デザインパターンの主な種類

Gang of Four（GoF）のデザインパターン
GoF のパターンは、以下の 3 つのカテゴリに分かれている。

生成パターン（Creational Patterns）
構造パターン（Structural Patterns）
振る舞いパターン（Behavioral Patterns）
GoF 以外にも、実際の開発ではさまざまなデザインパターンが提案されており、
以下のような他のカテゴリにも分類されている。

オープン・クローズド原則に基づくパターン

Extension Patterns（拡張パターン）：既存のクラスに新しい機能を追加する方法に関するパターン
並列処理パターン

Concurrency Patterns（並列パターン）：複数のスレッドやプロセスで同時に動作する場合に、データの整合性やスレッドセーフを考慮するためのパターン
例：Thread Pool Pattern（スレッドプールパターン）
データベースパターン

データベースに特化したデザインパターンも存在する。データの格納方法やデータベースとのやり取りに関するパターン
例：Data Mapper（データマッパーパターン）
エンタープライズ・アーキテクチャパターン

大規模なエンタープライズシステムにおいて、アプリケーションの設計に関するベストプラクティスを提供するパターン
例：Model-View-Controller (MVC)（モデル・ビュー・コントローラーパターン）

代表的なデザインパターン
GoF のデザインパターン以外にも多くのデザインパターンが存在するが、以下に代表的なものを挙げる

1. 生成パターン（Creational Patterns）
Singleton（シングルトン）：クラスのインスタンスが 1 つしか存在しないことを保証する。
Factory Method（ファクトリーメソッド）：オブジェクト生成の責任をサブクラスに委譲。
Abstract Factory（抽象ファクトリーパターン）：関連するオブジェクト群を生成するインターフェースを提供。
Builder（ビルダーパターン）：複雑なオブジェクトの生成過程を分離し、柔軟に構築する。
Prototype（プロトタイプ）：既存のオブジェクトをコピーして新しいオブジェクトを生成する。
2. 構造パターン（Structural Patterns）
Adapter（アダプターパターン）：異なるインターフェースを持つクラスをつなぐ。
Decorator（デコレーターパターン）：オブジェクトの機能を動的に拡張する。
Facade（ファサードパターン）：複雑なサブシステムへの簡易なインターフェースを提供。
Composite（コンポジットパターン）：オブジェクトを木構造で扱い、個々のオブジェクトとその集合を同一視する。
Proxy（プロキシパターン）：他のオブジェクトへのアクセスを制御する代理人を提供。
3. 振る舞いパターン（Behavioral Patterns）
Observer（オブザーバーパターン）：状態の変化を他のオブジェクトに通知する。
Strategy（ストラテジーパターン）：アルゴリズムをクラスとして切り替え可能にする。
Command（コマンドパターン）：操作をオブジェクトとしてカプセル化する。
Iterator（イテレーターパターン）：集合の要素を順番にアクセスする方法を提供。
State（ステートパターン）：オブジェクトの状態によって振る舞いを変える。
その他のデザインパターン
Data Access Object (DAO)：データベースアクセスのための設計パターンで、データ操作のロジックを分離する。
Model-View-Controller (MVC)：ユーザーインターフェースの設計パターンで、ビジネスロジックと表示ロジックを分離する。
Microkernel Architecture：最小限の機能を提供するカーネルを使い、プラグインを追加する形で機能を拡張するアーキテクチャパターン。

まとめ
デザインパターンは非常に多くの種類があり、それぞれのパターンが特定の問題を解決するための手法を提供する。GoF の 23 のデザインパターンはその基礎的なものであり、他にもさまざまな特定の領域に特化したデザインパターン（例えば、並列処理、データベース、エンタープライズアーキテクチャなど）が存在する。状況に応じて適切なパターンを選択することが、効果的なソフトウェア設計には欠かせない。