**UML概要：**

プログラムを作るには設計書が必要です。設計書はプログラムの構成を関わる人々に分かるように作るのが重要。

モデリング：抽象化された概念を基に、システムの全体像を分かりやすく表現することです。

* システムの対象となるユーザーの業務を理解する必要がある
* ユーザーの業務とみっちり繋がるところをモデルの中心に置いて、変わりやすい部分を外側に置くのが重要
* UML（Unified Modelling Language）はモデルの表記方法を統一・標準化された言語です。
  + 表現力が高い
  + 分析、設計と実装する環境を正確に伝わります
  + 分かりやすい

開発の現場の課題：

* 要求の高度化と多様化
  + 利用者が求める機能の高度化と多様化
* システムの大規模化と複雑化
  + システムの拡張（scaling）
  + 他のシステムとの連携、それに伴う複雑さ
* 関係者間での情報交換の難しさ
  + 喋る言語の違い
  + 複数人の情報共有・交換する難しさ

開発の統一：

* 表記法：UML
* 開発プロセス：Waterfall型, Spiral型, 反復型
* 考え方：Object指向

Object指向：

* システム構成の中心となる物をオブジェクトとして扱う
* オブジェクト指向は、オブジェクト同士が協調動作でシステムの処理が進む
* オブジェクトとは、物理的・概念的に分けられるもの・ことを表す
  + 目に見える物：カメラ、人
  + 目に見えないもの：顔情報、追跡情報、データベース情報
  + 属性（データ）と操作（関数）を含む
* オブジェクトの抽象化した概念はクラスと言う
* クラスを用いてオブジェクトを作成される

Encapsulation（カプセル化）:

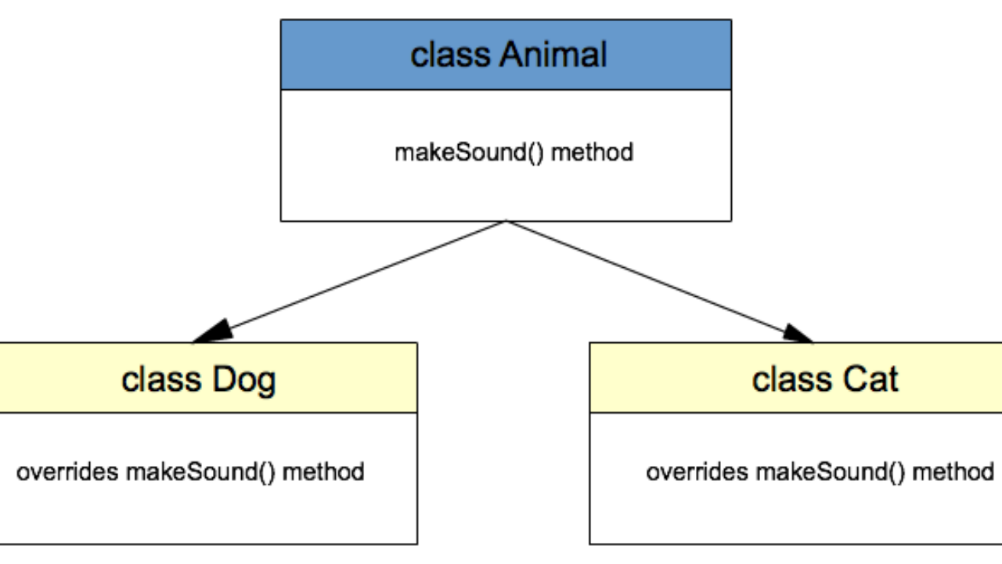
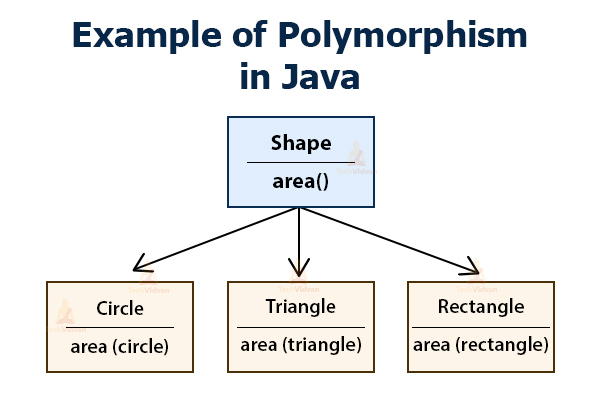
* プログラムをモジュール化して、公開部分（Public）と非公開部分（Private）に分ける仕組みです
* 非公開部分は公開されている操作のみでアクセス出来る
  + 外部から直接アクセスを否定される

Inheritence（継承）：

* あるクラスが持っている機能を引継ぎ、更に独自・特化した機能を持つことです
* 親のクラスで一般的機能があって、子のクラスで親の機能と共に独自機能がある
  + 先生クラス（親）と教授クラス・准教授クラス

Polymorphism：

* 同じ機能を持つ関数が入力データによって、適当な対応すること、または同じ入力データに対してそれぞれ異なる操作をすることです



UMLの基本：

* オブジェクト指向を基に、システム設計を表記する方法です
* 様々な設計書をUMLで表現出来る
* プログラミング言語に依存しない、相互対応することが出来る
* 図の体系：
  + 構造図：クラス図、オブジェクト図
  + 振る舞い図：ユースケース図、シーケンス図