

## Design Pattern

### Unit: I

#### Introduction to Design Pattern


Course Details  
(B. Tech. 5<sup>th</sup> Sem)



Ibrar Ahmed  
(Asst. Professor)  
CSE Department



# Faculty Introduction

<b>Name</b>	<b>Ibrar Ahmed</b>	
<b>Qualification</b>	M. Tech. (Computer Engineering)	
<b>Designation</b>	Assistant Professor	
<b>Department</b>	Computer Science & Engineering	
<b>Total Experience</b>	4 years	
<b>NIET Experience</b>	1 years	
<b>Subject Taught</b>	Design & Analysis of Algorithm, Data Structures, Artificial Intelligence, Soft Computing, C Programming, Web Technology, Discrete Mathematics.	

# Evaluation Scheme

## B. TECH (CSE) Evaluation Scheme

Session 2020-21	Third Year	SEMESTER V							ESC (3)	PCC (14)	ELC (6)	PW (1)			
Sl. No.	Subject code	Subject	Periods			Evaluation Schemes				End Semester		Total	Credit	Course Type	
			L	T	P	CT	TA	TOTAL	PS	TE	PE				
1		Design Thinking -II	2	1	0	30	20	50		100		150	3	ESC	
2	20CS501	Database Management System	3	1	0	30	20	50		100		150	4	PCC	
3	20CS502	Web Technology	3	0	0	30	20	50		100		150	3	PCC	
4	20CS503	Compiler Design	3	1	0	30	20	50		100		150	4	PCC	
5		Python Web development with Django Design Pattern	3	0	0	30	20	50		100		150	3	ELC	
6			3	0	0	30	20	50		100		150	3	ELC	
7	P20CS501	Database Management System Lab	0	0	2				25		25	50	1	PCC	
8	P20CS502	Web Technology Lab	0	0	2				25		25	50	1	PCC	
9	P20CS503	Compiler Design Lab	0	0	2				25		25	50	1	PCC	
10		Internship Assessment	0	0	2				50			50	1	PW	
11		Constitution of India / Essence of Indian Traditional Knowledge	2	0	0	30	20	50		50		100	0	NC	
12		MOOCs for Honors degree													

## UNIT-I: Introduction of Design Pattern

Describing Design Patterns, Design Patterns in Smalltalk MVC, The Catalogue of Design Patterns, Organizing The Catalog, How Design Patterns solve, Design Problems, How to Select a Design pattern, How to Use a Design Pattern. Principle of least knowledge.

## UNIT-II: Creational Design Pattern

A Case Study: Designing a Document Editor

Creational Patterns: Abstract Factory, Builder , Factory Method, Prototype , Singleton Pattern,

## UNIT-III: Structural Design Pattern

Structural Pattern Part-I, Adapter, Bridge, Composite.

Structural Pattern Part-II, Decorator, Facade, Flyweight, Proxy.

## UNIT-IV: Behavioral Design Patterns Part: I

Behavioral Patterns Part: I, Chain of Responsibility, Command, Interpreter, Iterator Pattern.

Behavioral Patterns Part: II, Mediator, Memento, Observer, Patterns.

## UNIT-V: Behavioral Design Patterns Part: II

Behavioral Patterns Part: III, State, Strategy, Template Method, Visitor, What to Expect from Design Patterns.



# Branch Wise Application

1. Real time web analytics
2. Digital Advertising
3. E-Commerce
4. Publishing
5. Massively Multiplayer Online Games
6. Backend Services and Messaging
7. Project Management & Collaboration
8. Real time Monitoring Services
9. Live Charting and Graphing
10. Group and Private Chat

# Course Objective

In this semester, the students will

Study how to shows relationships and interactions between classes or objects..

Study to speed up the development process by providing well-tested, proven development/design paradigms.

Select a specific design pattern for the solution of a given design problem.

Create a catalogue entry for a simple design pattern whose purpose and application is understood.

# Course Outcomes (COs)

**At the end of course, the student will be able to:**

**CO1 : Construct a design consisting of collection of modules.**

**CO2 : Exploit well known design pattern such as Factory, visitor etc.**

**CO3 : Distinguish between different categories of design patterns.**

**CO4 : Ability to common design pattern for incremental development.**

**CO5 : Identify appropriate design pattern for a given problem and design the software using pattern oriented architecture.**

# Program Outcomes (POs)

**Engineering Graduates will be able to:**

**PO1 : Engineering Knowledge**

**PO2 : Problem Analysis**

**PO3 : Design/Development of solutions**

**PO4 : Conduct Investigations of complex problems**

**PO5 : Modern tool usage**

**PO6 : The engineer and society**

# Program Outcomes (POs)

**Engineering Graduates will be able to:**

**PO7 : Environment and sustainability**

**PO8 : Ethics**

**PO9 : Individual and teamwork**

**PO10 : Communication**

**PO11 : Project management and finance**

**PO12 : Life-long learning**

# COs - POs Mapping

CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	2	3	3	-	-	-	-	-	-	-
CO2	3	2	3	2	3	-	-	-	-	-	-	-
CO3	3	2	3	2	3	-	-	-	-	-	-	-
CO4	3	2	3	2	3	-	-	-	-	-	-	-
CO5	3	2	3	3	3	-	-	-	-	-	-	-
AVG	2.8	2.0	2.8	2.4	3.0	-	-	-	-	-	-	-

# Program Specific Outcomes(PSOs)

S. No.	Program Specific Outcomes (PSO)	PSO Description
1	PSO1	Understand to shows relationships and interactions between classes or objects of a pattern.
2	PSO2	Study to speed up the development process by providing well-tested, proven development
3	PSO3	Select a specific design pattern for the solution of a given design problem
4	PSO4	Create a catalogue entry for a simple design pattern whose purpose and application is understood.

# COs - PSOs Mapping

CO.K	PSO1	PSO2	PSO3	PSO4
CO1	3	-	-	-
CO2	3	3	-	-
CO3	3	3	-	-
CO4	3	3	-	-
CO5	3	3	-	-



# Program Educational Objectives (PEOs)

Program Educational Objectives (PEOs)	PEOs Description
PEOs	To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and provide sustainable solutions for real-life problems using state-of-the-art technologies.
PEOs	To have a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors and to face the global challenges.
PEOs	To have an effective communication skills, professional attitude, ethical values and a desire to learn specific knowledge in emerging trends, technologies for research, innovation and product development and contribution to society.
PEOs	To have life-long learning for up-skilling and re-skilling for successful professional career as engineer, scientist, entrepreneur and bureaucrat for betterment of society.

# Result Analysis(Department Result & Subject Result & Individual result

Name of the faculty	Subject code	Result % of clear passed
Mr. Sanjay Nayak		

# Pattern of Online External Exam Question Paper (100 marks)

Printed page: ....

Subject Code: .....

Roll No: 

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

**NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA**

**(An Autonomous Institute Affiliated to AKTU, Lucknow)**

**B.Tech./MBA/MCA/M.Tech (Integrated)**

**(SEM:.....THEORY EXAMINATION(2020-2021))**

**Subject .....**

**Time: 2 Hours**

**Max. Marks: 100**

# Pattern of Online External Exam Question Paper (100 marks)

		<b>SECTION – A</b>	<b>[30]</b>	<b>CO</b>
<b>1.</b>	<b>Attempt all parts- (MCQ, True <u>False</u>)</b>	<b>Three Question From Each Unit</b>	<b>[15×2=30]</b>	
		<b>UNIT-1</b>		
	<b>1-a.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-b.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-c.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
		<b>UNIT-2</b>		
	<b>1-d.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-e.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-f.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
		<b>UNIT-3</b>		
	<b>1-g.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-h.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-i.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
		<b>UNIT-4</b>		
	<b>1-j.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-k.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-l.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
		<b>UNIT-5</b>		
	<b>1-m.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-n.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>1-o.</b>	<b><u>Question-</u></b>	<b>(2)</b>	

# Pattern of Online External Exam Question Paper (100 marks)

		<b><u>SECTION – B</u></b>	<b>[20×2=40]</b>	<b>CO</b>
<b>2.</b>	<b>Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in Glossary) <u>Question</u> from Unseen passage - Four Question From Unit-I</b>		<b>[4×2=08]</b>	<b>CO</b>
	<b>Glossary- (Required words to be written)</b>			
	<b>2-a.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>2-b.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>2-c.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>2-d.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
<b>3.</b>	<b>Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in Glossary) <u>Question</u> from Unseen passage - Four Question From Unit-II</b>		<b>[4×2=08]</b>	<b>CO</b>
	<b>Glossary- (Required words to be written)</b>			
	<b>3-a.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>3-b.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>3-c.</b>	<b><u>Question-</u></b>	<b>(2)</b>	
	<b>3-d.</b>	<b><u>Question-</u></b>	<b>(2)</b>	

# Pattern of Online External Exam Question Paper (100 marks)

<b>4.</b>	<b>Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in <u>Glossary</u>) Question from Unseen passage - Four Question From Unit-III</b>		<b>[4×2=08]</b>	<b>CO</b>
	<b>Glossary- (Required words to be written)</b>			
<b>4-a.</b>	<u>Question-</u>		(2)	
<b>4-b.</b>	<u>Question-</u>		(2)	
<b>4-c.</b>	<u>Question-</u>		(2)	
<b>4-d.</b>	<u>Question-</u>		(2)	
<b>5.</b>	<b>Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in <u>Glossary</u>) Question from Unseen passage - Four Question From Unit-IV</b>		<b>[4×2=08]</b>	<b>CO</b>
	<b>Glossary- (Required words to be written)</b>			
<b>5-a.</b>	<u>Question-</u>		(2)	
<b>5-b.</b>	<u>Question-</u>		(2)	
<b>5-c.</b>	<u>Question-</u>		(2)	
<b>5-d.</b>	<u>Question-</u>		(2)	
<b>6.</b>	<b>Attempt all Four parts. Fill in The Blanks, Match the pairs (From the Data Given in <u>Glossary</u>) Question from Unseen passage - Four Question From Unit-V</b>		<b>[4×2=08]</b>	<b>CO</b>
	<b>Glossary- (Required words to be written)</b>			
<b>6-a.</b>	<u>Question-</u>		(2)	
<b>6-b.</b>	<u>Question-</u>		(2)	
<b>6-c.</b>	<u>Question-</u>		(2)	
<b>6-d.</b>	<u>Question-</u>		(2)	

# Pattern of Online External Exam Question Paper (100 marks)

<b>SECTION – C</b>				
<b>7</b>	<b>Answer any 10 out of 15 of the following, Subjective Type Question, Three Question from Each Unit</b>		<b>[10×3=30]</b>	<b>CO</b>
		<b>UNIT-1</b>		
	7-a.	<u>-Question-</u>	(3)	
	7-b.	<u>-Question-</u>	(3)	
	7-c.	<u>-Question-</u>	(3)	
		<b>UNIT-2</b>		
	7-d.	<u>-Question-</u>	(3)	
	7-e.	<u>-Question-</u>	(3)	
	7-f.	<u>-Question-</u>	(3)	
		<b>UNIT-3</b>		
	7-g.	<u>-Question-</u>	(3)	
	7-h.	<u>-Question-</u>	(3)	
	7-i.	<u>-Question-</u>	(3)	
		<b>UNIT-4</b>		
	7-j.	<u>-Question-</u>	(3)	
	7-k.	<u>-Question-</u>	(3)	
	7-l.	<u>-Question-</u>	(3)	
		<b>UNIT-5</b>		
	7-m.	<u>-Question-</u>	(3)	
	7-n.	<u>-Question-</u>	(3)	
	7-o.	<u>-Question-</u>	(3)	

- Student should have knowledge of object oriented analysis and design.
- Knowledge of Data structure and algorithm.
- knowledge of Programing language such as C/C++ etc.
- Good problem solving Skill .



## YouTube /other Video Links

- <https://youtu.be/rI4kdGLaUiQ?list=PL6n9fhu94yhUbctIoxoVTrkIN3LMwTCmd>
- <https://youtu.be/v9ejT8FO-7I?list=PLrhzvIcii6GNjpARdnO4ueTUAVR9eMBpc>
- <https://youtu.be/VGLjQuEQgkI?list=PLt4nG7RVVv1h9lxOYSOGI9pcP3I5oblbx>

- Describing Design Patterns.
- Design Patterns in Smalltalk MVC.
- The Catalogue of Design Patterns.
- Organizing The Catalog.
- How Design Patterns solve.
- Design Problems.
- How to Select a Design pattern.
- How to Use a Design Pattern.

# Unit I Objective

In Unit I, the students will be able to find

- Definitions of terms and concepts.
- The idea of a pattern.
- The origins of design patterns.
- Patterns in software design.
- Scope of development activity: applications, toolkits, frameworks.
- The Portland Pattern Repository.

## Topic : Describing design patterns

- In this topic, the students will gain , The idea of a pattern and the definition of terms and concepts, what are the role of patterns and how they are used in software design.

# Design Pattern - Overview

- Design patterns represent the best practices used by experienced object-oriented software developers.
- Design patterns are solutions to general problems that software developers faced during software development.
- These solutions were obtained by trial and error by numerous software developers over quite a substantial period of time.

## What is Gang of Four (GOF)?

- In 1994, four authors Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides published a book titled **Design Patterns - Elements of Reusable Object-Oriented Software** which initiated the concept of Design Pattern in Software development.
- These authors are collectively known as Gang of Four (GOF). According to these authors design patterns are primarily based on the following principles of object orientated design.

## principles of object orientated design:-

- Program to an interface not an implementation.
- Favor object composition over inheritance.

## Usage of Design Pattern:

Design Patterns have two main usages in software development.

### 1. Common platform for developers:-

Design patterns provide a standard terminology and are specific to particular scenario. For example, a singleton design pattern signifies use of single object so all developers familiar with single design pattern will make use of single object and they can tell each other that program is following a singleton pattern.



## Usage of Design Pattern:

Design Patterns have two main usages in software development.

### 2. Best Practices:-

Design patterns have been evolved over a long period of time and they provide best solutions to certain problems faced during software development. Learning these patterns helps unexperienced developers to learn software design in an easy and faster way.

## Types of Design Patterns:-

- As per the design pattern reference book **Design Patterns - Elements of Reusable Object-Oriented Software** , there are 23 design patterns which can be classified in three categories: Creational, Structural and Behavioral patterns.
- We'll also discuss another category of design pattern: J2EE design patterns.

## Types of Design Patterns:-

### 1. Creational Patterns:-

These design patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new operator. This gives program more flexibility in deciding which objects need to be created for a given use case.

## Types of Design Patterns:-

### 2. Structural Patterns:-

These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.

## Types of Design Patterns:-

### 3. Behavioral Patterns:-

These design patterns are specifically concerned with communication between objects.

### 4. J2EE Patterns:-

These design patterns are specifically concerned with the presentation tier. These patterns are identified by Sun Java Center.

## Topic : Design Patterns in Smalltalk MVC

- In this topic, the students will gain , The idea of design pattern in Smalltalk MVC, they understand what are the model ,view and controller defined by small talk.

# Design Patterns in Smalltalk MVC

Smalltalk MVC is defined in Design Pattern as:-

MVC Consists of three kinds of objects. The **Model** is the application object, the **View** is its screen presentation, and the **Controller** defines the way the user interface reacts to user input.

MVC (I'll always use that term to refer to the Smalltalk form) has the following structure:

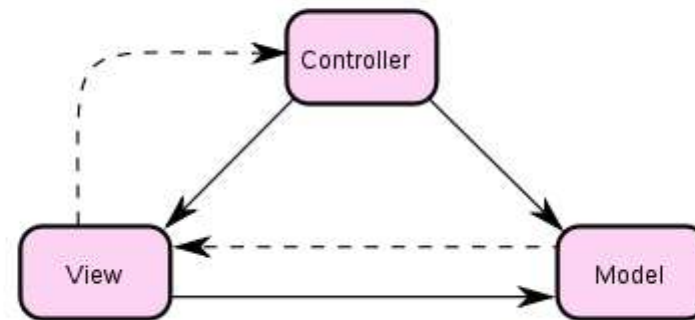
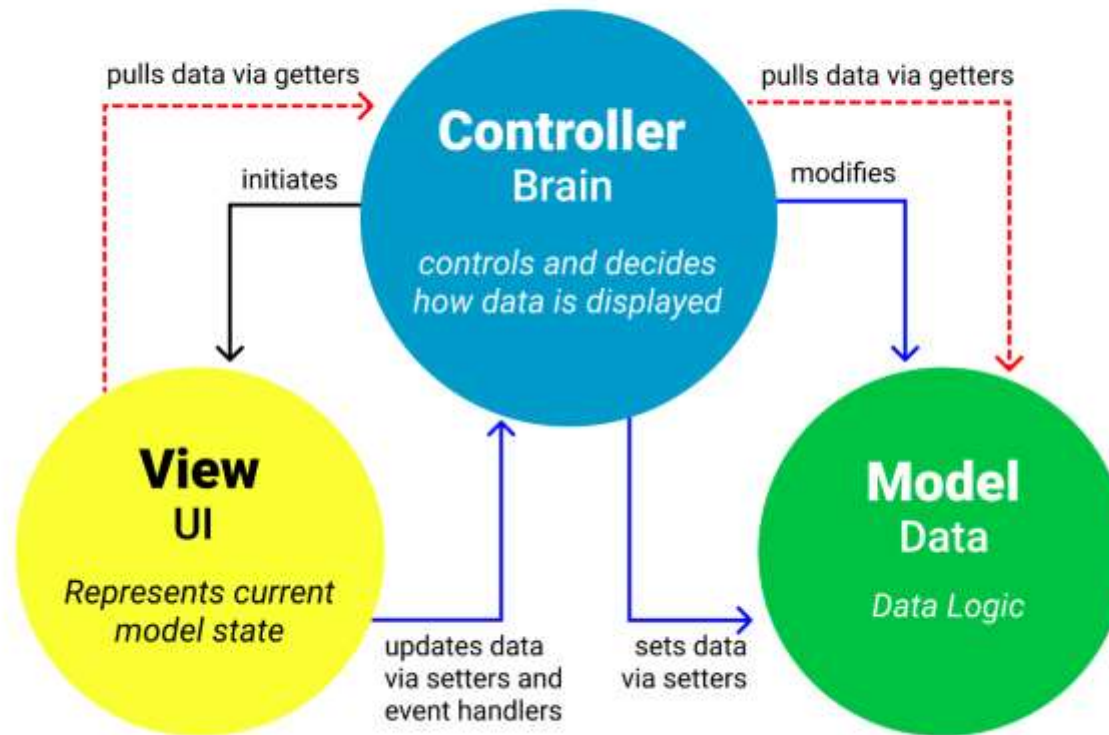


figure: Smalltalk MVC<sup>4</sup>

## MVC Architecture Pattern





MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns:-

Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

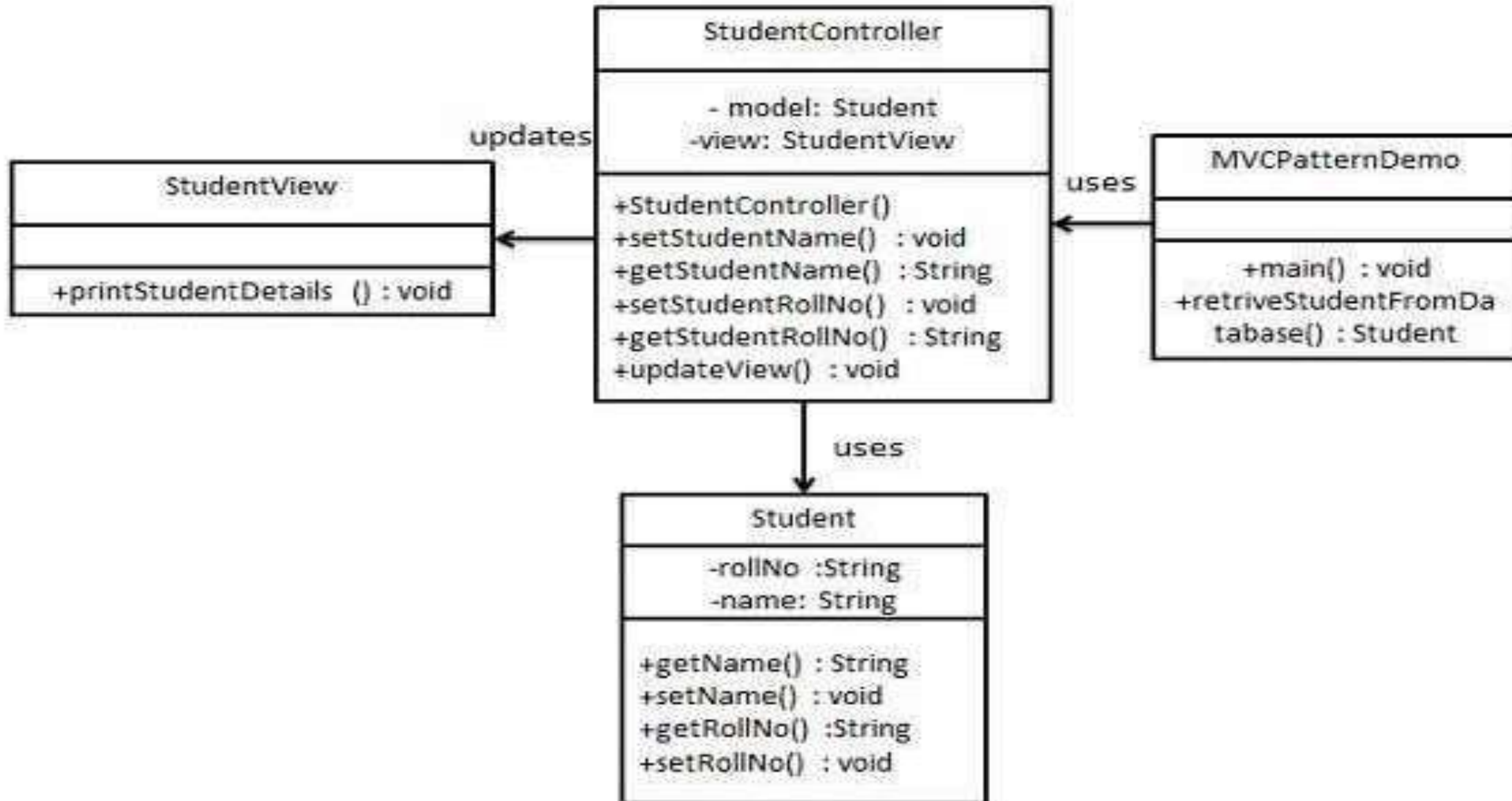
View - View represents the visualization of the data that model contains.

Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

## Implementation:-

- We are going to create a Student object acting as a model. Student View will be a view class which can print student details on console and Student Controller is the controller class responsible to store data in Student object and update view Student View accordingly.
- MVC Pattern Demo, our demo class, will use Student Controller to demonstrate use of MVC pattern.

# Design Patterns in Smalltalk MVC Cont.....



## Step 1 :-

Create Student object Model

(Student.java)

```
public class Student {  
    private String rollNo;  
    private String name;  
  
    public String getRollNo() {  
        return rollNo;  
    }  
  
    public void setRollNo(String rollNo) {  
        this.rollNo = rollNo;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

## Step 2:- Create View (StudentView.java)

```
public class StudentView {  
    public void printStudentDetails(String studentName, String studentRollNo){  
        System.out.println("Student: ");  
        System.out.println("Name: " + studentName);  
        System.out.println("Roll No: " + studentRollNo);  
    }  
}
```

## Step 3:-

### Create Controller (StudentController.java)

```
public class StudentController {  
    private Student model;  
    private StudentView view;  
  
    public StudentController(Student model, StudentView view){  
        this.model = model;  
        this.view = view;  
    }  
}
```

## Step 3:-

Create Controller

(StudentController.java)

```
public void setStudentName(String name){  
    model.setName(name);  
}  
  
public String getStudentName(){  
    return model.getName();  
}  
  
public void setStudentRollNo(String rollNo){  
    model.setRollNo(rollNo);  
}  
  
public String getStudentRollNo(){  
    return model.getRollNo();  
}  
  
public void updateView(){  
    view.printStudentDetails(model.getName(), model.getRollNo());  
}  
}
```

## Step 4:-

Use the Student Controller methods to demonstrate MVC design pattern (MVCPatternDemo.java)

```
public class MVCPatternDemo {  
    public static void main(String[] args) {  
  
        //fetch student record based on his roll no from the database  
        Student model = retrieveStudentFromDatabase();  
  
        //Create a view : to write student details on console  
        StudentView view = new StudentView();  
  
        StudentController controller = new StudentController(model, view);  
  
        controller.updateView();  
    }  
}
```



## Step 4:-

Use the Student Controller  
(MVCPatternDemo)

```
//update model data
controller.setStudentName("John");

controller.updateView();
}

private static Student retrieveStudentFromDatabase(){
    Student student = new Student();
    student.setName("Robert");
    student.setRollNo("10");
    return student;
}
}
```

## Step 5

Verify the output.

```
Student:  
Name: Robert  
Roll No: 10  
Student:  
Name: John  
Roll No: 10
```

## Topic : The Catalog of Design Patterns

- In this topic, the students will gain , The idea of design pattern in The catalog grouped by intent, complexity, and popularity. The catalog contains all classic design patterns and several architectural Patterns that are used in software engineering.

# The Catalog of Design Patterns

- The Design Patterns are organized into a form of a catalog.
- These Design Patterns collectively assist in software engineering by finding objects, specifying objects implementations, objects interfaces, determining objects granularity, implementing reuse mechanisms, etc. The Intents specify what the design pattern does.
- Some of the patterns with their names and intents are as follows

- **Abstract Factory:** It Indicates what factory is to be instantiated, provides an interface to create families of objects(related / dependent) without any specification of their concrete classes.
- **Adaptor:** It Adapt or converts an interface of a class into another one according to the client expectation and hence, overcomes the problem of incompatible interfaces thereby enabling the classes to work together.
- **Bridge:** It Separates abstraction from its implementation to make them independent.

- **Builder:** It Separates the complex objects constructions from their representation in order to create different representations with the same construction process.
- **Chain Of Responsibility:** It Enables the handling of command objects by passing them to other objects by using the logic present in the processing of objects. In other words, Its decouples sender and receiver by formatting a chain of receiving objects to pass the request until the request is handled by an object.
- **Command:** It encapsulates the action and its parameters and hence, enables to parameterize the different requests of the clients such as long or queue requests. It also assists undoable operations.

- **Composite:** It represents the objects in a tree structure where each object represents the same interface. This enables clients to treat individual objects and their compositions uniformly.
- **Decorator:** It Adds additional functionality to a class at runtime. This Enables flexibility to subclass for adding functionality.
- **Facade:** It creates a simplified/unified interface of existing interfaces in the subsystems so as to handle common tasks easily.
- **Factory Method:** It Focuses on objects creation of specific implementation. lets the subclass decide as to which class to be instantiated.

- **Flyweight:** It Performs sharing of common objects properties by a large number of objects to save space.
- **Interpreter:** It Deals with the implementation of a specified computer language that solves specific problems. It interprets sentences in language by representing the grammar of language along with an interpreter.
- **Iterator:** It Enables sequential aggregate objects elements by hiding their underlying representations.
- **Mediator:** It provides a unified interface to the set of interfaces in a subsystem. It provides loose coupling which enables objects to refer to each explicitly and also varies objects interaction independently.



**Memento:** It supports the rollback mechanism by enabling the objects, to restore to their previous state without violation of encapsulation.

**Observer:** Whenever an object changes its state, it raises an event that notifies other objects and updates them automatically. This defines a one-to-many dependency between the objects.

**Prototype:** Here Prototypical instance determines the type of objects to be created. Further new objects are created by cloning this prototype.

**proxy:** It provides an illusion by applying placeholder to other objects in order to have control over it.

**Singleton:** It Provides restrictions on instantiating a class to a single object and also makes it globally accessible.

**State:** It Permits an alteration in the object's behavior with alteration in its state that is allows objects type to change at runtime.

**Visitors:** It Describes the skeleton of a program, enables subclasses to define some steps of the algorithm, and also to redefine certain steps without affecting the structure of the algorithm.

**Strategy:** It defines the Family of algorithms and their selection based upon the clients.

## Topic : Organizing the Catalog

- In this topic, the students will learn how to organized the catalog As there are many design patterns, So they can organize or classify patterns to learn them faster.

# Organizing the Catalog

- Design patterns vary in their granularity and level of abstraction. Because there are many design patterns, we need a way to organize them. This section classifies design patterns so that we can refer to families of related patterns. The classification helps you learn the patterns in the catalog faster, and it can direct efforts to find new patterns as well.
- We classify design patterns by two criteria. The first criterion, called purpose, reflects what a pattern does. Patterns can have either creational, structural, or behavioral purpose. Creational patterns concern the process of object creation. Structural patterns deal with the composition of classes or objects. Behavioral patterns characterize the ways in which classes or objects interact and distribute responsibility.

# Organizing the Catalog

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

# Organizing the Catalog

Purpose / Scope		Purpose		
		Creational (5)	Structural (7)	Behavioral (11)
Scope	Class	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Topic : Solving real world problems with design patterns , selection and usage of design patterns.

- In this topic, the students will learn how to solve real world problems with design patterns, what are the approaches and Patterns that help in real world problem , how to choose best pattern and what are uses of the patterns.

# Solving real world problems with design patterns

Design patterns are conceptual solutions to solve common redundant problems in software engineering. However, learning them is not easy as literature or tutorials on the Internet often introduce them with theoretical examples. This talk gives you a slightly different approach by introducing some of the most useful design patterns with practical code samples to solve real world problems.

1. **Creational Patterns:** Abstract Factory and Simple Factory
2. **Structural Patterns:** Adapter and Composite
3. **Behavioral Patterns:** Mediator



# Selection and Usage of design patterns

- As there are 23 patterns in the catalog it will be difficult to choose a design pattern. It might become hard to find the design pattern that solves our problem, especially if the catalog is new and unfamiliar to you.
- Below is a list of approaches we can use to choose the appropriate design pattern

## 1. Consider how design patterns solve design problems:

Considering how design patterns help you find appropriate objects, determine object granularity, specify object interfaces and several other ways in which design patterns solve the problems will let you choose the appropriate design pattern.

## 2. Scan intent sections:

Looking at the intent section of each design pattern's specification lets us choose the appropriate design pattern.

## 3. Study how patterns interrelate:

The relationships between the patterns will direct us to choose the right patterns or group of patterns.

## 4. Study patterns of like purpose:

Each design pattern specification will conclude with a comparison of that pattern with other related patterns. This will give you an insight into the similarities and differences between patterns of like purpose.

## 5. Examine a cause of redesign:

Look at your problem and identify if there are any causes of redesign. Then look at the catalog of patterns that will help you avoid the causes of redesign.

## 6. Consider what should be variable in your design:

Consider what you want to be able to change without redesign. The focus here is on encapsulating the concept that varies.

# How to use a design patterns

Below steps will show you how to use a design pattern after selecting one:

**1. Read the pattern once through for a overview:**

Give importance to the applicability and consequences sections to ensure that the pattern is right for your problem.

**2. Study the structure, participants and collaborations sections:**

Make sure to understand the classes and objects in the pattern and how they relate to one another.

**3. Look at the sample code section to see a concrete example of the pattern in action:**

Studying the code helps us to implement the pattern.

## 4. Choose names for pattern participants that are meaningful in the application context:

The names in the design patterns are too abstract to be directly used in the application. Include the participant name into the name that appears in the application which makes the pattern implementation explicit. For example, if you use strategy pattern for developing UI, use UIStartegy or GridStrategy as class names.

## 5. Define the classes:

Declare their interfaces, inheritance relationships and instance variables, which represent the data and objet references. Identify the effected classes by the pattern and modify them accordingly.

## 6. Define application-specific names for the operations in the pattern:

Use the responsibilities and collaborations as a guide to name the operations. Be consistent with the naming conventions. For example you can always prefix an operation with “Create-” to denote a factory method.

## 7. Implement the operations to carry out the responsibilities and collaborations in the pattern:

The implementation section provides hints to guide us in the implementation. The examples in the sample code section can also help as well.

# Principle of Least Knowledge

- The Principle of Least knowledge:- also known as **The law of Demeter**, or more precisely, the Law of Demeter for Functions/Methods (LoD-F) is a design principle which provides guidelines for designing a system with minimal dependencies. It is typically summarized as “Only talk to your immediate friends.”
- What this means is a client should only have knowledge of an objects members, and not have access to properties and methods of other objects via the members. To put it in simple terms you should only have access to the members of the object, and nothing beyond that. Think if it like this: if you use more than 1 dot you are violating the principle.

# Principle of Least Knowledge

- A perfect example of The Principle of Least Knowledge is in a Cairngorm Model Locator implementation. The Cairngorm Model Locator violates the Principle of least knowledge for good reason – it simply would not be practical to write wrapper methods for every object on the Model Locator. This is the main drawback of the Principle of least Knowledge; the need to create wrapper methods for each object, which are more formally known as Demeter Transmogrifies.
- The goal of good software design is to minimize dependencies, and by carefully following the guidelines provided by The Principle of Least Knowledge this becomes much easier to accomplish.



- **SOLID Principles:**
- **SOLID principles** were the Object-Oriented principles introduced by Robert C. Martin in his paperwork “Design Principles and Design patterns” in the year 2000. The acronym for SOLID goes as follows:
- **S - Single Responsibility Principle (SRP):** The single responsibility principle ensures that every class or module should be accountable and responsible for only one functionality. There should be one and only one reason for changing any class.
- **O - Open Closed Principle (OCP):** Every class is open for extension but closed for modification. Here, we are allowed to extend the entities behaviour by not modifying anything in the existing source code.
- **L - Liskov Substitution Principle (LSP):** LSP principle states that the objects can be replaced by the subtype instances without affecting the correctness of the program.
- **I - Interface Segregation Principle (ISP):** The ISP principle states that we can use as many interfaces specific to the client’s requirements instead of creating only one general interface. Clients should not be forced to implement the functionalities that they do not require.
- **D - Dependency Inversion Principle:** Here, the high-level modules should not be dependent on the lower level modules or concrete implementations. Instead, they should be dependent on the abstractions

## **Q 1 - What is Gang of Four (GOF)?**

- A** - Four authors of Book 'Design Patterns - Elements of Reusable Object-Oriented Software' are known as Gang of Four (GOF).
- B** - Gang of Four (GOF) is a name of a book on Design Patterns.
- C** - Gang of Four (GOF) is a Design Pattern.
- D** - None of the above.

## **Q 2 - Event handling frameworks like swing, awt use Observer Pattern?**

- A** - false
- B** – true

## **Q 3 - Which of the following pattern a request is wrapped under an object as command and passed to invoker object?**

- A** - Proxy Pattern
- B** - Chain of Responsibility Pattern
- C** - Command Pattern
- D** - Interpreter Pattern

**Q 4 - Which of the following describes the Command pattern correctly?**

- A - In this pattern a class represents functionality of another class.
- B - This pattern creates a chain of receiver objects for a request.
- C - This pattern provides a way to evaluate language grammar or expression.
- D - In this pattern a request is wrapped under an object as command and passed to invoker object.

**Q 5 - Which of the following describes the MVC pattern correctly?**

- A - In this pattern, a visitor class is used which changes the executing algorithm of an element class.
- B - This pattern is used to separate application's concerns.
- C - This pattern is used to decouple presentation tier and business tier.
- D - This pattern is used in EJB persistence mechanism.

1. What is Software Design Patterns.
2. What is the Classifying Relationships between Software Design Patterns.
3. Define textual Analysis Technique
4. Elaborate Structural Modeling in detail.
5. Discuss GRASP with example.

## YouTube /other Video Links

- <https://youtu.be/rI4kdGLaUiQ?list=PL6n9fhu94yhUbctIoxoVTrkIN3LMwTCmd>
- <https://youtu.be/v9ejT8FO-7I?list=PLrhzvIcii6GNjpARdnO4ueTUAVR9eMBpc>
- <https://youtu.be/VGLjQuEQgkI?list=PLt4nG7RVVv1h9lxOYSOGI9pcP3I5oblbx>

**1. Design patterns are divided into three fundamental groups.**

- ☐ Behavioral Patterns
- ☐ Creational Patterns
- ☐ Structural Patterns
- ☐ J2EE Patterns

**2. Four authors \_\_\_\_\_ are collectively known as Gang of Four (GOF).**

- ☐ Erich Gamma
- ☐ Richard Helm
- ☐ Ralph Johnson
- ☐ John Vlissides
- ☐ Gavin King

**3. Which design pattern provides a single class which provides simplified methods required by client and delegates call to those methods?**

- ☐ Adapter pattern
- ☐ Builder pattern
- ☐ Facade pattern
- ☐ Prototype pattern

**4. Which design pattern suggests multiple classes through which request is passed and multiple but only relevant classes carry out operations on the request?**

- ☐ Singleton pattern
- ☐ Chain of responsibility pattern
- ☐ State pattern
- ☐ Bridge pattern

## Top 10 design pattern interview questions

1. What are design patterns?
2. How are design patterns categorized?
3. Explain the benefits of design patterns in Java.
4. Describe the factory pattern.
5. Differentiate ordinary and abstract factory design patterns.
6. What do you think are the advantages of builder design patterns?
7. How is the bridge pattern different from the adapter pattern?
8. What is a command pattern?
9. Describe the singleton pattern along with its advantages and disadvantages.
10. What are anti patterns?



# Expected Questions for University Exam

- What are design patterns?
- How are design patterns categorized?
- Explain the benefits of design patterns in Java.
- Describe the factory pattern.
- Differentiate ordinary and abstract factory design patterns.
- What do you think are the advantages of builder design patterns?
- How is the bridge pattern different from the adapter pattern?
- What is a command pattern?
- Describe the singleton pattern along with its advantages and disadvantages.
- What are anti patterns?

# Recap of Unit

**Till Now we understand,** The idea of a pattern and the definition of terms and concepts, The idea of design pattern in **Smalltalk MVC**, what are the model ,view and controller defined by small talk, **design pattern in The catalog**, we learn how to **organized the catalog**, learn how to **solve real world problems** with design patterns, what are the approaches and Patterns that help in real world problem , how to **choose best pattern and what are uses of the patterns**.  
The Principle of Least knowledge, also known as **The law of Demeter**, or more precisely, the Law of Demeter for Functions/Methods (LoD-F) is a design principle which provides guidelines for designing a system with minimal dependencies.