

# 用字と用語

60151039 東 裕之輔

# 資料

## ▶ ずっと受けたかったソフトウェアエンジニアリングの新人研修 2章「用字と用語」

### ▶ 監修

宇治則孝

### ▶ 著者

- ・大森久美子
- ・岡崎義勝
- ・西原 琢夫



<http://www.amazon.co.jp/dp/4798119342>

# はじめに

---

- ▶ システム開発では、各種の分析や設計の結果を文書に書き残すのが一般的である。
- ▶ それは成果物と呼ばれる。
- ▶ 成果物のほとんどは文書である。



# はじめに

---

- ▶ 成果物はなぜ文書なのか？
- ▶ 大学では、設計者とプログラマーが同一人物であることが多い。
- ▶ しかしシステムが一定の大きさになると、設計者とプログラマーは別の人であることが一般的。
- ▶ 会社も別ということもある。



# はじめに

---

- ▶ よって情報の引き継ぎが必要である。
- ▶ そのために用いられるのが文書である。



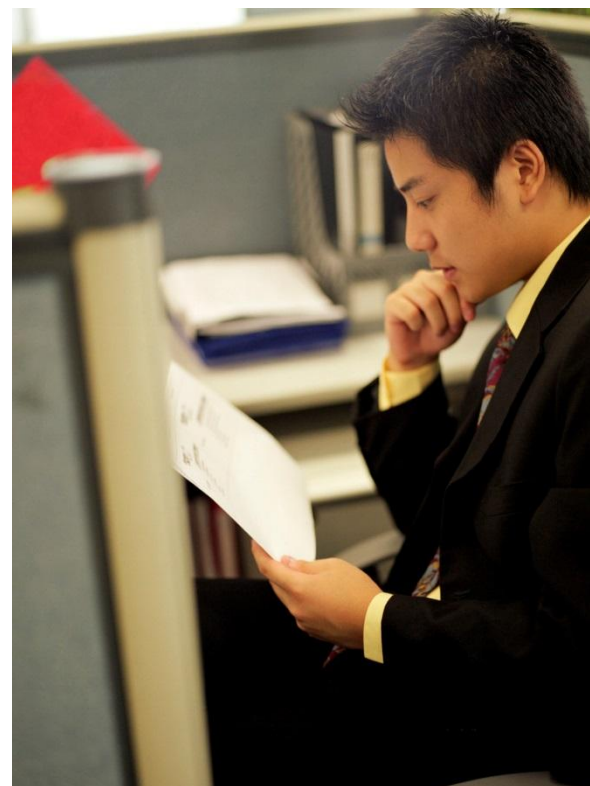
# 情報を引き継ぐ文書

---

- ▶ 正確に効率よく情報を引き継ぐ必要がある。
- ▶ 読みやすい文書。
- ▶ 分かりやすい文書。



- ▶ 文書の書き方が重要である。



# 読みやすい文書

---

- ▶ 読みやすい文書を作るルールの中の1つとなる「用字と用語の統一」

1. 音引き
2. 句読点
3. 拗音
4. カタカナ表記

- ▶ これらは不明確なので統一させる。
  - ➡ 分野別でルールが存在する場合がある。



# 音引きの統一

---

## 語尾の音引き

| 語尾に音引きなし | 語尾に音引きあり |
|----------|----------|
| コンピュータ   | コンピューター  |
| サーバ      | サーバー     |
| プログラマ    | プログラマー   |

## 語中の音引き

| 語中の音引きなし  | 語中に音引きあり   |
|-----------|------------|
| イーサネット    | イーサーネット    |
| ヘッダフォーマット | ヘッダーフォーマット |



# 句読点の統一

- ▶ 分野別のルールに合わせた句読点の統一。
- ▶ このシステムは、最新のCPUを使用しています。
- ▶ このシステムは、最新のCPUを使用しています.

| 分野    | 句点 | 読点 | 音引き      |
|-------|----|----|----------|
| 企業    | 、  | 。  | しない      |
| 学会    | ,  | .  | しない      |
| JIS規格 | ,  | 。  | しない      |
| 書籍    | 、  | 。  | 書籍により異なる |

※, . は全角

# カタカナ表記

---

| カタカナ | 日本語 |
|------|-----|
| テスト  | 試験  |
| パス   | 合格  |
| リソース | 資源  |

| 分野    | 原語or日本語       |
|-------|---------------|
| 企業    | カタカナ          |
| 学会    | カタカナと日本語の使い分け |
| JIS規格 | 日本語に翻訳したもの    |
| 書籍    | カタカナ          |



# 平音と拗音の使い分け

---

同じ原語をそのままカタカナで表記する場合でも、拗音や促音の書き方が異なる。

| 平音として書く | 拗音として書く |
|---------|---------|
| ソフトウェア  | ソウトウエア  |
| スリーウェイ  | スリーウェイ  |
| デジタル    | ディジタル   |

分野別のルールはないので、適切な表記を確認し、自分で統一する。

---



# 分かりやすい文書

---

- ▶ システム開発では、情報の引き継ぎに文章だけではなく、図も多く使用される。
- ▶ 以下の代表的な図やチャート記法を説明していく。
- ▶ UML(記法)
- ▶ クラス図
- ▶ シーケンス図
- ▶ ユースケース図
- ▶ フローチャート
- ▶ HCPチャート



# UML(Unified Modeling Language)

---

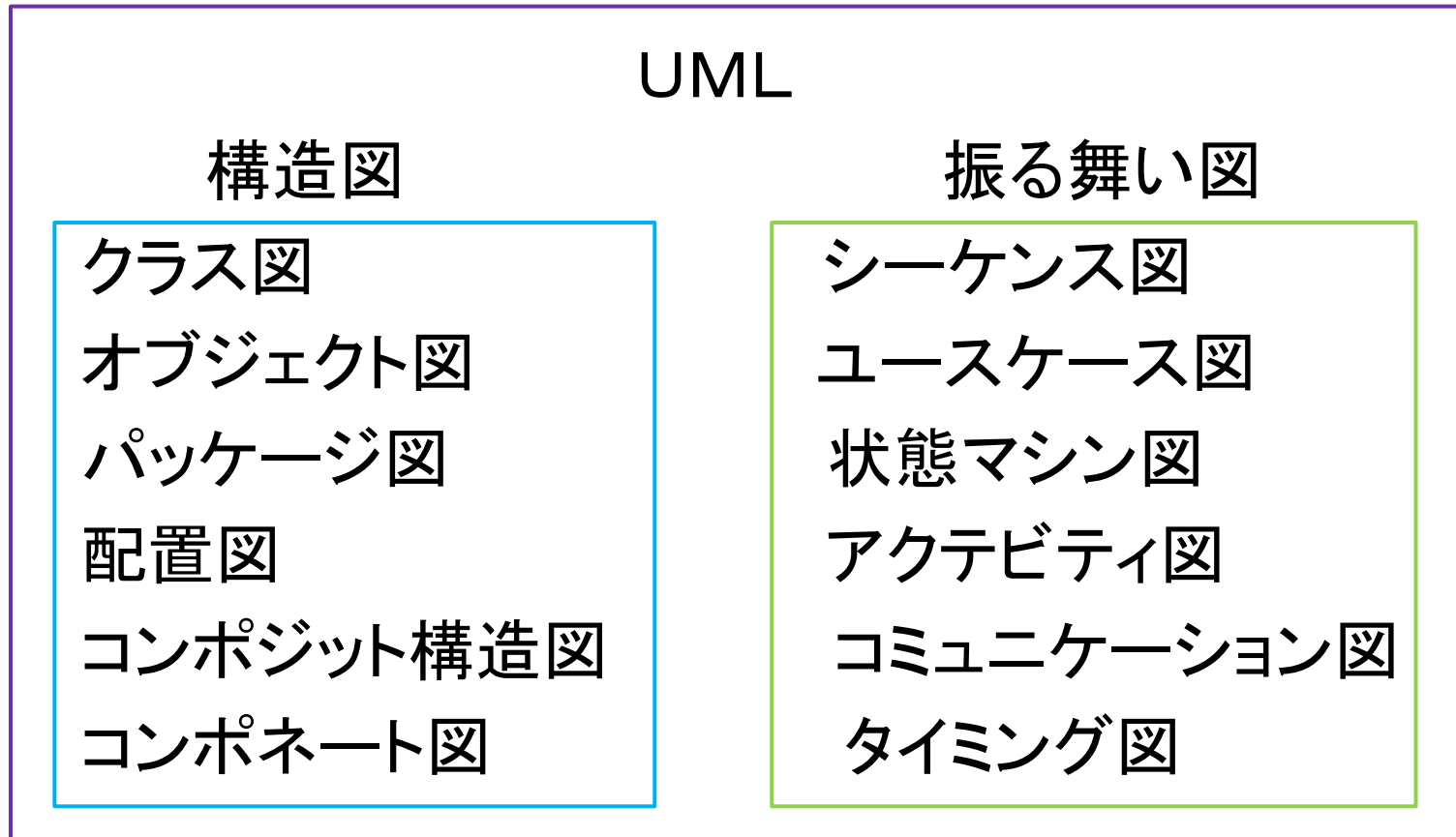
- ▶ オブジェクト記法に基づいて、オブジェクトをモデル化するための記法である。
- ▶ 現在は汎用的なモデリング言語として使われている。
- ▶ UMLの中には複数のダイアグラム(図)がある。



# UMLダイアグラム

---

- ▶ UMLには、構造図と振る舞い図が存在する。



# UMLダイアグラム

---

- ▶ 構造図
  - ▶ 静的な情報
  - ▶ 時間の流れと関係なくその場の状況を表す情報。
- ▶ 振る舞い図
  - ▶ 動的な情報
  - ▶ 時間の流れに沿って変化する状況を表す情報。



# UMLダイアグラム

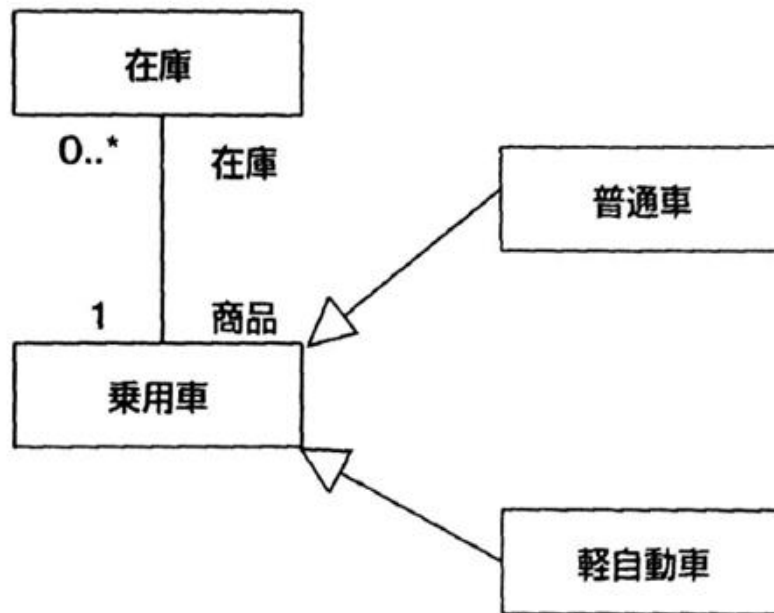
---

- ▶ 全てのダイアグラムを使うわけではない。
- ▶ よく使われる代表的なダイアグラムを説明していく。
- ▶ 構造図
  - クラス図
- ▶ 振る舞い図
  - シーケンス図
  - ユースケース図



# UML-クラス図

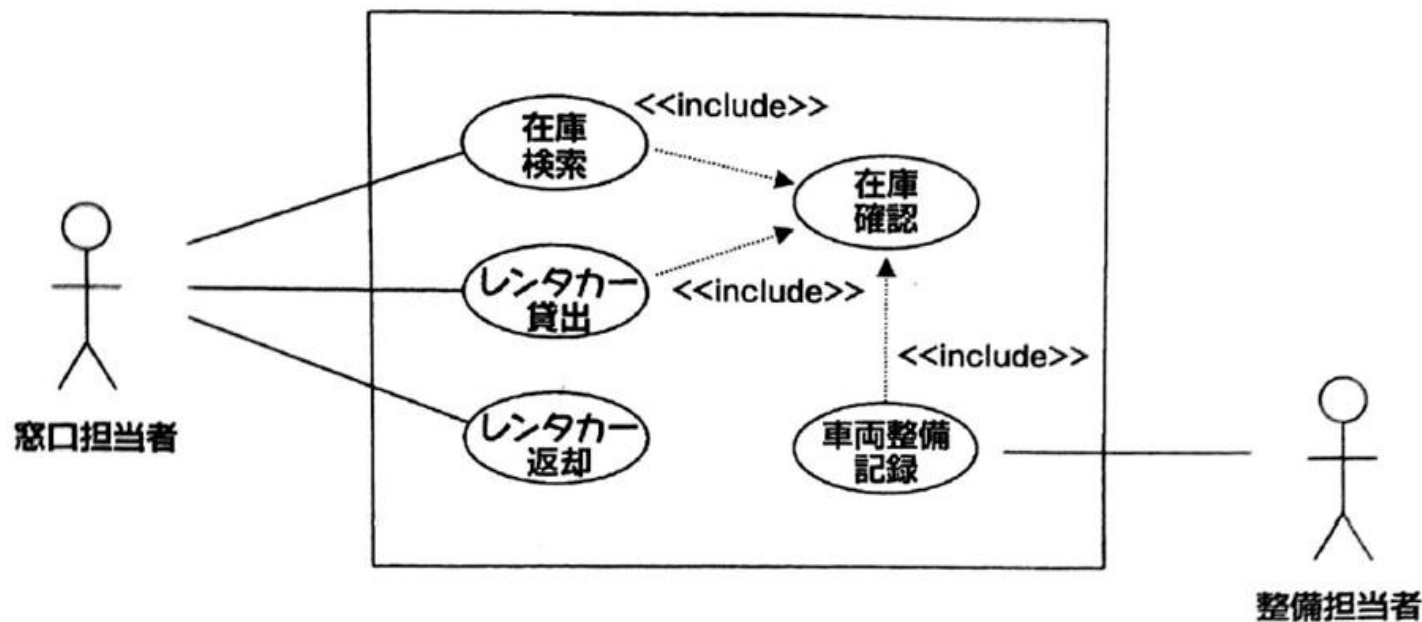
- ▶ どのようなクラス(オブジェクト)が存在し、どのような静的関係を持つかを表したものの。



「乗用車」には「普通車」と「軽自動車」があり、「乗用車」は0台以上の「在庫」がある、という関係を表している

# UML-ユースケース図

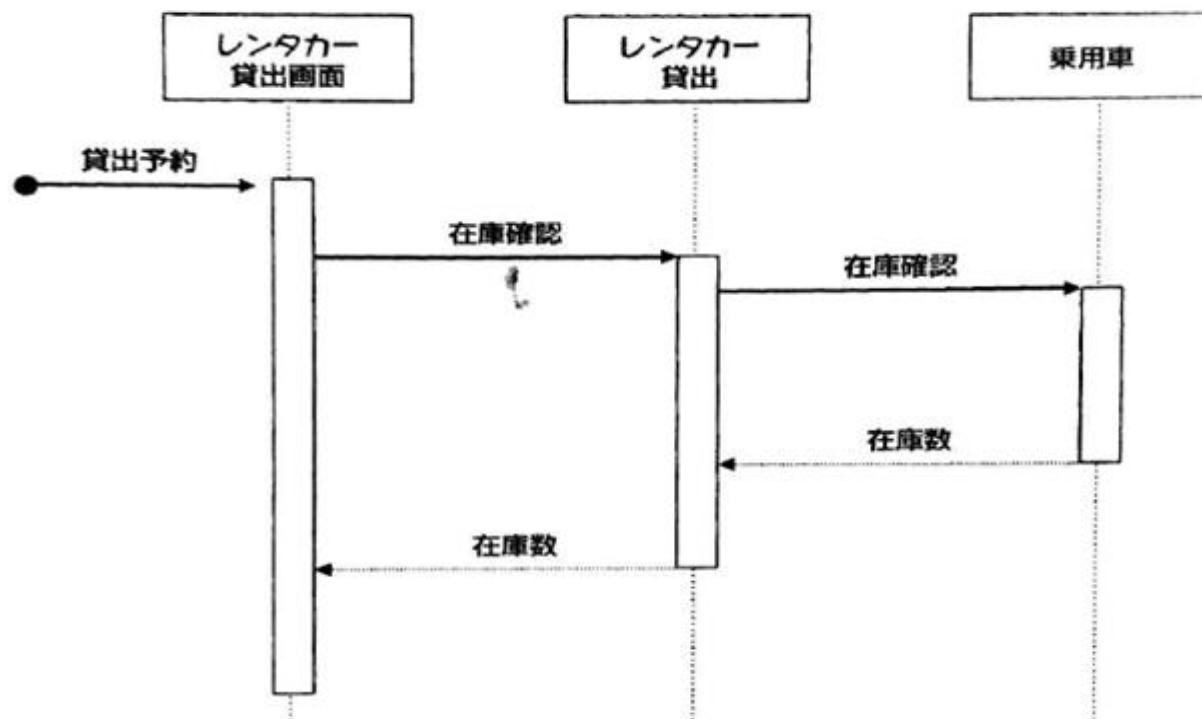
- ▶ システムに関連する要素(利用者や他のシステム)を動的に表したものの。



窓口担当者および整備担当者と、それを使用するレンタカー貸出システムの機能の関係を表している

# UML-シーケンス図

- ユースケースを実現するために必要な、オブジェクト間の動的なやり取りを表したもの。



「レンタカー貸出画面」が「貸出予約」メッセージを受信したら、「レンタカー貸出」に「在庫確認」メッセージを送り、さらに「乗用車」に「在庫確認」メッセージを送り、そこで「在庫数」が返され、さらに「在庫数」をレンタカー貸出画面に返す、という振る舞いを表している

# フローチャート

---

- ▶ 簡潔な図記号を使って処理やデータの流れを視覚的に分かりやすい形で描く図法である。
- ▶ プログラム処理の流れ以外に、業務プロセスや手続きの流れを記述する目的でも使われる。
- ▶ 近年では、細かい図記号を使わず、簡単に書くケースも増えてきている。
- ▶ 図は省略。



# HCPチャート

---

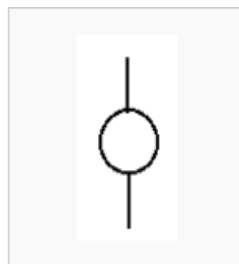
- ▶ 構造化設計に適した図を定規なしで描けるようにNTTで考案されたもの。
- ▶ 構造化を意識した図記号
- ▶ 構造化分析や構造化設計との親和性が高い。



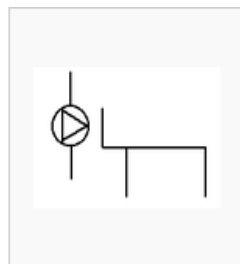
# HCPチャート図記号



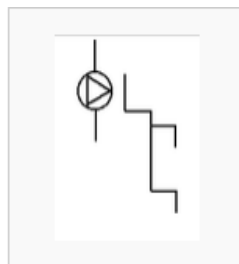
処理の始まり



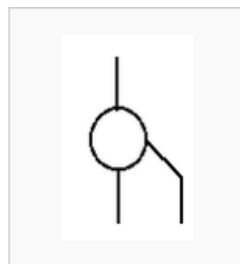
(普通の)処理



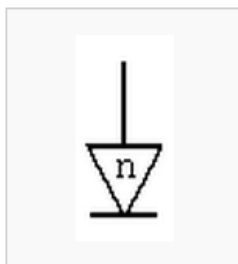
条件分岐(横長の書き方)



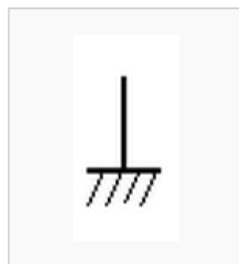
条件分岐(縦長の書き方)



例外条件による分岐



n個の上位レベルの処理の終了(nは数字)



最上位レベルの処理の終了



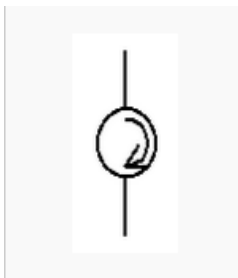
エラーによる脱出



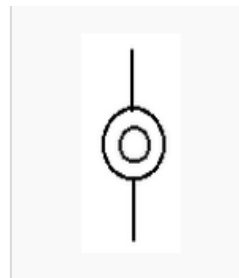
処理の終了



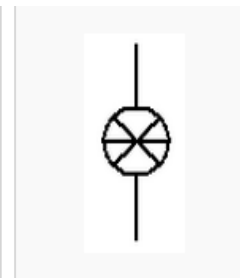
データ



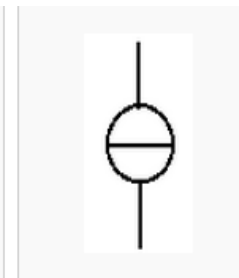
ループ処理



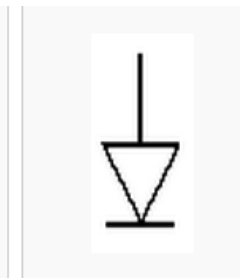
手続き呼び出し(サブルーチン等のコール)



マクロ呼び出し



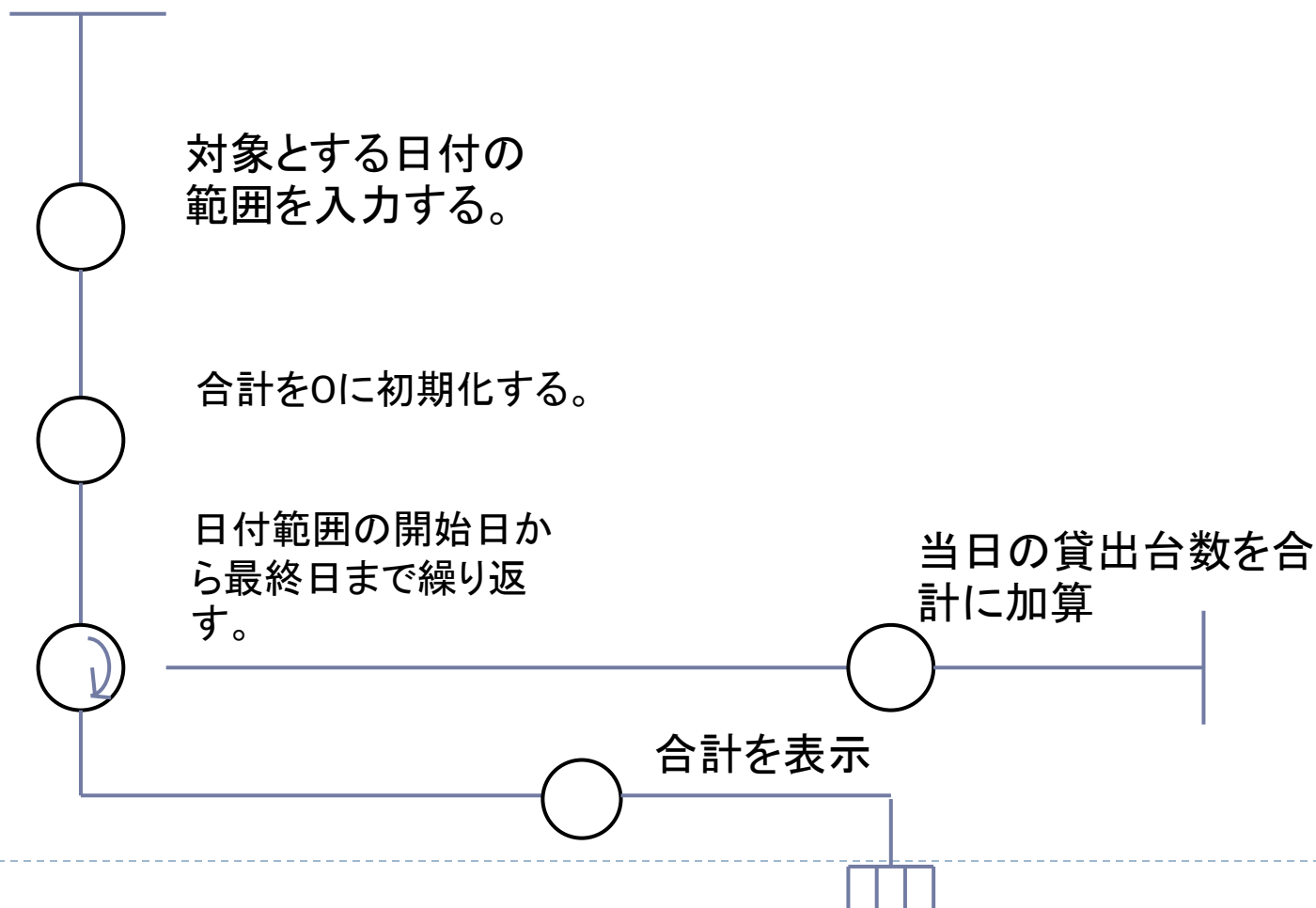
モジュール化(詳細を別紙などに記載するとき)



上位レベルの処理の終了

# HCPチャート

- ▶ 処理内容 「入力した期間の合計貸出台数を計算する。」



# HCPチャートの特徴

---

- ▶ フローチャートと比べると
- ▶ 条件分岐文ではyesとno以外、3分岐以上も表現できる。
- ▶ C言語におけるswitch-case文にも対応しているのでプログラミングしやすい。
- ▶ 処理内容を図形の中ではなく、周囲に書くため、チャート全体をコンパクトにできる。
- ▶ 手で描くことができる。



# まとめ

---

- ▶ システム開発において、読みやすい文書、分かりやすい文書。
- ▶ 句読点、音引きなどを、定められたルールに統一。
- ▶ 紹介したUMLダイアグラム、フローチャート、HCPチャートなどの図を使う。

