizations and associations.

3. Graphically a relationship is rendered as a path, with different kinds of lines used to distinguish the kinds of relationships.

Dependency:

A dependency is used relationships that states that a change in specification of one thing (for eg: class EVENT) may affect another thing that uses it (for eg: class WINDOW), but not necessarily the reverse.

→ Graphically, a dependency is rendered as a dashed directed line



Generalization:

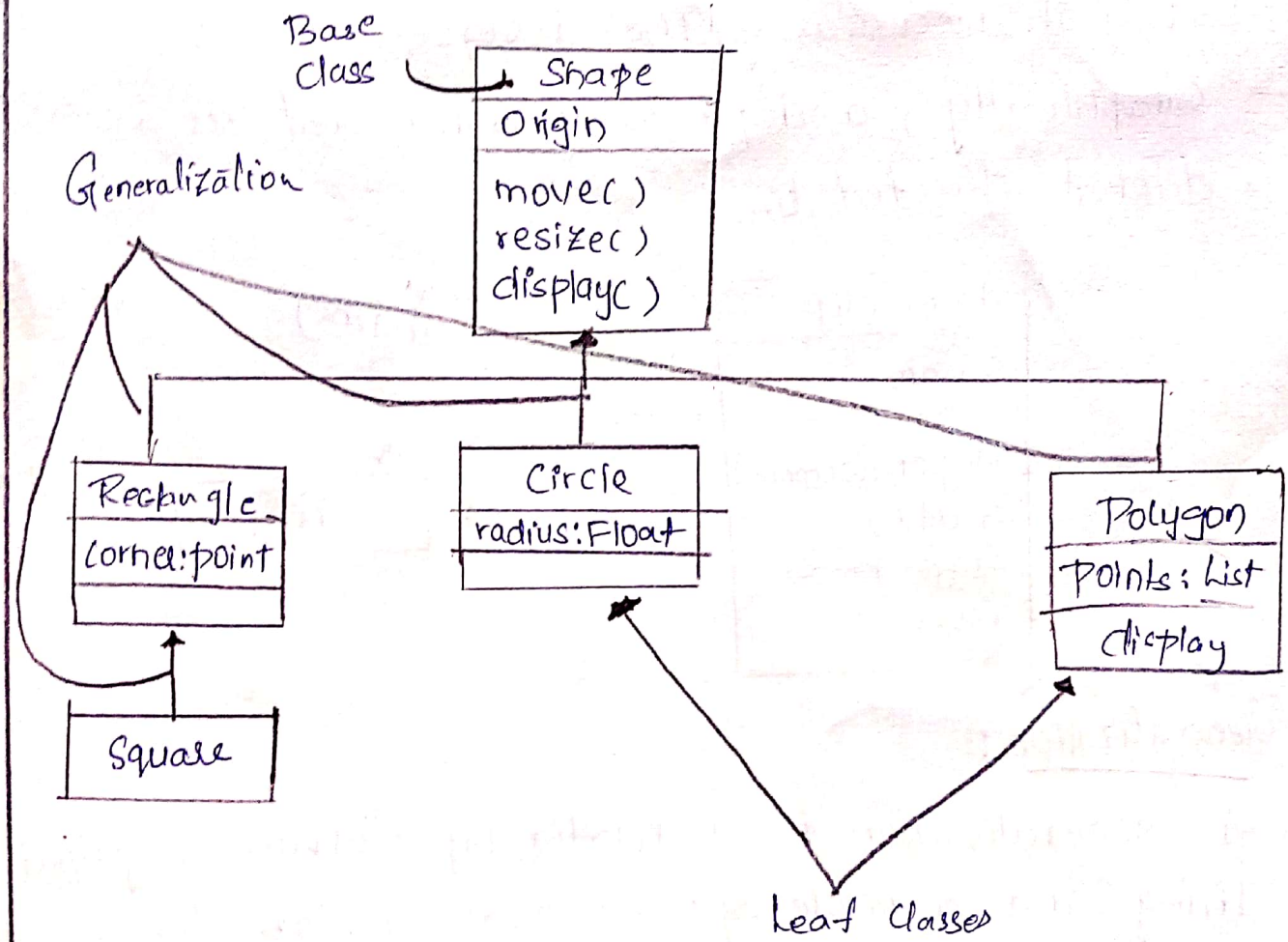
1. A Generalization is a relationship between a general thing and a more specific kind of that thing (Subclass)

2. Generalization is sometimes called as "is-a-kind-of"

2. a relationship : one thing (like the class Bag Window) is-a- kind-of a more general thing (Eg :- Window class)

3. Generalization means that objects of the child may be used anywhere the paset may appear, but not the reverse.

4. In other words, generalization means that the child is substituable for the parent. A child inherits the properties of its parents, especially their attributes and operations.

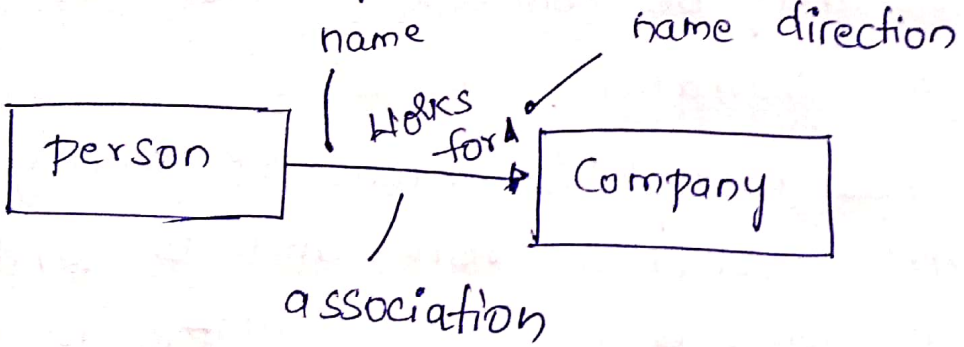


Association:

1. An association is a structural relationships that specifies that objects of one thing are connected to objects of another. Given an association connecting two classes, you an object of the other class and vice versa.
2. It's quite legal to have both ends of an association circle back to the same class. This means that given an object of the objects of the same class.

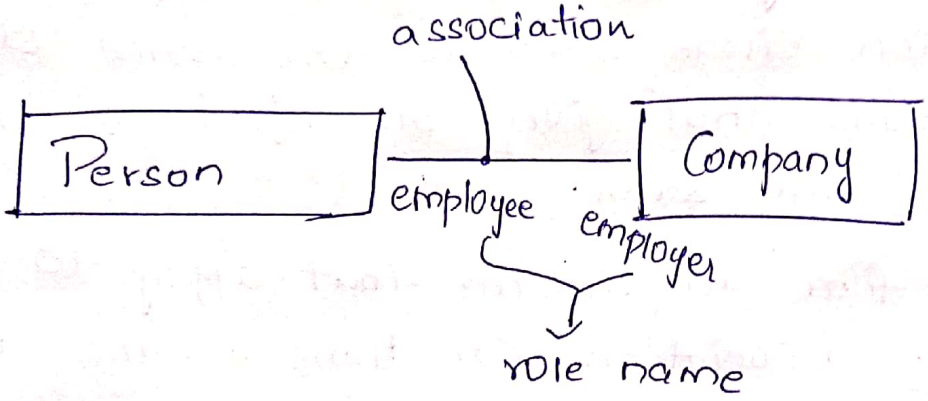
There are four adverisments that apply to association

Name : An association can have a name, and you use that name to describe the nature of the relationship, So that there is no ambiguity about its meaning, you triangle that points in the direction you intend to read the name as shown in figure.



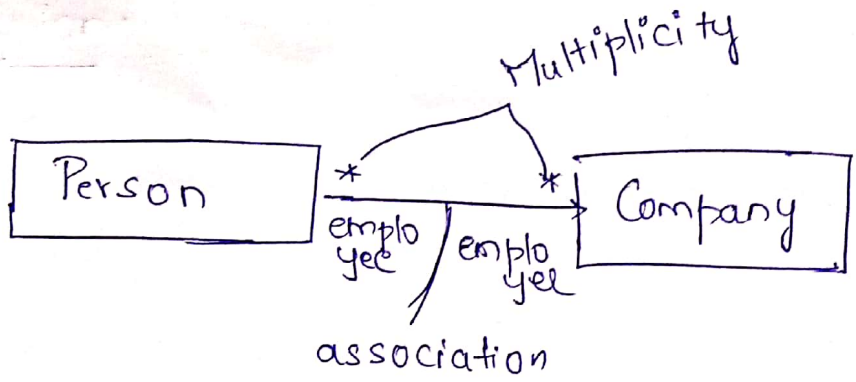
Role

1. When a class participates in an association, it has a specific role that it plays in that relationships
2. A role is just the face the class at the near end of the association presents to the class at the other end of the application.



*Multiplicity:

1. An association represents a structural relationshing among objects. In many modelling situations. It's important for you to state how many objects may be connected across as instance of an association.
2. This "how many" is called the multiplicaty of an associations role, and is written as an expression that evaluate to a range of values or an explict values as in figule



Aggregation

1. A plain association between two classes represents a structural relation between peers, meaning that both classes are conceptually at the same level, no more important than the other.

2. Sometimes you will want to model a "whole part" relationship, which one class represents a larger thing ("the whole") which consists of smaller things. The kind of relationship is called aggregation, which represents a "has – a" relationship.