Software Design Pattern

In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.

Creational

These design patterns are all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.

- **Abstract Factory**
- **Builder**
- **Factory Method**
- **Object Pool**
- **Prototype**
- **Singleton**

Structural

These design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

- **Adapter**
- **Bridge**
- **Composite**
- **Decorator**
- **Facade**
- **Flyweight**
- **Private Class Data**
- **Proxy**

Behavioral

These design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects.

- **Chain Of Responsibility**
- **Command**
- **Interpreter**
- **Iterator**
- **Mediator**
- **Memento**
- **Null Object**
- **Observer**
- **State**
- **Strategy**
- **Template Method**
- **Visitor**

- Don't Use Design Pattern At First.
- You Should Get To Design Pattern After Refactoring.
- Don't Memorize Design Patterns, Just Learn Context.
- Use It If You Need It.

Books To Read

- Head First Design Pattern
- Design Patterns: Elements of Reusable Object-Oriented Software (GOF – Gang Of Four)
- Head First Object Oriented Analysis & Design
- https://sourcemaking.com
- https://refactoring.guru