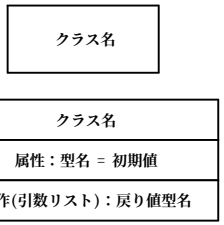
chapter-1

UML1.3記法一覧

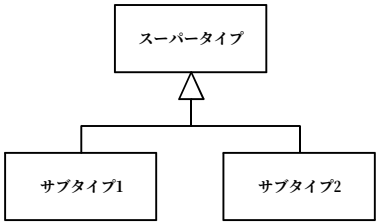
■ クラス



■ 可視性

```
+public
#protected
-private
$static
```

■ 汎化



■ 制約　　{制約の記述}

■ プロパティ　　{タグ＝値}

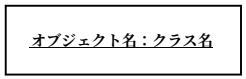
■ ステレオタイプ　　<< ステレオタイプ名>>

ステレオタイプ名や制約には，独自のものを使ってかまいません．プロジェクトやグループでいくつか定義するのが一般的です．

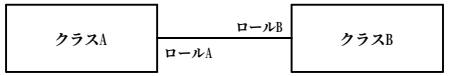
クラスにステレオタイプを付けると，メタモデル上のクラスにサブクラスを作成したことになります．

■ ノート

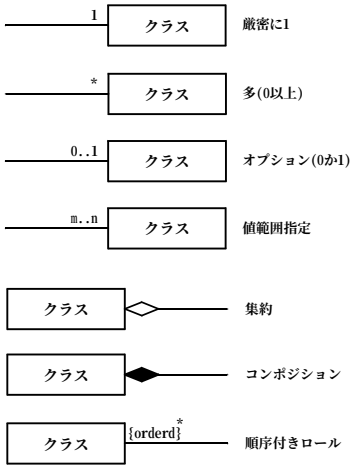
■ オブジェクト



■ 関連

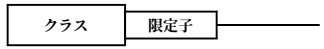


■ 多重度



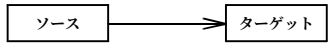
概念モデリングの際には特に多重度に敏感になりましょう．
重要なヒントが隠されていることがあります．

■ 限定子付き関連

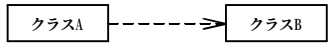


1対多の関連が，限定子によって1対1になります．
多側を一意に識別するためのキーです．

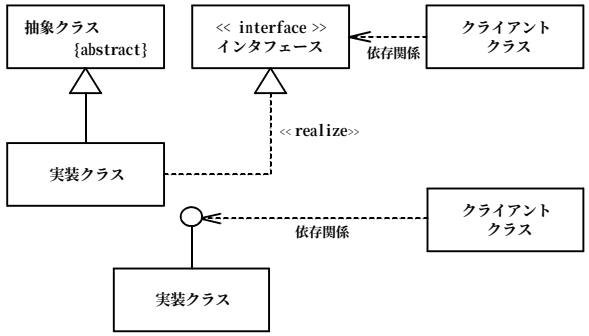
■ 誘導可能性



■ 依存関係

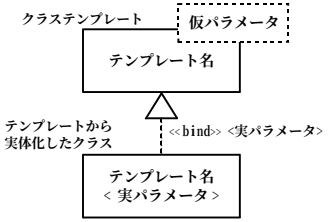


■ クラス図：インタフェース



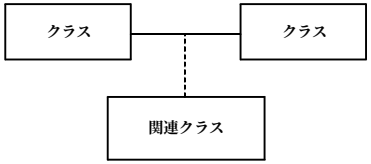
インタフェースには白丸のアイコンを使うと見やすく描くことができます．
<<interface>>ステレオタイプと同義です．

■ クラス図：パラメータ化されたクラス

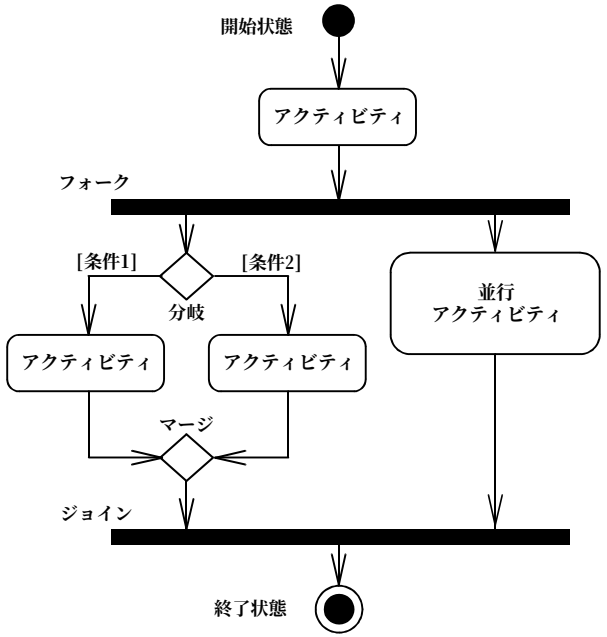


C++のtemplateのような総称性（genericity）を表現します．

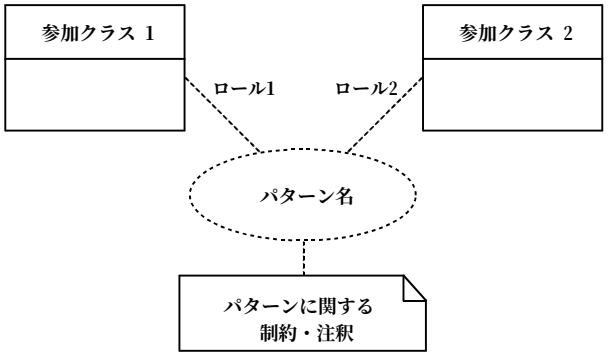
■ 関連クラス



■ アクティビティ図

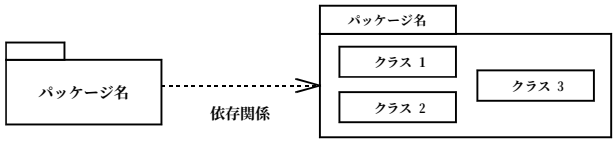


■ パターンの記法（パラメータ化コラボレーション）

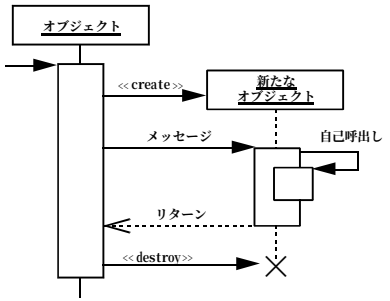


デザインパターンのような，名前の付いた典型的設計パターンの表出を表現します．

■ パッケージ図

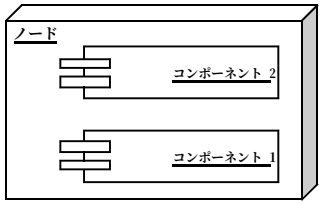


■ シーケンス図



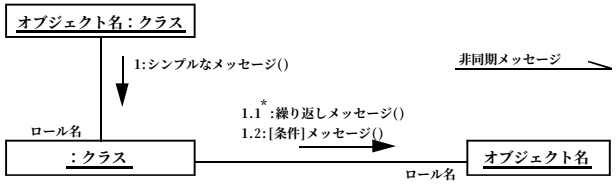
シーケンス図はオブジェクト同士の動的なメッセージ伝達を，時間の流れを縦軸にして表現しています．

■ 配置図



ノードとは物理的な計算機を指します．コンポーネントは，ファイル，実行形式，ライブラリなどです．

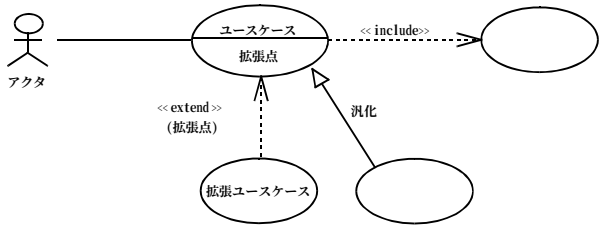
■ コラボレーション図



非同期メッセージとはリターンを待たない呼び出しです．呼び出し元はブロックせずに次の処理に進むことができます．

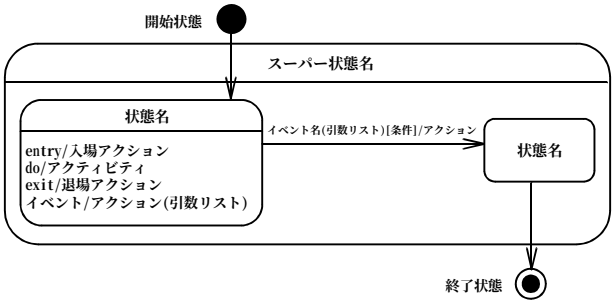
コラボレーション図は，シーケンス図と同じ内容を表現しますが，オブジェクトを空間的に配置することで，協調関係を表すのに適しています．

■ ユースケース図

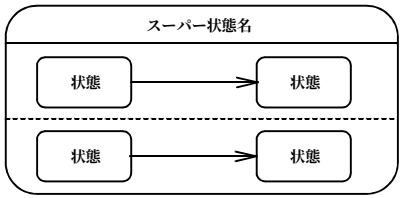


ユースケースの拡張点をオーバーライドしたものが，拡張ユースケース（<<extend>>）です．複数のユースケースから共通して利用されるユースケースをとり出し，<<include>>で呼び出すことができます．さらに，汎化によってユースケースのバリエーションが表現できます．

■ ステートチャート図



■ 並行状態



ユースケーステンプレート

ユースケース名	[ユースケースの名前]
目的	[ユースケースを実行することで達成する目的]
アクタ	[ユースケースの関与者]
事前条件	[ユースケースを実行するための前提条件]
主シナリオ	[目的が達成されるまでの処理手順]
事後条件	[ユースケースが実行されることで満たされる条件]
代替シナリオ	[主シナリオ実行中に発生した例外に対する処理]
関連ユースケース	
著者	
作成日/更新日	
備考	[どの項目にも分類できなかったことを記述．メモや注意点など]

ユースケースはあまり細かくブレークダウンすべきではありません．また，ユーザが分かる言葉で記述すべきです．ユースケースを切り出す際には，1つのユースケースに明確な1つの目的（ゴール）を持ち，その目的の達成を持ってアクタが満足するようにします．このユースケース切り出しの指針を，“Actor's Satisfaction”といいます．

ステレオタイプ

名前	適用先	説明
<<actor>> (アクタ) 	クラス (Class)	システムと相互作用する外部オブジェクト群.
<<bind>> (バインド)	依存性 (Dependency)	テンプレートに実パラメータをバインドした実体.
<<call>> (呼び出し)	依存性 (Dependency)	操作の呼び出し.
<<create>> (生成)	操作 (Operation)	クラシファイヤのインスタンス生成.
<<destroy>> (破棄)	操作 (Operation)	クラシファイヤのインスタンス破棄.
<<facade>> (ファサード)	パッケージ (Package)	このパッケージは，他のパッケージのモデル要素のみ参照する．他のパッケージの「パブリックビュー」を提供するために使う.
<<framework>> (フレームワーク)	パッケージ (Package)	フレームワーク・パッケージ.
<<global>> (グローバル)	パッケージ (Package)	対応するパッケージの可視が，グローバルスコープであることを示す.
<<interface>> (インタフェース)	クラス (Class)	このクラスは，他のクラスによって提供されたサービスを定義するために使う操作を定義する.
<<library>> (ライブラリ)	コンポーネント (Component)	このコンポーネントは，静的もしくは動的なライブラリを表現する.
<<process>> (プロセス)	クラシファイヤ (Classifier)	このクラシファイヤは，ヘビーウェイトなアクティブクラスを示す.
<<refine>> (洗練)	依存性 (Dependency)	この依存性は，2つの要素が異なるセマンティクスの抽象レベルにあることを示す.
<<signal>> (シグナル)	クラス (Class)	遷移を誘発するために使われる名前のシグナルを定義するイベントクラス.
<<system>> (システム)	パッケージ (Package)	システム全体を記述する．パッケージ階層のルートである.
<<thread>> (スレッド)	クラシファイヤ (Classifier)	このクラシファイヤは，ライトウェイトなアクティブクラスを示す.
<<trace>> (トレース)	依存性 (Dependency)	この依存性は，2つの要素が同じ概念をセマンティクスの異なるレベルまたはビューの相違点から表現することを示す．要件を追跡するため，またはモデル要素の変化を追跡するために使われる.
<<type>> (型)	クラス (Class)	このクラスは，オブジェクトに適用できる操作群を示す．このクラスは属性，操作，関連を持つことがあり，実装は持たない.
<<boundary>> (境界) 	クラス (Class)	このクラスは，システム外のアクタとシステム内のオブジェクトとのインタフェースである.
<<entity>> (エンティティ) 	クラス (Class)	このクラスは，受動的で，データベースやファイル等により永続化されることが多い.
<<control>> (制御) 	クラス (Class)	このクラスは，オブジェクトの相互作用を制御するオブジェクトを定義する.

制約

名前	適用先	説明
{abstract}	クラス (Class)	抽象クラスであり，インスタンス化されることはない.
{active}	オブジェクト (Object)	このオブジェクトは制御のスレッドを所有する.
{complete}	汎化 (Generalization)	すべてのサブタイプは指定され，いくつかは制御され，削除されるが，追加のサブタイプは許されない.
{frozen}	関連の終端 (Association end)	オブジェクトが生成され，初期化された後に，リンクはオブジェクトから追加，削除，移動できない.
{guarded}	操作 (Operation)	この操作は並列スレッドから同時に呼ばれることがあり，1つのスレッドとのみ通信できる.
{incomplete}	汎化 (Generalization)	すべてのサブタイプが指定されるわけではないが，いくつかは省略され，削除されるが，追加のサブタイプは許される.
{or}	関連 (Association)	どれか1つの関連が使われる．全く使われない可能性もある.
{xor}	関連 (Association)	どれか1つの関連のみが必ず使われる.
{ordered}	関連の終端 (Association end)	対応する要素は，複製が禁止される順序を持った集合になる.
{overlapping}	汎化 (Generalization)	サブタイプはさらなるサブタイプによって重複して継承されることがある.
{sorted}	関連の終端 (Association end)	対応する要素は，内部の値に基づいてソートされる.
{transient}	クラス役割 (Class role)	モデル要素は，相互作用の実行の間，作成され，破棄される.
{unordered}	関連の終端 (Association end)	対応する要素は，複製が禁止された順序のない集合になる.