Name: Shankar Acharya
Roll No. 35
Subject: OOSD

(2016 – Fall)

Q.1 a) What is object-oriented analysis and design? Support your answer with suitable example.

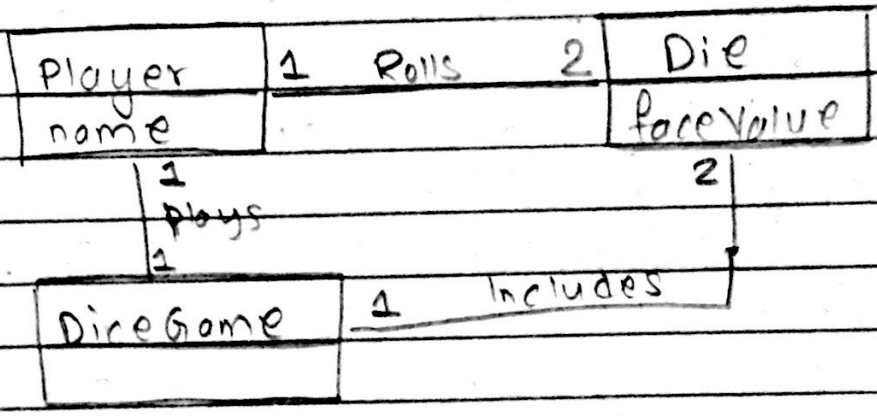
⇒ Object Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects. So, Object Oriented analysis and design (OOAD) is a popular technical approach 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 stakeholder 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

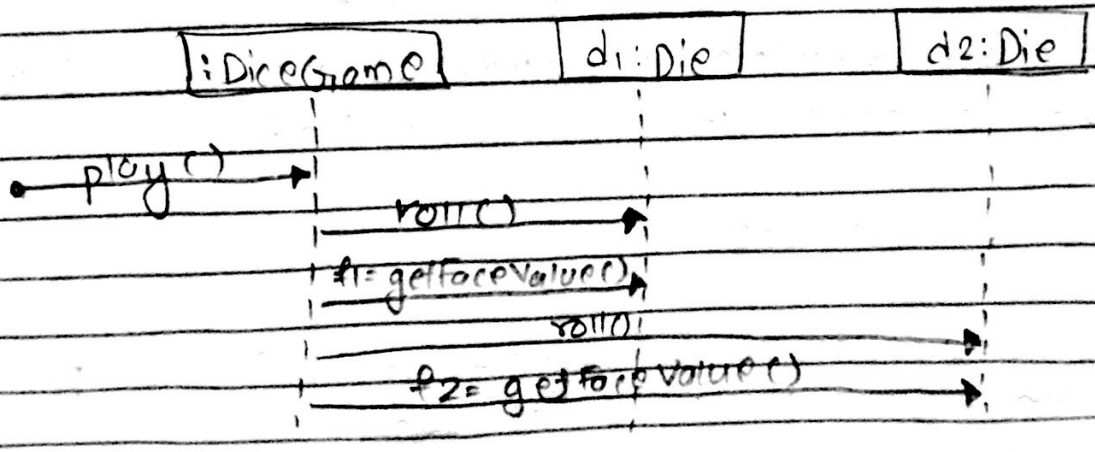
- Player requests to roll the dice. System presents results: If the dice face value totals seven, player wins; otherwise, player loses.

* Define a Domain Model

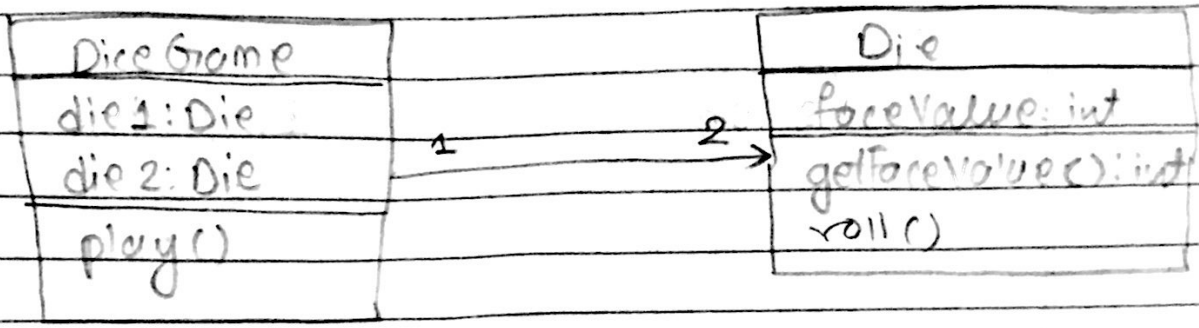


So, domain model helps in visualization of the concepts or mental models of a real-world domain. So, it is also called Conceptual Object Model

* Assign Object Responsibility and Draw interaction diagra Q.1



* Define Design class Diagrams:



Hence OO designs and languages can support a lower representational gap between the s/w components and our mental ~~age~~ modes of a domain.

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

⇒ 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

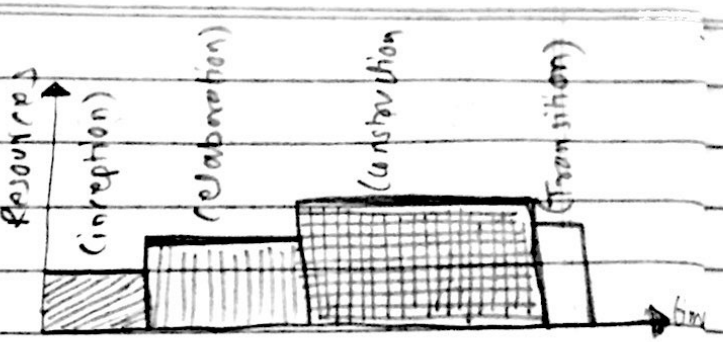


Fig: Graph showing time vs resources in UP

* 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 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 ~~fully~~ 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.

⇒ The System development is process of ~~engineering~~ 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~~ of conceptual model as a result of system modeling that describes and represents system.

As like in model development, first 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 ~~if it~~ 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 if new component is needed then it is attached, like the same we a sub-system is added into system when a new component is needed. Hence, we can state that system development is model development.

Q.2b) Discuss the strength and weakness of Object-Oriented and procedural programming with the help of Banking transaction example.

=> Object-oriented ~~is a~~ 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 messages when called upon to request a sepcific 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 and speed.

Similarly Procedural programming which at times has been referred to as inline programming takes a more top down approach to programming. They takes on applications by solving problems from the top of 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 the code.

For example, let us take banking transaction

OO Programming	Procedural Programming

main ( )
{
~~Customer = new~~ /* some codes */
Customer C1 = new Customer(acc_no);
~~e+.~~
Transaction t = new Transaction()
t.makeTransaction(C1, amount);
t.flush();
/* some codes */
}
	main()
	{
	Struct ~~C++~~ Customer C1;
	/* some codes */
	C1.acc-no = acc-no;
	/* some codes */
	makeTranxn(C1, amount);
	/* some codes */
	
	}

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

⇒ Sy Design is the process of defining the architecture, modules, interfaces and data for a system to satisfy specified requirements. It can be seen as the application of systems theory to product development. There is some overlap with the disciplines of system analysis, system architecture and system engineering. System designing is one of the important phase of s/w development. As when a system is well designed, then it becomes more ~~pr~~ successful project and risk prone. Hence designing part must be handeled with more critical way as like a four dimensional view. Every possible critical thinking must be applied. When a design is well done, then it is possible to move to development phase.

The UML has been the standard language in object-oriented analysis and design. It is widely used for modelling s/w systems and is used for high designing non-software systems and organization. The design concepts provide the software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved. They are as follows.-

1) Abstraction
2) Refinement
3) Modularity
4) Software Architecture
5) Data hierarchy
6) Information hiding.

Q.4 a) Define design pattern. How is design pattern important? Is software development possible without applying design pattern?

⇒ A design pattern is a general repeatable solution to a commonly occuring problem in software design. A design pattern isn't 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 doesn't require specifics tied to a particular problem.
→ Allows developers to communicate using well-known, well understood names for sw interactions.

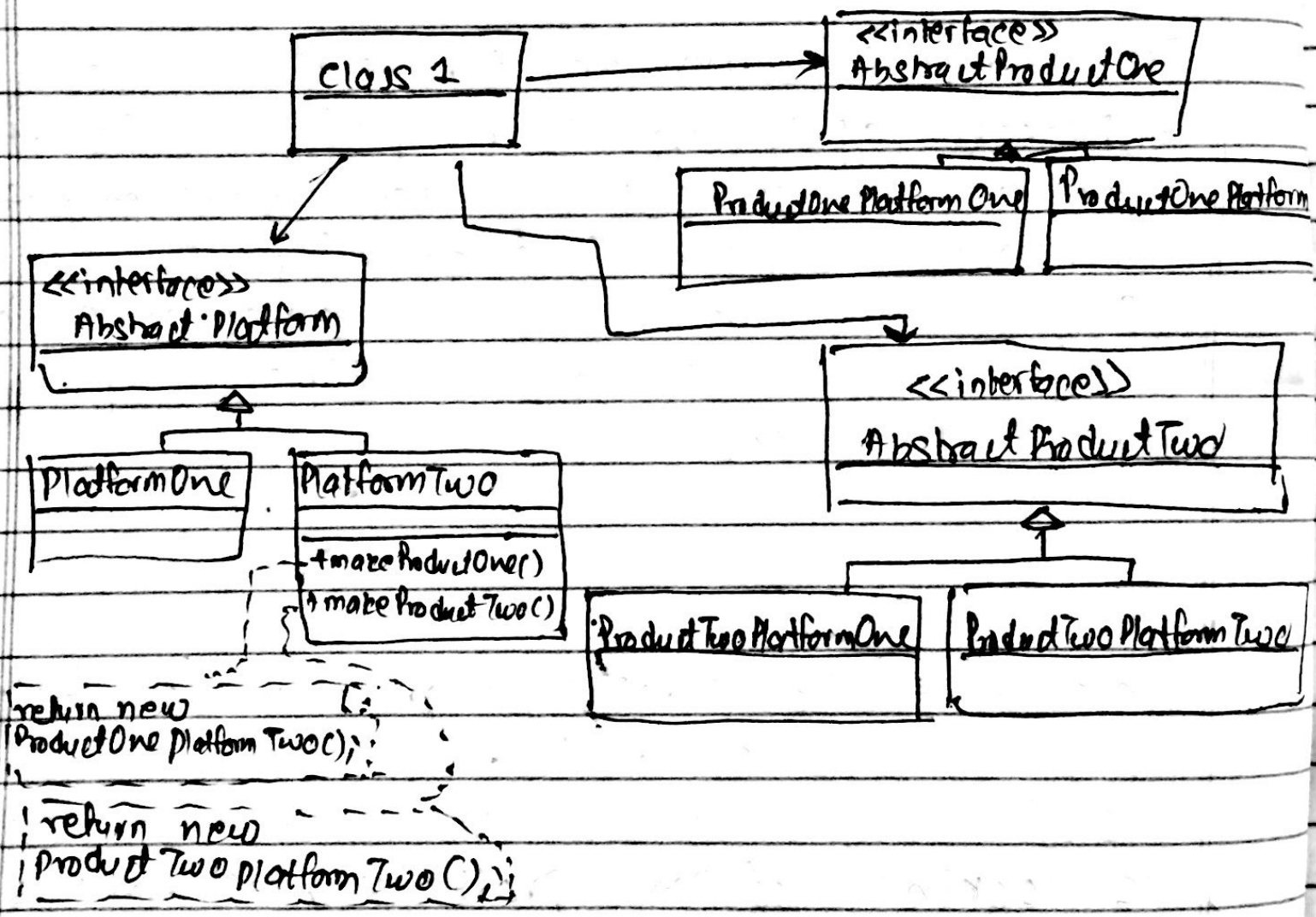
Yes, ~~soot~~ software development is possible without applying design pattern. As it is just a general solution to some ~~prob~~ common problem. But developers may choose to resolve using other methods. But using them highly helps in speeding up development.

2.4 b) Describe "suitable design pattern for following problem. "What is the best way to represent related objects (occurrence) in a class diagram?"
→ As looking at the problem, the best solution of design pattern is Abstract Factory. It is so because
→ It provides an interface for creating families of related & 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.a) Define design principle, design concept and design pattern. What are the disadvantages of design patterns?

=> 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 occuring problem 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 deceptivrly simple
→ They are validated by experince and discussion.

Q.5.b What is software architecture? Why do we need it?

⇒ 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.

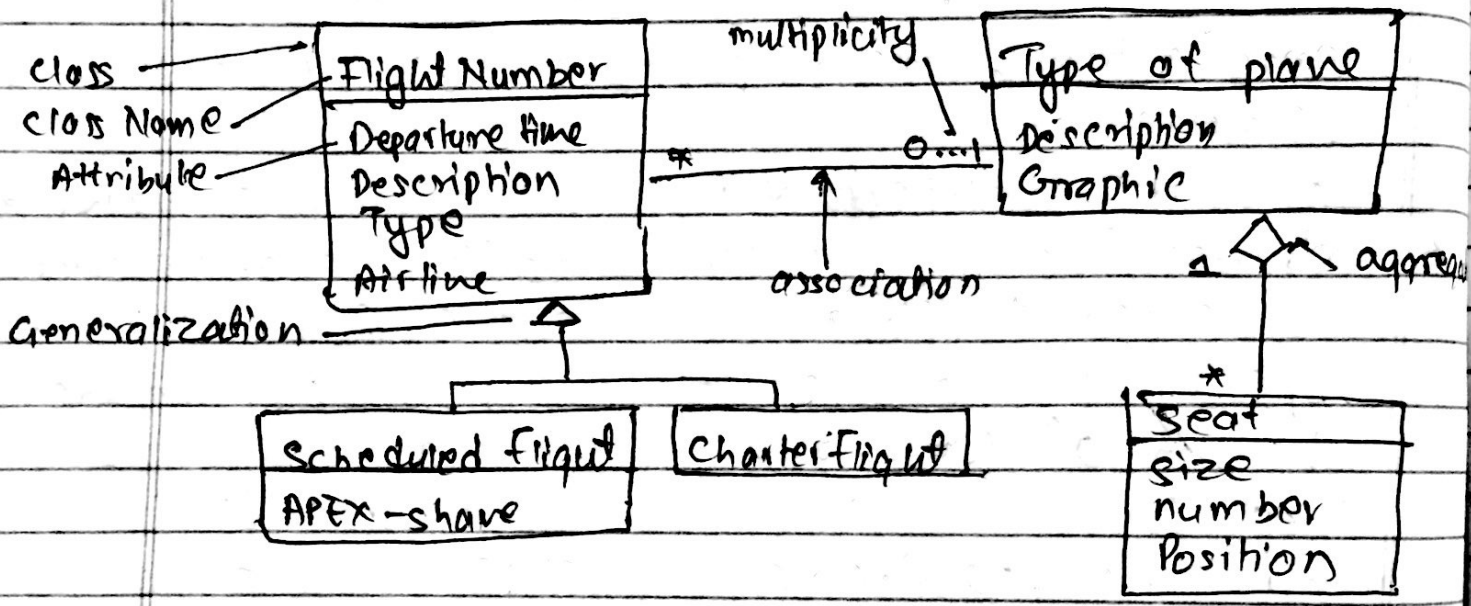
Software architecture is about making fundamental structural choices which are costly to change once implemented. Software architecture choices include specific structural options from possibilities in the design of software. For example, the system that control the space shuttle launch vehicle had the 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

Q.7c) Class Diagram

Class Diagram is 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.

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.

1.) It logically represents a business activity with specified outcome.
2) It is self-contained
3) It is a black box for its consumers.
4) 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

b) Discuss in short about MVC Architecture with suitable diagram

=> Model - View - Controller (MVC) architecture is a software architectural pattern for implementing good ~~p~~ software on computers. It divides a given application into three interconnected parts. This is done to seperate internal representations of information from the ways 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 languages

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.

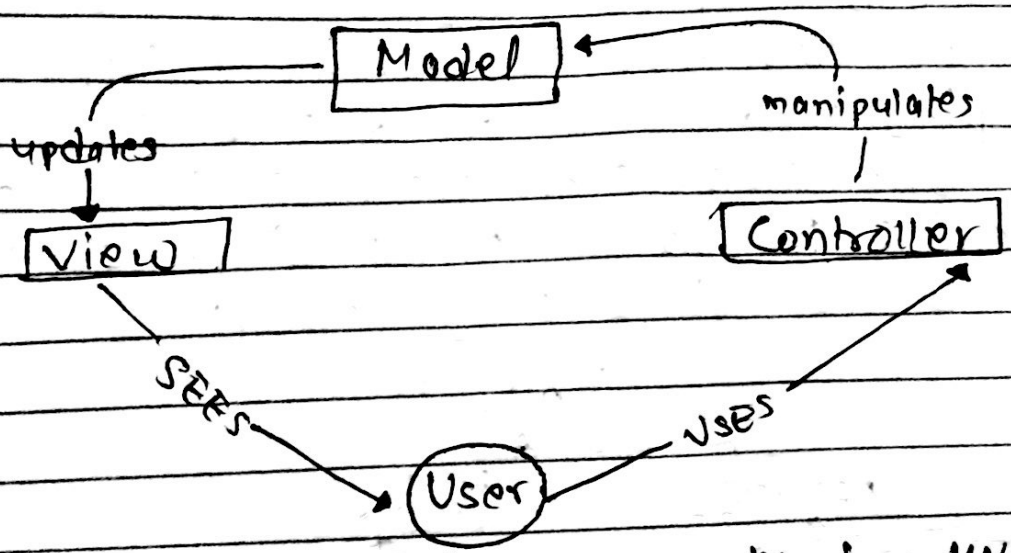


Fig:- Diagram of interactions with in MVC pattern.

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.

2.7.0) Player role Design pattern.

We know during development, we may need to assign different roles in different contexts of an application. An object may not need to play one or both roles simultaneously. So, it is desirable to improve encapsulation by 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:

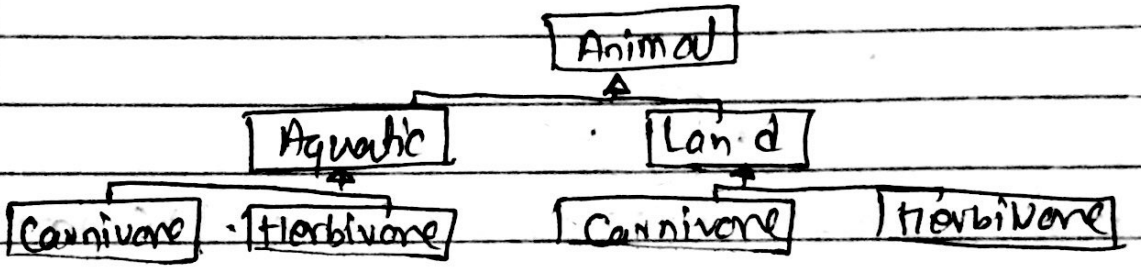


fig:- badly designed classes

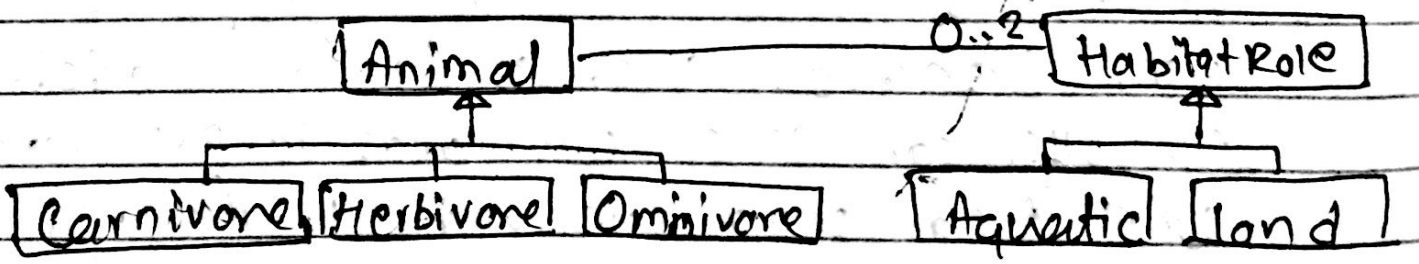


fig:- finely designed classes.