Java 言語は最も普及しているオブジェクト指向プログラミング言語（OOPL）です。
OOPL：Object Oriented Programing Language

1. オブジェクト指向プログラミング言語の特徴

- カプセル化
- 継承
- 多態性（多相性、ポリモーフィズム）

加えて Java の特徴

- クラスベース（プロトタイプベースに対して）： new() でインスタンスを生成する方式
- ガベージコレクション：C++にはない

2. OOPL の特徴を活かす実装

- デザインパターン

3. OOPL を活かす設計

- オブジェクト指向設計
- アナリシスパターン

4. オブジェクト指向プログラミング（OOP）言語に必要な機能は、主に以下の 4 つです：

4.1 カプセル化（Encapsulation）：

データとそれに関連するメソッドを 1 つの単位（クラス）にまとめ、外部からデータを直接アクセスできないようにします。
これにより、データの保護と内部の実装の隠蔽が可能になります。
アクセス修飾子（public, private, protected, なし）を使って、データの可視性を制御します。

4.2 継承（Inheritance）：

既存のクラス（スーパークラス）を基に新しいクラス（サブクラス）を作成する機能です。
サブクラスはスーパークラスの特性を継承し、さらに独自の特性を追加することができます。
再利用性とコードの簡潔さを高め、階層的な関係を表現します。

4.3 ポリモーフィズム（Polymorphism）：

同じ操作を異なるデータ型のオブジェクトに対して行うことができる機能です。
これにより、同じメソッド名でも異なる実装を持つことができます。
オーバーライド（メソッドの再定義）やオーバーロード（同じメソッド名で異なる引数リストを持つ複数のメソッド）を利用します。

4.4 抽象化（Abstraction）：

必要な情報だけを取り出し、詳細を隠すことで、システムの複雑さを管理しやすくします。
抽象クラスやインターフェースを使って、共通の特性や動作を定義し、具体的な実装をサブクラスや実装クラス

に任せます。

5. リファクタリングしやすくなる（継続的な洗練を可能にする）

データと機能を分離したシステムよりも容易になる。ただし、それなりの設計や経験は必要。

「良いモデル」と「なんとかなっていたモデル」

例

Java 8 から日付・時刻の新しい API が追加されました。

Instant, LocalTime, LocalDate, LocalDateTime, ZonedDateTime などです。

これらは古い API である Date, Calendar などを改善したものです。

例

SpringSecurity（ライブラリ、フレームワーク）

WebSecurityConfigurerAdapter が不要になったため、5.7 で非推奨化、6.0 で削除になりました。