Year : 2016 (FALL)

1(a) What is Object-Oriented analysis and design? Support your answer with suitable example.

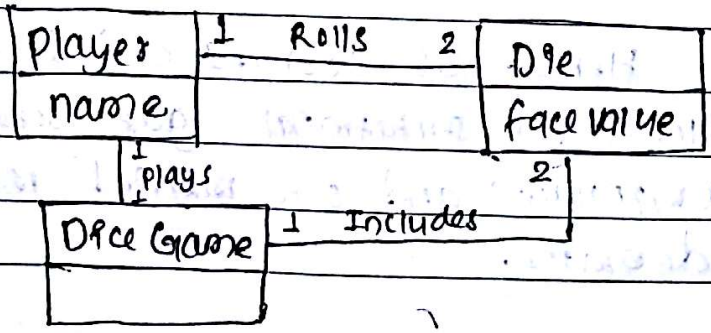
Sol) Object Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects. So, Object oriented analysis and design (OOAD) is a popular technical approach for analyzing and designing an application, system or business by applying object oriented programming, as well as using visual modelling throughout the development life cycles to foster better stake holder communication and product quality.

For example, consider a 'dice game' in which software simulates a player rolling two dice. If total is seven, he wins else lose.

Define use case.

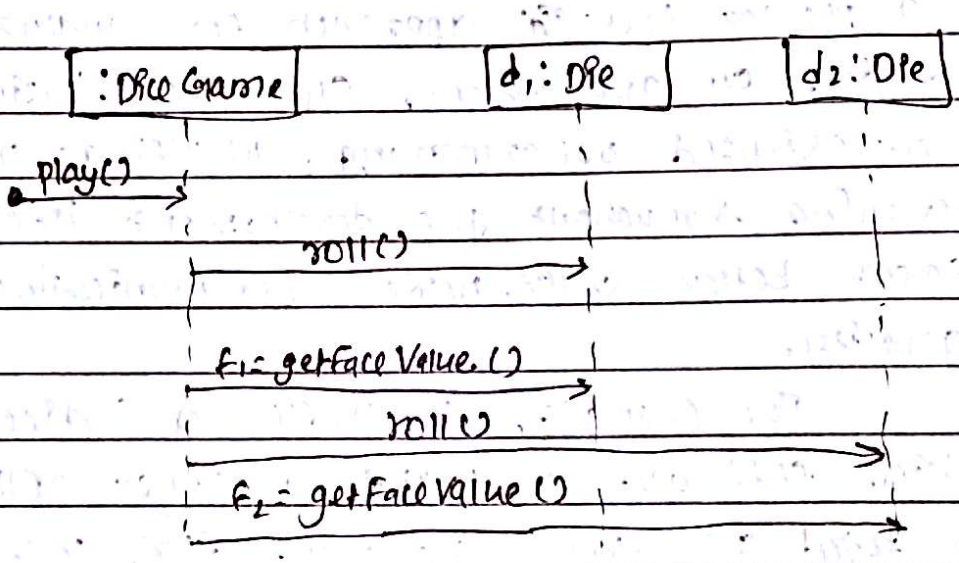
- Player requests to roll the dice. System presents results: If the dice face value totals seven, player wins; Otherwise, player loses.

Define a Domain Model

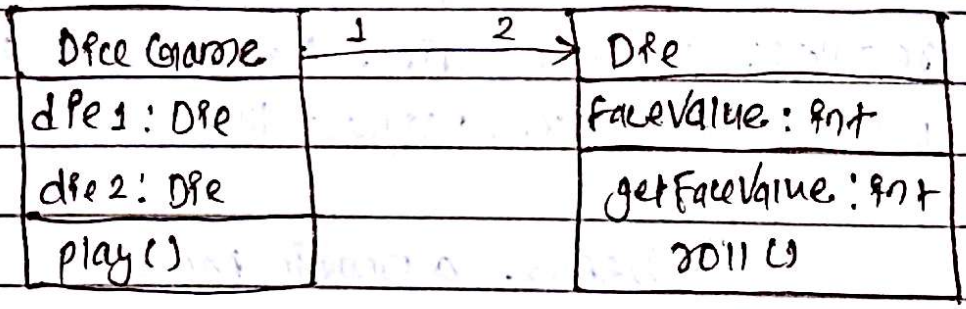


so, domain model helps in visualization of the concepts or mental models of a real-world domain. so, it is also called conceptual object model.

Assign object Responsibility and Draw interaction diagram.



Define Design class Diagram



Hence OO designs and languages can support a lower representational gap between the software components and our mental modes of a domain.

1(b) What are phases of Unified process? Describe in short.

Soln → The Unified process is a popular iterative and incremental software development process framework. It is the best known and extensively documented refinement of the Unified process. It is not just only a process but rather an extensively used framework which should be customized for specific organizations or projects. The phases of unified process are enlisted below:

(i) Inception
(ii) Elaboration
(iii) construction
(iv) Transition

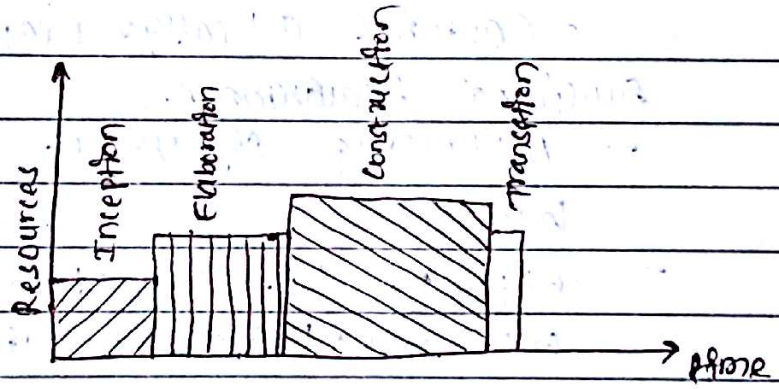


Figure: Graph showing time Vs resources in UP.

* Inception phase:

Inception is the smallest phase in project, and ideally is short phase. The following are typical goals for the inception phase:

- Establish

- prepare preliminary project schedule and cost estimate.
- Feasibility
- Buy or develop it

The life cycle objective milestone marks the end of the inception phase.

Elaboration

The goal of elaboration phase is to establish the ability to build new system given the financial constraints, and other kinds of constraints that the development project faces. The task in elaboration phase includes:

- Capturing a healthy majority of remaining functional requirement.
- Addressing significant risk on ongoing basis.
- expanding the candidate architecture into full architectural baseline.

And major milestone associated with elaboration phase are:

- most of the functional requirements for new system ~~has~~ have been captured in. in the use case modes.
- architectural baseline is complete which will serve as solid foundation for ongoing development.

Construction:

The primary goal of construction phase is to build a system capable of operating successfully in beta customer environments. During construction, the team performs tasks that involve building the system iteratively and incrementally making sure that the viability of the system is always evident in executable form.

Transition:

The primary goal of transition phase is to roll out the fully functional system to customers. During transition, the project team focuses on correcting defects and modifying the system to correct previously unidentified problems.

2 (a) "System development is model development". Do you agree? Justify.

Sol'n → The system development is process of system engineering, information system and software engineering to describe a process of planning, creating, testing and deploying an information system. Where as model development is the formulation of conceptual model as a result of system modeling that describes and represents systems.

As like in model development, first

Date
Page

Of all we specify domain for which the model is to be developed. Then, the model is developed as per the need, it is tested for it's output. Similarly a system is developed as in the same way. It goes through number of tests and improvements. As like: a model of new component is needed then it is attached, like the same way a sub-system is added into system when the new component is needed. Hence, we can state that "system development is model development."

2(b) Discuss the strength and weakness of object-oriented and prioedural programming with the help of banking transaction example.

Sol? The strength of object-oriented programming are:

- Modularity for easier troubleshooting.
- Reuse of code through inheritance
- Flexibility through polymorphism.
- Effective problem solving.

The weakness of OOP are:

- programs are of large size.
- it requires a lot of work to create a project.

- Object oriented programs are slower than other programs. programs demand more system resources, thus slowing the program down.

The strength of procedural programming are:
- Excellent for general purpose programming.
- No-need to reinvent the wheel as well tried and tested coding algorithms are available.
- Good level of control without having to know precise target CPU details.
- portable source code.

The weakness of procedural programming are:
- Lack of a cognitive structure for understanding the relationship between all the procedures and functions and data.
- programmers need to specialize in a specific procedural programming language.
- It is difficult to relate with the real world projects.
- It is difficult to maintain, if the code grows larger.
- lack of security.

For example, let us take banking transaction:

OO programming	procedural programming

main()
{
Customer c1 = New Customer(acc-no);
/* some codes */
Transaction t = New Transaction()
t.makeTransaction(c1, amount);
t.flush();
/* Some codes */
}
	main()
	{
	Struct Customer c1;
	/* some codes */
	c1.accno = acc-no;
	/* Some codes */
	makeTransn(c1, amount);
	/* some codes */
	}

3(a) Describe work flow for capturing requirement as use cases, including the participating workers and their activities.

Sol? Use case model allows developer and customer to agree on requirements, i.e. conditions and capabilities to which the system must confirm a model of system containing actors and use cases and their relationships. Actor represents user type. Each type of users may be represented by one or more actors often corresponds to worker in a business. A role of worker

defines what a worker does in a particular business process. The roles can be used to derive corresponding actors will play. Each use case represents a way the actors use the system. A use case specifies a sequence of actions, including alternatives of the sequence; that the system can perform. Example: Withdraw money (granted, denied, different amount etc). Use case instance is the execution of a usecase. It is one path through the use case. An sequence of interactions between an actor instance and the use case. Each worker in a business is an actor. During such process of enlisting actors, it should be possible to identify at least one user who can play the candidate actor- minimal overlap between roles.

Example: buyer, seller, accounting system etc

3(b) Design is four dimensional view of a system. Justify along with design concepts.

Sol? Software design is a process through which requirements are translated into a representation of software. From a project management point of view, software design can be conducted in two main steps:

① preliminary Design:

concerned with the transformation of requirements into data and software architecture.

(9) Detail Design:

Focuses on refining the architectural representation and lead to detailed data structure and algorithmic representation of software.

The three main design activities concerned in Design phase are: Data Design, Architectural Design and procedural Design. In addition, many modern applications have a distinct interface design activity.

- Data design is used to transform the information domain into data structures.
- Architectural Design is used to develop a modular program structure and represent the control relationships between modules.
- procedural design is used to transform structural components into a procedural description of the software.
- Interface Design establishes the layout and interaction mechanisms for human-machine interaction.

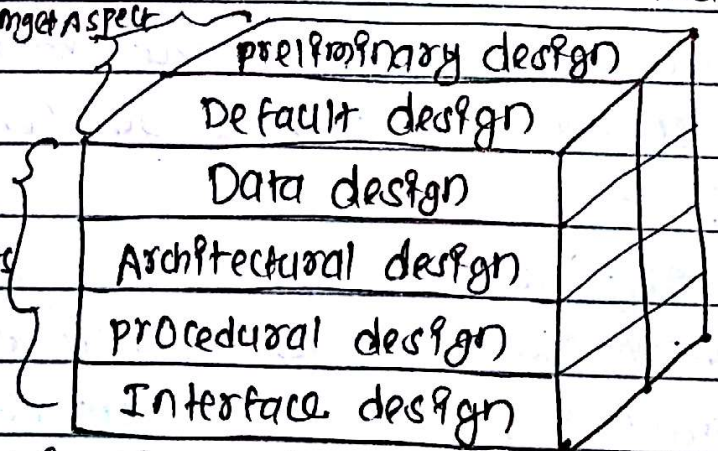


Figure: Relationship betn technical & Mgt. Aspect of design.

4@ Define design pattern. How is design pattern important? Is software development possible without applying design pattern?

Sol? → A design pattern is a general repeatable solution to a commonly occuring problem in software design. A design pattern is not finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.

The design pattern are important because:

- They can speed up development process by providing tested, proven development paradigms.
- Reusing it helps to prevent subtle issues that can cause major problems and improve code readability for coders.
- They provide general solutions, documented in a format that does not require specification to a particular problem.
- Allows developers to communicate using well-known, well understood names for software interactions.

Yes, software development is possible without applying design pattern. As it is just a general solution to some common problem. But developers may choose to resolve using other methods. But using them highly helps in speeding up development.

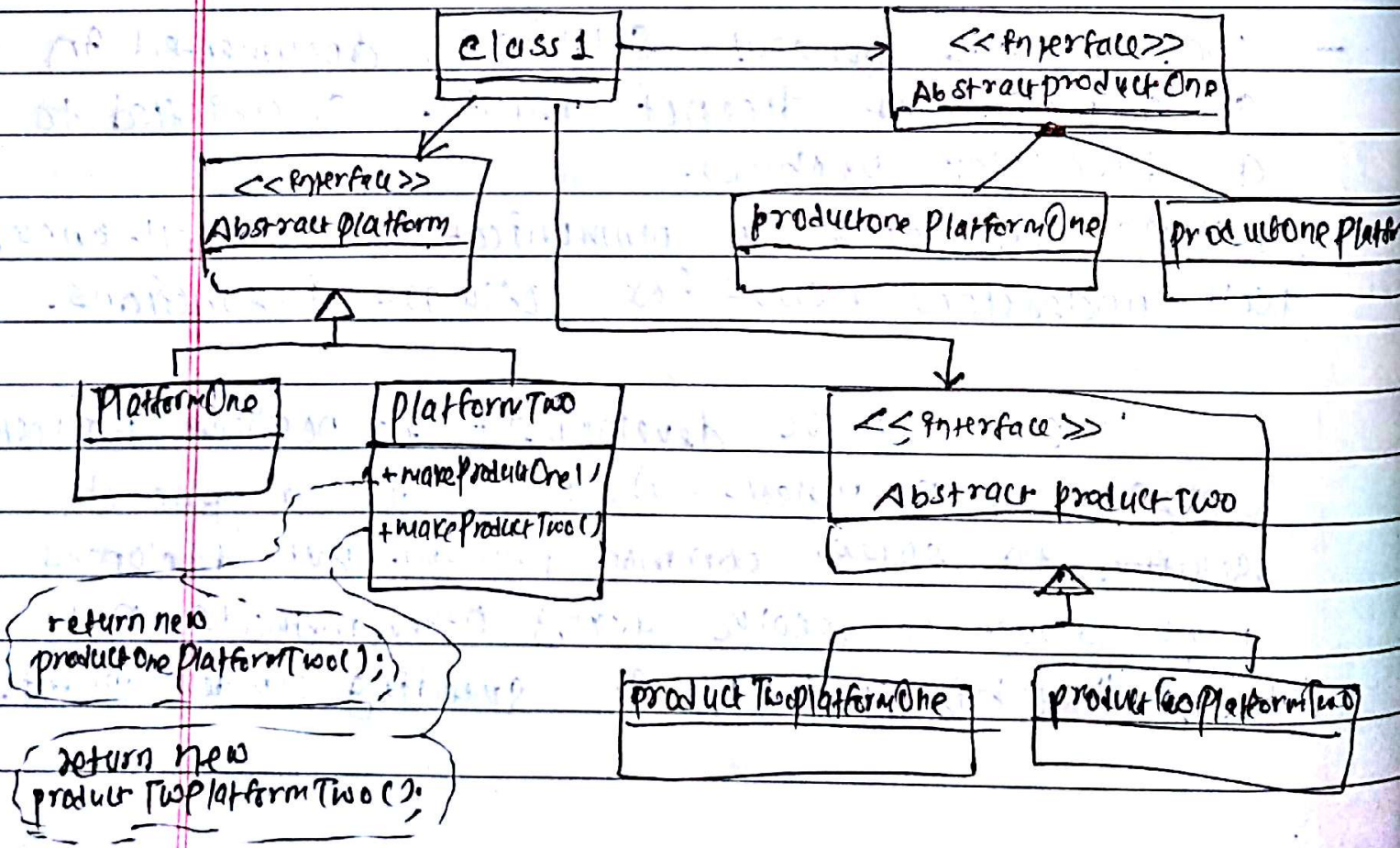
4(b) Describe suitable design pattern for following problem. "What is the best way to represent related objects (occurence) in a class diagram?"

Sol? → As looking at the problem, the best solution or design pattern is Abstract Factory. It is so because

→ It provides an interface for creating families of related or dependent object without specifying their concrete classes.

→ A hierarchy that encapsulates: many possible "platforms and the construction of a suite of "products".

→ The new operator considered harmful, so, the structure will be,



5@ Define design principle, design concept and design pattern. What are the disadvantages of design patterns?

Sol? → Design principles represent a set of guidelines that help us to avoid having a bad design. The design principles are associated to Robert Martin who gathered them in "Agile Software Development: principles, patterns and practices". According to Robert martin there are 3 important characteristics of a bad design that should be avoided.

Rigidity:
It is hard to change because every change affects too many parts of the system.

Fragility
When you change unexpected parts of system break.

Immobility:
It is hard to reuse another application because it cannot be disentangled from current application.

Design concepts usually describes about the abstraction in system. How to provide abstraction between lower level and higher level. And hence, the types of abstraction are

① procedural Abstraction
② Data Abstraction

Design pattern is a general repetable solution, to a Commonly occurring problems in software design. It is description or template for how to solve a problem that can be used in many different situations.

The disadvantages of Design pattern are
- They do not lead to direct code reuse.
- They are complex in nature.
- They are deceptively simple
- They are validated by experience and discussion.

5(b) What is software architecture? Why do we need it?

Sol'n → Software architecture refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures. These structures are needed to reason about the software system. Each structure comprises software elements, relations among them, and properties of both elements and relations.

Software architecture is about making fundamental structural choices which are costly to change once implemented. Software architecture choices include specific structural options from possibilities, in the design of software. For example, the system that control the space shuttle launch vechicle had the requirement of being very fast and very reliable. Therefore an appropriate real-time computing language would need to be chosen.

We need software architecture for

- high productivity
- Better code maintainability
- higher adaptability
- Hype agnostic

6 (a) Describe software oriented architecture and design principles it helps to adhere.

— A software-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. The basic principles of service oriented architecture are independent of vendors, product and technologies. A service is a discrete unit of functionality that can be accessed remotely and acted upon

and updated independently, such as retrieving a credit card statement online. A service has four properties according to one of many definitions of SOA.

(i) It logically represents a business activity with specified outcome.

(ii) It is self-contained.

(iii) It is a black box for it's consumers.

(iv) It may consist of other underlying services.

The service oriented design principles may be broadly categorized as follows:

- Standardized service contract
- Service loose coupling
- service abstraction
- service reusability
- Service autonomy
- Service statelessness
- Service discoverability

6(b) Discuss in short about MVC Architecture with suitable diagram.

→ Model-View-Controller (MVC) architecture is software architectural pattern for implementing good software on computers. It divides a given application into three interconnected parts. This is done to separate internal

representation of information from the way information is presented to and accepted from the user. the MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

Traditionally used for desktop GUIs, this architecture has become popular for designers in web applications and even mobile, desktop and other clients. popular programming language like Java, C#, Ruby; PHP and others have popular MVC framework that are currently being used in web application development straight out of the box.

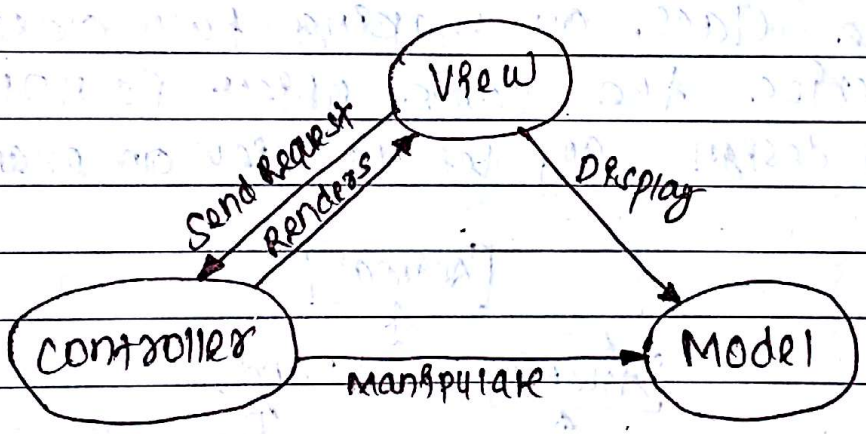


Fig: MVC Architecture

The model stores data that is retrieved according to commands from the controller and displayed in the view. A view generates new output to the user based on changes in the model. A

controller can send commands to the model's state. It can send commands to it's associated view to change the view's presentation of the model.

7 (a) Player role design pattern:

We know during development, we may need to assign different roles in different context of an application. An object may not need to play one or both roles simultaneously, so it is desirable to improve encapsulation keeping information related to only one role in one class, or making two classes to describe the same object is not good OO design so, let us view an example:

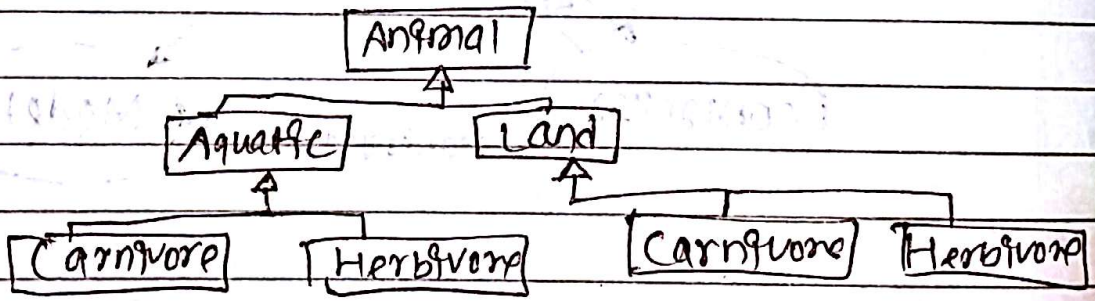


Figure: Badly designed classes.

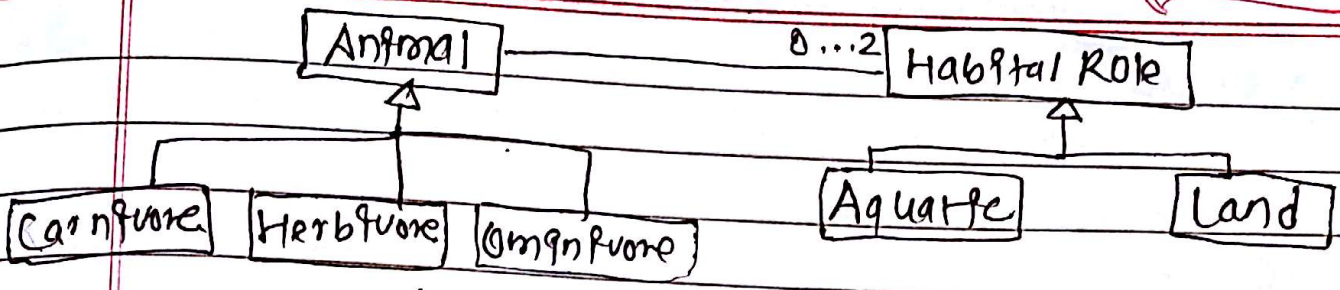
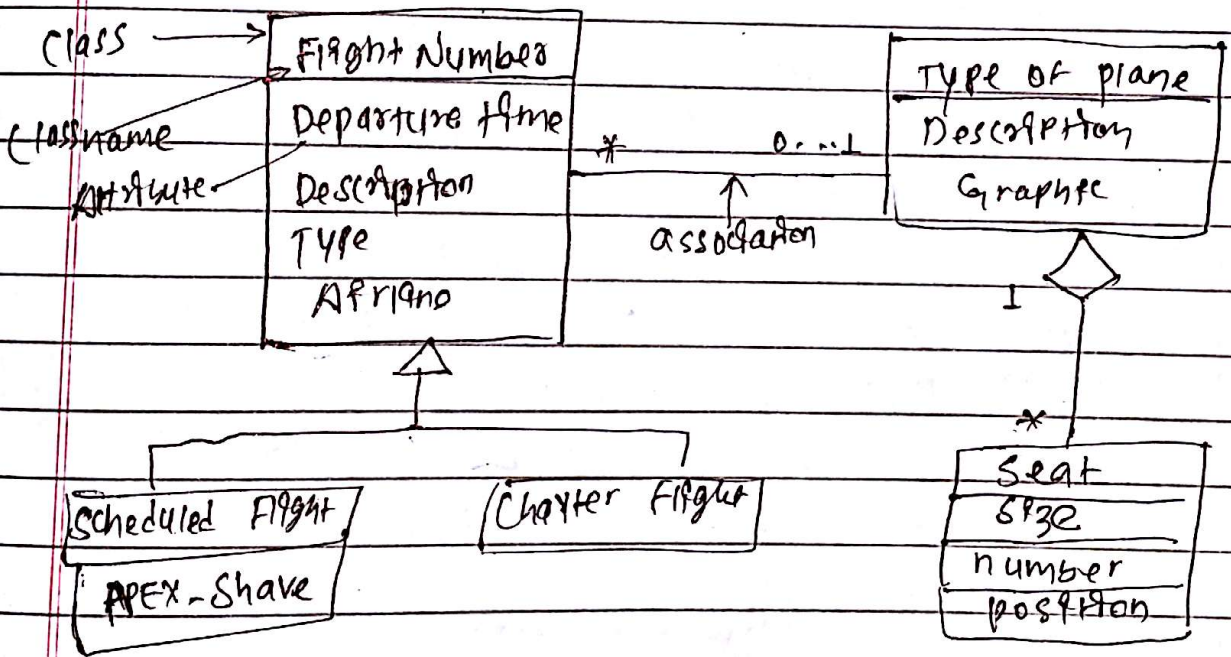


Fig: Finely designed classes.

7© Class Diagram

Class Diagram as a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.



A class represents a relevant concept from the domain, a set of persons, objects, or ideas that are depicted on the IT system.

Year : 2015 (Spring)

1@ What is UML? Is UML an object oriented supported tool to design object oriented systems. Support your example with a suitable example.

Sol'n Unified modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. So, UML is an important aspect involved in object oriented software development.

The UML is a tool for specifying s/w systems. standardized diagrams types to help us describe and visually map a software system design and structure. Many article discuss great feature of UML, but just a few of them describe experimental results with UML in real-life projects. In order to quantitatively present the advantages of UML over common way of software development.

A group of students who took (attended the course during their university educational on C++, SQL basic, Java basic and C were divided into two subgroup. Among them, group (A) took UML courses of presentation while the group (B) didn't attend any UML sessions. Now, two systems were to be designed. The non-UML group or group (B), having no knowledge on UML nor experiences in the object oriented

system design and development had to design the systems using previous knowledge and sense. so, they just made few tough descriptions of the system operation and some flow-diagrams on the flip side, UML group could use various tools of UML to understand and show the internal structure & architecture of the system. It created mostly use case sequence and activity diagrams and even class diagram that show classes, methods and attributes. Now, implementation as a final goal of every development process requires that many details got to be defined.

1(b) What are the activities of Unified process? Describe in short.

Sol^n → The unified process is a popular iterative and incremental software development process framework. It is best known and extensively documented refinement of the Unified process. It is not just only a process but rather an extensively used framework which should be customized for specific organizations or projects. The phases and activities of unified process are enlisted below:

(i) Inception
(ii) Elaboration
(iii) construction
(iv) Transition

Inception

Inception is the smallest phase in project and ideally is short phase. In inception phase idea is planned. The following are type or goals for the inception phase:

- Establish.
- prepare preliminary project schedule and cost estimate
- Feasibility
- Buy or develop it.

The life cycle objective milestone marks the end of the inception phase.

Elaboration

The goal of elaboration phase is to establish the ability to build new system given the financial constraints, and other kinds of constraints that the development project faces. The task in elaboration phase includes:

- Capturing a healthy majority of remaining functional requirement.
- Addressing significant risk on ongoing

basis.

- expanding the candidate architecture into full [redacted] architectural baseline.

And major milestone associated with elaboration phase are:

- most of the functional requirements for new system have been captured in the use case model.
- architectural baseline is complete which will serve as solid foundation for ongoing development.

construction

The primary goal of construction phase is to build a system capable of operating successfully in beta customer environments. During construction, the team performs tasks that involve building the system iteratively and incrementally making sure that the viability of the system is always evident in executable form.

Transition

The primary goal of transition phase is to roll out the fully functional system to customers. During transition the project team focuses on correcting defects and

modifying the system to correct previously unidentified problems.

2@ What is business modelling? What is the relationship between Business modeling requirement gathering in the development of a software? Take one example to prove your point.

Sol The process of business model design is part of business strategy. Business model design and innovation refer to the way a firm (or a network of firms) defines its business logic at the strategic level. In contrast, firms implement their business much at the operational level, through their business operation.

For many projects, deciding what needs to be done is more difficult than getting it done. It does not matter how talented a development team is, if they are not sent out to accomplish the right goals, they can't be expected to find success. That's why proper requirements are essential to produce high quality software.

The requirement can be obvious or hidden or unknown, expected or unexpected from client point of view. Requirement Gathering is one of the steps of requirement engineering process which includes: Feasibility requirement study,

Software requirement specification and Software requirement validation. This implies firstly the feasibility study analyzes whether the software product can be practically materialized terms of implementation, contribution of project organization cost constraints and as per values and objectives of the organization.

Today's S/w development methodologies are equipped with a combination of methods and technologies for business process engineering and requirements. However experienced has shown that the methods and technique deliver disappointing results when being applied independent of each other. There are two main obstacles to effective alignment of business process and the information system (IS) that support them:

(i) The IS are not developed with correct understanding of the business they are supposed to support.

(ii) Business process are not linked to the system requirements and thus envolve independent from the IS.

Hence, unless the two process. i.e. business modelling and requirement gathering go side-by-side or aligned, the development of a software is heavily affected after math.

2 (b) Discuss the strength and weakness of Object Oriented and procedural programming with the help of point of sale system.

Ans → Object oriented programming is problem solving technique which uses classes and objects to create models based on the real world environment. An OOP application may use a collection of objects which will pass message when called upon to request a specific service or information. The main advantage of OOP programming is it makes it easy to maintain and modify existing code as new objects are created inheriting characteristics from existing one. This makes adjusting program much simpler. But the major disadvantage of OOP is increase in size of program, effort speed.

Similarly, procedural programming which at time has been referred to as line programming takes more top-down approach to programming. They takes on applications by solving problems from the top to the code down to the bottom. They are faster

and size efficient but developer must edit every line of code that corresponds to the original change in code. For example: - let us take a point of sales system which comprises network operated by a main computer and linked to several checkout terminals with pos system.

— one can analyze sales data, figure out how well the item on your shelves sell and adjust purchasing levels accordingly.

— One can maintain a sales history to help adjust buying decisions for seasonal purchasing trends.

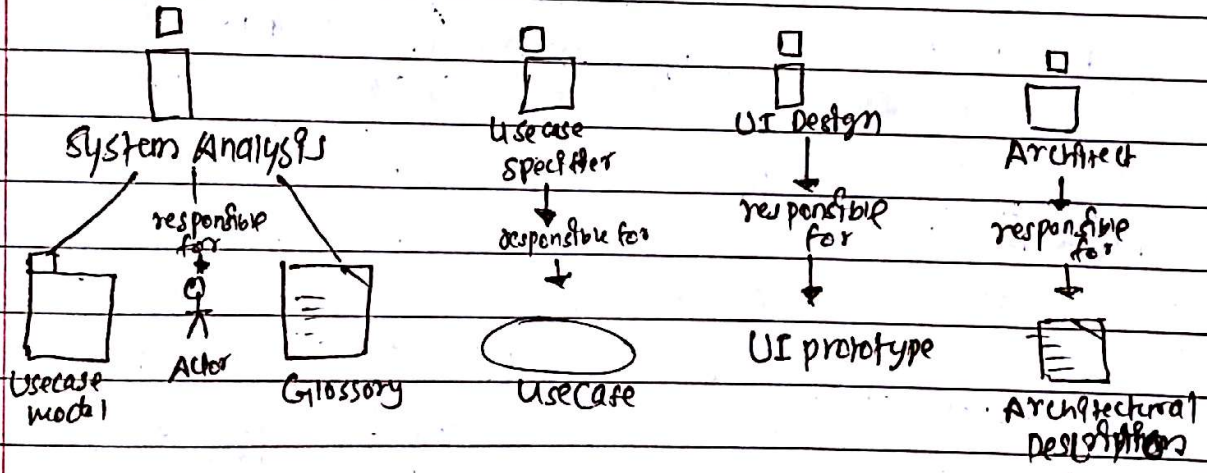
— One can improve pricing accuracy by integrating bar-code scanners and credit card authorization ability with pos system.

Q 3(a) What are use cases? Describe workflow for capturing requirement as use cases, including the participating workers and their activities.

Sol^n → In software and system engineering, a use-case is a list of actions or events steps typically defining the interactions between a role (known in UML as actor) and a system to achieve a goal. The actor can be a human or other external

system within the unified process, five workflow come across the set of four phases: Requirements, analysis, Design, Implementation, and testing. Each workflow is a set of activities that varies project workers perform.

The primary activities of the requirement work are aimed at building the use case model, which captured the functional requirements of the system being defined. The model helps the project stake to reach agreement on the capabilities of the system and the conditions to which it must conform.



3(b) Design patterns have to be applied using some programming language. Discuss the statement in the light of programming paradigm versus design patterns. Take any two patterns with an example to prove your point.

Sol? → Design pattern are as much lower level tool, providing proven models on how to organize code to gain specific functionality while not compromising the overall structure. In other words, a design pattern is a general, reusable solution to a commonly occurring problem in software design. It's not finished design that can be transformed directly into code. It is a descriptive template for how to solve a problem that can be used in many different situations.

Besides paradigms are other extreme, guiding the principles on how code is actually laid out, and they each require quite different mind sets to apply. For instance, procedural programming is mainly concerned about dividing the program logic in to functions and bundling those functions into modules. Object-oriented programming aims to encapsulate the data and the operations that manipulate the data into objects. Functional programming emphasis the use of functions instead of separating statements following one another, following side effects and state changes.

Some instances of programming paradigm includes :- Imperative - which allow side effect.

- declarative - which does not state the order in which operations execute.
- functional - which disallows side effect.
- Object oriented - which groups code together with state the code modifier.
- procedural - which groups code into functions and so on.

So design patterns are easier to understand, they are template of solution, some common design patterns are :-

- Factory method
- Abstract factory.

Factory method:

creates an object from a set of similar classes, on some parameter, usually a string. Example is: creation of a Message Digest object in Java.

Q) What is the role of design pattern? How is design pattern important? Highlight the role of software development with respect to the design pattern?

Sol) In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem. A design pattern is n't a finished design that can be transformed directly into code.

This is a common method of mathematical pattern analysis and such analysis is important for the following reasons:

- Understanding patterns allows examine to identify such pattern when they first appear.
- It follows someone to make educated guesses.
- Understanding patterns aid in developing mental skill.

To understand the role of design patterns, lets face an example of an adaptor. Adaptor pattern is one among those 23 patterns described in the book "Design pattern" by gang of four. Adaptor provides a solution to the scenario in which a client and server need to interact with one another but cannot because their interfaces are incompatible. To implement an adaptor, we create a custom class that has the interface provided by the server and defines the server operations in terms the client expect.

Hence, the role of design pattern in software development is very crucial to make timely and effective response to design issue and to build a better quality software. Design patterns have two major benefits:

- They provide you with a way to solve issues related to software development

using a proven solution.

(b) They make communication between designers efficient. Software professionals can head when they refer the name of the pattern used to solve the particular issue when discussing system design.

◁ (b) Write about structure and documentation of pattern.

Sol⁷ Design pattern documentation is highly structure. The patterns are documented from template that identifies the information needed to understand the software problem and the solution in terms of the relationship between the classes and objects necessary to implement the solution. There's no uniformity in the design pattern template declation. i.e. the authors prefer different style for their pattern templates. some want to be more expressive less structural while the rest prefer their pattern to be more precise and high grain instruction. An example of template from the authors of the first book "Design pattern" is as follow!

Term	Description
pattern-name	→ Describes the short expressive name
Intent	→ what the pattern does.
Also known as	→ List any synonym for the pattern
Applicability	→ List the situations where pattern is applicable.
Structure	→ Set of diagrams of the classes and objects that depict pattern.
participants	→ the classes and object that participates in design pattern.
Collaborations	→ Describes how the participants collaborate to carry out their responsibility.

The documentation of design patterns describes about the context in which the pattern is used, therefore the faces within the context that the pattern seeks to resolve and the suggested solution.

5(9) Define design principle, design concept and design pattern. What are the dis-advantages of design patterns?

Soln → Software design principles represent a set of guidelines that help us to avoid having bad design. According to Robert Martin there are 3 important characteristics of a bad design that should be

avoided:

(i) Rigidity
(ii) Fragility
(iii) Immobility

Other major principles includes

* DRY (Don't Repeat Yourself)

⇒ Try to avoid duplicates. For instance, imagine you have copied and pasted blocks of code in different part in your system, then there are lot's of same codes.

* Keep it simple

Most systems works best if they are kept simple rather than making them complex, therefore, simplicity should be a key goal in design and unnecessary complexity should be avoided. Software design is the process of implementing software solutions to one or more sets of problems. One of the main components of a software design is the requirement's analysis (SRA). SRA is a part of the software development process that lists specifications used in software engineering. The sets of fundamentals software design concepts includes the following:

(i) Abstraction

(i) Architecture
(ii) patterns
(iv) Modularity
(v) Information hiding
(vi) Functional independence
(vii) Refinement
(viii) Refactoring

The disadvantages of design patterns are as follows:

- Design patterns may increase or decrease the understandability of a design and implementation. They can decrease by adding indirection or increasing the amount of code.
- It does not lead to direct code reuse.
- Complex in nature
- They are deceptively simple.
- They are validated by experience and decisions.
- Teams may suffer from pattern overload.
- Integrating patterns into a software development process is a human-intensive activity.

5 (b) What is software architecture? Why do we need it?

Sol→ Software architecture is the process of designing the global organization of a software system, including dividing software into subsystems, deciding how these will interact, and determining their interfaces. The term 'software architecture' is also applied to the documentation produced as a result of the process.

Software engineers discuss all aspect of systems design in terms of the architectural model. The architectural model will often consider the overall efficiency, reusability and maintainability of the system. There are four main ~~reason~~ reason why we need to develop an architectural model:

(i) To enable everyone to better understand the system.
(ii) To allow people to work on individual pieces of the system in isolation.
(iii) To prepare for extension of the system.
(iv) To facilitate reuse and reusability.

6 (a) Describe Service oriented architecture and design principles it helps to adhere.

Soln A service-oriented architecture (SOA) is a style of software design where services are provided to the other components by application components through a communication protocol over a network. The basic principles of service-oriented architecture are independent of vendors, products and technologies. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online.

The principles of SOA are:

1. Explicit Boundaries
2. Shared contract and schema, not class.
3. policy-driven.
4. Autonomous
5. Wire formats, not programming languages APIs
6. Document-oriented.
7. Loosely couples
8. Standards-compliant
9. Vendor independent
10. Metadata-driven.

6(b) Discuss about client server and distributed architecture. Are they similar? Justify. What is the role of distributed architecture in today's world of automation? Explain.

Sol^n → client-server architecture is a network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print server) or network traffic (network servers).

On the other hand, distributed system architecture (DSA) is the ideal solution for ~~redep~~ integrating processes where those are multiple units, control rooms or geographically distributed locations. In distributed architecture components are hosted on different platforms.

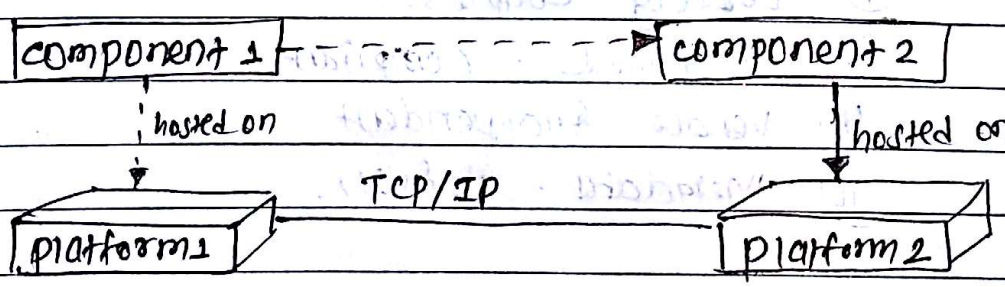


Figure: Distributed system Architecture

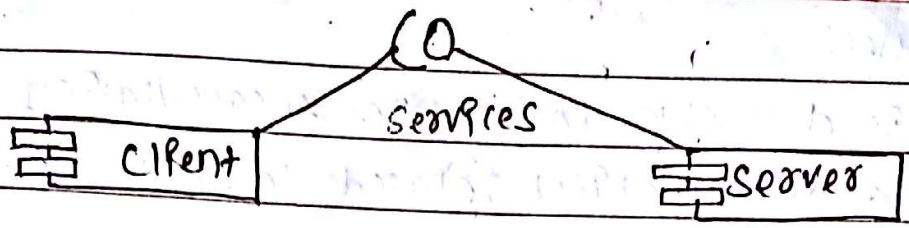


Figure: 1

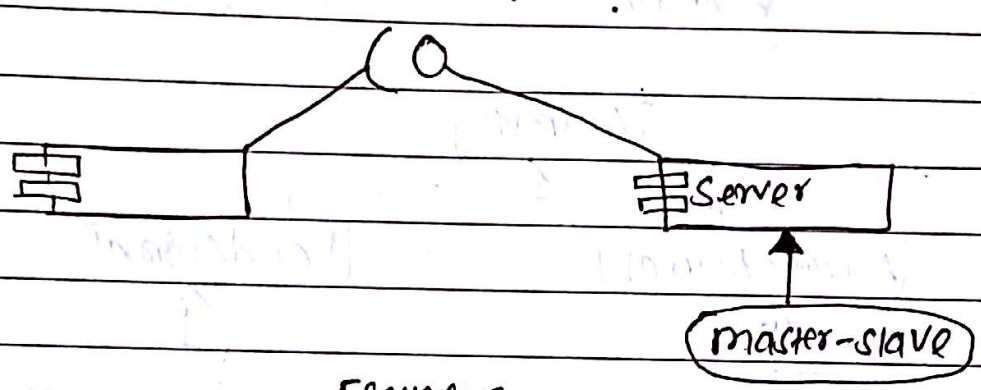


Figure: 2

7 (a) Player-Role design pattern

Soln → "studying software design patterns is an effective way to learn from the experience of others".

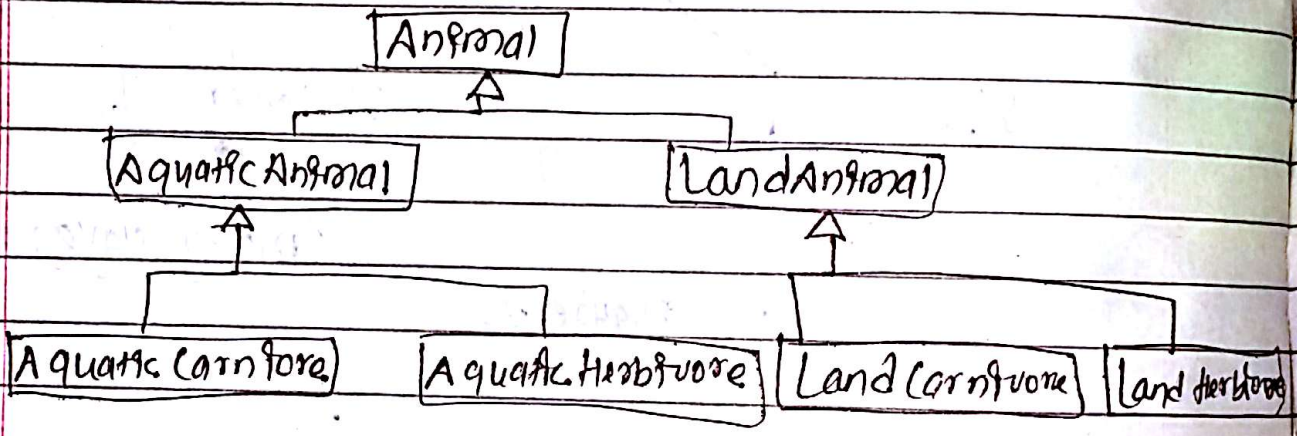
Context:

- an object may play different roles in different contexts of an application.
- an object may need to change roles throughout the course of the application.
- an object may not need to play one or both roles simultaneously.

Issues:

- It is desirable to improve encapsulation by keeping information related to only one role for one class.
- making two classes to describe the same object is not good OO design.

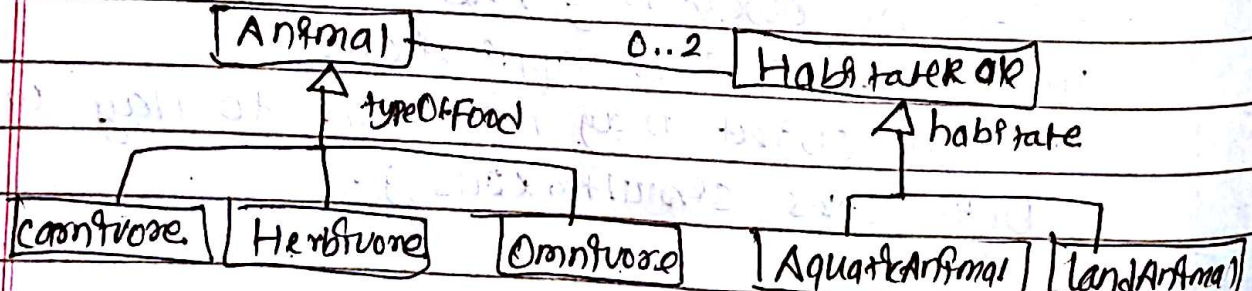
Example of class design of animals



problem:

Two or more objects are required if the animal needs to play different roles, like aquatic and land carnivore, which is not good OO design.

Player-role solution to Animals



Solutions:

An animal can be:
- Carnivore
- Herbivore
- Omnivore

An animal can have:
- No habitat role
- an aquatic habit role
- a land habitat role
- an aquatic and land habitat roles.

7(b)

Sol^n

Reuse Vs Reusability

code reuse is called software reuse. Software reuse is the process of creating new software systems from existing software components. Reuse has an enormous impact on productivity. The following elements of software reused are, Software specifications, designs, tests cases, data, prototypes, plans, documentation, frameworks and templates.

Reusability is the degree to which the artifacts can be reused. The ability to use all or the greater part of the same programming code or system design in another application. Reusability is the segment of source code that can be used to add new functionalities with slight

Or localizes code modifications when a change in implementation is required.
Reusability is the extent to which a software component is able to be reused.
Reusable modules and classes increase the prior testing, reduce implementation time and use has to have appropriate design and coding standards. The types of software Reuse:

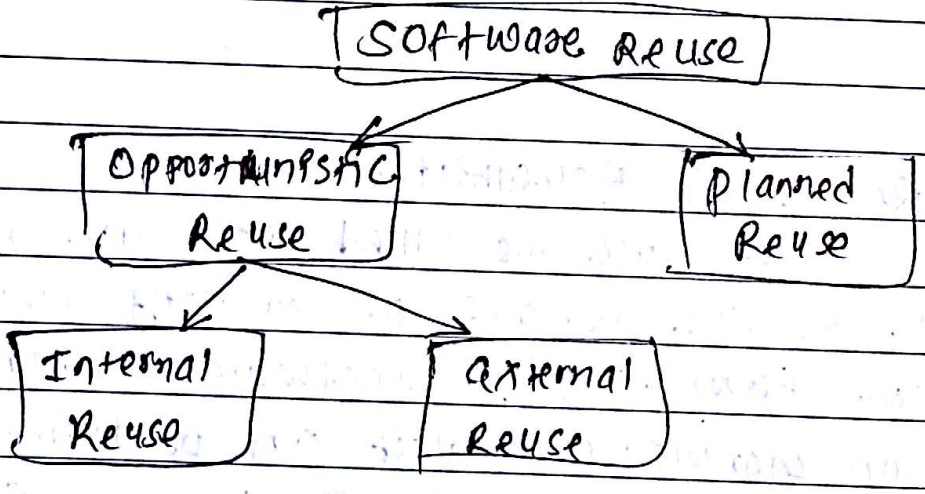


Fig: Types of Reuse.