SE2030 – Software Engineering

Department of Information Technology, Faculty of Computing

Year 2 semester 1 (2025)

Tutorial 9

This tutorial is designed to revise your knowledge of Design Patterns

Activity 01

a) What are the three main types of design patterns?

b) "INavigator" is a GPS-based road Navigation system. Read the description below and answer the questions. "INavigator" consists of a GPS receiver, which constantly sends a stream of $GPRMC sentences to a GPS class. A $GPRMC sentence consists of a navigation receiver warning, latitude, longitude, speed over ground, date, and magnetic variation. The GPS class continually reads, parses, and stores the $GPRMC sentences as records in a buffer. These records are then displayed to the user in three views. The three views are,

- a text view which displays the basic information in the GPS sentence including distance travelled and average speed,
- a compass view which displays the direction the user is moving on and
- a breadcrumb trail view which displays the trail as well as minimum height, maximum height, and the ascent (the difference between them).

1. If you were hired as a designer for this project, what would you suggest as the most appropriate design pattern to be used in implementing the system?
2. Justify your selection of the design pattern.
3. Explain your solution using a simple class diagram

Activity 02

You are tasked with designing a logging system for a software application. The system must ensure that only one instance of the logger exists throughout the application's lifetime. All parts of the program should share and access this single instance to record logs consistently. If any code attempts to create a new logger object, it should instead return the existing instance.

Which design pattern would you use to implement this solution? Explain why it is the most suitable pattern?

Activity 03

A ride-hailing application (like Uber) needs to calculate fares based on different strategies:

- Normal Fare during regular hours
- Peak Fare during high-demand times
- Discount Fare during promotional offers

The system should allow the application to choose the appropriate fare calculation method at runtime without modifying the core code.

a) Which design pattern is best suited for this problem?
b) Justify your answer.
c) Briefly explain how the Context, Strategy Interface, and Concrete Strategies would be used in this scenario.

Activity 04

A software company is developing a document editor application that supports different types of files, such as Word documents, PDF files, and Spreadsheets. Each file type requires different objects to open and edit them. The application should not directly instantiate these objects, as that would make the system difficult to maintain and extend when adding new file types. Instead, the design should provide a way to create objects dynamically, based on the type of file the user opens.

Which design pattern best solves this problem, and why?

Activity 05

You have been tasked with designing a feature for an online shopping platform that enables customers to purchase various items. Each item can be accompanied by optional add-ons, including gift wrapping, express delivery, and a personalized message. The platform needs to support adding multiple optional add-ons to any item dynamically. Also, each add-on has a specific cost, which should be reflected in the final price of the item. The system should be flexible enough to allow future add-ons to be added without requiring changes to the existing classes.

a) Suggest a suitable design pattern that you could use for the above scenario. Justify your answer.

b) Draw the class structure of the design pattern you identified in part (a) with appropriate methods for the above scenario.

Activity 06

An application settings manager is required to store and manage configuration data such as file paths, themes, and system settings.

- The settings must stay consistent throughout the entire application.
- If multiple instances of the settings manager are created, it could lead to conflicts and inconsistent data.
- The solution must ensure that only one instance of the settings manager exists, and it

should be easily accessible from anywhere in the application.
- The instance should be created only when it is first needed.

1. Which design pattern would you use to solve this problem? Explain your reasoning.
2. What are the essential components (class, constructor, method) required to implement this pattern?
3. Why must the constructor be private in this design?
4. Explain how the global access point works in this pattern and why it is important.
5. Suppose a developer mistakenly allows multiple instances of the settings manager to be created.
   - What problems could occur?
   - How does this pattern prevent those problems?

Self-Study Activities

Activity 01

A company is developing a configuration manager for a large application. The configuration manager should load settings only once when the application starts. Throughout the application, different modules must access the same configuration data without creating multiple copies. If a module tries to create a new configuration manager, the system should return the already existing one.

Which design pattern would be the most appropriate for this scenario? Justify your answer.

Activity 02

In a stock market monitoring system, multiple investors are interested in tracking the prices of different stocks. Whenever the price of a stock changes, all investors who have subscribed to updates for that stock should be notified immediately. The system should allow investors to subscribe or unsubscribe from updates dynamically.

a. Recommend the most suitable design pattern for the given scenario above and justify your recommendation.
b. Draw the class structure of the design pattern that you identified above in part a) with appropriate classes and methods for the above scenario.