システム開発におけるUMLの活用

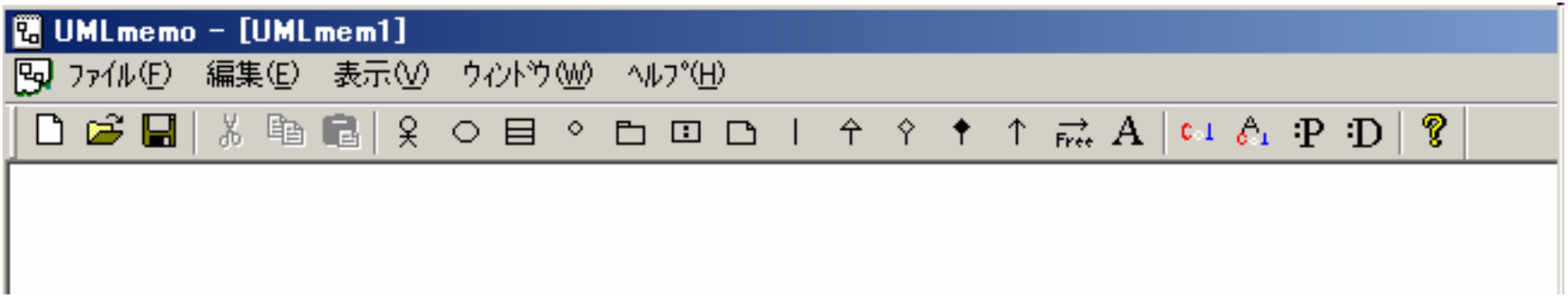
(情報システム開発論、第5回講義)

URL　http://homepage3.nifty.com/suetsuguf/
Email　fwhy6454@mb.infoweb.ne.jp

作成者　末次　文雄

参考：UMLmemo　ダウンロードサイト

- UMLのエディタ
- メモ帳感覚で利用できる
- 下記のアイコンをクリックして、作業域を使用



無料のダウンロードサイト：

(株)ベクター　　http://www.vector.co.jp/soft/win95/prog/se215535.html?site=n

目次（システム開発におけるUMLの活用）

注記：主として開発の上流工程を対象に説明します。

１．オブジェクト指向の特徴（復習）

１．１　オブジェクト指向の概念

①オブジェクト指向の考え方

・現実のモノ（オブジェクト）およびモノ同士の関係を、そのままソフトウェアで表現することによって、現実世界の仕組みを、コンピュータ上で再現するもの。

・現実のモノを中心とした考え方

　　－人間の発想、活動に近い考え方でシステムのモデリングを行うもの

　　－目的志向の即物的な考え方

・DOAとの大きな違いのある点

ーDOAが、データをモデリングの基本にしているのに対し、
ーOOは、データと操作を一体（オブジェクト）としてモデリングの基本にしている。

ーDOAでは、実体（エンティティ）間の関係は、データ間の関係であるが、
ーOOでは、オブジェクト間の関係は、メッセージのやりとりをして、状態を変化させている。

②オブジェクト、クラス

・オブジェクトは、特定のアプリケーションにおいて、一つ一つを識別すべき対象であり、
- ー実際に存在するモノ、概念
- ーその単位は、ERモデルでいうエンティティと近似している。
- ー固有の姿、形、性質などの属性を持つ
- ー固有の振る舞い（＝操作）を持つ
- ー他のオブジェクトとの間に固有の関係を持つ（＝関連）

・クラスは、類似性の高いオブジェクトから共通の特性を抜き出して、定義したもの。
- ーいわば、オブジェクトの集合であり、抽象化したもの。

③情報隠蔽、カプセル化、メッセージ

・オブジェクトの利用者は、メッセージを送ることにより
属性の検索などの操作の実行を依頼する。
これを情報隠蔽（カプセル化）という。
・通常は、オブジェクトの操作名をメッセージとする。

④　継承、汎化、特化

・下位クラスは、上位クラスの特性を引継ぐ
（継承という）
・特化とは、新に下位クラスを導き出すこと。
・汎化とは、定義済みのクラスの共通的な特性に
着目し、共通部分を取り出し、
上位クラスとして導き出すこと。

⑤多態性（ポリモアフィズム）

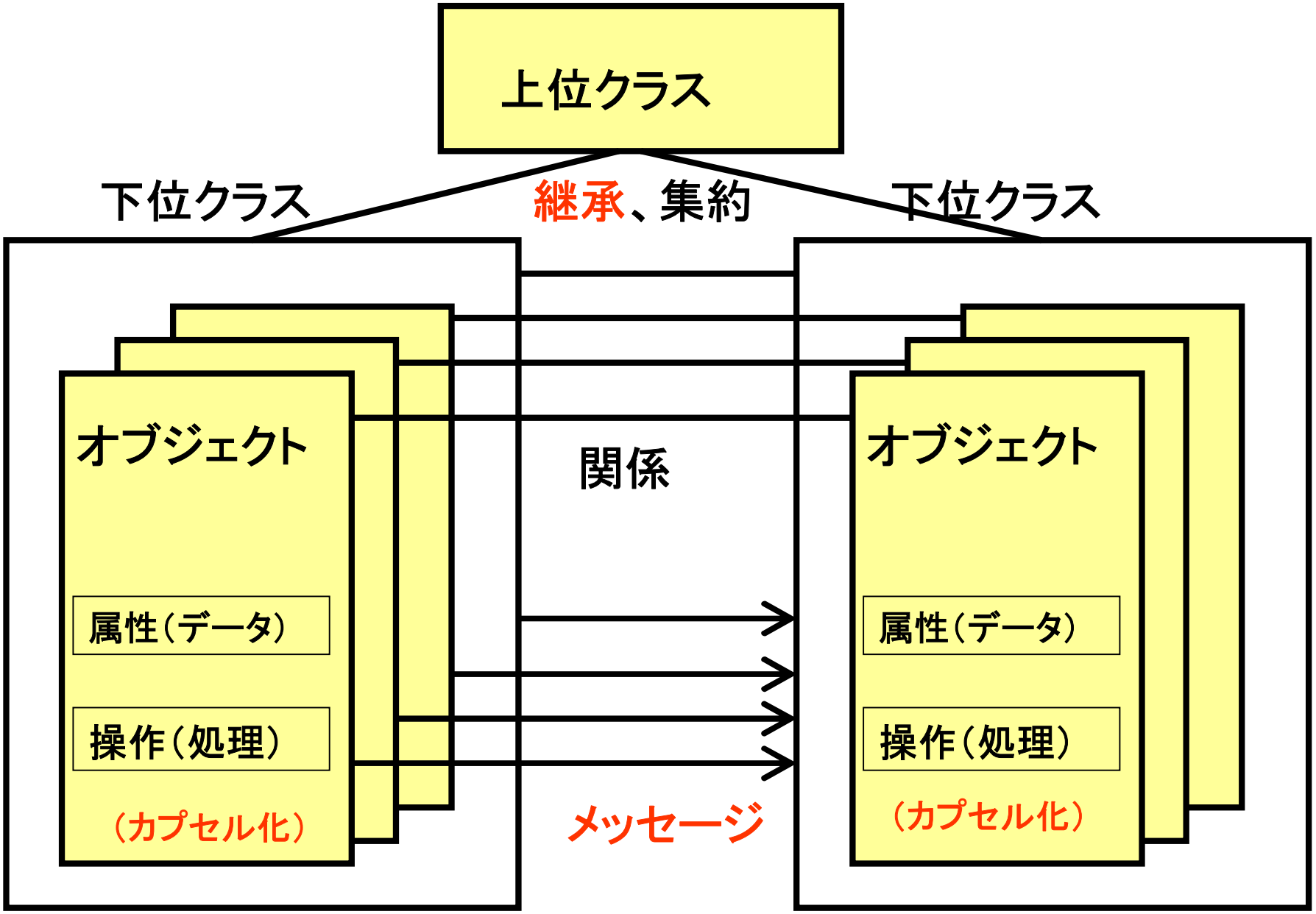
・継承によって、下位クラスは、上位クラスからの同じメッセージに対して、それぞれ下位クラスで定義した、異なる操作を実行できる。

⑥集約（aggregation）

・オブジェクトが他のオブジェクトを包含しているときに、二つのオブジェクトは、集約の関係にあるという。
（所有の関係ともいう）

・クラスが複雑な場合は、この集約の概念で、シンプルなクラスに分解することが出来る。

１．２オブジェクト指向の全体像



２．オブジェクト指向とUML

２．１　UMLの種類と用途

・UMLにおけるビュー

ービューとは、対象をモデリングする際の視点のことである。

ー特定の観点、立場から対象を俯瞰すること。

①ユースケース・ビュー

モデルの外部にいる使用者の観点に立ち、使用者に提供する機能を表現する。

②論理ビュー

モデリング対象の内部構造を表現するものであり、静的構造と動的構造から成る。

③コンポーネント・ビュー

実際に動作するプログラムの構成を表現する。

④その他実装に関するビュー

プロセス・ビュー、配置ビュー

▪UMLの分類と用途

			ダイアグラム	役割
機能モデル	機能構造	1	ユースケース図	システムの利用者から見た、外部機能、その順序を定義する。あわせてシステムの境界を定義。
論理モデル	静的な構造 （ｼｽﾃﾑ構造） （概念モデル）　（ｵﾌﾞｼﾞｪｸﾄﾓﾃﾞﾙ）	2	クラス図	概念や静的なクラス間相互関係を表現し、 モデルの構成部品の集まりである。
		3	オブジェクト図	システムのある時点でのオブジェクト状態のスナップショット。
		4	パッケージ図	モデル要素の階層的グルーピングであり、モデルの構成を管理する。（クラス図の特殊形）
	動的な構造 （振る舞い、状態）	5	シーケンス図(相互作用図)	オブジェクト間のメッセージ交換の時系列表現
		6	コラボレーション図（相互作用図）	オブジェクトの相互作用を直接に表現する。
		7	状態図（ステートチャート図）	オブジェクトの取りうる状態，遷移などライフサイクルを表現する。
		8	アクティビティ図	システムの動作の流れの表現。状態図の一種
実装モデル	物理的な構成	9	コンポーネント図	システムを構成する実行可能モジュールやソースコードの構造を表現
		10	配置図	システムを構成するマシンや装置（CPU、デバイス等）の分散配置を表現

２．２ UMLによるモデリングの効果

- どのようなシステム仕様を持っているか表現可能。
- システム化対象を目に見える形で表し、
  開発者間でモデルの共有化ができる。
- システムの実装に関しての指示になる。
- 将来、自動プログラム製造の可能性がある。
- システムの仕様を標準化された文書で表し、
  運用・保守時に利用できる。

２．３　開発工程とUML

以下に、どの開発工程で、どのUMLダイヤグラムを使用するかを示す。

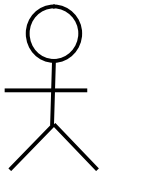
開発工程	使用するUMLダイヤグラム
要件定義	・ユースケース図、および記述書
外部設計 （分析フェーズ）	・クラス図、オブジェクト図、パッケージ図 ・シーケンス図 ・必要に応じて、コラボレーション図、 状態図、アクティビティ図 ・ユースケース図の詳細化、クラス仕様
内部設計 （設計フェーズ）	・コンポーネント図 ・配置図 ・追加項目をクラス図、シーケンス図等 に反映。クラス仕様書。

具体的なOO開発工程

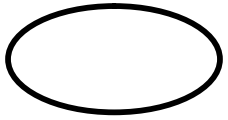
- 第４回の講義テキストの第４章を参照
  - ー要件定義（要求分析）
  - ー外部設計（システム分析）
  - ー内部設計（システム設計）

３．　UMLの表記法

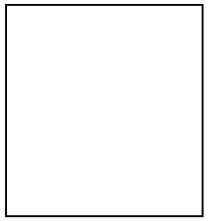
①ユースケース図、ユースケース記述



アクター:　システムの使用者、または関係する他システム。



ユースケース:　アクターにより起動されるシステム内の機能。
大きな単位であり、一連のシーケンスを含む。

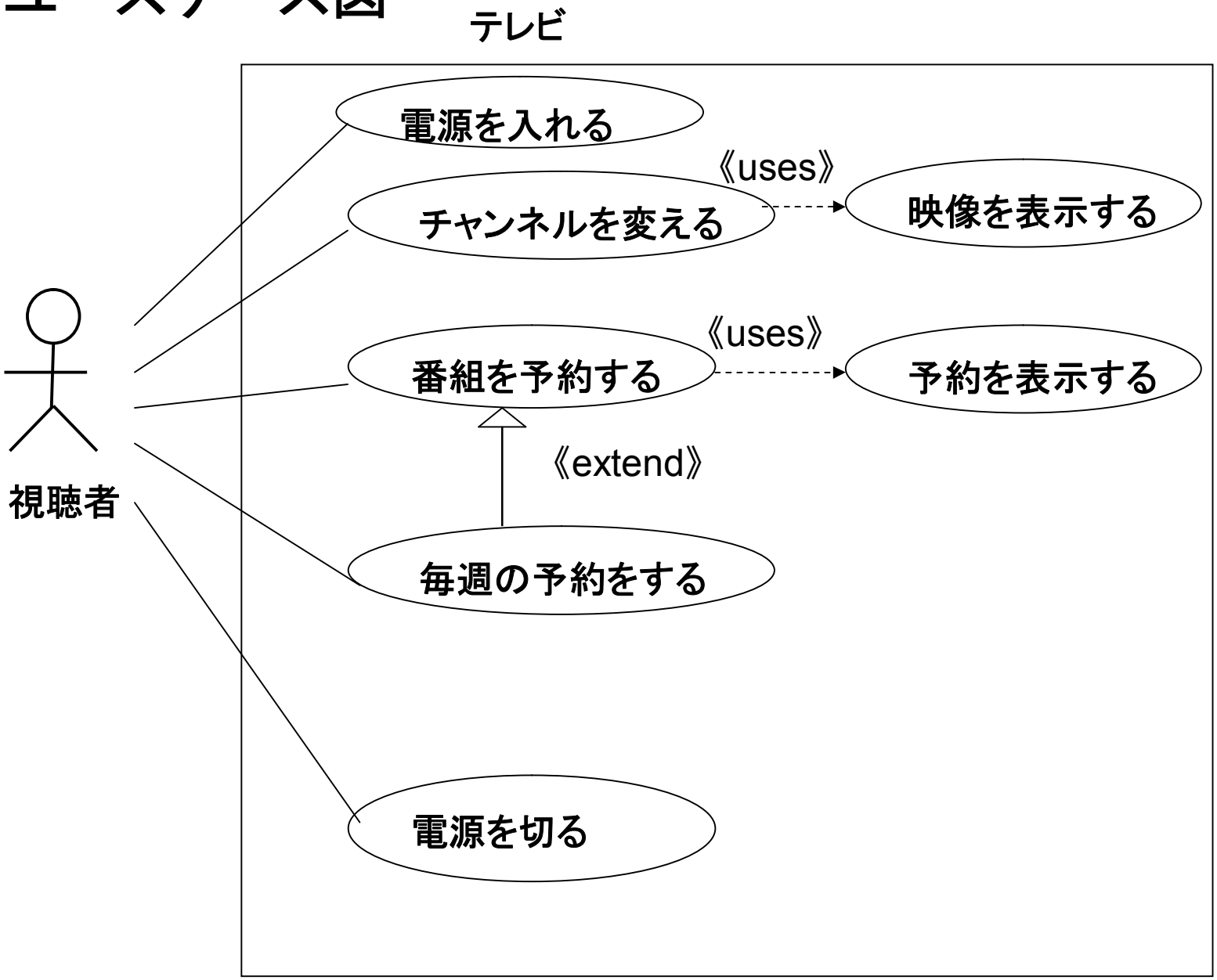


システム境界:　対象システムの範囲を示す。
アクターは境界の外部に位置する。

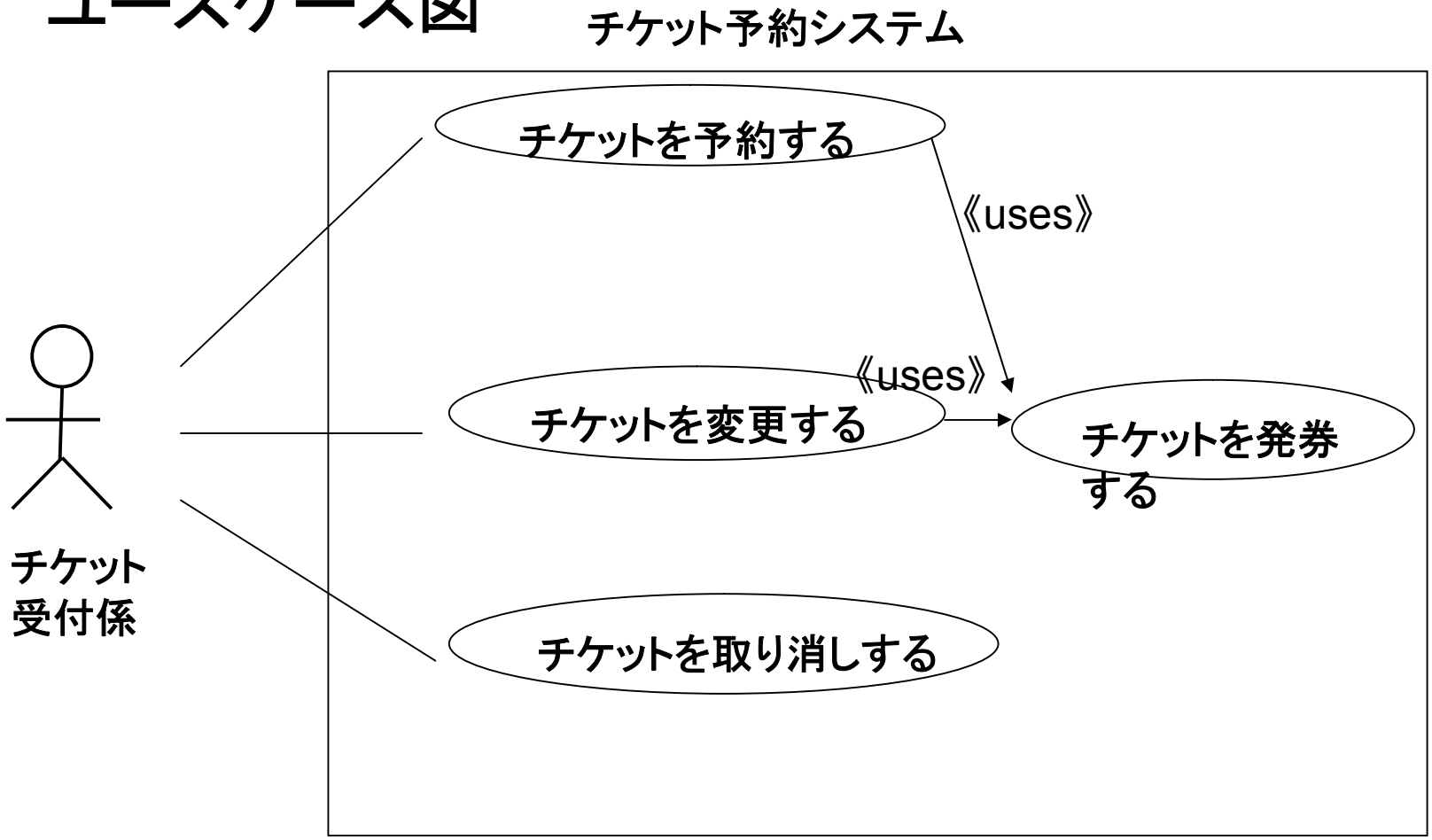
《uses》　**ユース関係**　:　他のユースケースを呼び出す関係を示す。

《extend》　**拡張汎化関係**　元のユースケースの拡張が出来る。

例示１　ユースケース図


テレビ
電源を入れる
《uses》
チャンネルを変える
映像を表示する
《uses》
番組を予約する
予約を表示する
《extend》
視聴者
毎週の予約をする
電源を切る

例示2　ユースケース図


チケット予約システム
チケットを予約する
《uses》
《uses》
チケットを変更する
チケットを発券する
チケットを取り消しする
チケット
受付係

例示３　ユースケース記述

・各ユースケース毎に、目的、概要を記述する。

ユースケースの記述を詳細化することにより、例外事項、モレなどの発見が進む。

ユースケース記述書

ユースケース概要を記入xxxxxxx

目的　　：　xxxxxxxxxx

概要　　：　xxxxxxxxxx

前提条件　　：　xxxxxxxx

制約条件　　：　xxxxxxxx

相互対話の順序（シナリオ、処理手順）

1.　xxxxxxxxxx

2.　Xxxxxxxxxx

システム、アクターがやることを区別

全体

ユースケース名称

アクターの説明

ユースケースの説明

（何をするのか、条件は無いか、例外は無いか・・・）

ユースケース毎に

例示４　ユーザーインターフェースの整理

・ユースケース図作成の最後に、ユーザーインターフェースを記述する。

ユーザーインターフェース記述書

（入出力画面、帳票の概要を記入する。）

入力画面A　：　xxxxx、xxxxxxを表示する

：　入力項目の枠を用意する

出力画面B　：　結果としてxxxxx、xxxxxxを表示する

入力画面C　：　次の入力枠を用意する

出力画面D　：　最終結果xxxxxxxおよび明細xxxxxxをプリンタに出す

② クラス図、オブジェクト図、パッケージ図

クラス名： 対象システムのクラス名を表示。

クラス名
属性
操作

《 ｽﾃﾚｵﾀｲﾌﾟ名 》 ｽﾃﾚｵﾀｲﾌﾟの名称

例えば、boundary、control、entityなど

オブジェクト名：クラス名　オブジェクト名の表示

パッケージ名：：クラス名　他のパッケージのクラス名

{ クラスの性格、特徴 }

クラス作成者名など属性を記入

属性 ： クラスが保有する属性を列挙する。

・属性の名前：ﾃﾞｰﾀ型＝初期値　通常は名前のみで可

・属性の名前の前にアクセス上の性質を表現できる

＋	public	他ｸﾗｽからも参照できる
－	private	自身からのみ参照できる
＃	protected	自身および継承ｸﾗｽから参照できる

操作：クラスの振舞いを列挙する

- **操作の名前**：操作名称（xxxxxをどのようにする）

  例示：ﾁｹｯﾄ発券、明細書の発行・・・

- 操作の名前の前にアクセス上の性質を表現できる

  （＋、－、＃）

- 名前の後ろに（引数のリスト）：返り値の型＝初期値

  を記入する

  例示：　＋getWindowWidth（　）：int

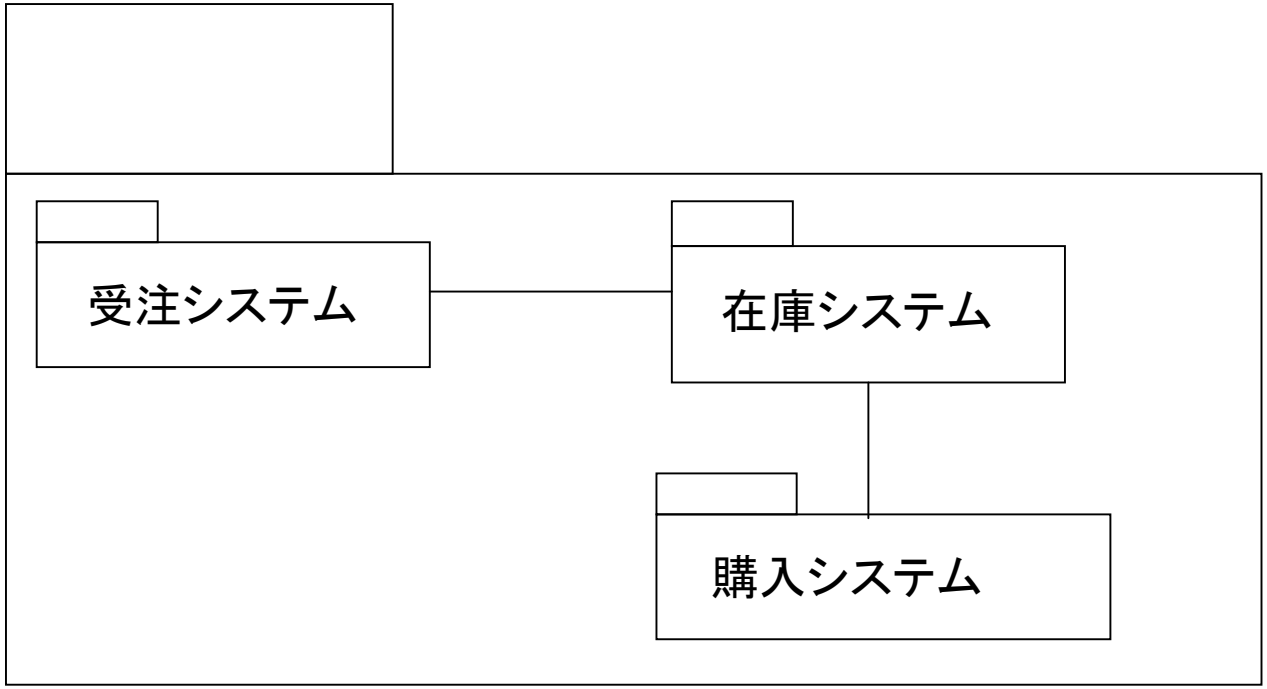
  　　＃setWindowSize（　）：void

- 分析、外部設計までは、通常、操作の名前だけを記述

パッケージ図

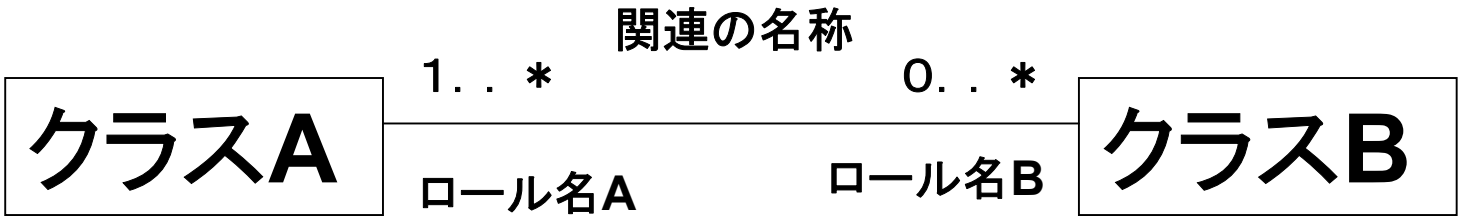
- 対象システムが大規模な場合、サブシステムに分割する。
- いくつかのクラスをグループ化して、複雑さを軽減する。

例示：



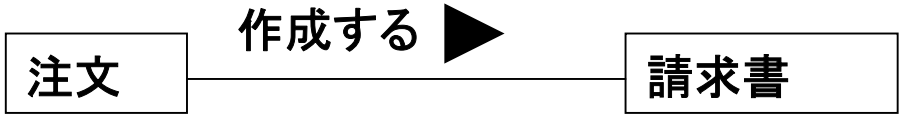
- パッケージ図には、シーケンス図、コラボレーション図、状態図、アクティビティ図を含むことが出来る。

③ クラス間の関連の表記方法



・関連の中央に関連名称と方向を記す （自明であれば省略）

例示：

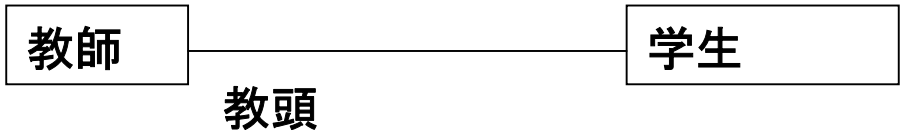


・多重度　クラスの間に、インスタンスがいくつの関連があるかを示す。

例示：　　下限値．．上限値　（　１　）（1．．10）（0．．＊）

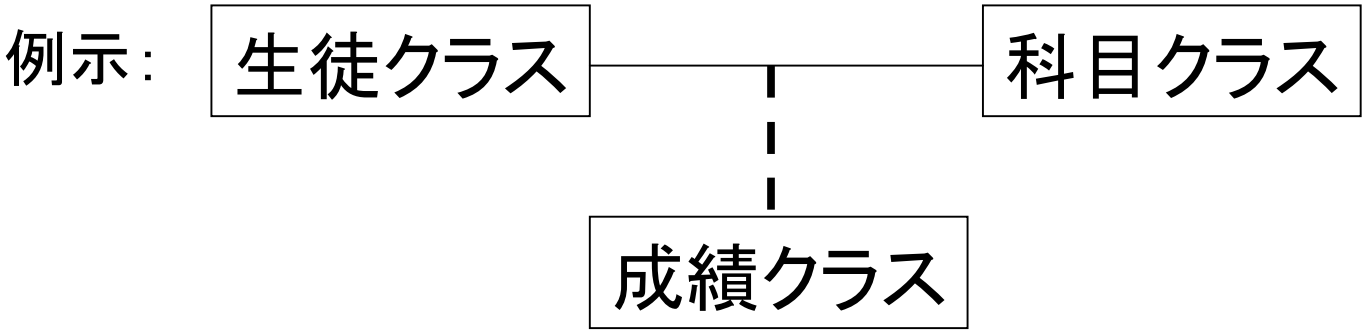
・ロール名は、他方のクラスから見ると、一方のクラスが、どのような役割をはたしているかを表す。

例示：



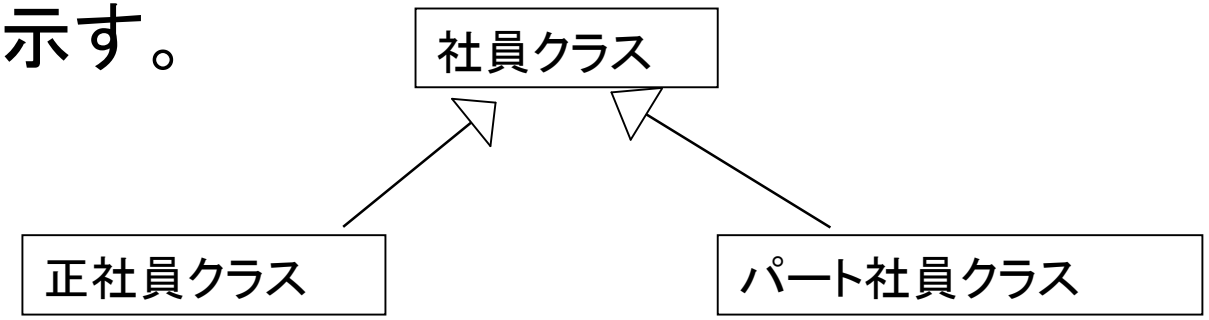
④関連クラス（association）

クラスとクラスの間に、関連する情報を保持するときに、関連クラスを設けることが出来る。



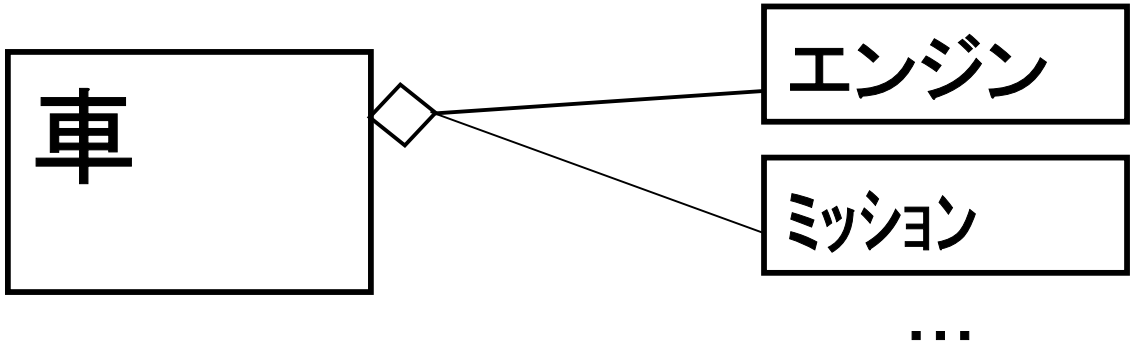
⑤クラスの継承、汎化／特化関係

クラスとクラスの間に、継承の関係があるときの関係を示す。



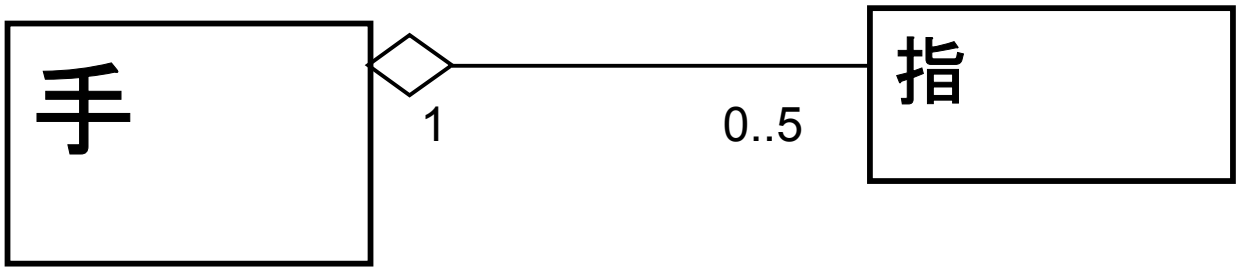
⑥クラス間の集約関係（共有集約）

クラス間に集約関係がある時の表示方法



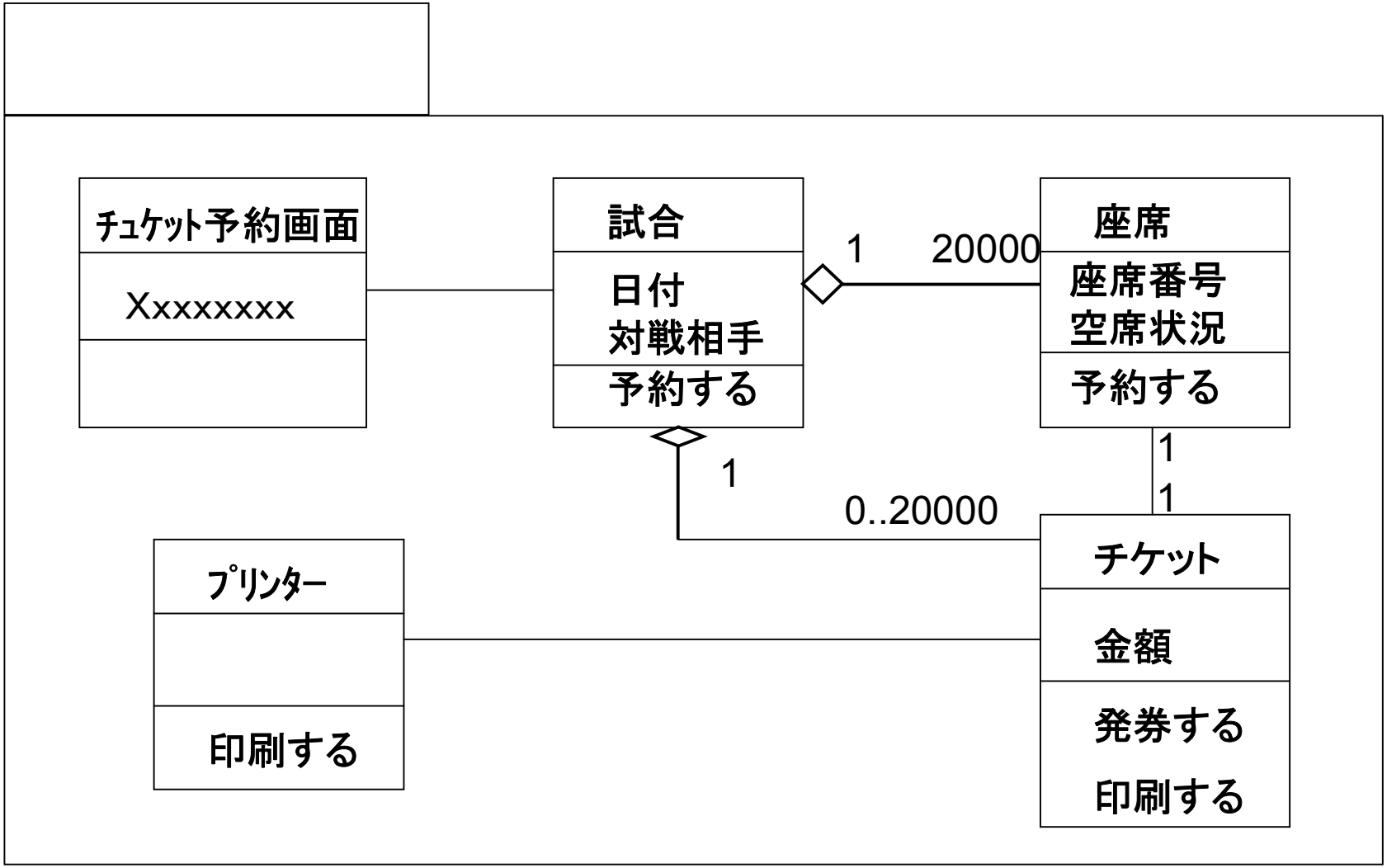
⑦コンポジション（複合集約）

集約関係の一種であるが、全体クラスと部分クラスのサイクルが一致するとか、全体クラスの多重度が「1」で有るような場合の表示方法

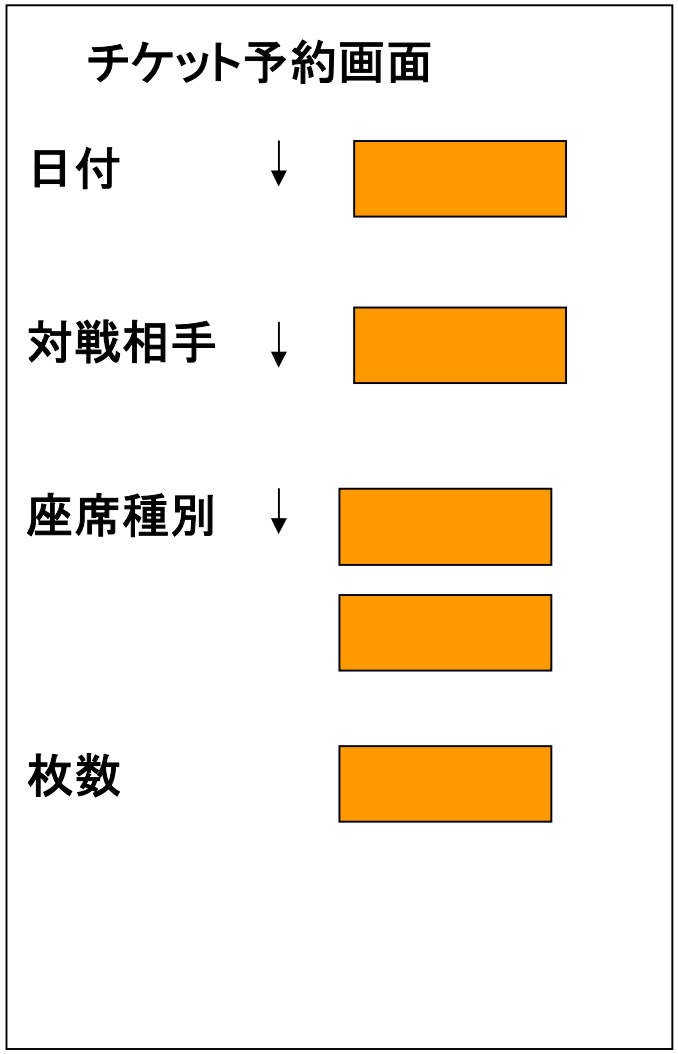


例示:　クラス図

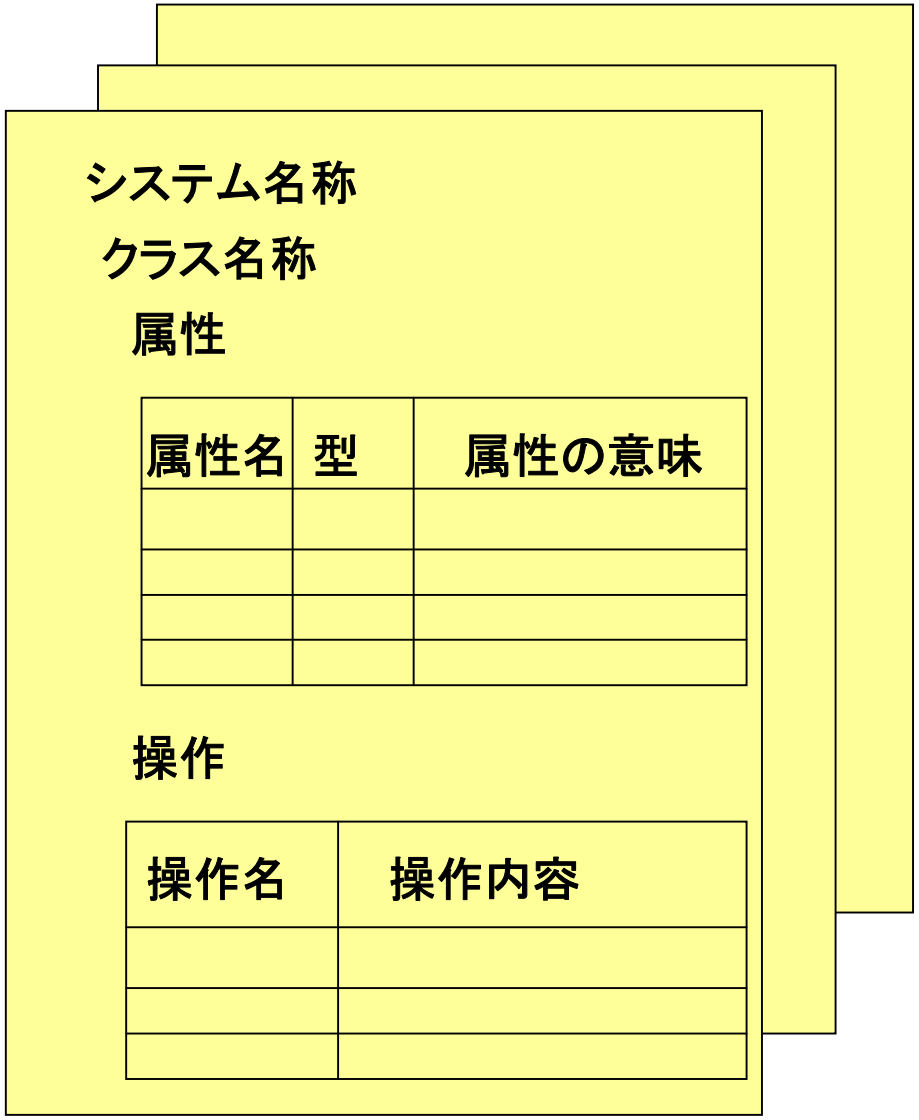
「試合のチッケト予約を受付けて、チケットを発券する」



例示：画面設計書

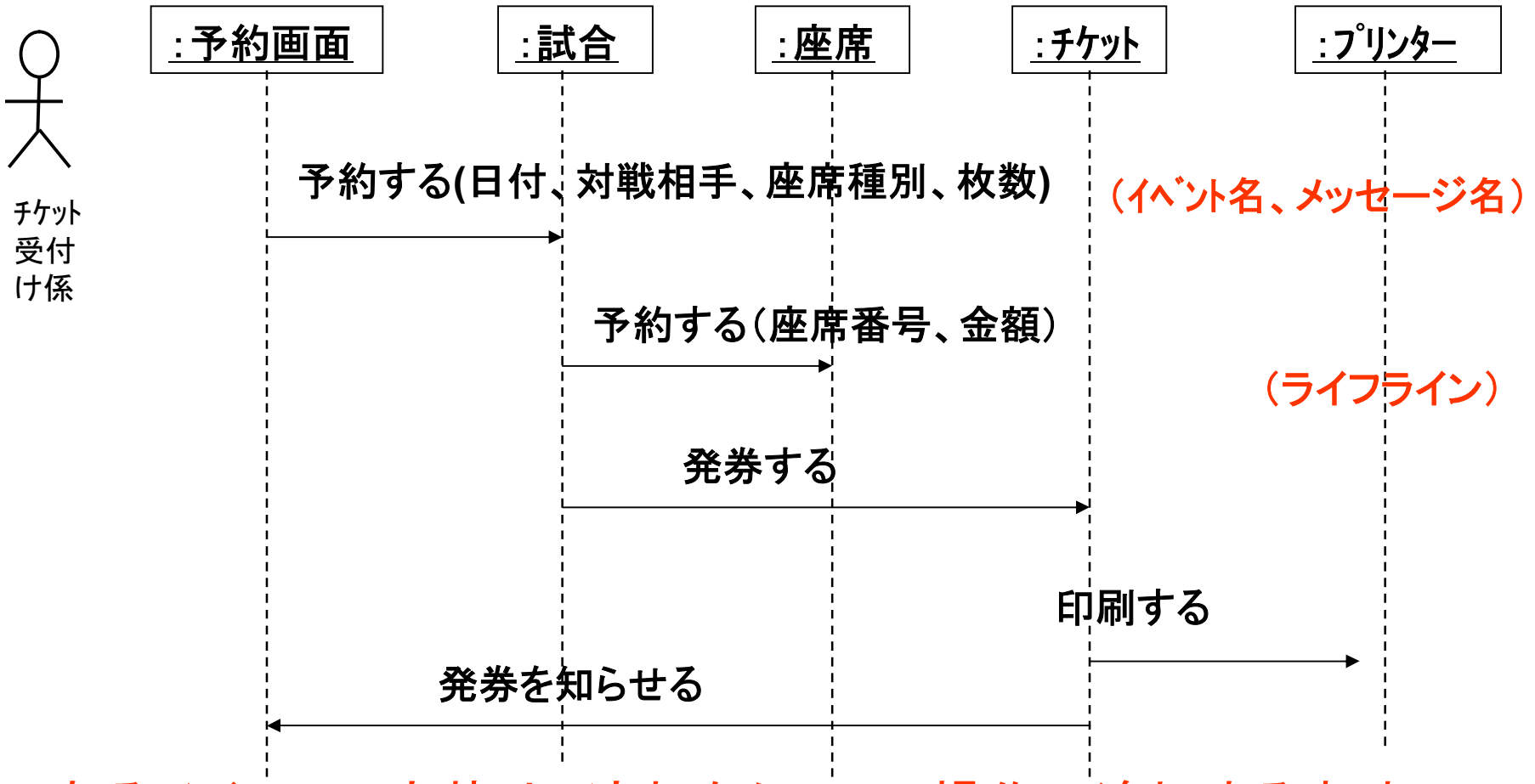


例示：クラス仕様書



⑧シーケンス図

・アクターを始点にして、システム内に存在するオブジェクト間のイベントの流れを表示する。（矢印は、クラス間の関係の意味）

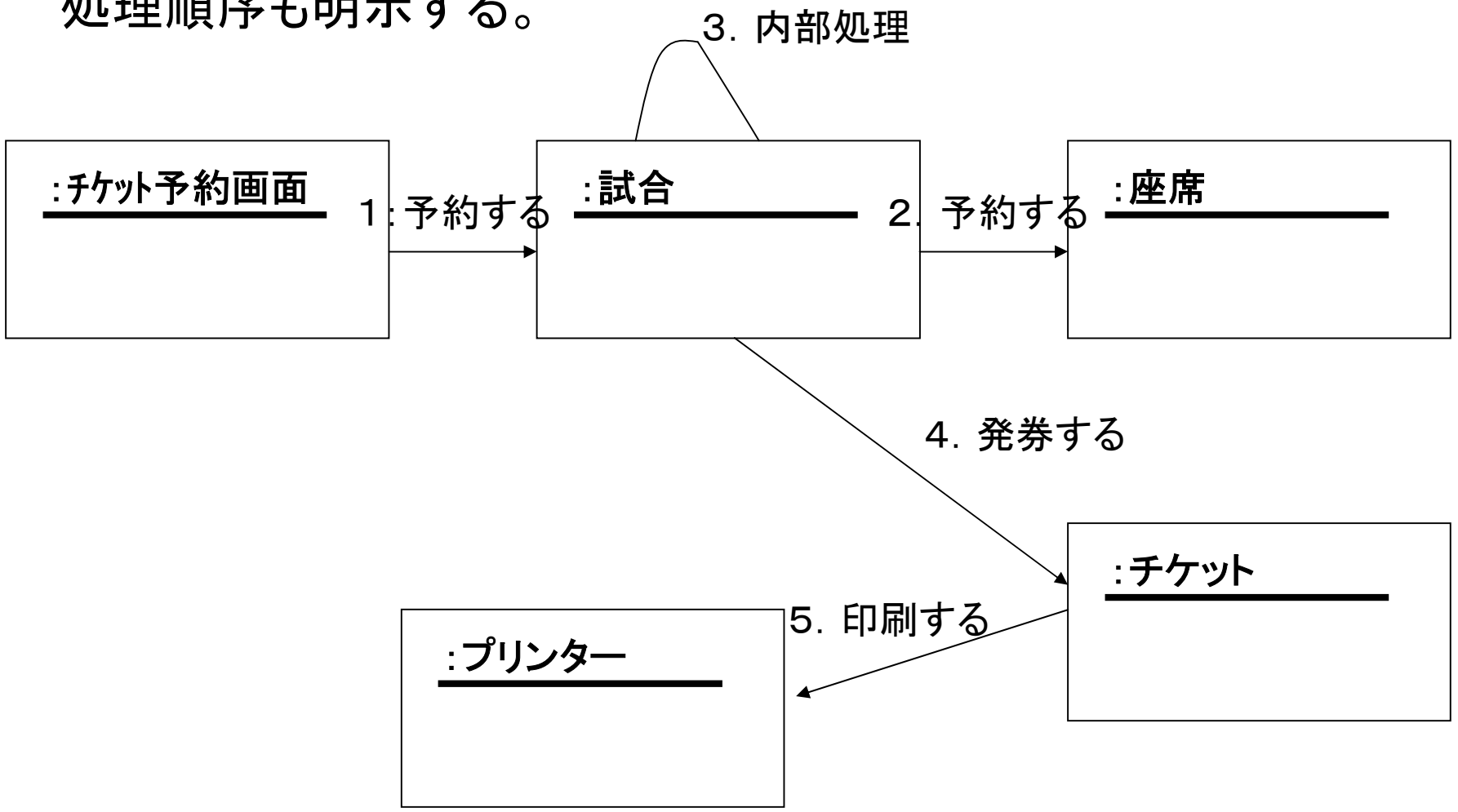


・あるイベントの実装は、流れをクラスの操作に追加する意味。

⑨　コラボレーション図

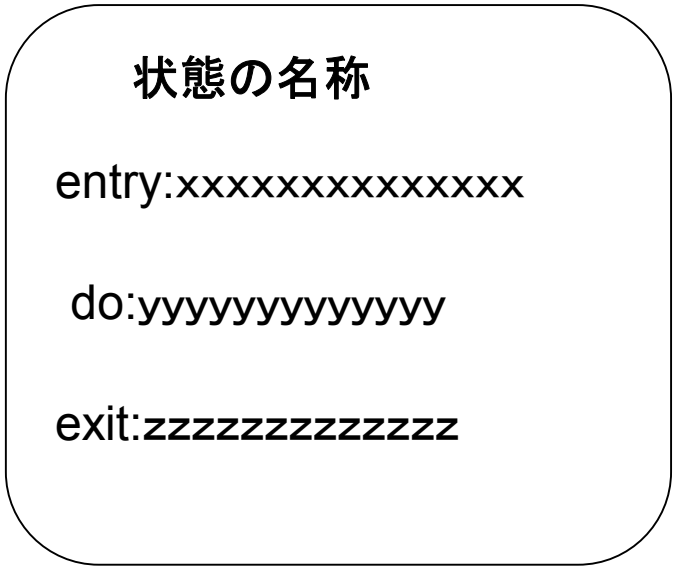
シーケンス図と同様に、オブジェクトの相互作用を記述し、

処理順序も明示する。



⑩状態図（ステートチャート図）

・オブジェクトの状態を記述する。（複雑なクラスで使用する。）
・具体的には、そのオブジェクトに対する外部からのイベントにより、
　オブジェクトがどのような状態になるのかを示す。
・ある状態から別の状態への遷移が分る。

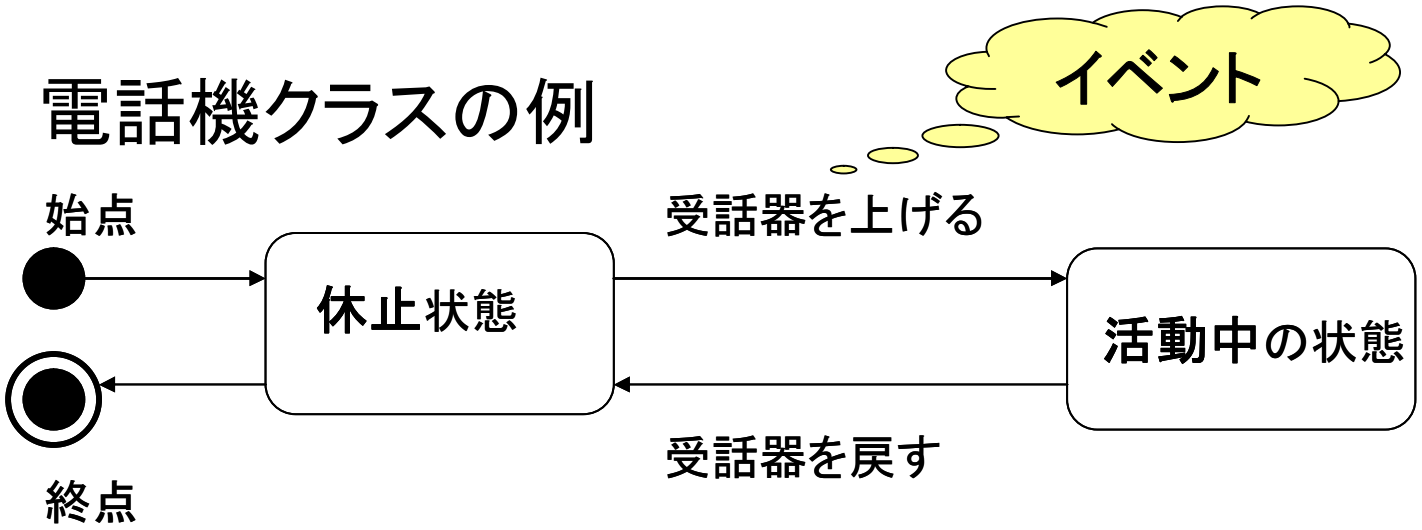


Entry　：　この状態になった時に実行する処理

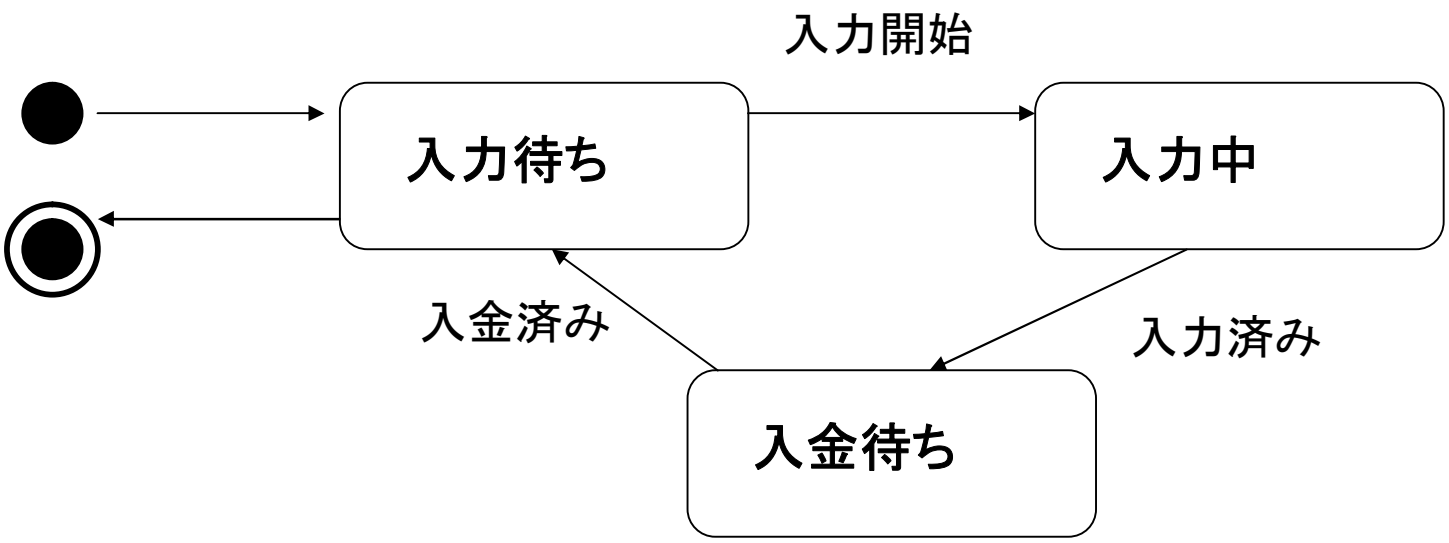
Do　：この状態になってから、抜け出すまでに
　　実行する処理

Exit　：この状態を抜け出る時に実行する処理

例示： 電話機クラスの例

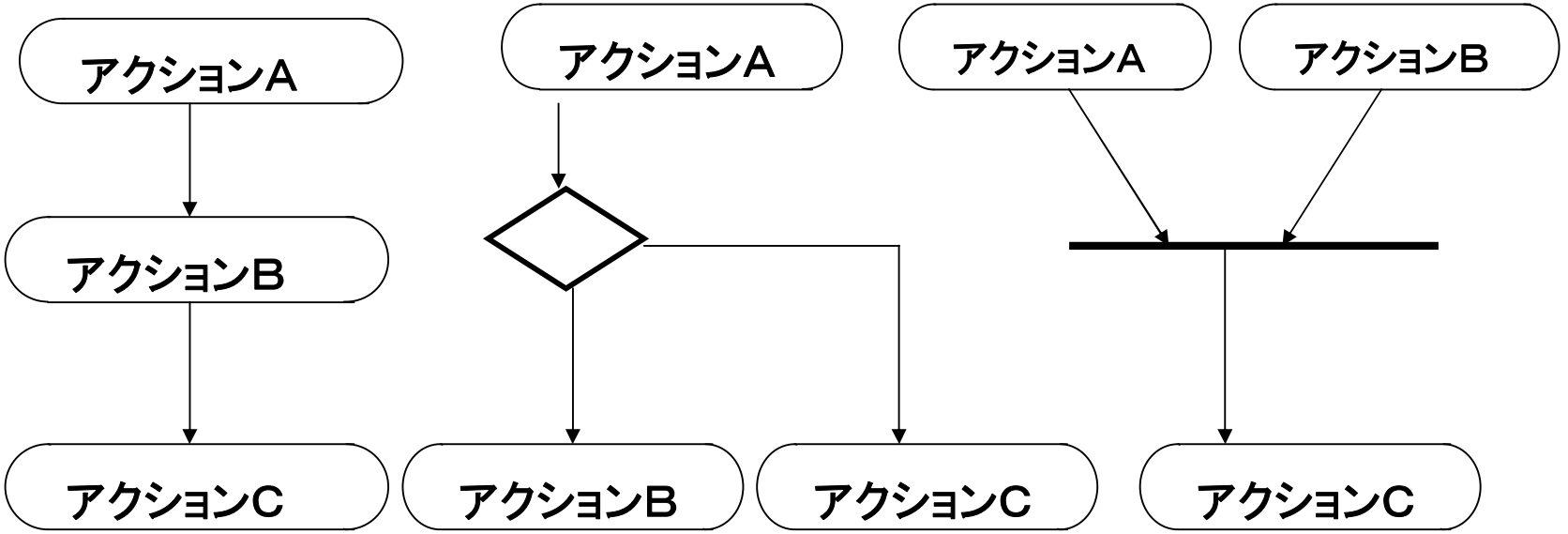


例示： POSレジスターの売上ユースケースの例

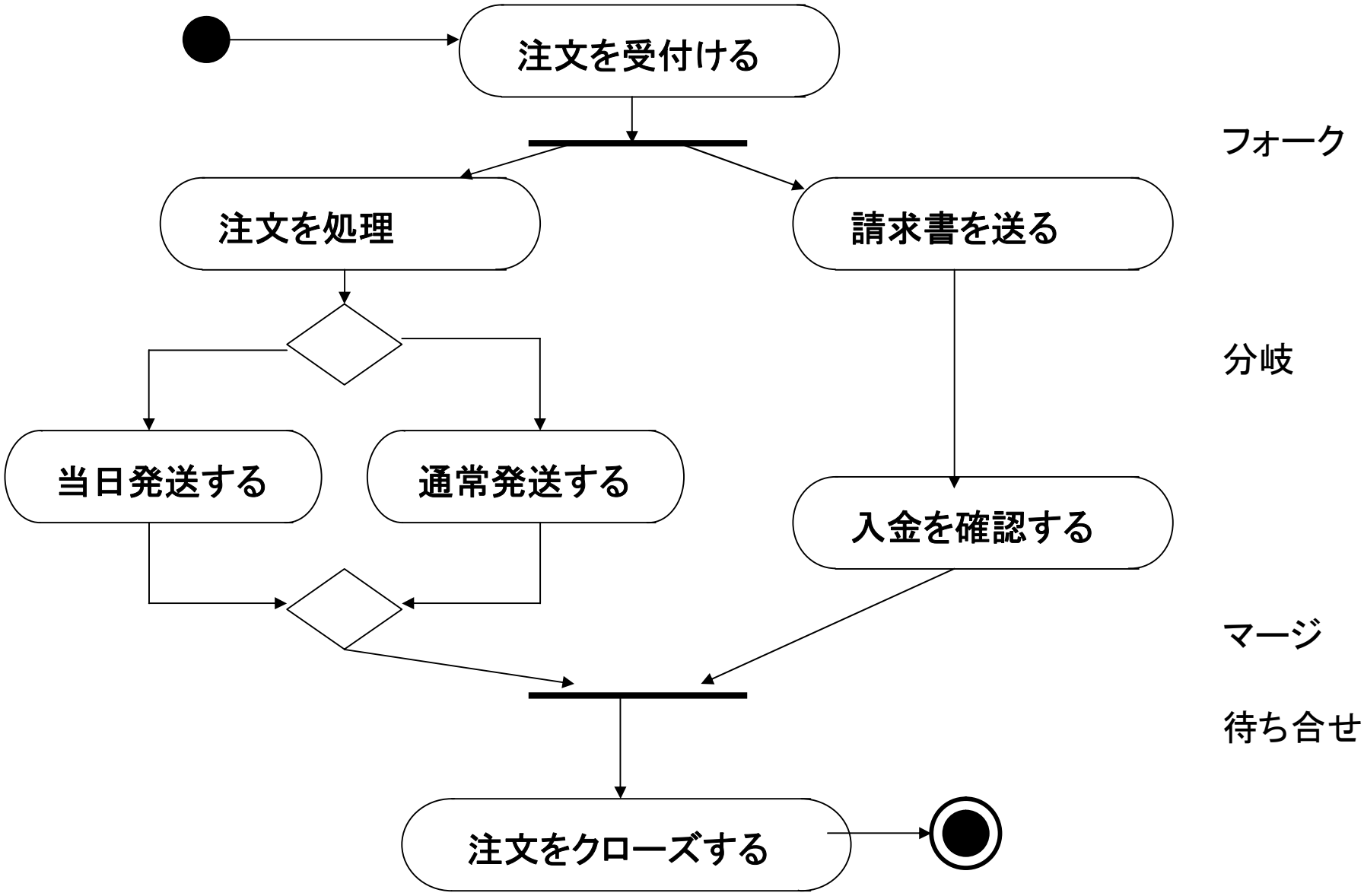


⑪アクティビティ図

・オブジェクトの振舞いを、従来のフローチャートの形式で表現する。
・ユースケースや、クラスの操作が実行するアクティビティの連続的な流れを表すことが出来る。
・実行順序、条件分岐、待ち合せを表示できる。
・状態図の一種である。
・多くの複雑な処理の制御が必要な場合に、有効である。



例示：本の注文のユースケースのアクティビティ図



４．UMLクラス図のステレオタイプ

４．１　OMT法によるクラス、オブジェクトの表記法

・OMTでは

－Object　Modeling　technique

－J.Ranbaughによる

－クラス、オブジェクトは、エンティティ（実体）のみを

対象とする。

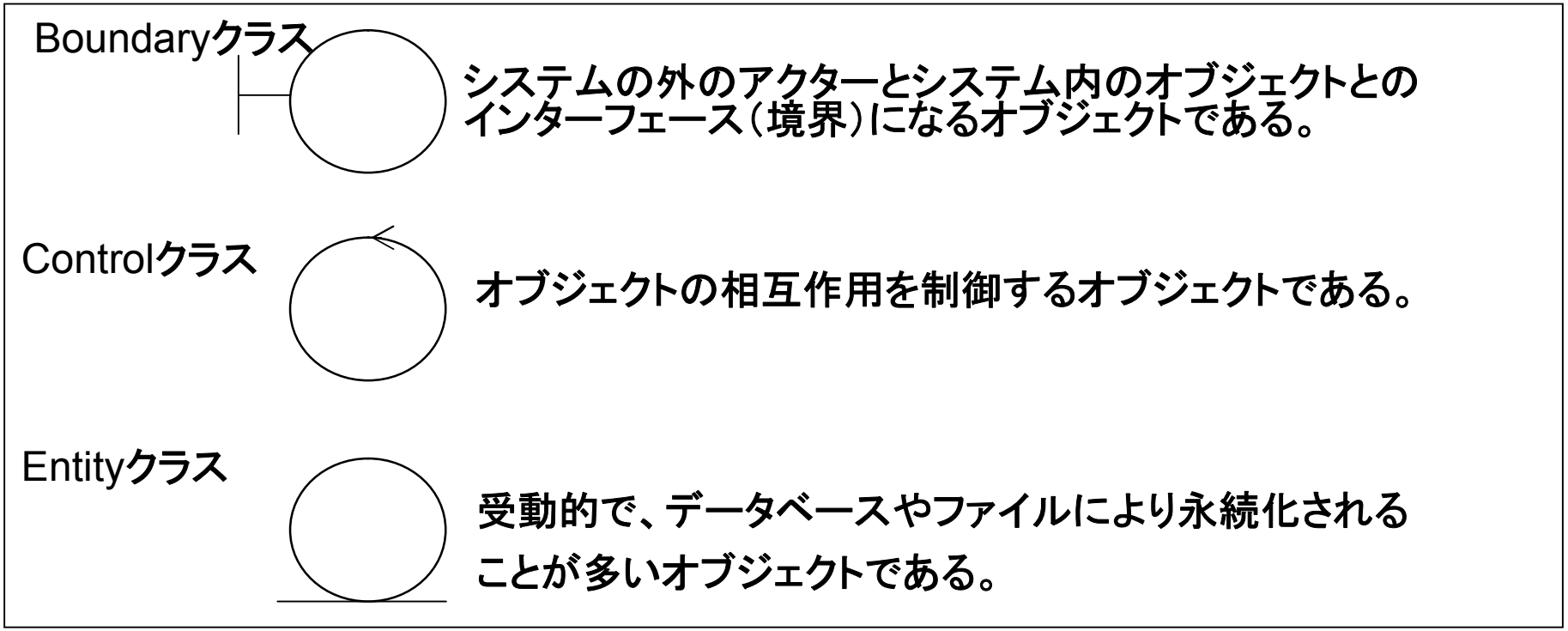
４．２ OOSE法によるクラス、オブジェクトの表記法

・OOSE

－Object Oriented Software Engineering

－J.jacobsonによる

－クラス、オブジェクトは、バウンダリー、コントロール、エンティティの３種類を表現できる。



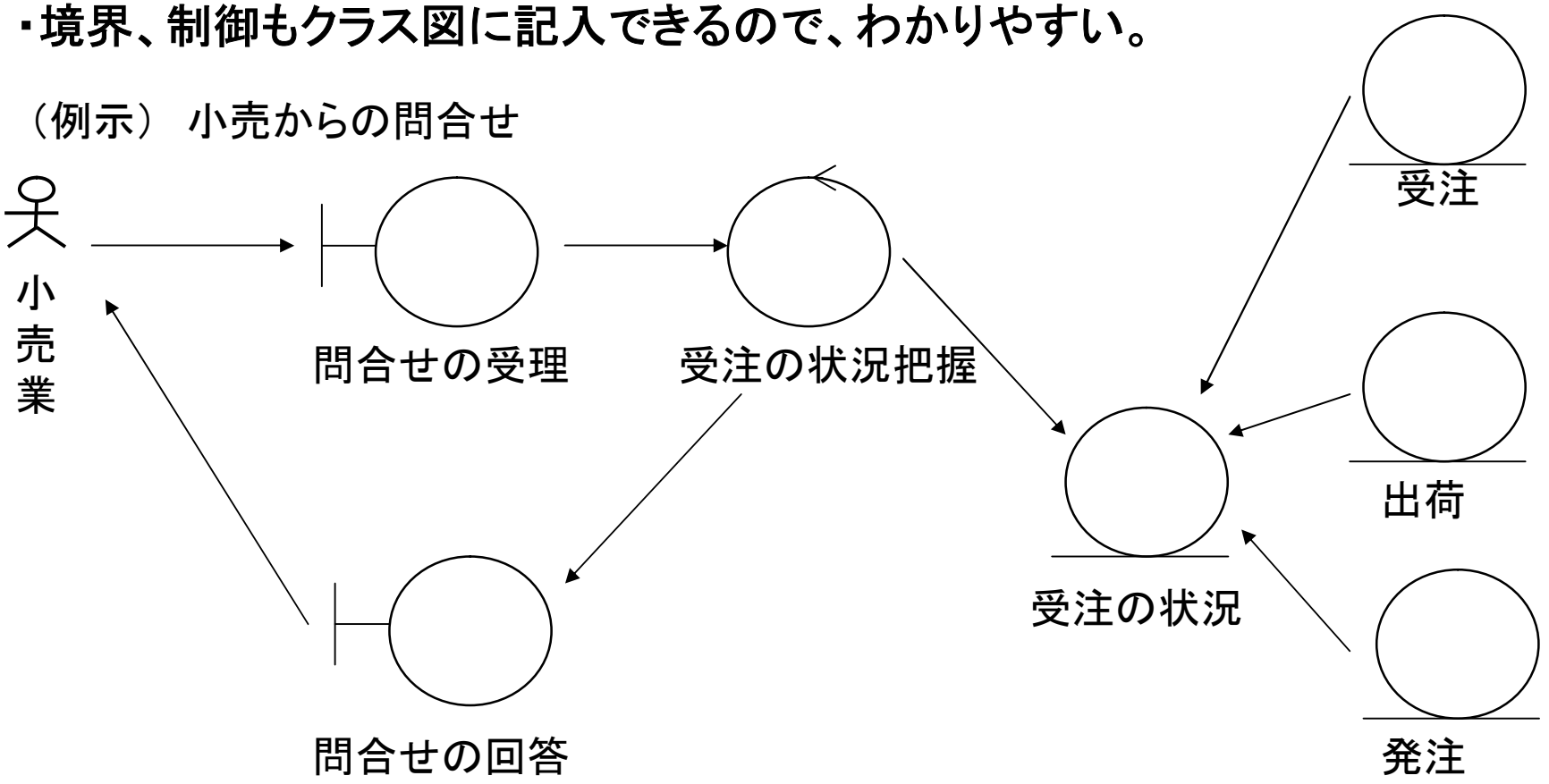
４．３　ステレオタイプのクラス図事例

・初期段階のクラス図、オブジェクト図を表記する方法。

・属性、操作は記入しないで、メモを取っておく。

・境界、制御もクラス図に記入できるので、わかりやすい。

（例示）　小売からの問合せ



５．UML使用上の留意点

①システム開発の目的、要求をクリアーにする

－目的、目標レベルの明確化

－対象の分析、要求分析

これらがクリアーで無いと、

機能（ユースケース）、

対象（クラス）　　を具体化できない。

目的、目標が明確でなければ、ある程度できても、

一般的な設計になり、問題解決にはならない。

②ユースケース図

・システム利用者の立場で機能（ユースケース）を取り出す。

・アクターの行為をよく分析する。

ーシステムに対して、双方向で何をするのか

（入力、変更、通知、システムから知らせ・・・）

・アクターとして、外部のシステム、装置も取り出す。

・ユースケース作図の前に、アクターの役割を記述する。

・主たる利用者を優先して取り出す。

（システム管理者などは副アクターとして後で追加。）

・要件定義段階では、機能を細かく分けない。

（例）支払い機能を、現金／カード／小切手での支払い

などには分けない。

チケットの印刷、請求書のプリントなども分けない。

③クラスの取り出し方

- 基本は、要求仕様書、ユースケース記述書を元にして、名詞句に着目して、クラス候補を取り出す。
  （クラス、属性の元になるもの）
- クラス候補を取出したあとで、対象領域のなかで重要なモノからクラスとし、それと関係の深いモノを見出す。
- 意味が重なっているものは、名前を一つに統合する。
- ユーザーインターフェース記述書から、インターフェースクラスを取り出す。（例示、チケット予約画面、チケットプリンター・・・）
- 要件定義段階では、実装段階で必要となるクラスは未だ取り出さない。（個々のミドルウェアなど・・・）
- 操作は、クラスの役割・責務を考えて、見出す。

・よくあるクラス設定の間違い例

X　大きなクラス1つで済ませる （大きなプログラムを作成するの同じで、製造・保守が大変である）

X　機能に着目して、機能の数だけクラスを設ける

例示　　チケット予約、チケット変更、チケットキャンセル

X　属性をクラスに格上げする

例示　　車・鉛筆・洋服・靴などのサイズ、数値、色など

△　データベースをクラスとする

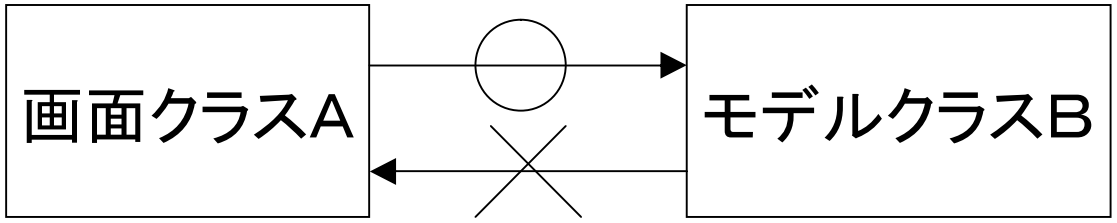
間違いでは無いが、内部設計の後半にクラスとして定義すればよい。

（追記）役割分担に着目したクラス設定法

- 凝集性を高く維持できるように責務を割当てる
  ー凝集性（ｺﾋｰｼﾞｮﾝ、cohesion）とは、
  　DOAでいうところのモジュール強度である。
  ー機能的強度、情報的強度が高い場合は、
  　クラスを分けない。もしくは、集約関係とする。
- モジュール間結合度の強いものは同じクラスに。
  ーデータ結合は、クラスとして独立させる。
- どのクラスに操作を任せるか（＝責務）という
  観点からの分析技法がある。
  ー責務の遂行に必要な情報をもっているクラスに
  　操作を任せる（エキスパート・パターン）

（追記2）画面クラスの使い方

- モデルクラスとビュークラスを切り離す。
  ーモデルはシステムの内部の論理（データ、処理）
  ービューは、システムの見かけの部分（画面、帳票）
  従って、画面クラスは独立させる。
- 画面は変更が頻繁であり、モデルクラスから
  画面クラスへの関連はもたせないこと。



- 画面クラスには、イベント駆動の操作を準備する。
  ーどのボタンが押されたら、どの操作を実行する。
- 画面クラスの操作では、通常、ライブラリー中の
  部品を活用する。（スクロール、表示、ボタン、画面形状）

④シーケンス図の作成

・基本は、ユースケース図、ユースケース記述書、
クラス図を元にして、
　　－アクターを起点に、関連があるクラス間の
　　－イベントの流れを順番に見つけ出すことである。

・煩雑さを避けるために、イベント毎に、シーケンス図は
独立させて作成する方がよい。

・よくあるシーケンス図の間違い（役割分担）
　X　一つのクラスに役割が全て集中している。
　X　上からのメッセージに対して、別なことを返している。
　X　受けたメッセージをそのまま下流に流している。

6. UMLのまとめ

基本は、機器の設計者、システムの設計者の立場で考える

ユースケース図

－要件定義段階で作成する。

－要求を分析し、ユーザーが何を要求しているのかを書き出す。

－その要求を実現するためには、どんな機能が必要かを取り出して書き出す。

－その機能をユースケース図、ユースケース記述、インターフェース記述にまとめる。

・制御機器の場合は、その機器の機能（働き）を取出す。

・ビジネスシステムの場合は、〇〇係／〇〇担当者の仕事内容。

・ クラス図

－外部設計段階で作成する。

－機器またはシステムの内部構造のこと。

－機器の場合は、機器の主要な構成部品である。

－ビジネスシステムの場合は、通常〇〇係／〇〇担当者がクラスに当たる。

・ 〇〇係／〇〇担当者の代わりにやるのはプログラムである。

－その機器の構成部品の役割が操作であり、〇〇係／〇〇担当者の頭の中にあるデータとかノートが属性であり、仕事の内容が操作である。

－それをクラス図、クラス仕様書のまとめる。

シーケンス図

－外部設計で作成する。

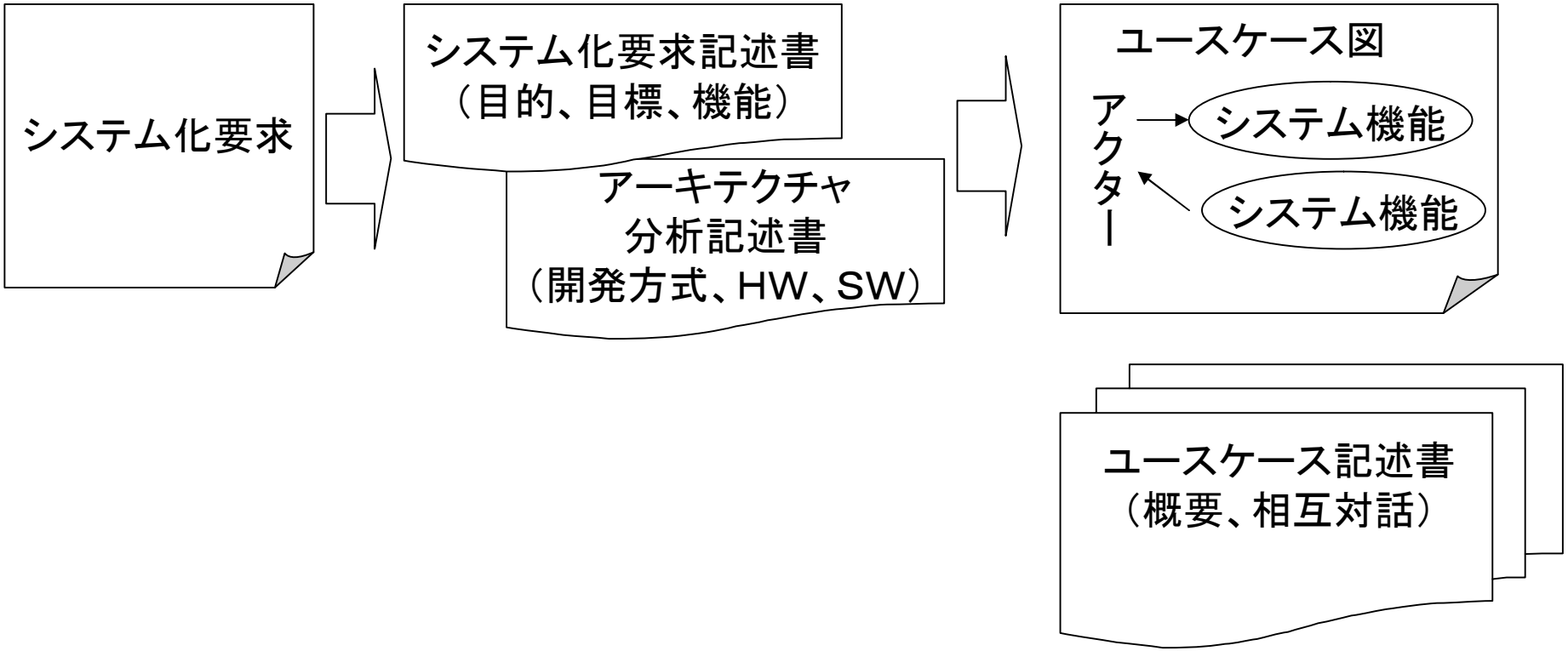
－機器の場合は、構成部品間の信号の流れ。

－ビジネスシステムでは、クラス間の依頼書／メッセージの流れをまとめたもの。

６．１ 要件定義のプロセス

・ユーザーのシステム要求を分析して、システム化要求をまとめる。

・それをもとにして、システム機能をユースケース図、ユースケース記述書にまとめる。

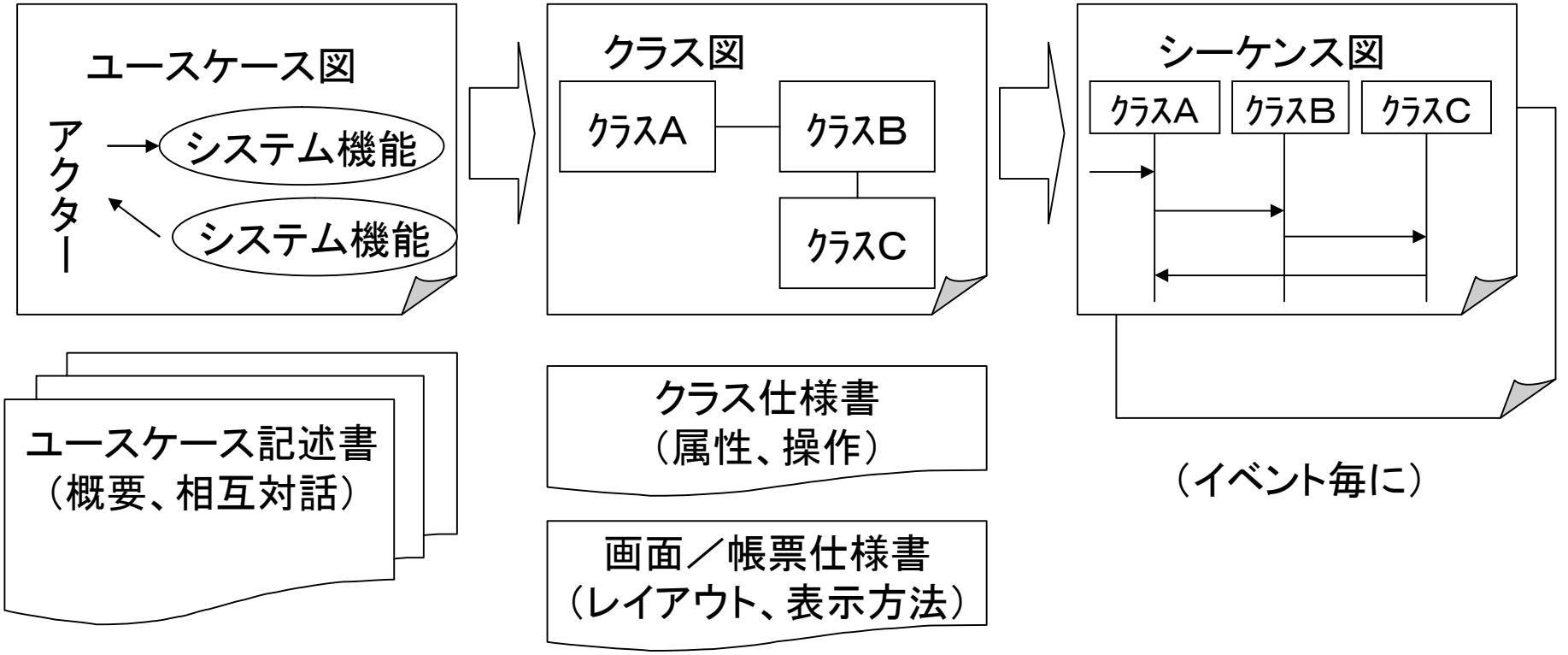


６．２ 外部設計のプロセス

・要件定義で作成したシステム機能定義をもとにして、オブジェクトモデルを作成する。

（静的モデル設計（クラス図）、クラス仕様書、画面／帳票仕様書、

動的モデル設計（シーケンス図））



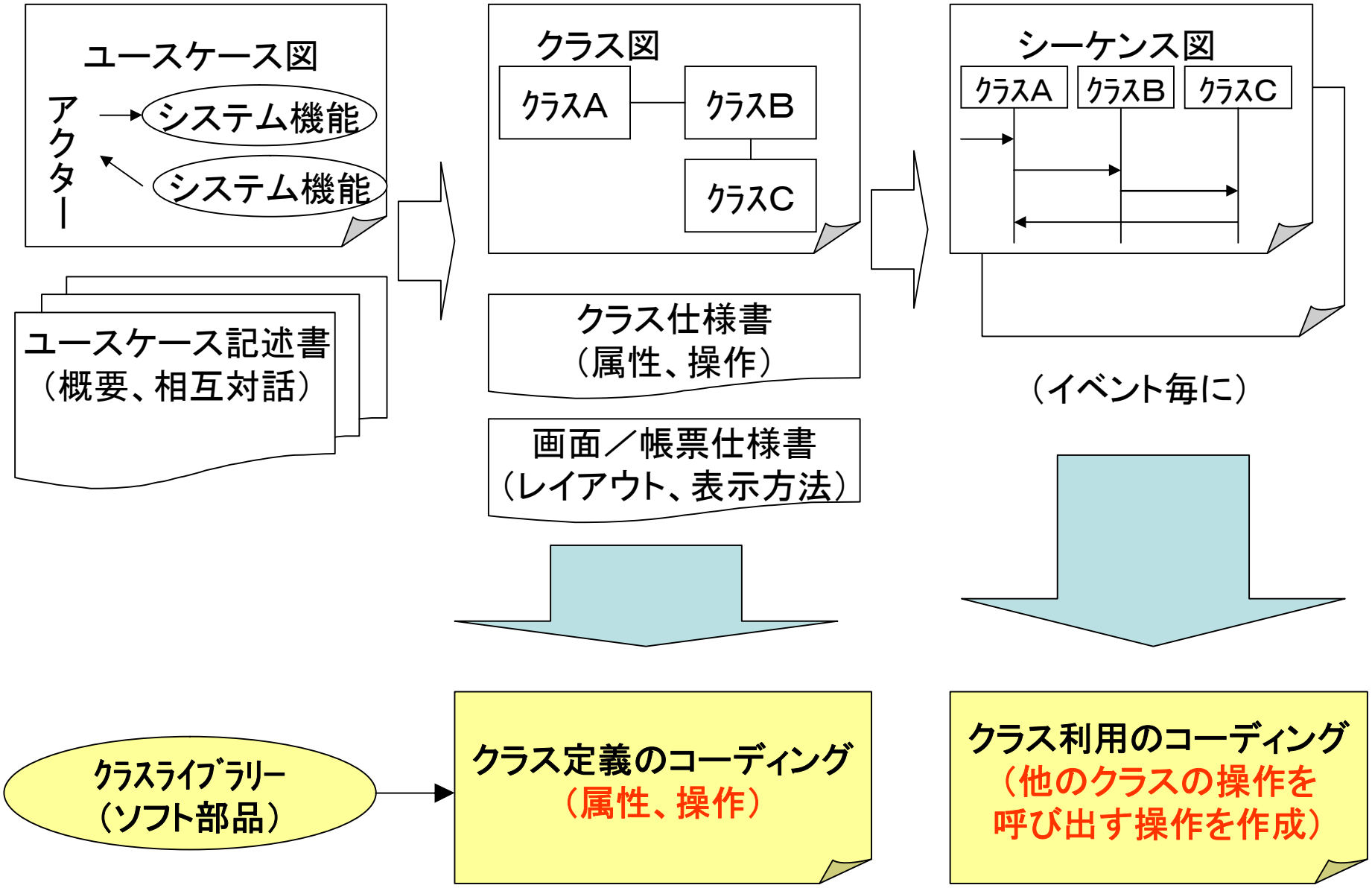
６．３ 内部設計のプロセス

・外部設計で作成したオブジェクト・モデルをシステム実装レベルまで具体化する。
（クラスの詳細化　＝　特化（下位クラス作成）、汎化（共通クラス作成））
（クラスの分解　　＝　インターフェースクラスの追加）
（画面クラスの詳細化　＝　画面利用時のモレを無くす）
（クラスの属性をもとにして正規化・ERモデル化によりデータベース構造）

（属性の追加・修正）
（操作の追加・修正）

・シーケンス図の詳細化→→→→→アプリケーションのプログラム設計になる。
（上記で変更したクラスを、シーケンス図上でも変更する）
（上記で変更した操作を、メッセージに反映する。）

６．４設計とプログラミングの関係



7. オブジェクト指向の課題

①ソフト部品の流用性の向上（メタ情報）

ー膨大なソフト部品の中から、必要な部品を容易に取出す手段を作る

ーソフト部品の流用時、必要な修正を容易にするための支援情報

ーソフト部品の適用領域、利用方法の標準化と仕組み作り

②分析・設計方法の統一と教育充実

③実施可能レベルのクラス決定法が必要

④上流から下流まで同じ名称のUML図を使うが、内容レベルは詳細化されており、別物である。（分析クラス図などの名称分け）

８．演習問題

機器などを想定

８．１　身近な事例をもとに、ユースケース、クラス図、シーケンス図を作成しなさい。
（例）電話機、洗濯機、携帯、テレビ・・・

８．２　以下のシステム化に当たって、ユースケース、クラス図、シーケンス図を作成しなさい。

「社員名簿システム：従来、帳簿式のものしかないので、システム化して、データメンテナンスのダブリ排除、各種の検索を効率化する。」

９．まとめとレポート課題

- 重要項目

ー開発工程におけるUMLの使用法
ークラスの求め方（決め方）
ーユースケース、クラス図、シーケンス図

- レポート課題（A4x1、２枚）
①８．１の問題の解答
②８．２の問題の解答

期限　　次回の授業開始時点
提出　　レポート用紙またはメール

10. 参考書、参照URL

- 中桐紀幸、「即戦UMLモデリング」(リックテレコム)
- ラーマン、「実践UML」(依田訳、ピアソン・エデュケーション社)
- 別所義夫「これで使えるオブジェクト指向」(電気通信協会)
- (株)永和「オブジェクトハンドブック」(ピアソンエデュケーション)
- ファウラー、スコット「UMLモデリングのエッセンス　第二版」
  (羽生田訳、翔泳社)

・情報処理学会、「情報処理1994年、5月号」
  ー特集「オブジェクト指向分析・設計」

・(株)オージス総研「10日でおぼえるUML入門教室」
  (翔泳社)

参考になるURL

- http://www.ogis-ri.co.jp/otc/hiroba/index.html
  オブジェクト広場
- http://ObjectClub.esm.co.jp/　UML、Java
- http://www.rational.co.jp/uml/　UML
- http://www.zzz.or.jp/~iizuka/EOO/index.html
  ○○のリンク集
- http://www1.u-netsurf.ne.jp/~kitada/3H/index.htm
  ○○の概念説明
- http://www.njk.co.jp/otg/　　○○全般