## 学習目標

この学習モジュールを受講すると，情報科学演習4の前提知識となる情報システムの開発プロセスとオブジェクト指向について学ぶことができます． 具体的には次のようなことができるように学習しましょう．

- 情報システム開発の「プロセス」を説明できる
- 情報システム開発における「ウォーターフォールモデル」と「反復モデル」の違いを説明できる．
- 「オブジェクト指向」で用いられる基本用語の説明ができる．

---

# 情報システムとは

## 情報システムの定義

「情報システム」は，人間の情報活動を支援するシステムであり，情報の蓄積，検索・加工，伝達のための仕組みです． コンピュータの使用は必須ではないので，コンピュータを使わない情報システムもあり得ます．ただし，通常は，コンピュータとネットワーク， およびそれを制御するソフトウェア，その運用体制までを含んだものを指します．

JIS(日本工業規格)の定義では，図１に示すように，情報システムは情報処理システムとこれに関連する「人的資源」， 「技術的資源」，「財的資源」などの組織上の資源とからなり，情報を提供し配布するものと定義されています．情報処理システムは， データ処理システム及び装置であって情報処理を行うもので，事務機器，通信装置などを含みます．データ処理システムは， データ処理を行う計算機，周辺装置及びソフトウェアです．

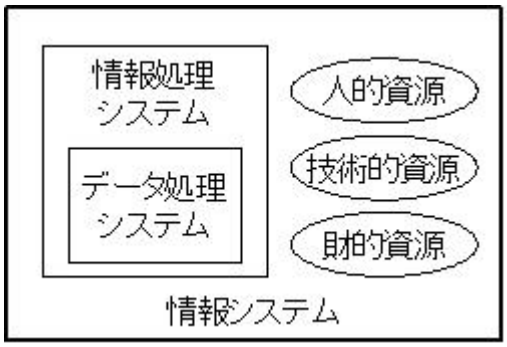



図１ 情報システムの構成

## 情報システムの例

情報システムの例として，以下のようなものがあります．

- オンラインショッピングシステム
- 病院会計システム・電子カルテシステム
- 銀行オンラインシステム
- タクシー配車システム
- 郵便物区分けシステム

他にも，炊飯器や携帯電話も情報システムと見ることができます

一方，ワープロ，表計算ソフト，テキストエディタなど，汎用の目的を持つアプリケーションは情報システムとは言えません． これらのソフトが何かの情報システムの一部に組み込まれることはあり得ます．

## 人を系に含んだシステム

情報処理システム(コンピュータやネットワーク機器，ソフトウェアの部分)のみを情報システムと呼ぶケースもありますが， このような狭い視点で情報システムを開発すると，利用者を無視した使いづらいシステムになったり，作業効率の悪いシステムになってしまうことがあります． 情報システムの設計や開発にかかわる人は，情報システムに人が存在することを常に意識することが重要になります．

---

# 情報システムの開発プロセス

情報システムのニーズが生じ，システムを構築し，運用をするまでの流れを「開発プロセス」と呼びます．情報システムの開発プロセスは， 様々なものが提唱されていますが，おおまかに見るとほぼ同じです．しかし，同じ作業でもいろいろな名称で呼ぶことができますし， どこまでを区切りとして捉えるかといった観点によっても，表現が異なることがあります．

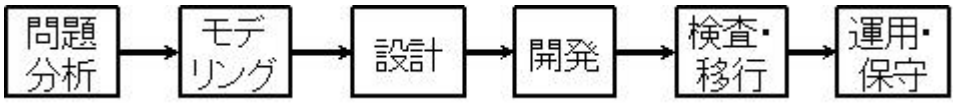



図２ 情報システム開発プロセスの一例

情報システムの開発プロセスを表現したものの一例を図２に示します．

問題分析

現状の作業や現状のシステムの問題点をあげ，それらを分析することで，どのようにシステム化をするかを考察します．分析をした結果， 情報システムを開発しないという結論に至ることもあります．

情報システムを開発することになった場合は，実現のための方向性について，いくつかの視点から検討します．これらの作業には，前提条件や制約条件の明確化， 人的資源とコンピュータ資源の役割分担，予算措置，開発期間などの算定なども含まれます．

モデリング

構築しようとする情報システムのモデルを作成します．この授業で，主に対象とする部分です．

対象となる業務内容を把握するために現状の業務をモデル化し，それを分析した上で，開発する情報システムをモデル化します．現場の取材や聞き取り調査を行い， ユーザや発注者と十分に話し合いを持ちながら，意思や想いにすれ違いが生じないように注意します．

設 計

開発する情報システムを設計します．設計は，一般に概要設計と詳細設計に分けることができます．概要設計ではシステムの全体構成(コンピュータシステム構成， ネットワーク，プラットフォーム，データベース，ソフトウェア開発環境)やユーザインタフェースの設計を行います．詳細設計は，データベースの詳細， プログラムの詳細，ネットワークの詳細を設計していきます．

開 発

プログラミング言語を用いたプログラミングとテストデータを用いたテスト，デバッグを行う工程です．定められた仕様に適合するようにプログラムを作成し， 十分なテストを行います．プログラミング１からプログラミング４などで学んだことが生かされる場面です．

検査・移行

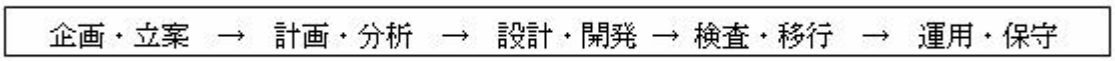
開発が終えたら，ユーザへの納品に向けて最終確認をするためのテスト(受け入れテスト)を実施します．検査に合格したら，業務にシステムを投入します． 万が一うまくいかない場合に備えて，いつでも元のシステムに戻せる体制をとりながら，移行を行うことが重要です．

運用・保守

情報システムが稼動すると運用に入ります．実際に使用してから明らかになる問題点も出てきますが，ソフトウェアの修正やシステムの改善で解決できる問題は， 保守によって修正します．簡単には解決できない問題は，次の情報システムのバージョンアップや，新しいシステムの開発によって解決することになります．

検査(テスト)自体は，各工程で繰り返し行うことに注意しましょう．つまり，問題分析を行ったら，分析結果が適切かどうかを評価します．モデリングや設計においても同様です． プログラム開発工程においては，部分プログラムが正しく動作するかどうか，各プログラムを結合した全体が正しく動作するかどうか，処理時間は要求を満たしているかどうかなどを随時テストする必要があります．

上で述べた開発プロセスを以下のように表現することもあります．




どのような表現をとっても，やるべき作業は同じです．それを具体的にどのように実施するかについては様々な方法論があります．

# ウォーターフォールモデルと反復モデル

情報システムの開発プロセスはいくつもの種類があります． 典型的な二つの用語，ウォーターフォールモデルと反復モデルを抑えておきましょう．

# ウォーターフォールモデル

「ウォーターフォールモデル」は，情報システムの開発プロセスで述べた問題分析から運用・保守までのプロセスを， しっかりと1回，実施するものです．問題分析を上位プロセスとして，滝を水が流れるように， 1段ずつ下に降りていくのがこのプロセスのイメージです．各プロセスにおいて，成果となる文書を完全な状態にまとめて， 次のプロセスに進みます．

したがって，開発中の仕様変更に柔軟に対応することは困難です．また，問題分析から設計において， 開発者とユーザの間で十分な意思疎通を行っていないと，システムができあがってから，実際にユーザが望むものとは異なっていたなどということも生じます． しかし，開発する情報システム像が明確な場合には，今日でも有用な手段です．

問題分析から運用・保守のプロセスを1回実施するモデルには，実装を底として工程をV字型で表したVモデルがあります． 分析や設計をそれに対応するテストと結びつけることを強調した手法です．

---

# 反復モデル

反復モデルは，情報システムの開発プロセスで述べた問題分析から運用・保守までのプロセスを何度も実施します． 1回目にはシステム全体を薄く仕上げ，2回以降で徐々にしっかりと仕上げていく開発モデルです． 個々の反復をイテレーションと呼びます．

情報システムのニーズが生じた際に，システムのイメージを掴むことが難しいものです．反復モデルでは， 1回目のイテレーションで，薄くともシステム全体が仕上がれば，ユーザもシステムのイメージを掴むことができ， ユーザの意図と完成するシステムのギャップを小さくすることが可能でし，その上で，システムの仕様を練り直すことができます． つまり，仕様変更に柔軟に対応することが可能です．また，1回目のイテレーションでシステムの問題点を見出すことで， リスクを早期に回避できます．

反復モデルとして，開発するシステムを複数のブロックに分割し， 一度の反復で１つのブロックを完成させていくインクリメンタルモデルもあります．重要なブロックを最初に開発することで， リスクの早期回避が可能になります．また，システムを漸増的に仕上げるので，仕様変更への対応も可能です．

この授業で学習するUMLを用いたオブジェクト指向開発では，反復型をとることが多いようです． また，反復型開発プロセスでは，繰り返し検証を行うことで品質を向上させることができるという利点もあります．

反復モデルの具体的なものとして，XPやUPなどがあります．

**XP(Extreme Programming)**は，Kent Beck氏らが提唱する開発プロセスで，テスト，実装を重視しており， ペアプログラミング，リファクタリングなど12のプラクティスが明示されています．12のプラクティスの中には， 週40時間以上働かない「40時間労働」(40-hour work)というのも含まれています．XPは小・中規模のソフトウェアの開発に向いた手法とされています． XPのように，ソフトウェア要求仕様の変更などの変化に対して機敏な対応でき、 顧客に価値あるソフトウェアを迅速に提供することを目的とするソフトウェア開発方法論の総称をアジャイルプロセスと呼びます．

**UP(Unified Process)(統一プロセス)**は，ユースケース駆動，アーキテクチャ中心，管理された反復，カスタマイズ可能， 4つのフェーズという特徴がある反復モデルです．4つのフェーズとは，方向付け，推敲，作成，移行であり，これらを繰り返すことでシステムを完成させます． ラショナルソフトウェア社が提唱する開発プロセスRUPはその具体例で，ビジネスモデリングからテストまでの一連の開発作業工程における成果物やガイドライン等が詳しく定義されています． UPは反復モデルですが，ヘビーな手法のため，通常，アジャイルプロセスとは呼ばれていません．

## 情報システム セルフテスト

次のページに進み，「情報システム」に関するセルフテストを受験して，理解を確認しましょう．

# オブジェクト指向の概念

UMLを習得するには，プログラミング３で習ったオブジェクト指向の考え方を知っていることが必須となりますので， 簡単に復習しておきましょう．

---

# オブジェクト指向の概要

オブジェクト指向XXでは，物理的，概念的にまとまった「もの」をオブジェクトとして定義していき，それらを中心にXXを進めます． XXには，プログラミング，モデリング，設計，開発などの用語が当てはまります．

オブジェクト指向における「オブジェクト」は， 属性(データ)と振る舞い(手続き)から構成されます．さらにオブジェクトに適切な名前が付けられて， そのオブジェクトが何に分類されるかを表すことができます．オブジェクトを定義するとき，そのオブジェクトがどのような「属性(データ)」と「振る舞い(処理手続き)」を持つのかを決める訳ですが，その際に，そのオブジェクトの債務，つまり，責任を持つ範囲を明確にすることが重要になります．

「オブジェクト指向」は，クラスとインスタンス，カプセル化とメッセージ送信，継承などから特徴付けられます．

---

## クラスとインスタンス

クラスはオブジェクトを抽象化して定義したものです．つまり，クラスはオブジェクトを概念として表したもので，クラスには，その概念が何という名前で，どのような属性を持ち，どのような振る舞いができるのかが定義されます．

インスタンスは，あるクラスに属する具体的なオブジェクトです．オブジェクト指向プログラミングでは，クラスに，そのクラスがどのようなデータと手続きを持つかを定義し，クラスからnew演算子でインスタンスを生成し，インスタンスを変数に入れて使用しましたね．オブジェクト指向モデリングでは，例えば概念として表現した「顧客」がクラスで，具体的な顧客としてのAさんがインスタンスといった関係になります．

多くの場合は，オブジェクトと言えば，具体的なインスタンスのことなのですが，オブジェクトは概念としてのクラスを指すこともありますし，具体的なオブジェクトとしてのインスタンスを指すこともあるので注意しましょう．

---

## カプセル化とメッセージ送信

カプセル化はオブジェクトの持つ属性や振る舞いを，オブジェクトの外から直接参照できるものと，参照できないものに明確に分けて，外部に公開した操作のみでアクセスを可能にする仕組みです．

カプセル化は「情報隠蔽」と呼ばれることがあり，属性や振る舞いを外から隠すことと言われる場合がありますが，外部に公開するインタフェースを明らかにする点の方が重要です．

外部からのインタフェースを明確にすることは，プログラミングにおいては，オブジェクト内のデータへの不法なアクセスを防いだり，インタフェースを変えずに実装を変えることができるなど，保守性や拡張性を高める利点があります．モデリングや設計においては，そのオブジェクトに対してどのような操作が行えるのかを，実装方法まで踏み込まずに定義していくことができます．

カプセル化によって，オブジェクトへのアクセスは，インタフェースとして公開されているメッセージを送信して行うことになります．オブジェクト同士でメッセージをやりとりすることで処理が進みます．このようなメッセージのやり取りをメッセージ送信と呼びます．

---

## 継 承

継承は，あるクラスの資産(属性と振る舞い)をすべて受け継ぐことです．オブジェクト指向プログラミングでは，継承における元のクラスをスーパークラス，資産を継承して新しく作成したクラスをサブクラスと呼びます．

Javaでは extends というキーワードを使って，サブクラスを作成しましたね．例えば，Java Appletを作成するときに，Appletというクラスを継承して，独自のJava Appletを作成するのでした．このように元から存在するクラスをスーパークラスとして，サブクラスを作成する場合があります．この様子を図３に示します．スーパークラスからサブクラスに，より特殊化することを特化（**specialization**）と言います．

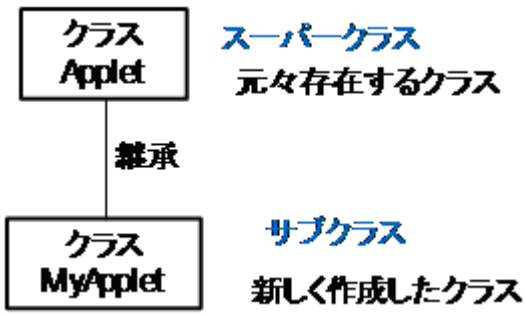



図3 **Applet**クラスを継承して独自のアプレットクラスを作成

一方，図4のように，似たようなクラスが存在する場合に，共通部分を抽出してスーパークラスを作成することでプログラムコードをシンプルにすることもあります．サブクラスからスーパークラスに，より一般化することを汎化（**generalization**）と言います．

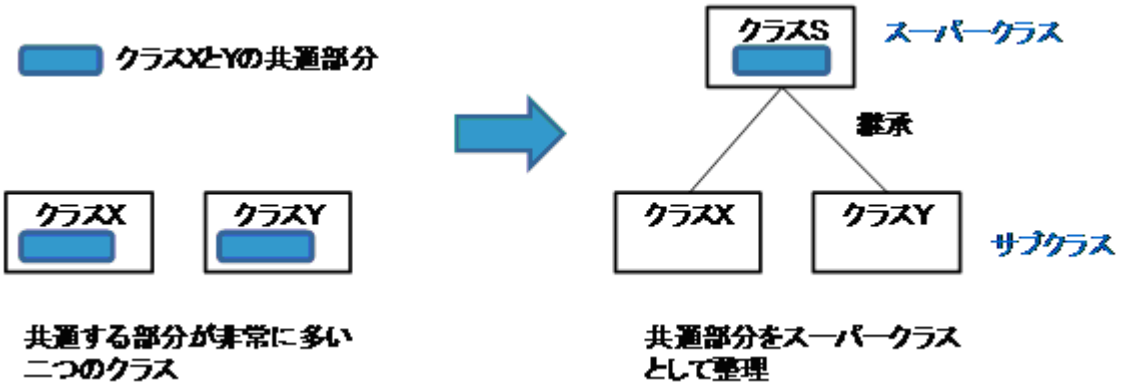



図４ 類似のクラスをまとめて一般化してスーパークラスを作成

モデリングの段階では，あるオブジェクトが持つ資産(属性と振る舞い)が明確にわかっていないことが多いので，資産を受け継ぐというよりは，概念として上位・下位の関係がある場合に，この継承の考え方を使います．例えば，「自動車」は「乗り物」の一種であるので，「乗り物」をスーパークラス，「自動車」をサブクラスとします．このような関係は，**is_a関係**と呼ばれます．「自動車は乗り物( An automobile is a vehicle)」ということです．

## 可視性

オブジェクト指向では，カプセル化によって，属性や振る舞いを公開にするか，非公開にするかを明確に定義しておくことが重要と述べました．公開するか，非公開とするかの指定を可視性と言います．Javaプログラムにおいては，以下の4つがあります．UMLにおいても同様な指定ができます．

| キーワード | 意 味 |
|---|---|
| public | 公開部．クラスの内外，どこからでもアクセス可能 |
| package-private | パッケージ内からのみアクセス可能．キーワードを書かなければこの指定となる |
| protected | 保護部．そのクラスとそのクラスのサブクラスからのみ，アクセス可能 |
| private | 非公開部．そのクラス内からのみ，アクセス可能 |

# オブジェクト指向 セルフテスト

次のページに進み，「オブジェクト指向」に関するセルフテストを受験して，理解を確認しましょう．

## まとめ

このモジュールでは，以下のことを学びました.

- 情報システムは人を系に含み，情報処理システム，人的資源，技術的資源，財的資源から構成される
- 情報システムの開発は，問題分析，モデリング，設計，開発，検査・移行，運用・保守といった「プロセス」をたどる
- 開発プロセスには，「ウォーターフォールモデル」と「反復モデル」があり，オブジェクト指向開発では「反復モデルが多用される
- 「オブジェクト指向」における用語
  - クラス，インスタンス
  - カプセル化，メッセージ送信
  - 継承，スーパークラス，サブクラス
  - 可視性，private, protected, public

---