Internal Assessment Test 1 – November 2021

Sub:	Software Architecture & Design Patterns					Sub Code:	17IS72
Date:	11/11/2020	Duration:	90 min's	Max Marks:	50	Sem. / Sec:	7 - A & B

Section -A Descriptive Answers

1. a) Define design pattern according to 'Christopher Alexander'. List the uses of design patterns.

b) Give the classification of design patterns in a neat tabular form and explain.

* Def. from Christopher Alexander :- "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

uses of design patterns :

(i) They make your Job easier
(ii) They ~~provide~~ help you analyze the more abstract areas of a program by providing concrete, well-tested solutions.
(iii) They encourage code reuse and accommodate change by supplying well-tested mechanisms for delegation & composition, and other non-inheritance based reuse technique.
(iv) They encourage more liable & maintainable code by following well-understood paths.
(v) They provide a common language & Jargon for progra-mmers.

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight (195) Observer State Strategy Visitor

* Design patterns are classified by two criteria:
  (1) purpose - Reflects what a pattern does
    - They can be:
      a) creational - concern the process of object creation.
      b) Structural - Deal with the composition of classes or objects.
      c) Behavioral - characterize the ways in which classes or objects interact and distribute responsibility
  (2) Scope - Specifies whether the pattern applies primarily to classes or to objects
    - class patterns - deal with relationships between classes and their subclasses which are established through inheritance so they are static - fixed at compile time
    - object patterns - deal with object relationships which can be changed at run-time and are more dynamic.

* Creational class pattern & object pattern -
  Creational class patterns defer some part of object creation to subclasses, while creational object patterns ~~use inheritance to compose~~ defer it to another object.

* Structural class patterns & object patterns -
  Structural class patterns use inheritance to compose classes, while the structural object patterns describe ways to assemble objects.

2.	What are the pitfalls, hints (or) techniques should a designer be aware, when implementing the design pattern? Explain.

- Design patterns solve many of the day-to-day problems object-oriented designers face, and in many different ways.
- problems & solutions using design patterns are as follows:

i) Finding Appropriate Objects:-

- Object-oriented programs (OOP) are made up of objects — which package both data and the procedures that operate on the data.
- Decomposing a system into objects is the hard part in object-oriented design.
- Many objects in a design come from the analysis model. But, object-oriented designs often end up with classes that have no counterparts in the real world.
- Strict modeling of the real world leads to a system that reflects today's realities but not necessarily tomorrow's.
- The abstractions that emerge during design are key to making a design flexible.
- Design patterns identify less obvious abstractions.

ii) Determining Object Granularity:-

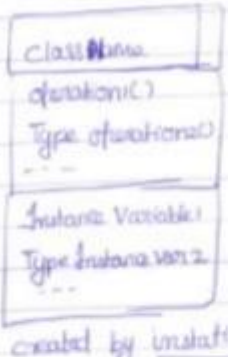
- Objects can vary tremendously in size and number.
- Design patterns address this issue.
  - Eg:- Facade pattern — describes how to represent subsystems as objects
    Flyweight pattern — describes how to support huge numbers of objects at the finest granularities

3) Specifying object interfaces:-

- Signature:- Every operation declared by an object specifies the operation's name, its parameters and return value. This is known as operation's signature.
- Interface:- The set of all signatures defined by an object's operations is called the interface to the object.
- Object's interface characterizes the complete set of requests that can be sent to the object. Any request that matches a signature in the object's interface may be sent to the object.
- Type:- Name used to denote a particular interface.
  Subtype & supertype — A subtype interface contains the interface of its supertype.
- Dynamic binding:- The run-time association of a request to an object and one of its operations is known as dynamic binding.
- polymorphism — It simplifies the definitions of clients, decouples objects from each other, and lets them vary their relationships to each other at run-time.
- Design patterns for interfaces:-
  - They define interfaces by identifying their key elements and the kinds of data that get sent across an interface.
  - They can specify relationships b/w interfaces.
  - Eg:- i) Memento pattern - describes how to encapsulate & save the internal state of an object so that the object can be restored to it's state later.

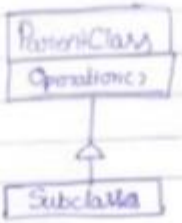
4) Specifying Object Implementation:

- An object's implementation is defined by its class.
- The class specifies the object's internal data and defines the operations the object can perform.



- Objects are created by instantiating a class i.e., an object is an instance of the class. This allocates storage for the object internal data and associates the operations with these data.

- Class inheritance — Defining new classes (Subclass) in terms of existing classes (Parent class).
  - This includes the definitions of all the data & operations that the parent class defines.
  - Objects that are instances of the subclasses will contain all data defined by the subclass and its parent classes and also perform all operations defined by this subclass and its parents.



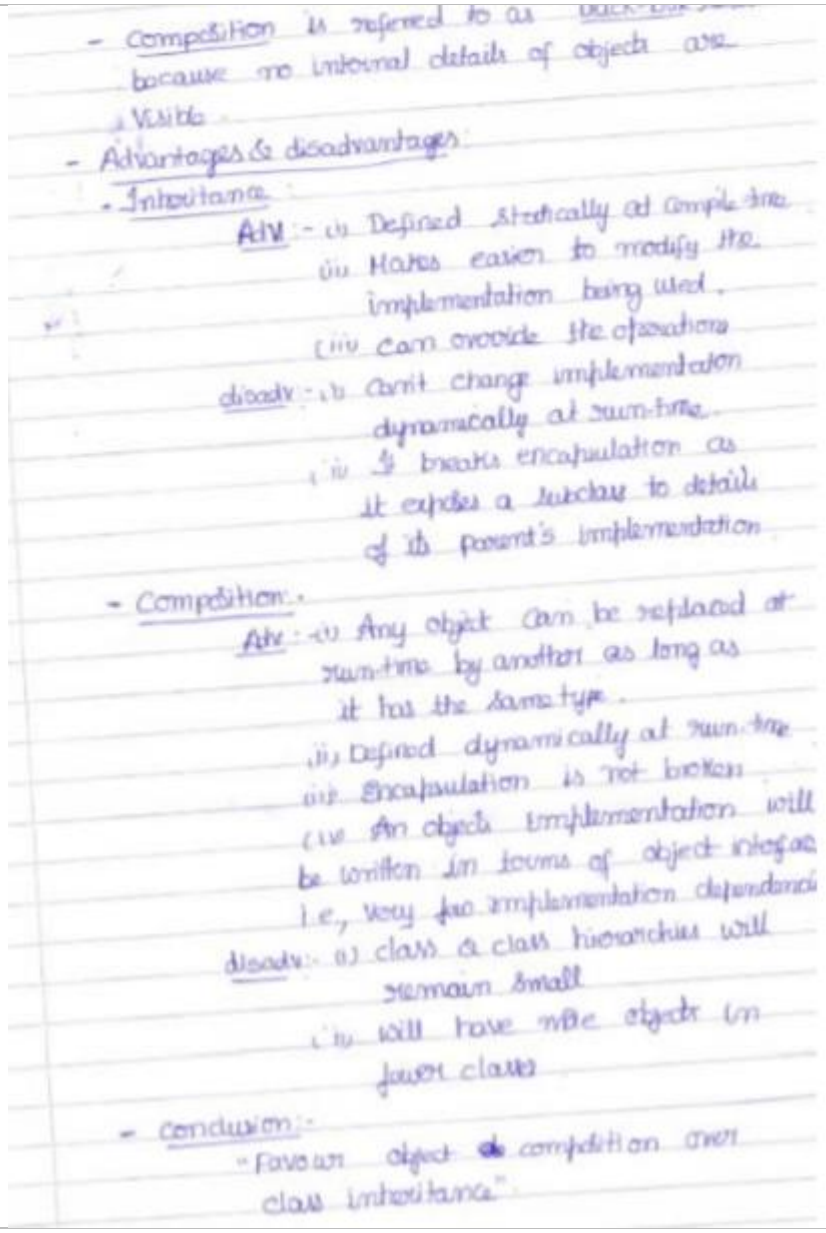
- Class Vs Interface Inheritance:
  - Class inheritance defines an object's implementation in terms of another object's implementation
  - Interface Inheritance describes when an object can be used in place of another.
  - Many of the design patterns depend on this distinction
  - Benefits:
    - Clients remain unaware of the specific types of objects they use
    - Clients remain unaware of the classes that implement these objects.

- Programming to an Interface, not an Implementation

- There are two benefits to manipulating objects solely in terms of the interface defined by abstract class:
  1) Clients remain unaware of the specific types of objects, they use, as long as the objects adhere to the interface that clients expect
  2) Clients remain unaware of the classes that implement these objects. Clients only know about the abstract class(es) defining the interface.
- These benefit reduces implementation dependencies between subsystems that leads to the following principle of reusable object-oriented design:
  "Program to an interface, not an implementation"

5) Putting Reuse Mechanisms to work:-

- Inheritance Vs Composition:-
  - Inheritance is often referred to as "white-box", as the internals of the parent classes are often visible to subclasses.



3.	Explain how Frameworks are different from design patterns in software architecture.

*Design Patterns in framework

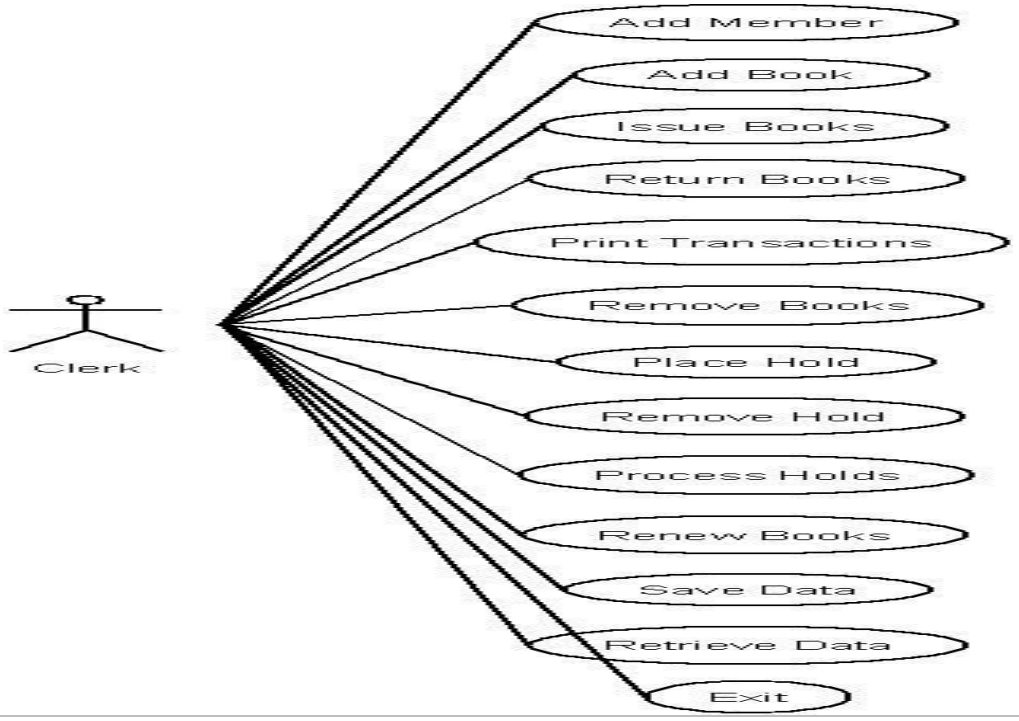
- A framework is a set of cooperating classes that make up a reusable design for a specific class of software
- framework can be customized to a particular application by creating application-specific subclasses of abstract classes from the framework
- Framework dictates the architecture of your application
- It emphasizes design reuse over code reuse
- In toolkit, we write the main body of the application and call the code we want to reuse. But in framework, we reuse the main body and write the code it calls
- Adv:- Build an application faster, easier to maintain and more consistent to their users.
- Mature frameworks usually incorporate several design patterns
- Differences b/w framework & design pattern
  a) Design patterns are more abstract than frameworks
  b) " " smaller architectural elements than framework
  c) " are less specialized than frameworks

Design patterns are most abstract than frameworks : Frameworks can be embodied in code, but only examples of patterns can be embodied in code. A strength of

frameworks is that they can be written down in programming languages and not only studied but executed and reused direcrtly. In contrast, the design pattern can be implemented, each time they are used. Design patterns also explain the intent, tradeoffs, and consequences of the design.
Design patterns are smaller architectural elements than framework : A typical framework contains several design patterns but the reverse is never true.
Design patterns are less specialized than frameworks : frameworks always have a particular application domain. A graphical editor framework might be used in factory simulation, but it won't be mistaken for a simulation framework. In contrast, the design patterns in this catalog, can be used in nearly any kind of application.

Frameworks are becoming increasingly common and important. They are the way Object-Oriented Systems achieve the most reuse.Larger Object-oriented applications will end up consisting of layers of frameworks that cooperate with each other. Most of the design and code in the application will come from or influednced by the frameworks it uses.

4. Design a Use Case diagram of Library System and explain any two use cases



Use-Case for Returning Books

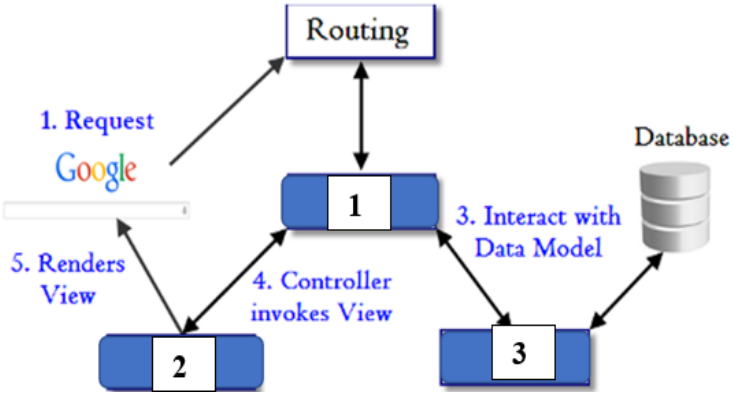
Actions performed by the actor	Responses from the system
1. The member arrives at the return counter with a set of books and gives the clerk the books.	
2. The clerk issues a request to return books.	
	3. The system asks for the identifier of the book.
4. The clerk enters the book identifier.	
	5. If the identifier is valid, the system marks that the book has been returned and informs the clerk if there is a hold placed on the book; otherwise it notifies the clerk that the identifier is not valid. It then asks if the clerk wants to process the return of another book.
6. The clerk notes whether there is a hold on the book and answers in the affirmative or in the negative. If there is a hold, the clerk sets the book aside.	
	7. If the answer is in the affirmative, the system goes to Step 3. Otherwise, it exits.

Use-Case for Issuing Books

Actions performed by the actor	Responses from the system
1. The member arrives at the check-out counter with a set of books and supplies the clerk with his/her identification number.	
2. Clerk issues a request to check out books.	
	3. The system asks for the user id.
4. Clerk inputs the user ID to the system.	
	5. If the ID is valid, the system asks for the ID of the book; otherwise it prints an appropriate message and exits the use-case.
6. The clerk inputs the identifier of a book that the user wants to check out.	
	7. If the ID is valid and the book is issuable to the member, the book is recorded as having been issued to the member; Member is recorded as having possession of the book, a due-date is generated and details of the transaction are displayed. If the book is not issuable, a suitable error message is displayed. The system asks if there are more books.
8. The clerk stamps the due-date, prints out the transaction (if needed) and replies positively or negatively.	
	9. If there are more books for checking out, the system goes back to Step 5; otherwise it exits.
10. The clerk stamps the due date and gives the user the books checked out. The customer leaves the counter.	

Section -B MCQs

1. MVC model for URL routing : Identfy Model, View, Controller blocks from the following figure



a) 1-Model,2-View,3-Controller

b) 1-View,2-Model,3-Controller

c) 1-Controller,2-View,3-Model

d) 1-View,2-Controller,3-Model

2. Match the following:

INTENT
I1. INTERFACES
I2. RESPONSIBILITY
I3. CONSTRUCTION
I4. OPERATIONS
I5. EXTENSIONS

PATTERNS
P1. Builder, Factory Method, Abstract Factory, Prototype, Memento
P2. Singleton, Observer, Mediator, Proxy, Chain of Responsibility, Flyweight
P3. Adapter, Facade, Composite, Bridge
P4. Decorator, Iterator, Visitor
P5. Template Method, State, Strategy, Command, Interpreter

a) I1-P1,I2-P2,I3-P3,I4-P4,I5-P5

b) I1-P3,I2-P4,I3-P1,I4-P2,I5-P5

c) I1-P3, I2-P2, I3-P1, I4-P5, I5-P4

d) I1-P1,I2-P3,I3-P5,I4-P2,I5-P4

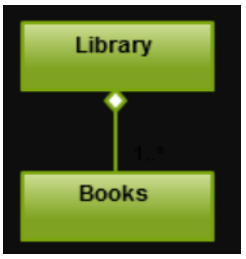
3. Aggregation (encapsulation) relationship is represented in UML notation by

a. Line with solid diamond at one end
b. Line with hollow diamond at one end
c. Line with an arrow at one end
d. Line without an arrow

4. Which diagram in UML is used to describe the physical components their distribution and association?

a. Component diagram
b. Object diagram
c. Deployment diagram

	d. Interction diagram
5.	In your opinion what is the best way to describe the relationship between these two classes: Customer, Address a. Customer is an Address b. Address is a Customer c. Customer has an Address d. Address has a Customer
6.	Which of the following statements is true? i. There are 5 views that are represented through the Unified Modelling Language (UML). ii. These 5 views in UML are represented through 9 UML diagrams. a. Only i is true b. Only ii is true c. Both i and ii are true d. None of them is true
7.	Which of the following is not a UML diagram? a. Class diagram b. Object Diagram c. Interface diagram d. Use case model
8.	The private members of the base class are visible in derived class but are not accessible directly. a) True b) False
9.	If a class is extending/inheriting another abstract class having abstract method, then _____ a) Either implementation of method or making class abstract is mandatory b) Implementation of the method in derived class is mandatory c) Making the derived class also abstract is mandatory d) It's not mandatory to implement the abstract method of parent class
10.	Which among the following best describes abstract classes? a) If a class has more than one virtual function, it's abstract class b) If a class have only one pure virtual function, it's abstract class c) If a class has at least one pure virtual function, it's abstract class d) If a class has all the pure virtual functions only, then it's abstract class
11.	Which of the following is a correct interface? a) interface A {void print() { }} b) abstract interface A { print(); } c) abstract interface A { abstract void print(); { }} d) interface A { void print(); }

12.	Given the following piece of code: public class School {public abstract double numberOfStudent(); which of the following statements is true? a) The keywords public and abstract cannot be used together. b) The method numberOfStudent() in class School must have a body. c) You must add a return statement in method numberOfStudent(). d) Class School must be defined abstract.
13.	What kind of Association is the following UML class diagram?  a) Aggregation b) Composition c) Inheritance d) Realization
14.	When would you use the GOF Factory method design pattern? a. To ensure that a certain group of related objects are used together b. To control the creation of an object based on a established interface c. To allow the concrete implementation to determine the subclass to construct. d. To abstract steps of construction of complex objects
15.	List the ways in which design pattern solve design problems. a)Finding appropriate Objects b) Determining object granularity c) Specifying object interfaces d) Specifying object implementation e) Putting reuse mechanism to work f) Relating Run-Time and Compile-Time structures g) Designing for change

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CO's to PO's & PSO's mapping

Name of the course	:	**Software Architecture & Design Patterns**	Sub Code	:	17IS72
Name of the Faculty/s	:	Mrs. Vidya.U	Sem & Sec	:	7thA,B

	Course Outcomes	Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO 1	Design and implement codes with higher performance and lower complexity	**2**	2	2	1	-	-	-	-	-	-	-	-	-	1	-	-	-
CO 2	Be aware of code qualities needed to keep code flexible	**1**	2	2	-	-	-		-	-	-	-	-	-	1	-	-	-
CO 3	Experience core design principles and be able to assess the quality of a design with respect to these principles	**2,5**	2	2	2	-	-	-	-	-	-	-	-	-	1	-	2	-
CO 4	Capable of applying these principles in the design of object oriented system	**4,5**	2	-	2	2	-	-	-	-	-	-	-	-	1	-	-	-
CO 5	Demonstrate an understanding of a range of design patterns. Be capable of comprehending a design presented using this vocabulary.	**3**	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-
CO 6	Be able to select and apply suitable patterns in specific contexts	**3,4**	2	2	-	-	-	-	-	-	-	-	-	-	1	-	-	-

PROGRAM OUTCOMES(PO), PROGRAM SPECIFIC OUTCOMES(PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		

PSO 1	Implement and maintain enterprise solutions using latest technologies.
PSO 2	Develop and simulate wired and wireless NW protocols for various network applications using modern tools.
PSO 3	Apply the knowledge of Information technology and software testing to maintain legacy systems.
PSO 4	Apply knowledge of web programming and design to develop web based applications using database and other technologies