Software Engineering

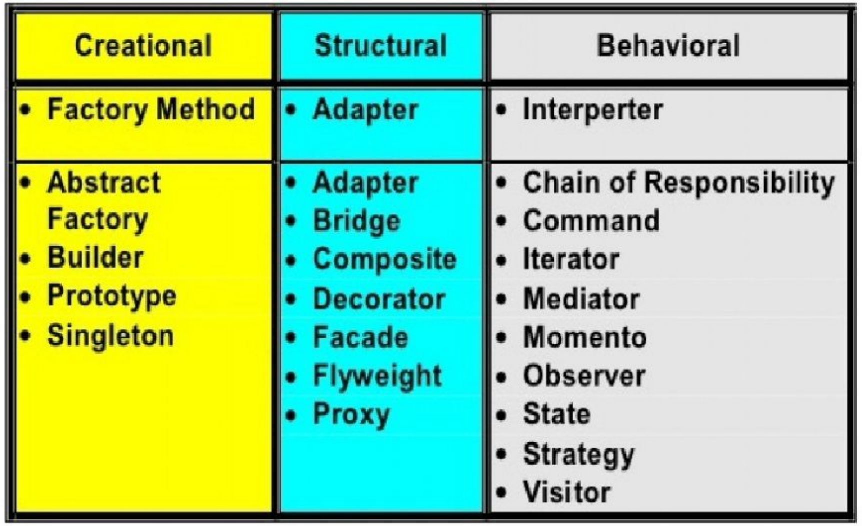
Lesson #11 - Practice

Lesson #11 - Practice

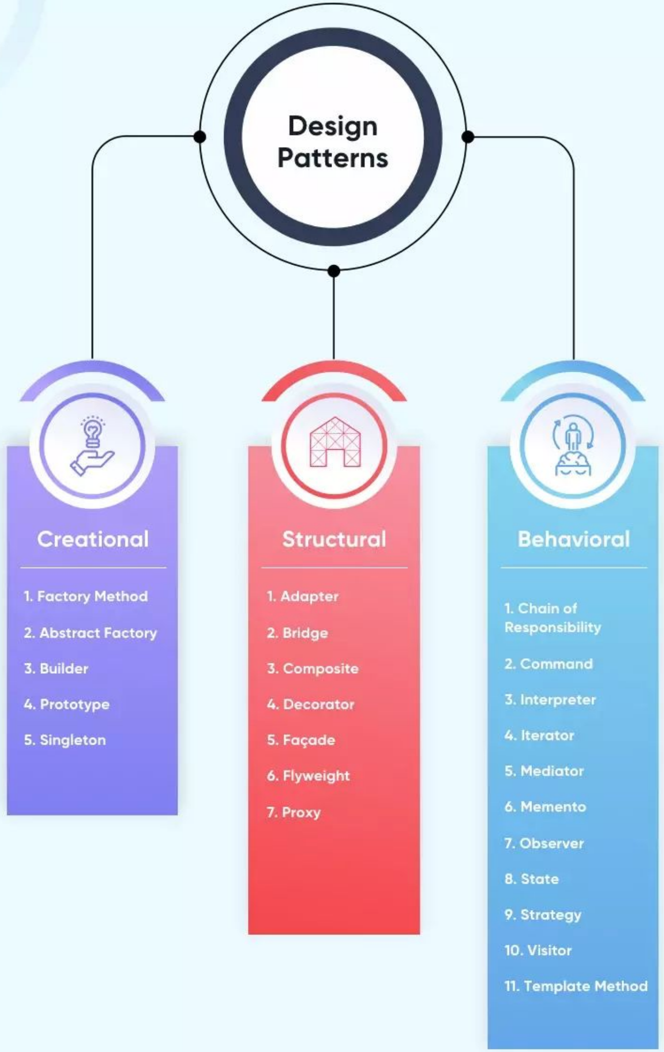
Agenda: Lesson #11 - Software Engineering - Practice

Agenda: Lesson #11 - Software Engineering - Practice

Design Patterns



Creational	Structural	Behavioral
• Factory Method	• Adapter	• Interperter
• Abstract Factory • Builder • Prototype • Singleton	• Adapter • Bridge • Composite • Decorator • Facade • Flyweight • Proxy	• Chain of Responsibility • Command • Iterator • Mediator • Momento • Observer • State • Strategy • Visitor



Design Patterns

Creational

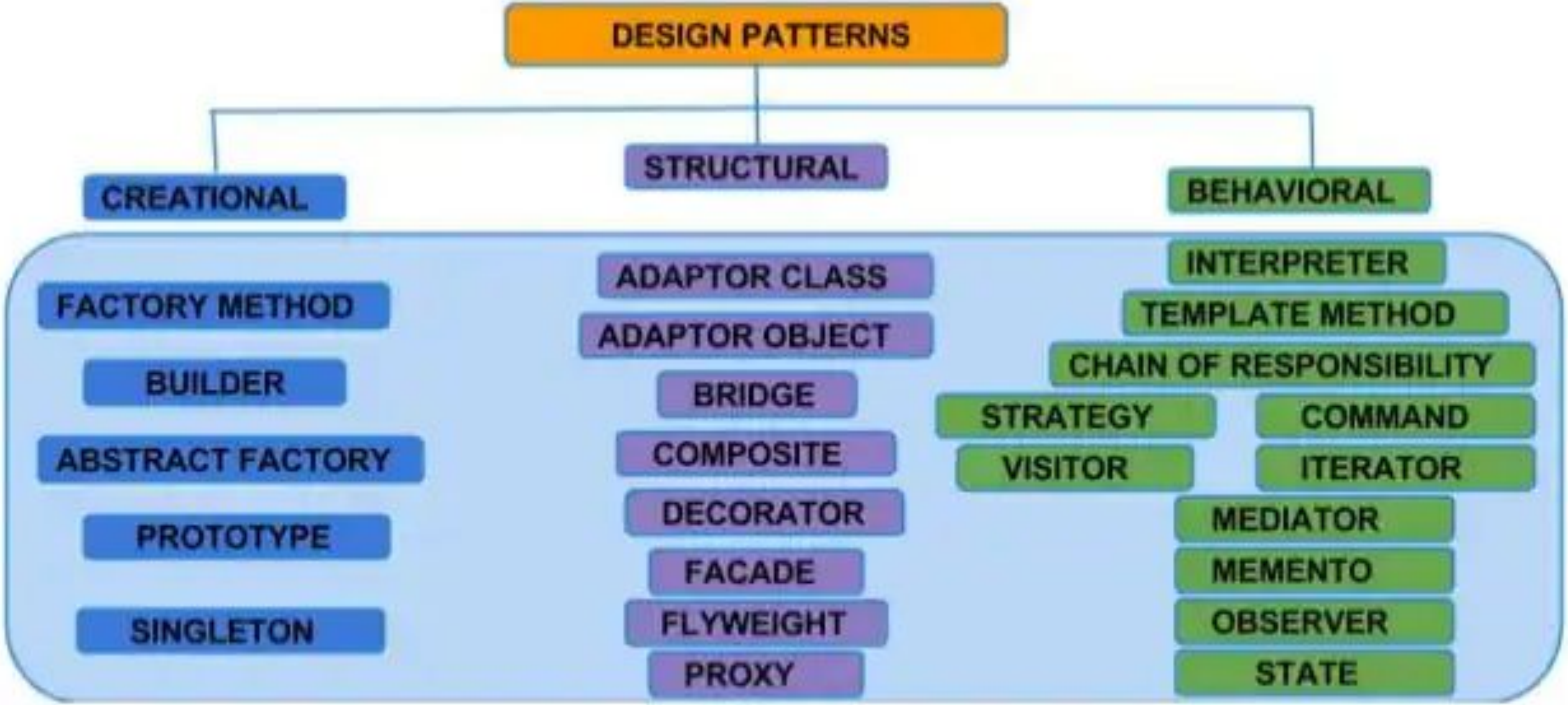
1. Factory Method
2. Abstract Factory
3. Builder
4. Prototype
5. Singleton

Structural

1. Adapter
2. Bridge
3. Composite
4. Decorator
5. Façade
6. Flyweight
7. Proxy

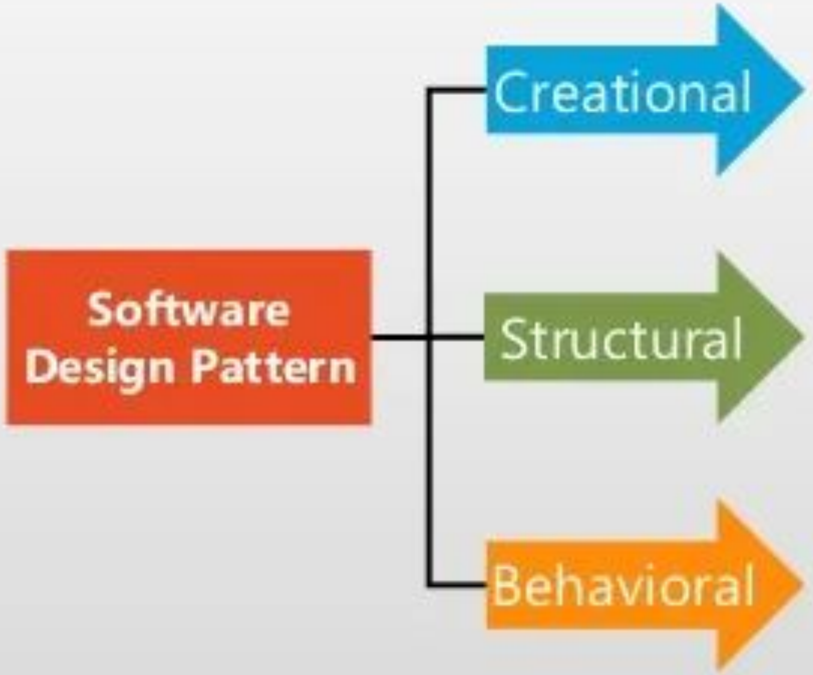
Behavioral

1. Chain of Responsibility
2. Command
3. Interpreter
4. Iterator
5. Mediator
6. Memento
7. Observer
8. State
9. Strategy
10. Visitor
11. Template Method


DESIGN PATTERNS
CREATIONAL
STRUCTURAL
BEHAVIORAL
FACTORY METHOD
BUILDER
ABSTRACT FACTORY
PROTOTYPE
SINGLETON
ADAPTOR CLASS
ADAPTOR OBJECT
BRIDGE
COMPOSITE
DECORATOR
FACADE
FLYWEIGHT
PROXY
INTERPRETER
TEMPLATE METHOD
CHAIN OF RESPONSIBILITY
STRATEGY
COMMAND
VISITOR
ITERATOR
MEDIATOR
MEMENTO
OBSERVER
STATE

Design Patterns

Types Of Design Patterns



- Used to construct objects such that they can be decoupled from their implementing system.
- Used to form large object structures between many disparate objects.
- Used to manage algorithms, relationships, and responsibilities between objects

Design Patterns

Designing object-oriented software is hard, and designing reusable object-oriented software is even harder

You must find pertinent objects, factor them into classes at the right granularity, define class interfaces and inheritance hierarchies, and establish key relationships among them

Your design should be specific to the problem at hand but also general enough to address future problems and requirements

Design Patterns

You also want to avoid redesign, or at least minimize it

Experienced object-oriented designers will tell you that a reusable and flexible design is difficult if not impossible to get "right" the first time

Before a design is finished, they usually try to reuse it several times, modifying it each time

Design Patterns

One thing expert designers know not to do is solve every problem from first principles. Rather, they reuse solutions that have worked for them in the past.

When they find a good solution, they use it again and again. Such experience is part of what makes them experts.

Agenda: Lesson #11 - Software Engineering - Practice

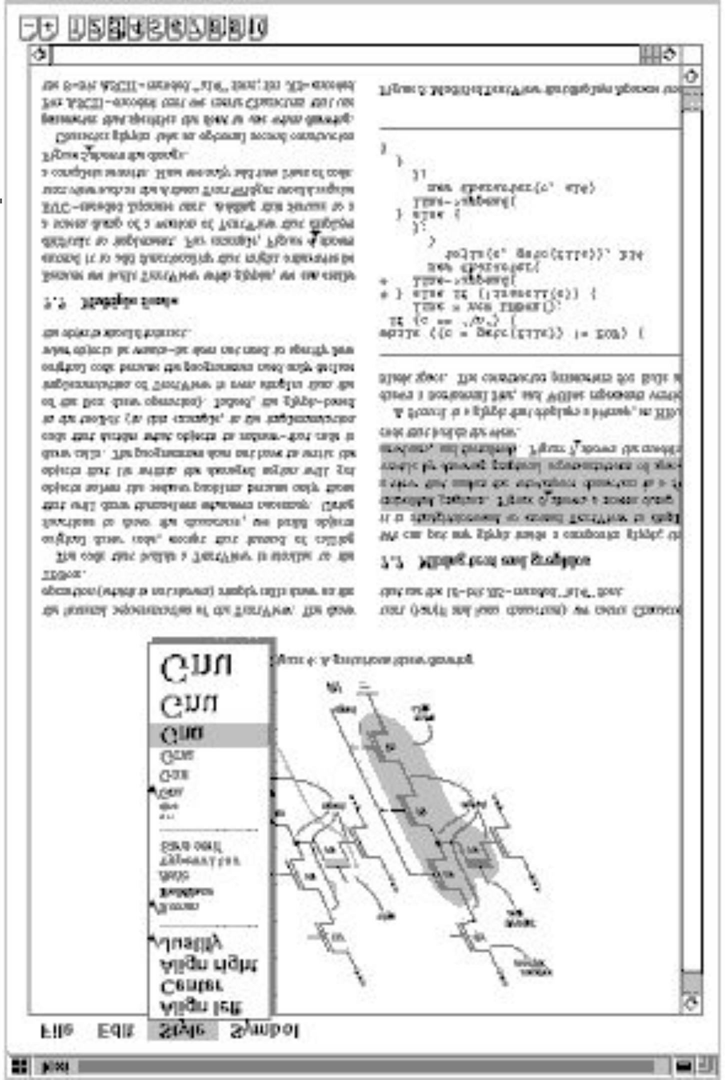
A Case Study: Design a Document Editor

This chapter presents a case study in the design of a "What-You-See-Is-What-You-Get" (or "WYSIWYG") document editor called Lexi.1

We'll see how design patterns capture solutions to design problems in Lexi and applications like it

A Case Study: Design a Document Editor

A Case Study: Design a Document Editor



Agenda: Lesson #11 - Software Engineering - Practice

1 Introduction

2 A Case Study: Designing a Document Editor

3 **Class Work**

4 Q & A

Class work

Software Engineering, 10. Global Edition

No classwork for today

Agenda: Lesson #11 - Software Engineering - Practice

Agenda: Lesson #11 - Software Engineering - Practice

Q & A