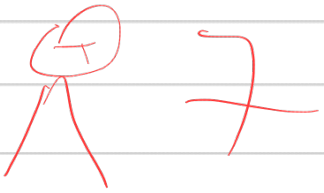
Head First Desing Patterns



Chapter 1 : Welcome to Design Patterns

The SimUDuck app: Introduction to the book's context, presenting an application called SimUDuck that simulates different types of ducks and their behaviors.

Joe thinks about inheritance...: Starts with a reflection on using inheritance to model different types of ducks and their behaviors.

How about an interface?: Introduction to the idea of using interfaces to define behaviors independently of the main class.

The one constant in software development: Emphasis on the constant change in software development and the need for flexible code to adapt to those changes.

Separating what changes from what stays the same: Discussion on the importance of separating parts of the code that are prone to changes from those that are stable.

Designing the Duck Behaviors: Exploration of how to design duck behaviors and how to use interfaces to represent these behaviors.

Testing the Duck code: The importance of testing the code to ensure that duck behaviors work correctly.

Setting behavior dynamically: Introduction to the concept of changing duck behaviors at runtime.

The Big Picture on encapsulated behaviors: Understanding how to encapsulate behaviors in separate classes to maintain cleaner and more modular code.

HAS-A can be better than IS-A: Discussion on how composition (HAS-A) can be a more flexible alternative to inheritance (IS-A) in some cases.

The Strategy Pattern: Introduction to the Strategy design pattern, which involves encapsulating a set of interchangeable algorithms.

The power of a shared pattern vocabulary: The importance of having a common vocabulary to discuss and communicate design patterns.

How do I use Design Patterns?: How to apply design patterns in real-world situations.

Tools For your Design Toolbox: Introduction to useful tools and concepts For software design.

Chapter 2: The Observer Pattern

The Weather Monitoring application: Introduction to the new "Weather Monitoring" application and its need to notify different devices when the weather changes.

Meet the Observer Pattern: Presentation of the Observer design pattern, allowing objects to subscribe and receive notifications of changes in another object.

Publishers + Subscribers = Observer Pattern: Explanation of how the Observer pattern works with a publisher and multiple subscribers receiving updates.

Five minute drama: a subject For observation: A dramatized example of how the Observer pattern would work in an everyday situation.

The Observer Pattern defined: Definition and details of the Observer pattern, including the relationship between subjects and observers.

The power of Loose Coupling: Importance of low object dependency in the Observer pattern.

Designing the Weather Station: Design of the weather station and its components using the Observer pattern.

Implementing the Weather Station: Practical implementation of the Observer pattern in the weather station.

Using Java's built-in Observer Pattern: How to use the Observer pattern implementation provided by the java.util.Observable class in Java.

The dark side of java.util.Observable: Limitations and disadvantages of using java.util.Observable to implement the Observer pattern.

Design patterns Gamma

Chapter 1 : Introduction

What is a pattern?

Patterns are proven solutions to common problems in software design.

Elements of a pattern

A pattern consists of a name, problem, solution, and consequences.

Classification of patterns

Patterns are grouped into creational, structural, and behavioral.

Design problems

Common challenges like Flexibility and maintenance.

Three types of problems

Problems categorized into creation, structure, and behavior.

Definition of terms

Key definitions to understand the concepts.

Chapter 2: Design Patterns and Object-Oriented Programming

Why patterns?

Advantages such as improved design and solution reuse.

How to choose between patterns?
Selection based on problem nature and requirements.

What makes a design good?
Key Features: Flexibility and comprehensibility.

Making an object-oriented design work
Strategies for effectiveness in object-oriented systems.

Chapter 3 : Creational Patterns

Singleton Pattern

Unique instance of a class and global access.

Prototype Pattern

Object copies for simplified creation.

Builder Pattern

Step-by-step construction of complex objects.

Abstract Factory Pattern

Creation of families of related objects.

Factory Method Pattern

Object creation in subclasses.

Adapter

Bridge

Proxy

Chain of responsibility

Command

Composite

Decorator

Facade

Flyweight

Interpreter

Iterator

Mediator

Memento

Observer

State

Strategy

Template method

Visitor