

**TOPIC TITLE: DESIGN PRINCIPLES****LEARNING OBJECTIVES:**

At the end of the topic session, the students should be able to:

LO1: Compare the scope of applicability of the different design principles; and

LO2: Apply design principles when designing object-oriented systems.

**MATERIALS/EQUIPMENT:**

- Computer
- LCD projector
- File/s (03 Design Principles and Patterns)
  - 03 Handout 1.pdf
  - 03 LCD Slides 1.ppsx
  - 03 Laboratory Exercise 1.pdf
  - 03 Laboratory Exercise 1 Answer Key.pdf
  - 03 Skills Checklist.pdf
- Software requirement
  - MS PowerPoint
- Whiteboard marker and eraser

**TOPIC PREPARATION:**

- The instructor is encouraged to research materials that will help supplement the topics in this session.
- Log on to eLMS to obtain a copy of **03 Handout 1** which will be used in this session.
- Review **03 Handout 1**. Check the handout together with the slide presentation to ensure that topics will be discussed cohesively. Some slides only contain images that support the content of the handout. Thus, a careful review of related materials is required.
- Encourage the students to take notes.
- Provide additional examples other than the ones provided in the handout and slides, if needed.
- Motivate the students to engage in all class activities and let them feel they are important. Religiously follow all activities as these are geared towards the achievement of the course learning outcomes.
- Anticipate possible questions that students might raise during the discussion.

**PRESENTATION OVERVIEW:**

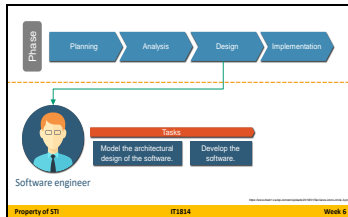
A. Introduction	20 min
B. Instructional Input	
<i>Overview of Design Principles</i>	30 min
a. Define design principles and SOLID principles.	
b. Explain the benefits of using the SOLID principles in software design.	
<i>Single Responsibility Principle</i>	30 min
a. Describe how the Single Responsibility Principle (SRP) helps in developing clean code.	
b. Explain how a software design architecture violates SRP and what problems may occur on that design.	
c. Explain how to implement SRP.	
<i>Open-Closed Principle</i>	30 min
a. Describe how the Open-Closed Principle (OCP) helps in developing clean code.	
b. Explain how a software design architecture violates OCP and what problems may occur on that design.	
c. Explain how to implement OCP.	
<i>Liskov Substitution Principle</i>	30 min
a. Describe how the Liskov Substitution Principle (LSP) helps in developing clean code.	
b. Explain how a software design architecture conforms to LSP.	
<i>Interface Segregation Principle</i>	30 min
a. Describe how the Interface Segregation Principle (ISP) helps in developing clean code.	
b. Explain how a software design architecture violates ISP and what problems may occur on that design.	
c. Explain how to implement ISP.	
<i>Dependency Inversion Principle</i>	30 min
a. Describe how the Dependency Inversion Principle (DIP) helps in developing clean code.	
b. Explain how a software design architecture violates DIP and what problems may occur on that design.	
c. Explain how to implement DIP.	
C. Generalization	55 min
D. Evaluation	40 min
E. Application	300 min
F. Assignment	5 min
Total duration	600 min

## TOPIC PRESENTATION:

### A. Introduction

1. Before starting the discussion, arrange the students by their assigned groupings from Prelim. They will perform the activity after discussing all the topics by group.

#### Slide 1



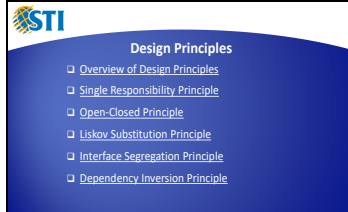
2. Display **Slide 1** of **03 LCD Slides**. Then, tell the following statements:

After software analysis, where we gather and analyze requirements, the next phase on the development of a software project is software design. In this phase, we should ask ourselves, “How are we going to build our software?”

When we develop our software system, we want our system to be successful. The system should be working properly, free from any errors, and flexible when new requirements have occurred to achieve a successful software system.

In this topic, we will discuss how to build a clean, successful, and flexible software system.

#### Slide 2

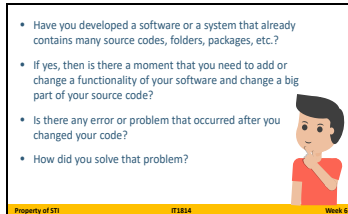


3. Show **Slide 2** to present the topic coverage.
4. Ask the students if they have any clarifications before proceeding to the topic.

### B. Instructional Input

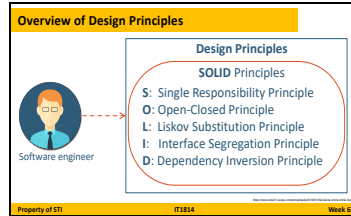
#### Overview of Design Principles

#### Slide 3



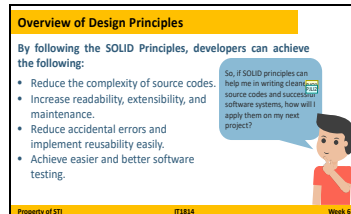
1. For this subtopic, refer the students to **Page 1** of **03 Handout 1**.
2. Show **Slide 3** and ask at least three (3) students to answer each given question:
  - Have you developed a software or a system that already contains many source codes, folders, packages, etc.?
  - If yes, then is there a moment that you need to add or change a functionality of your software and change a big part of your source code?
  - Is there any error or problem that occurred after you changed your code?
  - How did you solve that problem?
3. Tell the students the following statements:

## Slide 4



When you develop software, you encounter some unwilling code. For example, these codes can be a long method, a long class, or a dirty code. In this topic, you will learn basic solutions about such kind of problems. You will also learn the SOLID principles, including the implementations of each.

## Slide 5

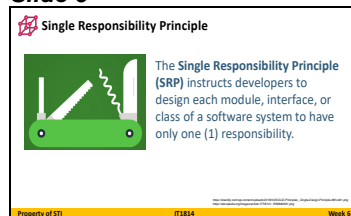


4. Show **Slide 4**. Define design principles and SOLID principles.
5. Show **Slide 5**. Explain the benefits of using SOLID principles in software design.
6. Tell the students that on the proceeding topics, you will discuss each of the SOLID principles and how to implement them.
7. Ask the students the following guide questions to check their understanding before proceeding to the topic:

- What is the use of design principle?
- What are SOLID principles?

**Note:** Acknowledge and assess the answers coming from the class. Provide comments and recommendations to the students about their answers.

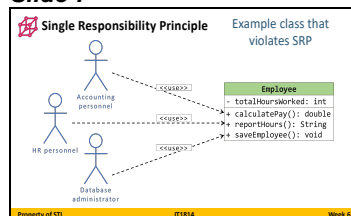
## Slide 6



## Single Responsibility Principle

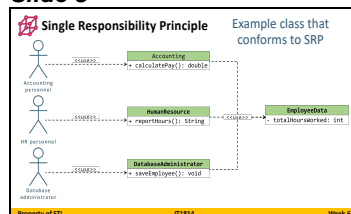
1. For this subtopic, refer the students to **Pages 1–2** of the handout.
2. Show **Slide 6**. Describe how the Single Responsibility Principle (SRP) helps in developing clean code.

## Slide 7



3. Show **Slide 7**. Explain how the given software design architecture example violates SRP and what problems may occur on this design.
4. Show **Slide 8**. Ask at least three (3) students to explain how the given class diagram conforms to SRP. Explain how the given example solution to the design implements the SRP.

## Slide 8




5. Ask the students the following guide question to check their understanding before proceeding to the topic:
- What does SRP suggest to developers to create a good design architecture?

**Note:** Acknowledge and assess the answers coming from the class. Provide comments and recommendations to the students about their answers.

## Slide 9

**Open-Closed Principle**



The Open-Closed Principle (OCP) states that software modules, interfaces, or classes should be open for extension but closed for modification.

Property of STI IT1814 Week 5

## Open-Closed Principle

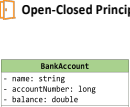
1. For this subtopic, refer the students to **Page 3** of the handout.
2. Show **Slide 9**. Describe how the Open-Closed Principle (OCP) helps in developing clean code.
3. Show **Slide 10**. Explain how the given software design architecture example violates OCP and what problems may occur on this design.
4. Show **Slide 11**. Ask the students to convert the given class diagram into a Java program. Explain how the given example solution to the design implements OCP.
5. Ask the students the following guide question to check their understanding before proceeding to the topic:

- What does OCP suggest to developers to create a good design architecture?

**Note:** Acknowledge and assess the answers coming from the class. Provide comments and recommendations to the students about their answers.

## Slide 10

**Open-Closed Principle**



**Disadvantages of this design:**

- For every new added methods, the unit testing of the software should be done again.
- When a new requirement is added, the maintenance and adding function may take time.
- Adding a new requirement might affect the other functionalities of software even if the new requirement works.

Property of STI IT1814 Week 5

## Slide 11

**Open-Closed Principle**

Example class that conforms to OCP

```

classDiagram
    class BankAccount {
        - name: string
        - accountNumber: long
        - balance: double
        + newSavingsAccount(): void
        + newCheckingAccount(): void
        + deposit(): void
        + withdraw(): double
        + getBalance(): double
    }
    class CheckingAccount {
        - serviceCharge: double
        + applyServiceCharge(): void
        + withdraw(): double
    }
    class SavingsAccount {
        - interestRate: double
        + computeInterest(): void
        + withdraw(): double
    }
    BankAccount <|-- CheckingAccount
    BankAccount <|-- SavingsAccount
  
```

Property of STI IT1814 Week 5

## Liskov Substitution Principle


1. For this subtopic, refer the students to **Pages 3–4** of the handout.
2. Show **Slide 12**. Describe how the Liskov Substitution Principle (LSP) helps in developing clean code.
3. Show **Slide 13**. Explain how the given example design conforms to LSP.
4. Ask the students the following guide question to check their understanding before proceeding to the topic:

- What does LSP suggest to developers to create a good design architecture?

**Note:** Acknowledge and assess the answers coming from the class. Provide comments and recommendations to the students about their answers.

## Slide 12

**Liskov Substitution Principle**



The Liskov Substitution Principle (LSP) suggests that when creating new derived class of an existing class, make sure that the derived class can be substitute for its base class.

Property of STI IT1814 Week 5

## Slide 13

**Liskov Substitution Principle**

Example class that conforms to LCP

```

classDiagram
    class Billing
    class License {
        + calculateFee(): void
    }
    class PersonalLicense {
        - users: int
    }
    class BusinessLicense {
    }
    Billing --> License
    License <|-- PersonalLicense
    License <|-- BusinessLicense
  
```

```

License applicant = new License();
applicant = new PersonalLicense();
applicant = new BusinessLicense();
  
```

Property of STI IT1814 Week 5

## Slide 14

**Interface Segregation Principle**

Example class that violates ISP

```

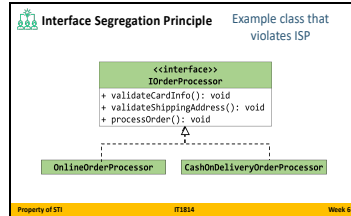
classDiagram
    class IOrderProcessor {
        + validateCardInfo(): void
        + validateShippingAddress(): void
        + processOrder(): void
    }
    class OnlineOrderProcessor
    class CashOnDeliveryOrderProcessor
    IOrderProcessor <|-- OnlineOrderProcessor
    IOrderProcessor <|-- CashOnDeliveryOrderProcessor
  
```

Property of STI IT1814 Week 5

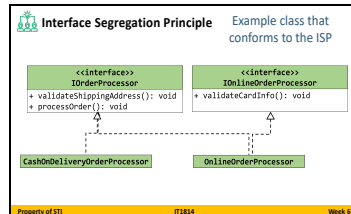
## Interface Segregation Principle

1. For this subtopic, refer the students to **Pages 4–5** of the handout.
2. Show **Slide 14**. Describe how the Interface Segregation Principle (ISP) helps in developing clean code.

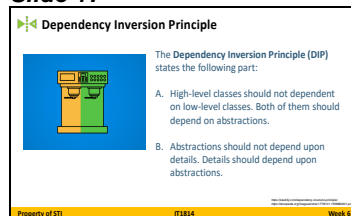
## Slide 15



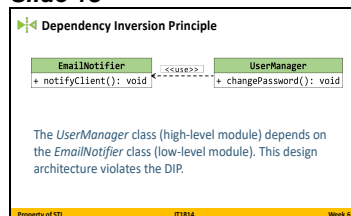
## Slide 16



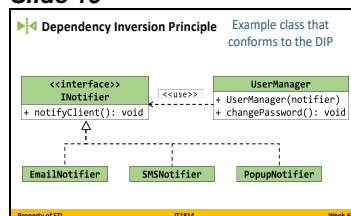
## Slide 17



## Slide 18



## Slide 19



## Slide 20



3. Show **Slide 15**. Explain how the given software design architecture example violates ISP and what problems may occur on this design.
4. Show **Slide 16**. Ask the students to convert the given class diagram into a Java program. Explain how the given example solution to the design implements the ISP.
5. Ask the students the following guide question to check their understanding before proceeding to the topic.

- What does ISP suggest to developers to create a good design architecture?

**Note:** Acknowledge and assess the answers coming from the class. Provide comments and recommendations to the students about their answers.

## Dependency Inversion Principle

1. For this subtopic, refer the students to **Pages 5–6** of the handout.
2. Show **Slide 17**. Describe how the Dependency Inversion Principle (DIP) helps in developing clean code.
3. Show **Slide 18**. Explain how the given software design architecture example violates DIP and what problems may occur on this design.
4. Show **Slide 19**. Ask at least three (3) students to explain how the given class diagram conforms to DIP. Explain how the given example solution to the design implements the DIP.
5. Ask the students the following guide question to check their understanding before closing the topic:

- What does DIP suggest to developers to create a good design architecture?

**Note:** Acknowledge and assess the answers coming from the class. Provide comments and recommendations to the students about their answers.

## C. Generalization



## Steps 1–6

**Assessment:** Group Presentation

**Learning Objective(s):** LO1 and LO2

1. Show **Slide 20**. Ask each group to prepare a piece of paper.
2. Instruct them to write their names and their answer on their paper. Tell them that this activity will exercise their decision-making skills by

selecting what SOLID principle will be implemented for a certain project.

3. Allot 20 minutes for the groups to perform the seatwork.
4. Call on at least three (3) groups to present their answers to the class. Process their answers using the following questions:
  - a. What design principle(s) did your group choose in designing the architecture of the software system project?
  - b. Explain how you considered this design principle to implement on the project.

**Note:** Acknowledge and assess the answers coming from the class. Provide comments and recommendations to the students about their answers.

5. Use the following rubric for grading the presentation of each group:

**GRADING RUBRIC:**

Criteria	2 points	3 points	4 points	5 points
<b>Presentation (x2)</b>	Only one (1) member of the group is active in the presentation.	Only a few of the group members are active in the presentation.	Some of the group members are active in the presentation, showcasing a comprehensive knowledge of their work.	All group members are active in the presentation, showcasing a comprehensive knowledge of their work.
<b>TOTAL</b>				<b>10 POINTS</b>

6. Collect each group's seatwork for grading and recording. Return their seatwork as their reference or reviewer once their scores are recorded.
7. Ask the students if they have any clarifications.

**D. Evaluation**



**Steps 1–2**

**Assessment:** eLMS Quiz

**Learning Objective(s):** LO1

1. Before facilitating the laboratory exercise, ask the students to access their eLMS on their respective computers and answer **03 eLMS Quiz 1** under the *Design Principles* module.
2. Give them 25 minutes to answer the quiz then ask them to submit their quiz after answering.

## E. Application



### Steps 1–2

**Assessment:** Hands-on activity

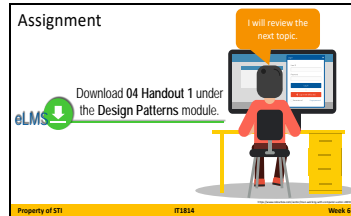
**Learning Objective(s):** LO2

1. Distribute **03 Laboratory Exercise 1** to the students and discuss its content. The activities are required to be performed by students with a partner. Allow the students to select their partner within their group only.
2. Next, ask the students for questions or clarifications. Address their queries accordingly. Give them ample time to do the activities.
3. Collect and save the students' activities.

## F. Assignment

1. Show **Slide 21**. Then, ask the students to log on to their eLMS and download **04 Handout 1** under the *Design Patterns* module to be used for the next session.

### Slide 21



## REFERENCES

- Design Principles (n.d.). In *OODesign.com Object Oriented Design*. Retrieved from <https://www.oodesign.com/design-principles.html>
- Dooley J. (2017). *Software development, design and coding: With patterns, debugging, unit testing, and refactoring* (2<sup>nd</sup> ed.). Retrieved from <https://books.google.com.ph/books?id=LGRADwAAQBAJ&dq=Software+Development,+Design+and+Coding:+With+Patterns,+Debugging,+Unit+Testing,+and+Refactoring>
- Joshi, B. (2016). *Beginning SOLID principles and design patterns for asp.net developers*. California: Apress Media, LLC.
- Martin, R. (2017). *Clean architecture. A craftsman's guide to software structure and design*. Retrieved from <https://archive.org/details/CleanArchitecture>