基礎情報処理

Information Processing Basics
UML2

2004年11月25日

高等教育研究開発推進センター
小山田耕二

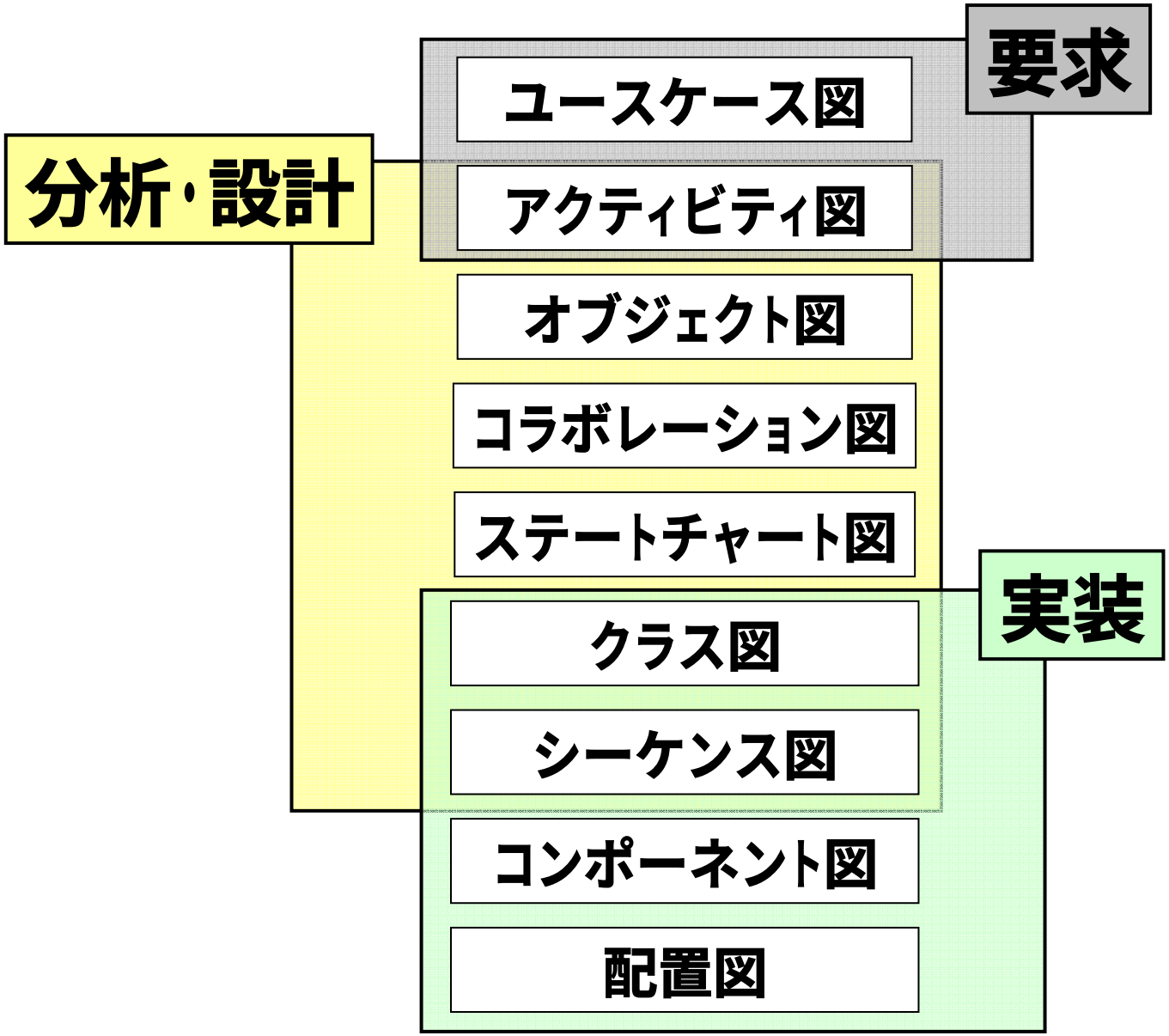
Outline

1. コンピュータとはなにか
2. ディジタル情報の世界
3. 論理回路からコンピュータまで1
4. 論理回路からコンピュータまで2
5. プログラム基礎1
6. プログラム基礎2
7. UML1
8. UML2
9. データ構造とアルゴリズム
10. コンピュータネットワーク
11. 情報倫理
12. さまざまな情報処理

8 UML 2

8.1 **オブジェクト図**
8.2 **コラボレーション図**
8.3 **ステートチャート図**
8.4 **クラス図**
8.5 **シーケンス図**
8.6 **コンポーネント図**
8.7 **配置図**

設計工程とUML


要求
ユースケース図
分析・設計
アクティビティ図
オブジェクト図
コラボレーション図
ステートチャート図
実装
クラス図
シーケンス図
コンポーネント図
配置図

8.1 オブジェクト図 (1)

□オブジェクト

「具体的なモノ」のこと
(人、車、PC、情報など現実世界にあるモノ)
オブジェクト名、属性値、状態で構成される

□表記法

四角の箱で表し、以下のように記述

オブジェクト名／ロール：クラス名 [状態名]
属性名＝値

例)

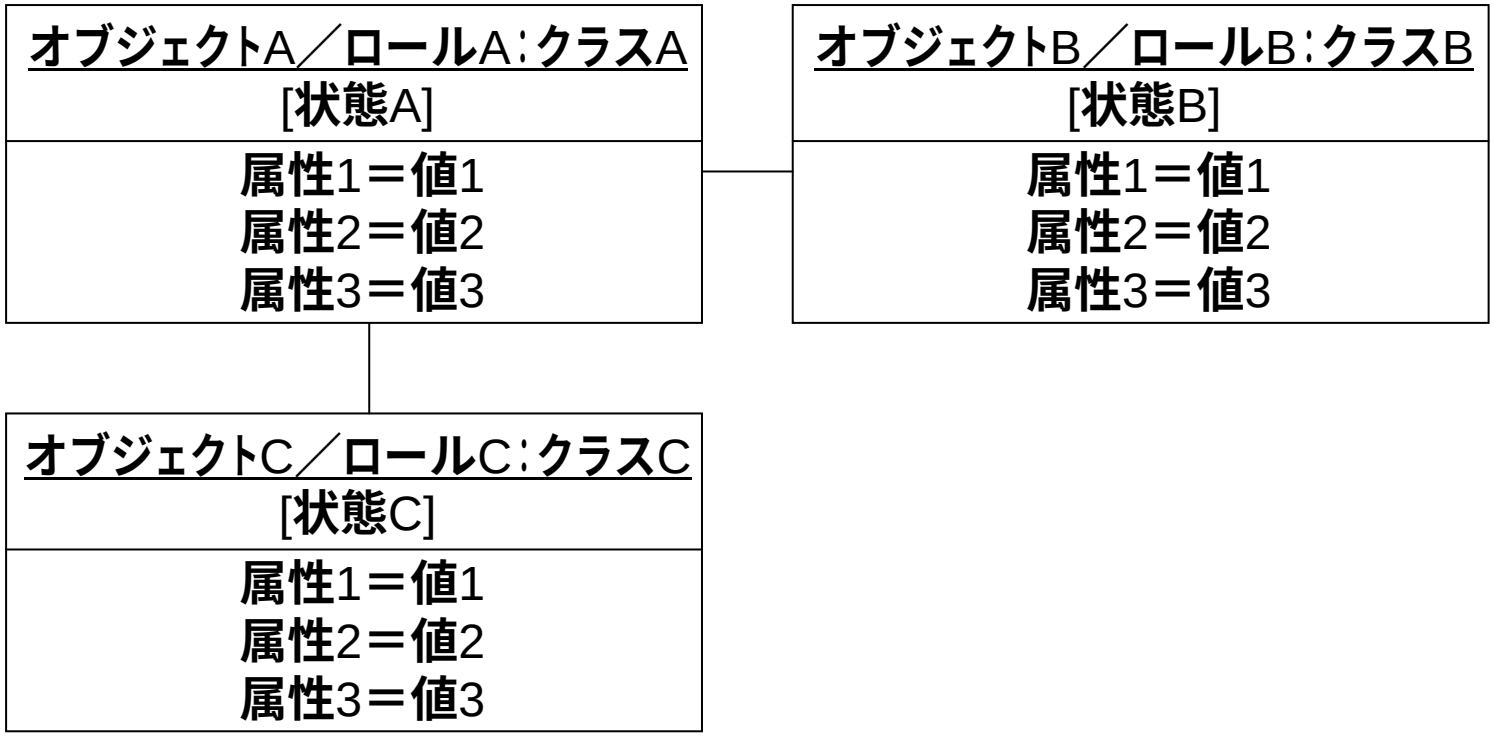
オブジェクト名／ロール：クラス名 [状態名]
属性名＝値

8.1 オブジェクト図 (2)

分析・設計

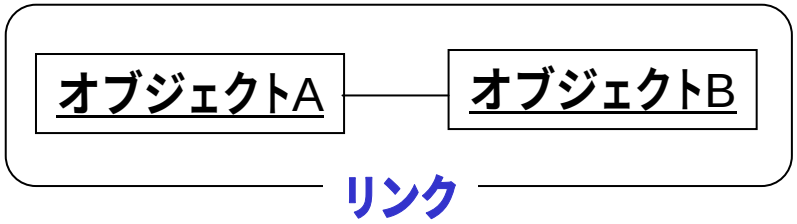
□オブジェクト図

オブジェクト間の静的な関係を把握するために作成
システムのある時点におけるスナップショットを表す



8.1 オブジェクト図 (3)

□オブジェクト図の要素



分析・設計

8.1 オブジェクト図 (4)

□オブジェクト図を使ったモデリング

「世界中の酒」の通信販売を行っている兄弟社は、
手書きの受注伝票を廃止して、販売管理システム
に受注情報を登録することになった。
ここでは、受注関連の部分だけを考える。

□業務概要

・受注担当者は、注文の電話を受け、顧客情報、商品情報、受注情報を
システムに登録する。
・登録する情報は、受注した商品ごとに分け、それぞれ顧客情報、商品情報、
受注情報を登録する。
・受注担当者は登録の際に、入力した情報を確認して確定を行う

分析・設計

8.1 オブジェクト図（5）

□シナリオ

常連客であり、酒の小売業を営むAさん（顧客番号：034、住所：京都市左京区吉田XXX、電話番号：075-753-XXXX）から、本日以下の注文が入ったとする。

焼酎「キョウダイ」（商品コード001、単価980円）：200本
ワイン「ブジョレー」（商品コード026、単価1280円）：300本
日本酒「寒ばしり」（商品コード003、単価4280円）：50本

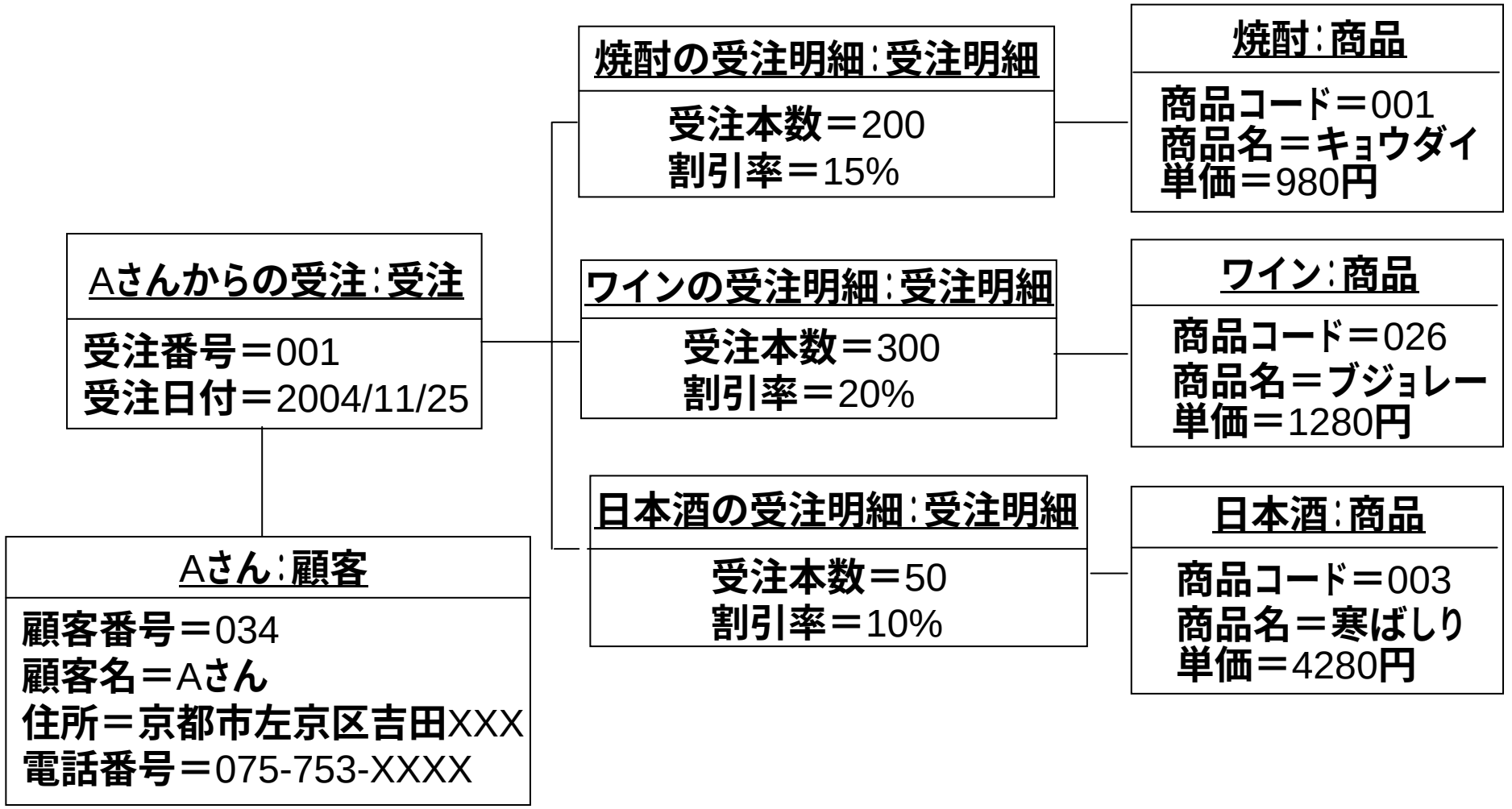
常連客であるAさんの焼酎、ワイン、日本酒の割引率はそれぞれ15%、20%、10%である。
業務概要をオブジェクト図を用いてモデル化する。

（ヒント）
「Aさん：顧客」、「Aさんからの受注：受注」、
「焼酎の受注明細：受注明細」、「ワインの受注明細：受注明細」、
「日本酒の受注明細：受注明細」、
「焼酎：商品」、「ワイン：商品」、「日本酒：商品」

これらの関係を考える

8.1 オブジェクト図(6)

□オブジェクト図を使ったモデリング

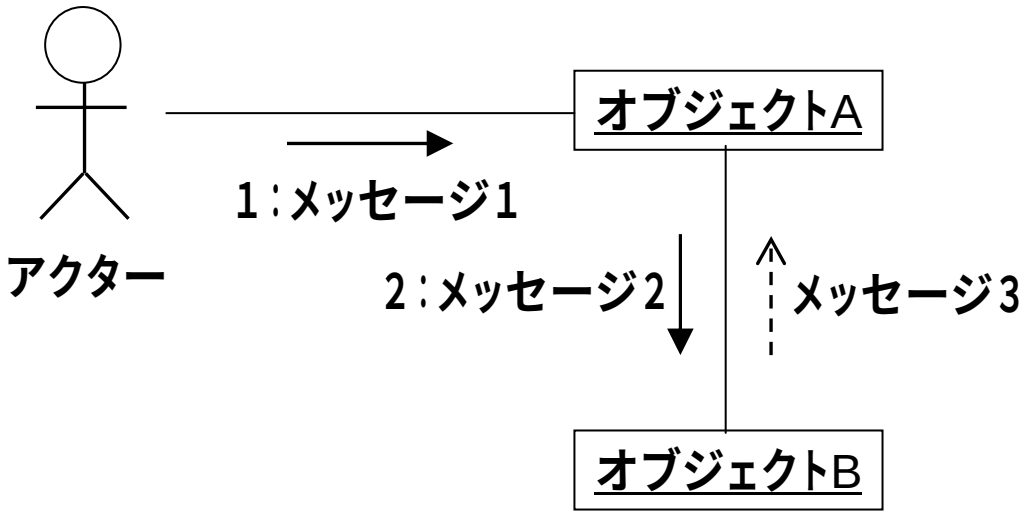


分析・設計

8.2 コラボレーション図 (1)

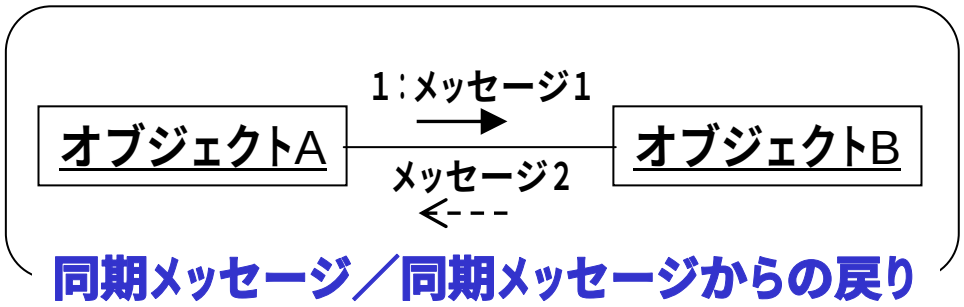
□コラボレーション図

「相互作用図」の一種
オブジェクト間の関係を重視する

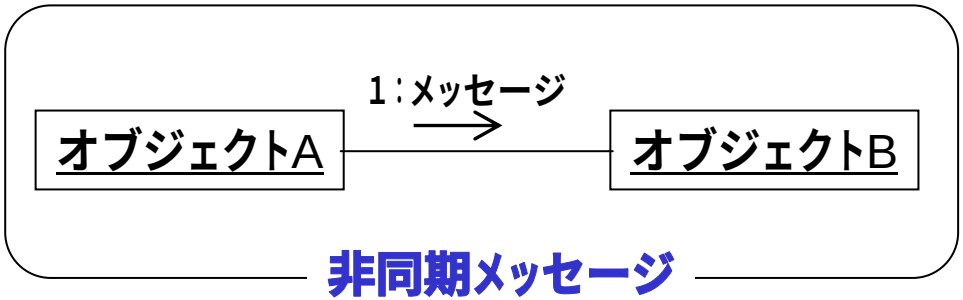


8.2 コラボレーション図 (2)

□コラボレーション図の要素 (1)



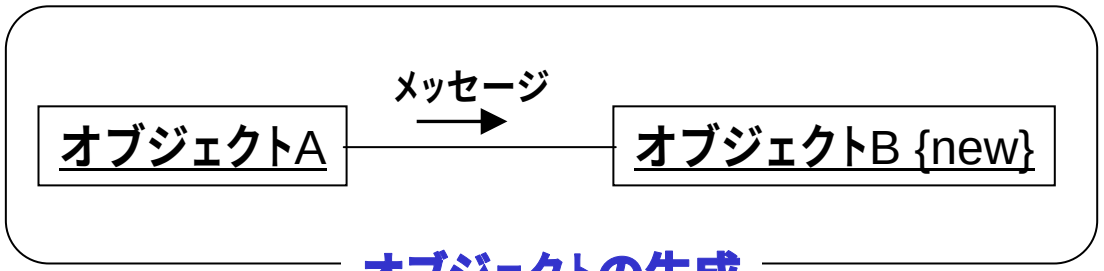
同期メッセージ／同期メッセージからの戻り



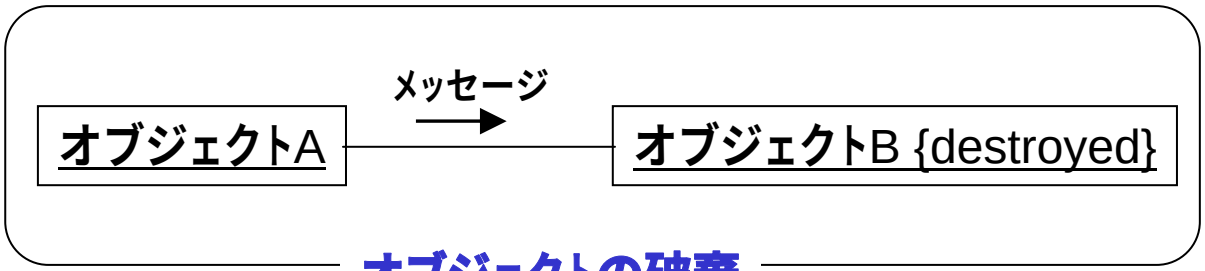
非同期メッセージ

8.2 コラボレーション図 (3)

□コラボレーション図の要素 (2)



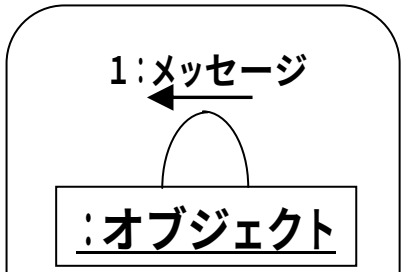
オブジェクトの生成



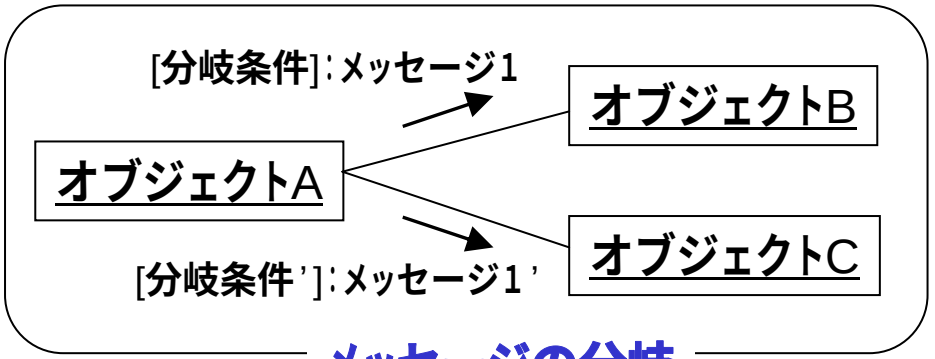
オブジェクトの破棄

8.2 コラボレーション図 (4)

□コラボレーション図の要素 (3)

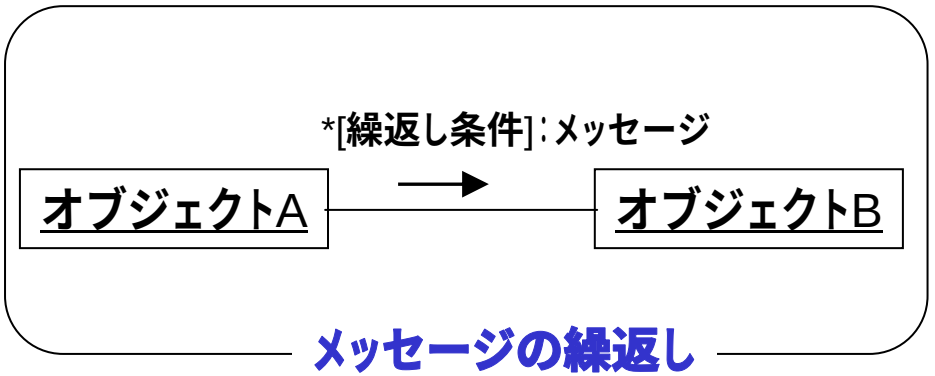


自己メッセージ



メッセージの分岐

パターン



メッセージの繰返し

8.2 コラボレーション図 (5)

□コラボレーション図を使ったモデリング

「世界中の酒」の通信販売を行っている兄弟社は、
販売管理システムを導入することになった。
ここでは、受注情報登録処理を実現するオブジェクト間の
相互作用だけを考える。

8.2　コラボレーション図 (6)

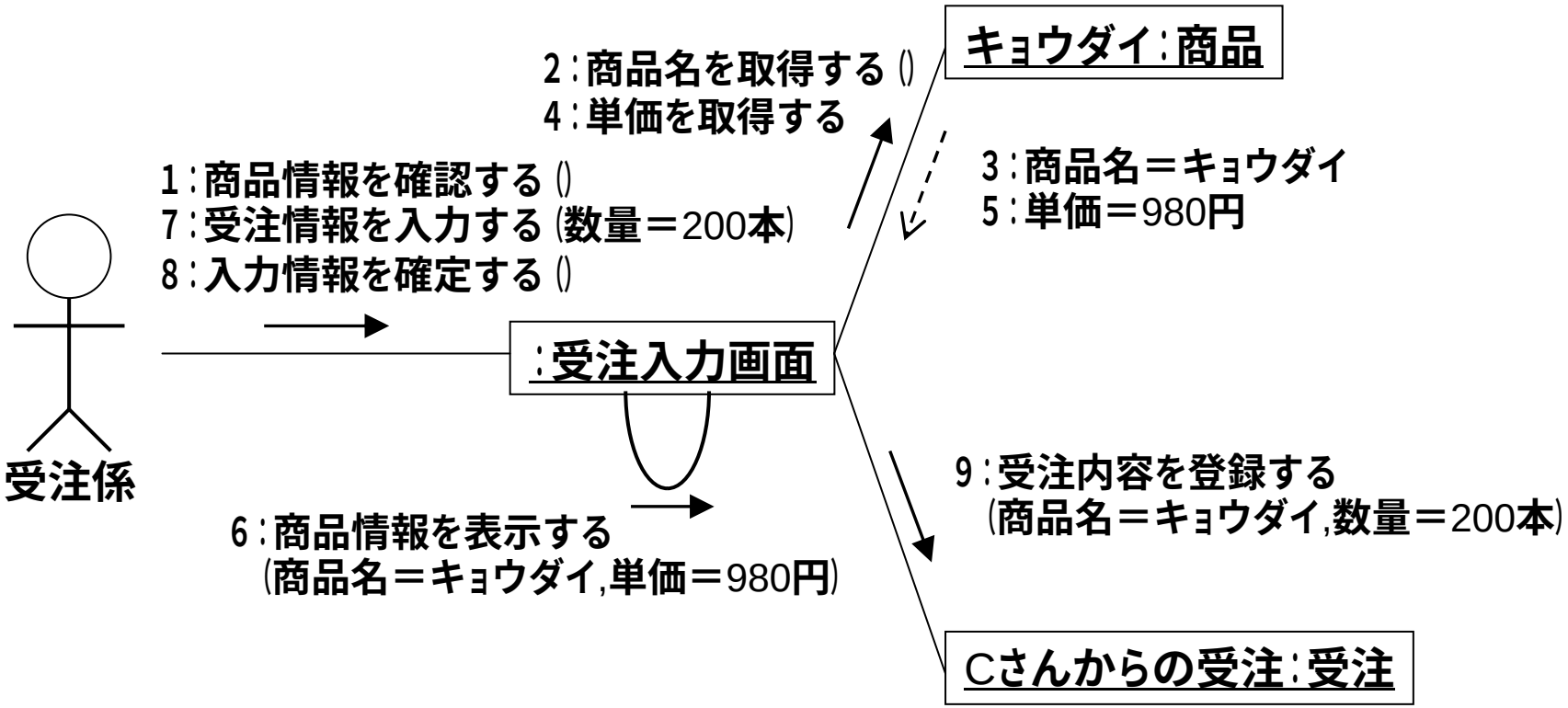
□シナリオ

受注係は、Cさんからの注文「焼酎キョウダイ200本」に応じて、
1．商品情報を確認し、
2．商品名をデータから取得する
3．データからは商品名「キョウダイ」が返される
4．単価情報をデータから取得する
5．商品単価が入力画面に返される
6．商品単価が入力画面に表示される
7．受注情報を入力する
8．入力情報を確定する
9．受注内容が登録される

(ヒント)
「受注係」、「受注入力画面」、「商品」、「受注」の関係を考える

8.2 コラボレーション図 (7)

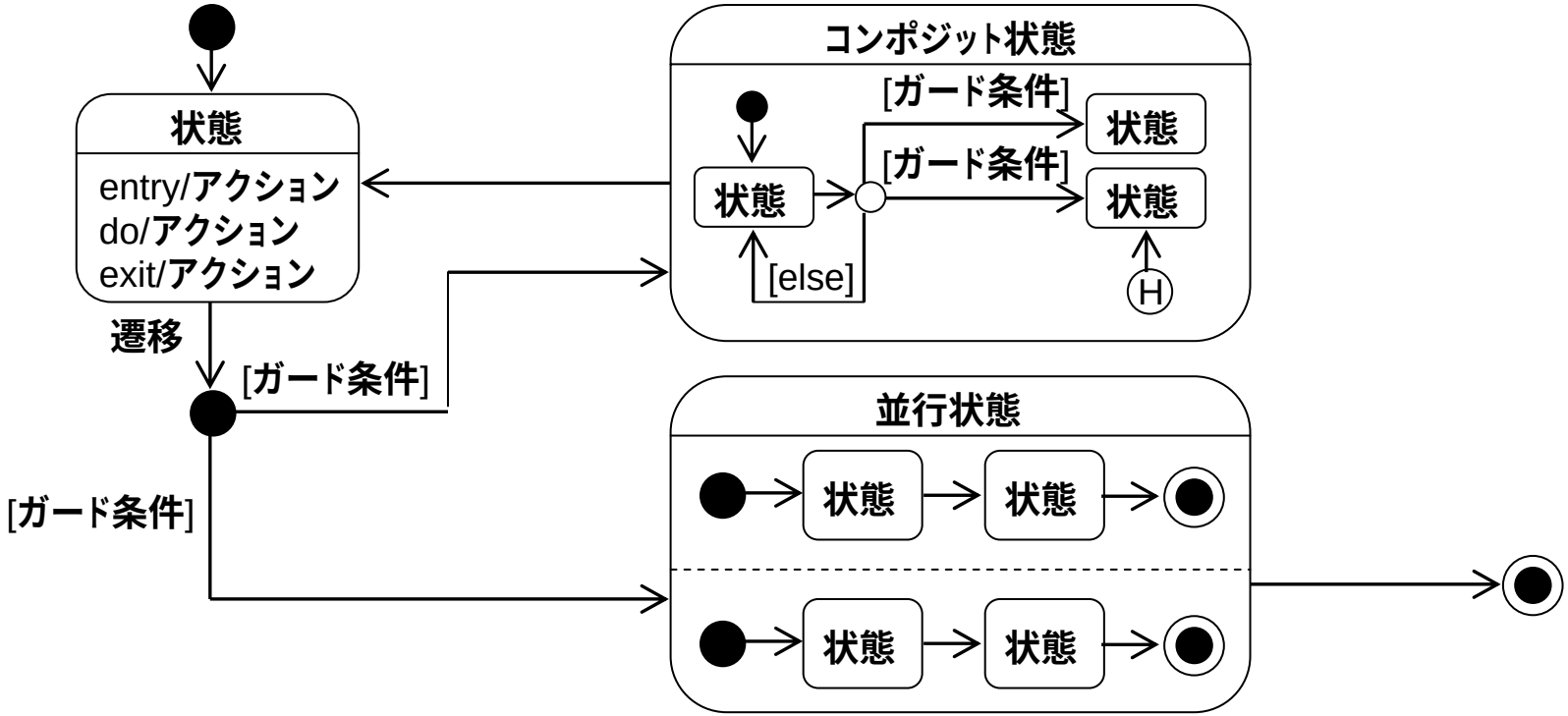
□コラボレーション図を使ったモデリング



8.3 ステートチャート図 (1)

□ステートチャート図

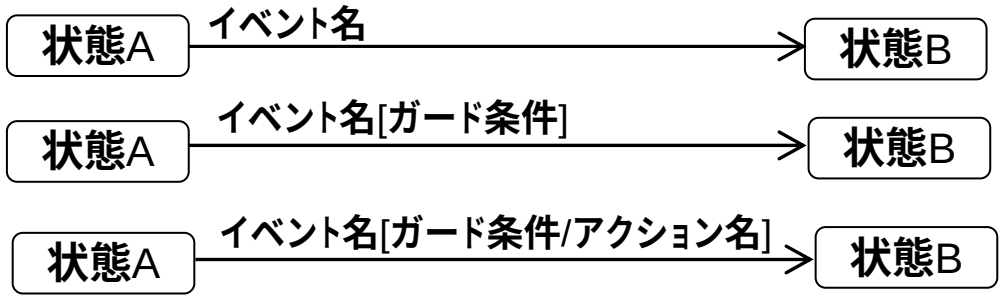
1つのオブジェクトの状態がイベントの発生や時間経過とともにどのように変化するかを表す。
特定のオブジェクトに注目して、生成から消滅までのライフサイクルをモデル化する。



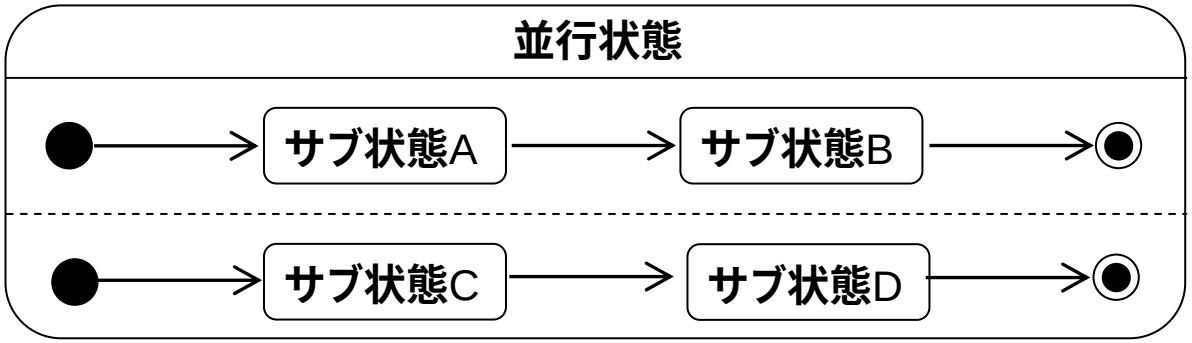
8.3 ステートチャート図 (2)

□ステートチャート図の要素 (1)

状態



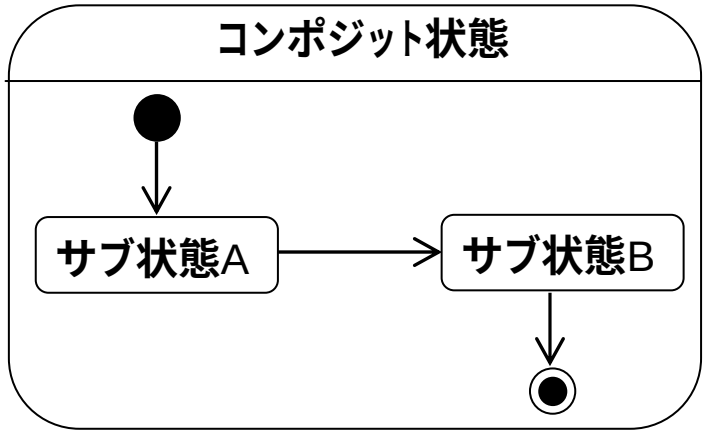
遷移



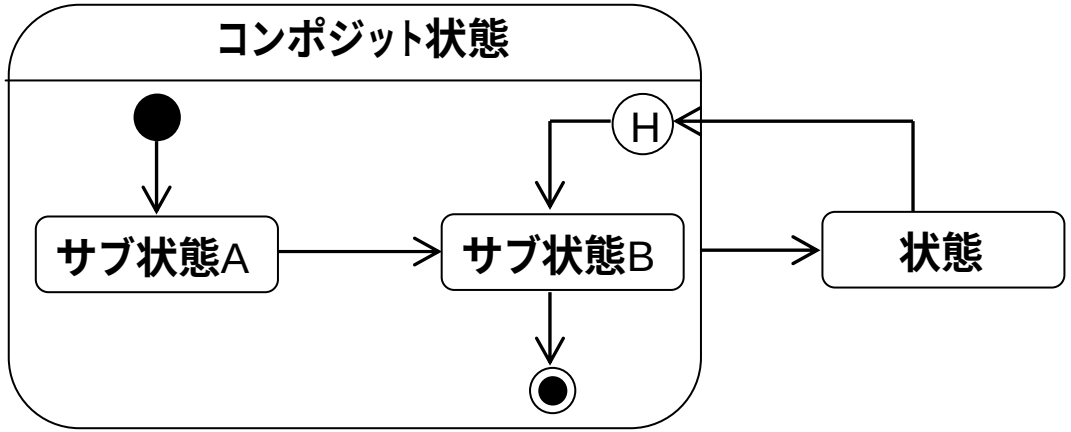
並行状態

8.3 ステートチャート図 (3)

□ステートチャート図の要素 (2)



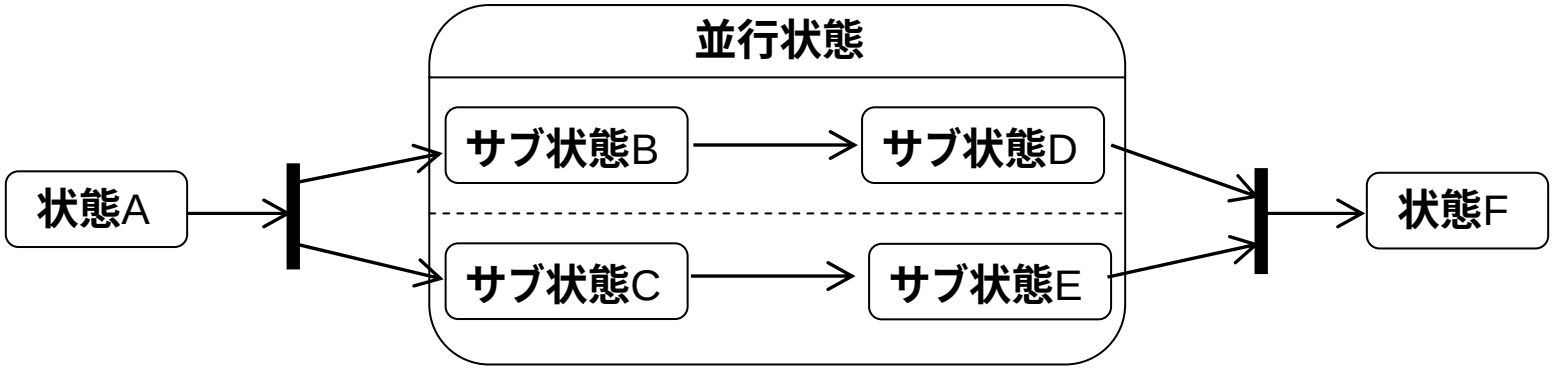
コンポジット状態



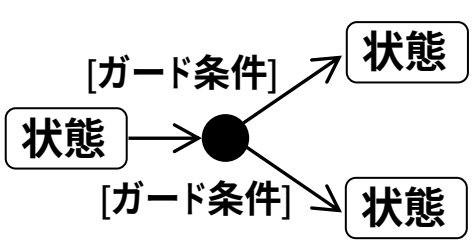
履歴状態指示子

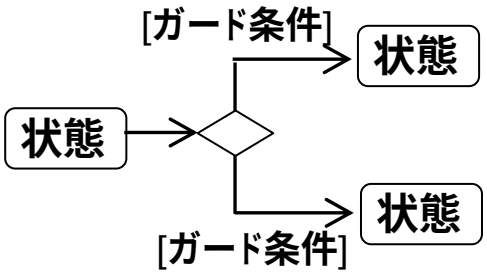
8.3 ステートチャート図 (4)

□ステートチャート図の要素 (3)

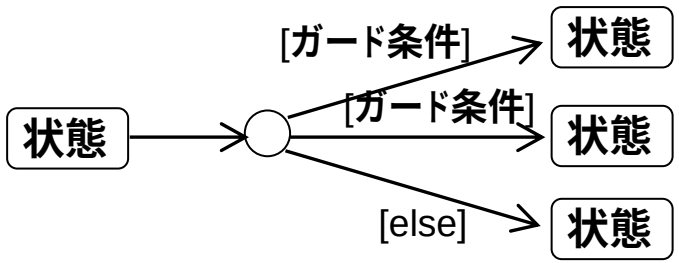


同期バー





連結点



動的選択点

8.3 ステートチャート図 (5)

□ステートチャート図を使ったモデリング

「世界中の酒」の通信販売を行っている兄弟社は、
販売管理システムを導入することになった。
ここでは、オブジェクト「受注」の状態遷移を考える。

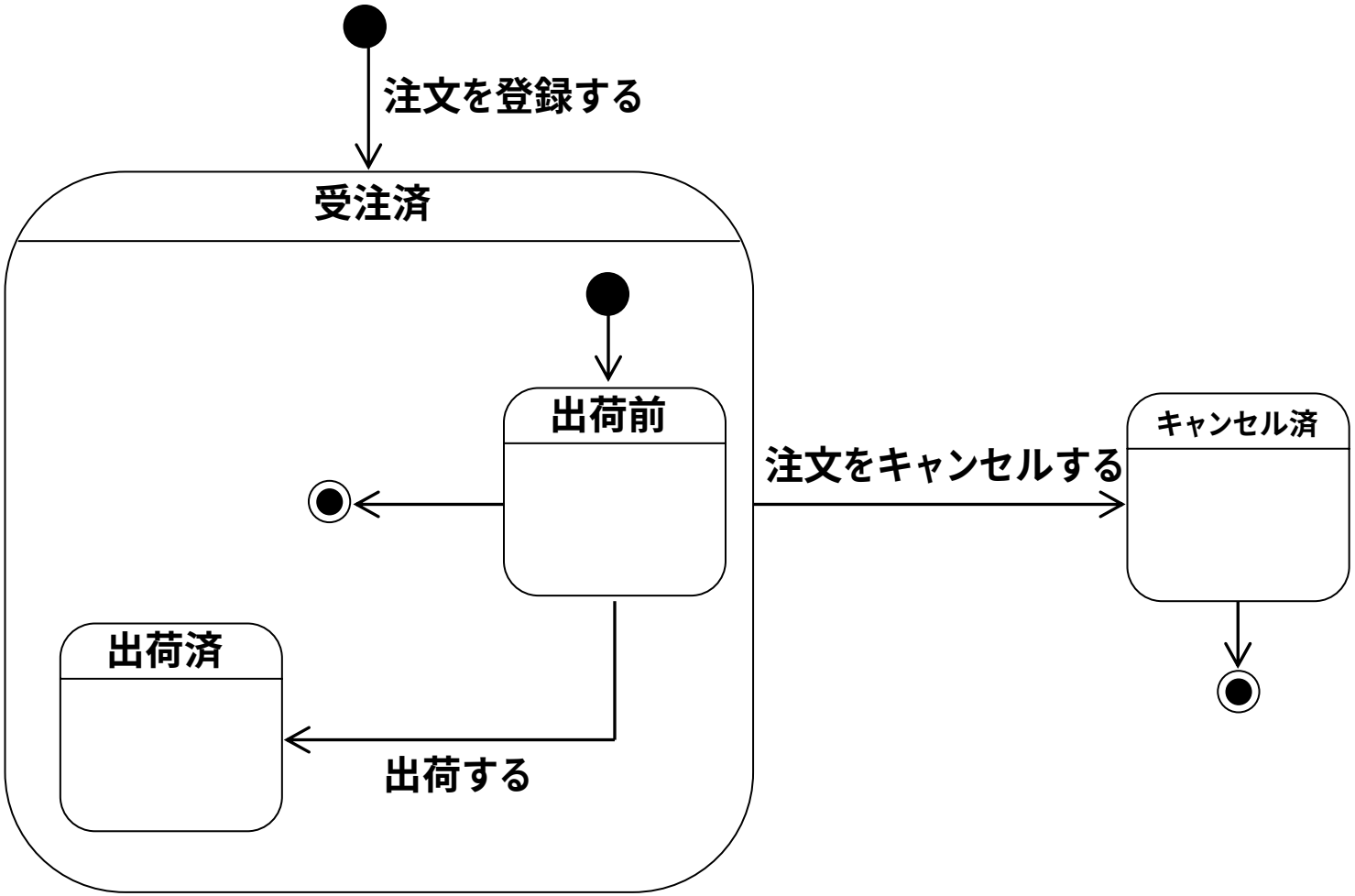
8.3 ステートチャート図 (6)

□シナリオ

「注文を登録する」と注文は「受注済」となる。
受注注文は「出荷前」と「出荷済」に遷移する。
場合によっては、受注後に「キャンセル」になる。
この場合、詳細な条件は記述しないものとする。

8.3 ステートチャート図 (7)

□ステートチャート図を使ったモデリング



実装

8.4 クラス図 (1)

□クラス

オブジェクトの共通的要素を抽象化し、それを枠組みとして定義したもの
クラスとオブジェクトは「雛形」と「その実体」という関係

□表記方法

クラス名
属性1：型＝初期値 属性2：[多重度 ordered]
操作（パラメータ：型＝デフォルト値）：戻り値の型

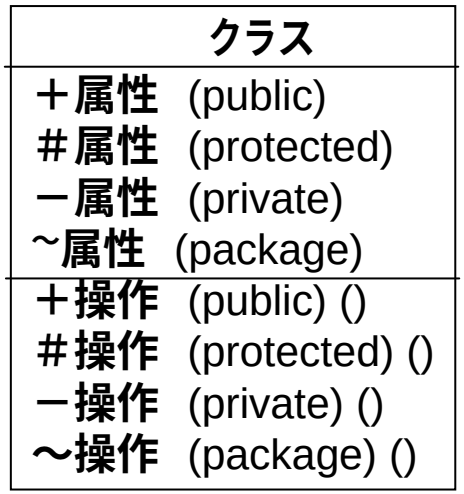
8.4 クラス図 (2)

□クラス図

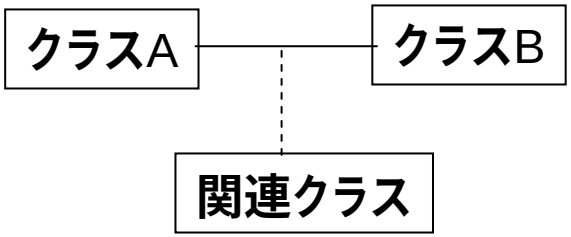
「クラス」というオブジェクト指向の設計単位を用いて
システムの静的な構造 (モデル) を表す

8.4 クラス図 (3)

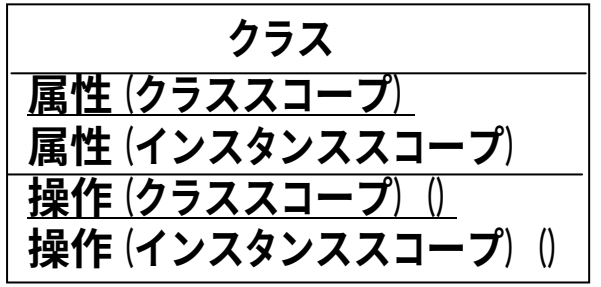
□クラス図の要素 (1)



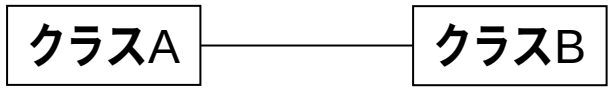
可視性



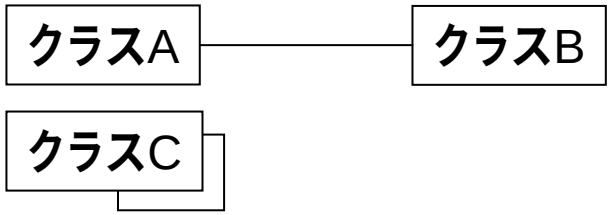
関連クラス



スコープ



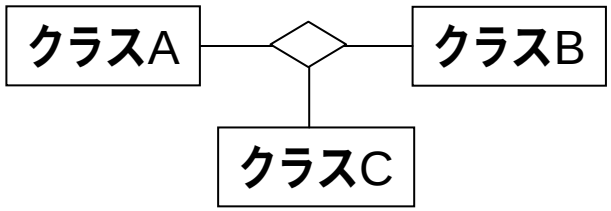
関連



二項関連

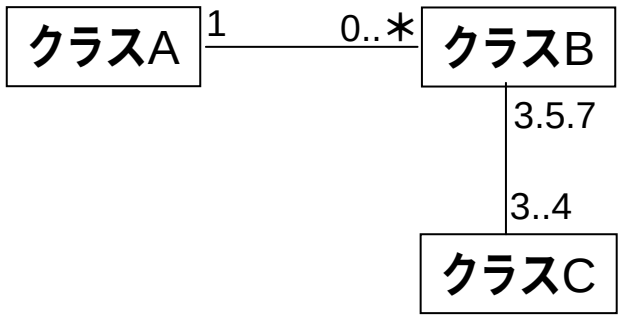
8.4 クラス図 (4)

□クラス図の要素 (2)

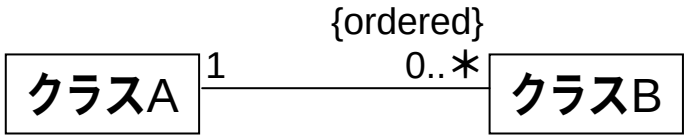


N項関連

関連名



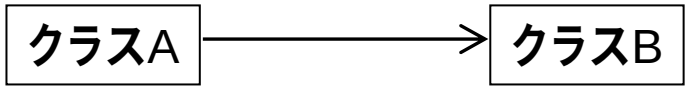
多重度



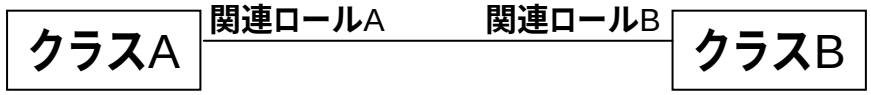
順位付け

8.4 クラス図 (5)

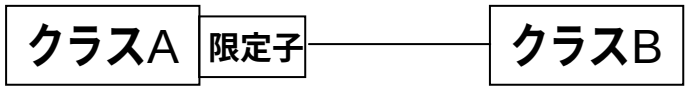
□クラス図の要素 (3)



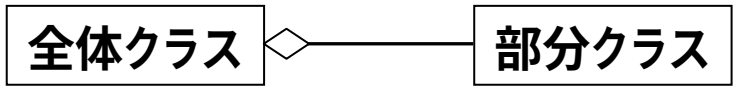
誘導可能性



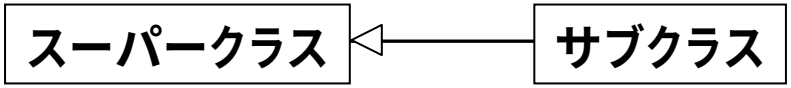
関連ロール



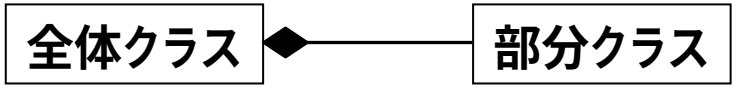
限定子



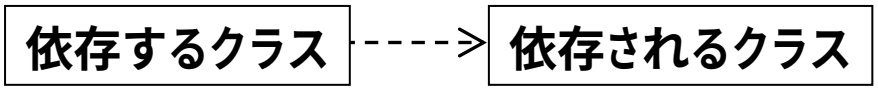
集約



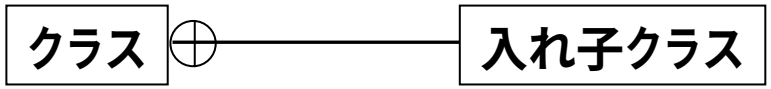
汎化



コンポジション



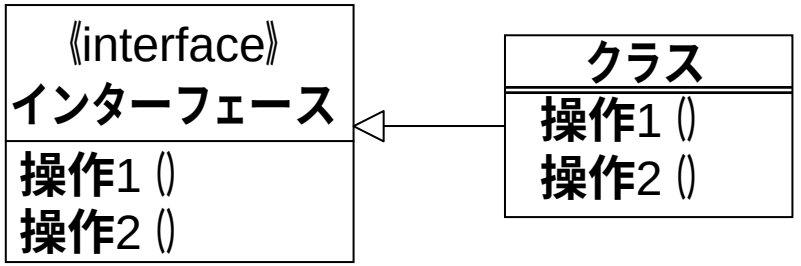
依存

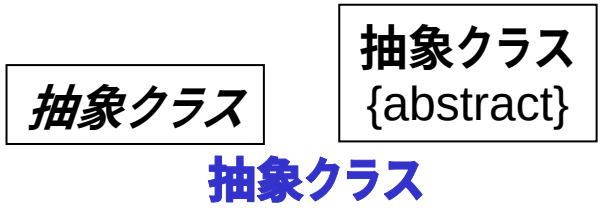


入れ子クラス

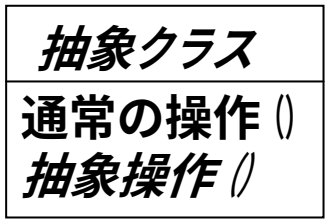
8.4 クラス図 (6)

□クラス図の要素 (4)

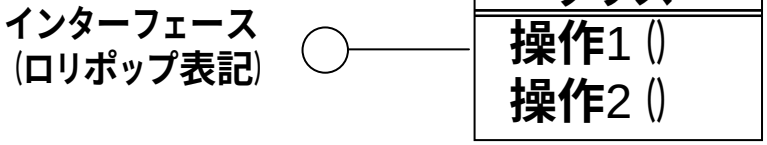




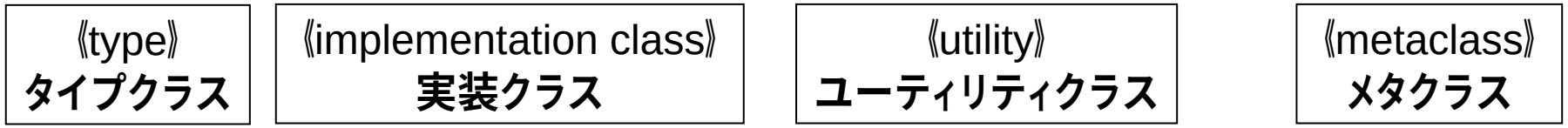
抽象クラス



抽象操作



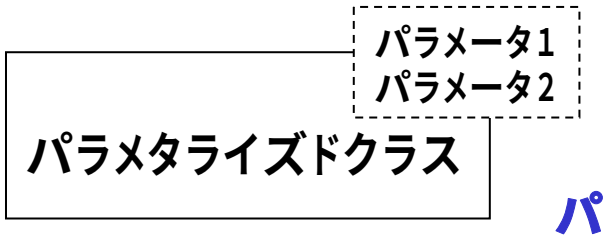
実現



タイプクラスと実装クラス

ユーティリティクラス

メタクラス



パラメタライズドクラス

8.4 クラス図 (7)

□クラス図を使ったモデリング

「世界中の酒」の通信販売を行っている兄弟社は、
顧客に対する柔軟な割引価格の提示により売り上げを
伸ばそうとしている。時期や顧客によって割引率の異なる
商品情報を扱うために受注から在庫の確認、受注明細の
登録までを管理するシステムを構築することになった。

8.4 クラス図 (8)

□シナリオ

受注係は、受注入力画面に受注内容、顧客情報を入力する。
商品の倉庫における在庫状況が確認され、受注明細が決定するものとする。

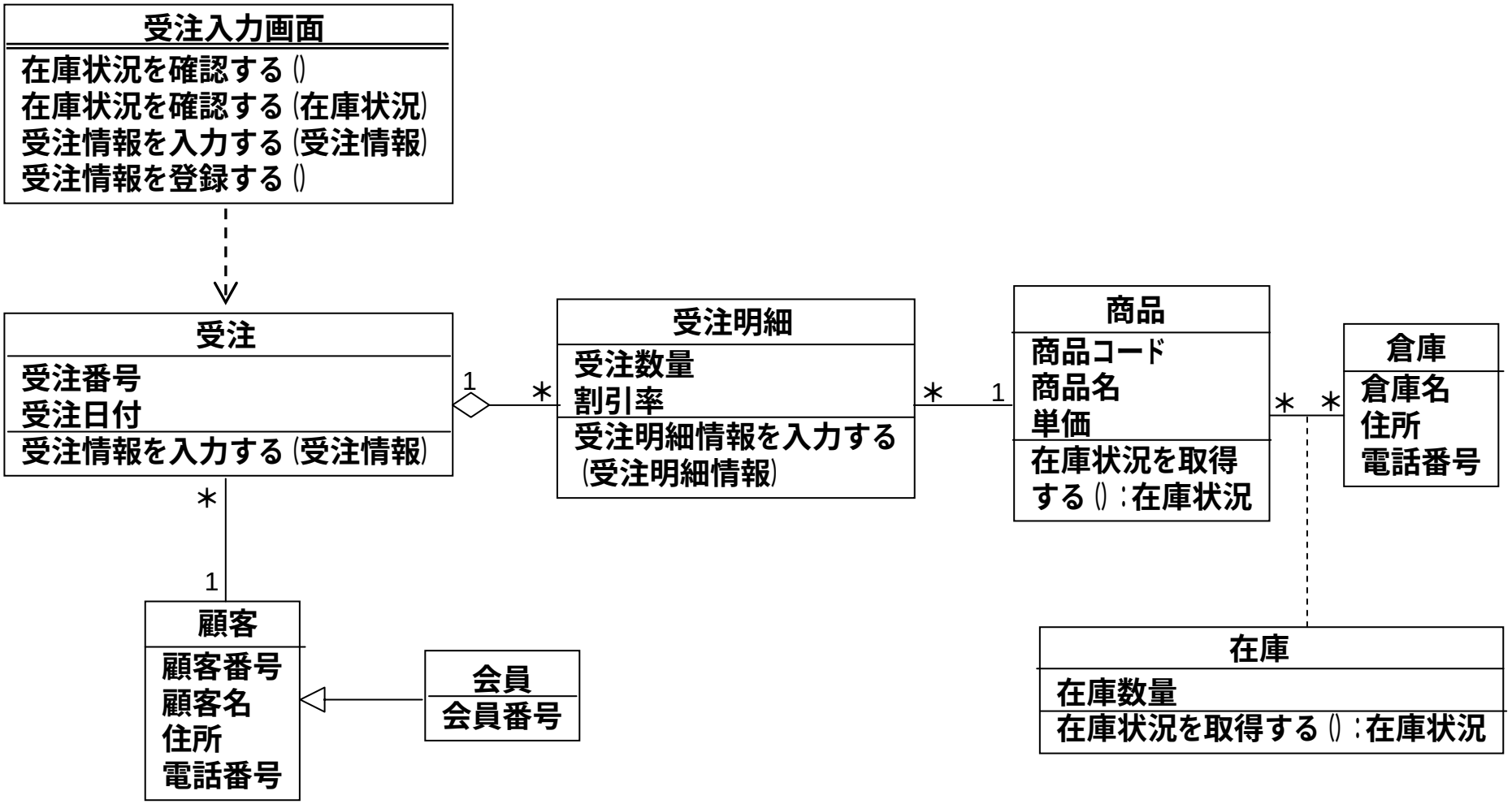
□クラス仕様

各クラスの仕様は以下のとおりとする。

- 受注入力画面：在庫状況を確認する ()、在庫状況を確認する (在庫状況)
  受注情報を入力する (受注情報)、受注情報を登録する ()
- 受注：受注番号、受注日付、受注情報を入力する (受注情報)
- 受注明細：受注数量、割引率、受注明細情報を入力する (受注明細情報)
- 商品：商品コード、商品名、単価、在庫状況を取得する ()
- 倉庫：倉庫名、住所、電話番号
- 在庫：在庫数量、在庫状況を取得する ()
- 顧客：顧客番号、顧客名、住所、電話番号
- 会員：会員番号

8.4 クラス図 (9)

□クラス図を使ったモデリング



8.4 クラス図（10）

□クラス図とコーディング

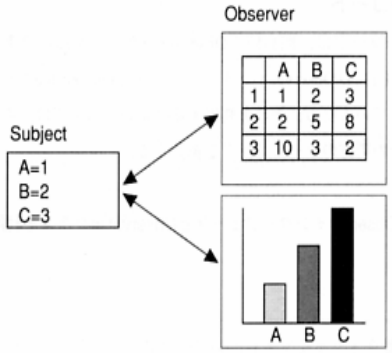


図1-55　Observerパターンの例

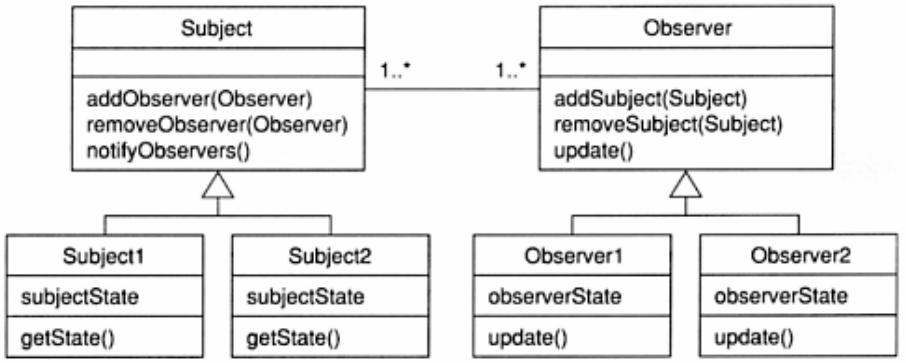


図1-56 Observerパターンの構造

```
import java.util. *;
public class Subject
{
  private Vector observers = new Vector();
  public void addObserver (MyObserver o) {
     Observers. addElement(o);
  }
  ・・・・・・・
}
```

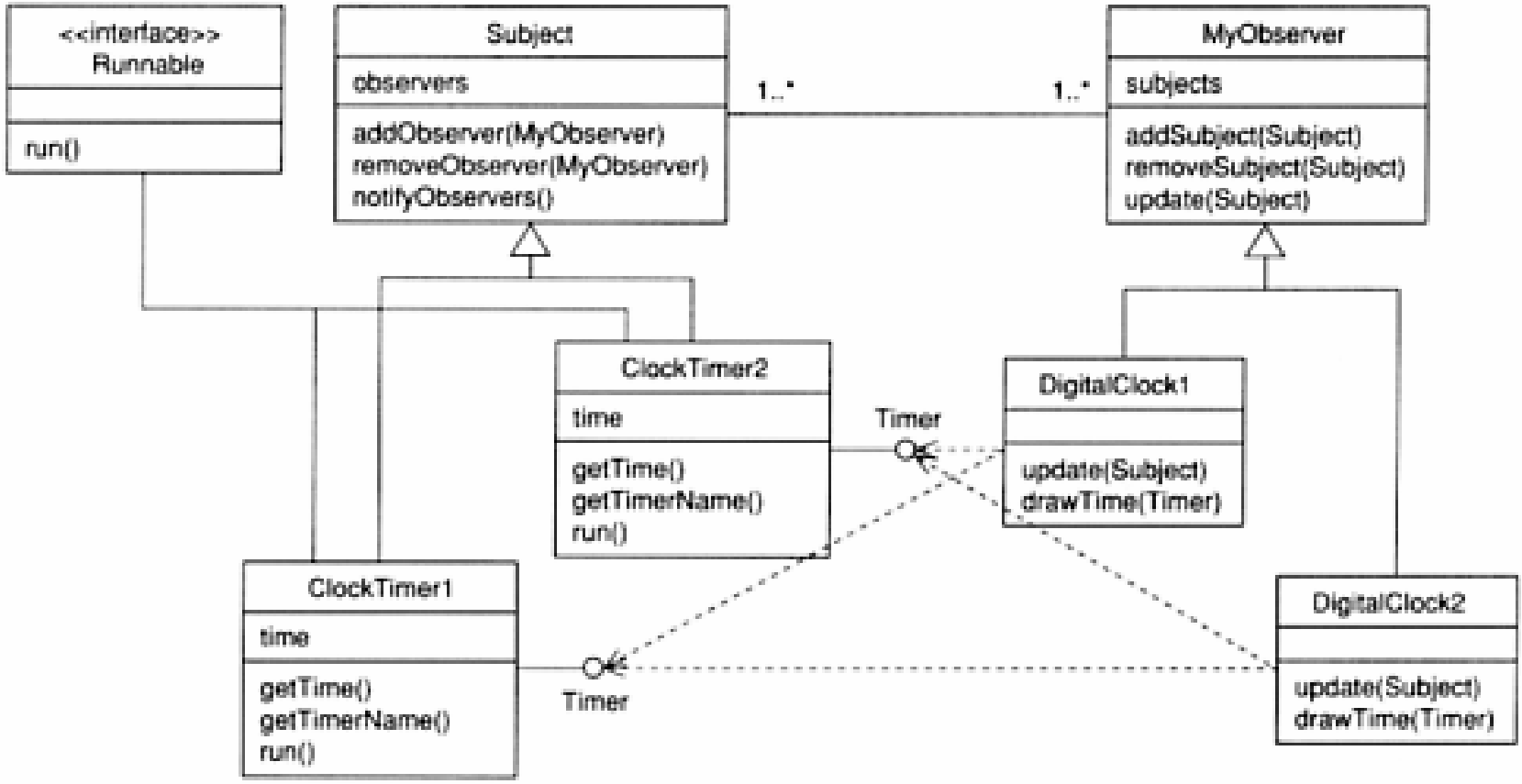


図1-57　ClockTimerとDigitalClock

Subjectクラス

```
import java.util. *;
public class Subject
{
  private Vector observers = new Vector();
  public void addObserver (MyObserver o) {
      Observers. addElement(o);
  }
  public void removeObserver ⑴MyObserver o) {
      Observers. removeElement(o);
  }
  public void notifyObservers() {
      Enumeration e = Observers. elements();
       While (e.hasMoreElements() ){
           MyObserver o = (MyObserver)e.nextElement();
        o.update(this);
            }
    }
}
```

Timerインターフェース

```
public interface Timer
{
   public int getTime();
   public String getTimerName();
}
```

ClockTimer1クラス

```
public class ClockTimer1 extends Subject implements
Runnable,Timer
{
	private Thread timer;
	private int time = 0;
	private String timerName = "ClockTimer1";
	public ClockTimer1() {
		timer = new Thread(this);
		timer.start();
	}
	public int getTime() {
		return time;
	}
	public String getTimerName() {
		return timerName;
	}
	public void run() {
		While(true) {
			try {
				timer.sleep(1000);
			}catch(Exception e) {
			}
			time = time+1;
			notifyObservers();
		}
	}
}
```

ClockTimer2クラス

```
public class ClockTimer2 extends Subject implements
Runnable,Timer
{
        protected Thread timer;
        protected int time = 0;
        private String timerName = "ClockTimer2";
        public ClockTimer2 () {
              timer = new Thread(this);
              timer.start();
        }
        public int getTime() {
              return time;
        }
        public String getTimerName() {
              return timerName;
        }
        public void run() {
              While(true) {
                    try {
                          [①]
                    }catch(Exception e) {
                    }
                          [②]
                    notifyObservers();
              }
        }
}
```

MyObserverクラス

```
lmport java.util.*;
public class MyObserver
{
      protected Vector Subjects = new Vector();
      public MyObserver() { }
      public MyObserver(Subject s) {
            s.addObserver(this);
            subjects.addElement(s);
      }
      public void addSubject(Subject s) {
            s.addObserver(this);
            subjects. addElement(s);
      }
      public void removeSubject(Subject sS) {
            s.removeObserver(this);
            subjects. removeElement(s);
      }
      public void update(Subject changedSubject) {
      }
}
```

DigitalClock1クラス

```
public class DigitalClock1 extends MyObserver
{
    private String name="DigitalClock1";
    public DigitalClock1(Subject s) {
        super(s);
    }
    public void update(Subject changedSubject) {
        drawTime ( (Timer)changedSubject);
    }
    public void drawTime(Timer timer) {
        System.out.println(name+":"+timer.getTimerName()+
        "="+timer.getTime() );
    }
}
```

DigitalClock2クラス

```
public class DigitalClock2 extends MyObserver
{
    private String name="DigitalClock2";
    public DigitalClock2 (Subject s) {
        super(s);
    }
    public void update(Subject changedSubject) {
        drawTime ( (Timer)changedSubject);
    }
    public void drawTime(Timer timer) {
        System.out.println(name+":"+timer.getTimerName()+
        "="+timer.getTime() );
    }
}
```

ObserverSampleクラス (メインプログラム)

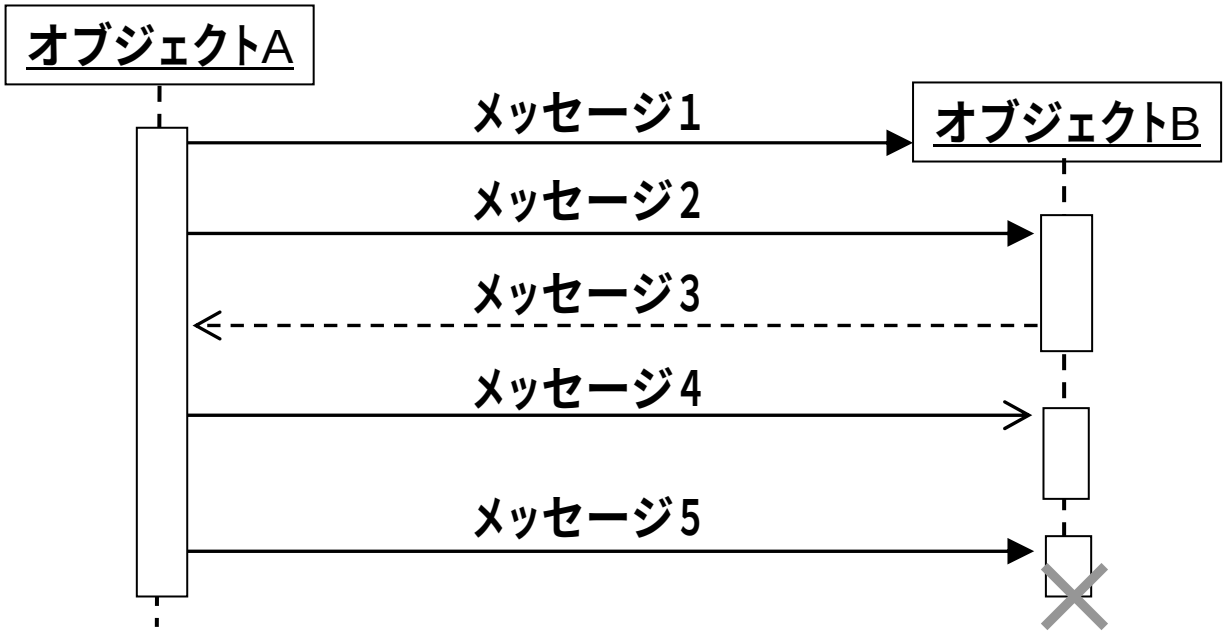
```
public class ObserverSample
{
    public static void main(String [ ] args)
    {
        ClockTimer1 timer1 = new ClockTimer1( );
        ClockTimer2 timer2 = new ClockTimer2( );
        DigitalClock1 dc1 = new DilgitalClock1(timer1);
        dc1.addSubject(timer2);
        DigitalClock2 dc2 = new DigltalClock2(timer1);
    }
}
```

実装

8.5 シーケンス図 (1)

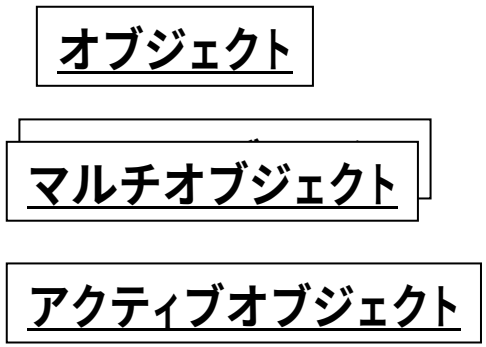
□シーケンス図

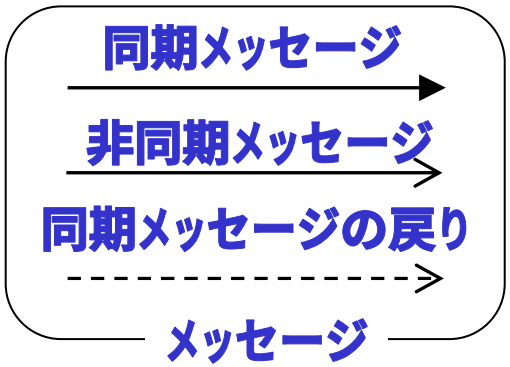
「相互作用図」のこと
オブジェクト間のメッセージのやり取りを時系列で表す
やりとりする内容・手順・時間を明確にする

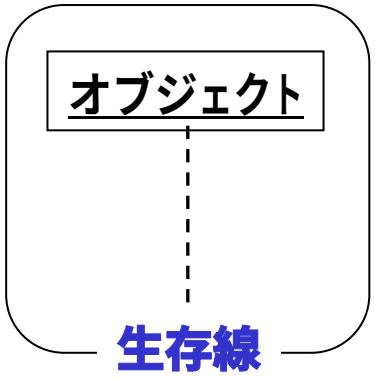


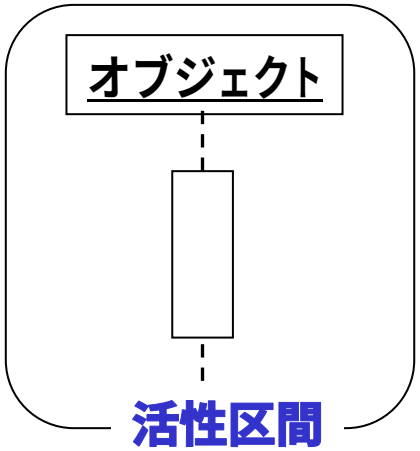
8.5 シーケンス図 (2)

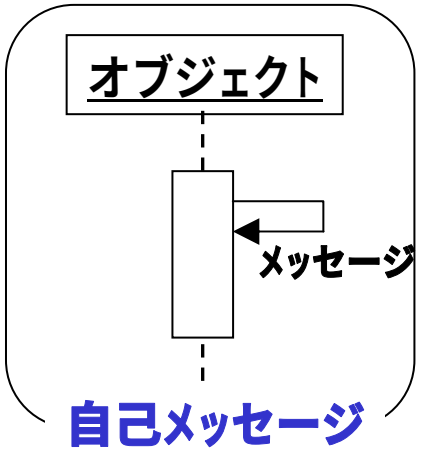
□シーケンス図の要素 (1)

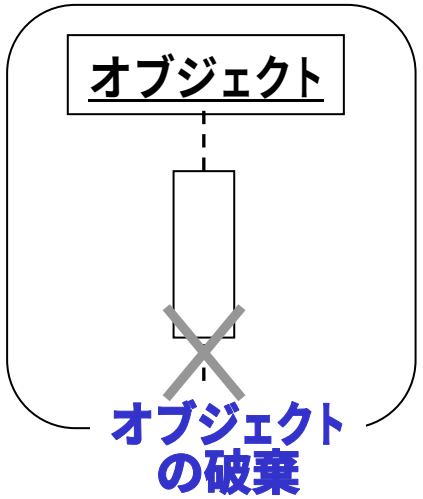






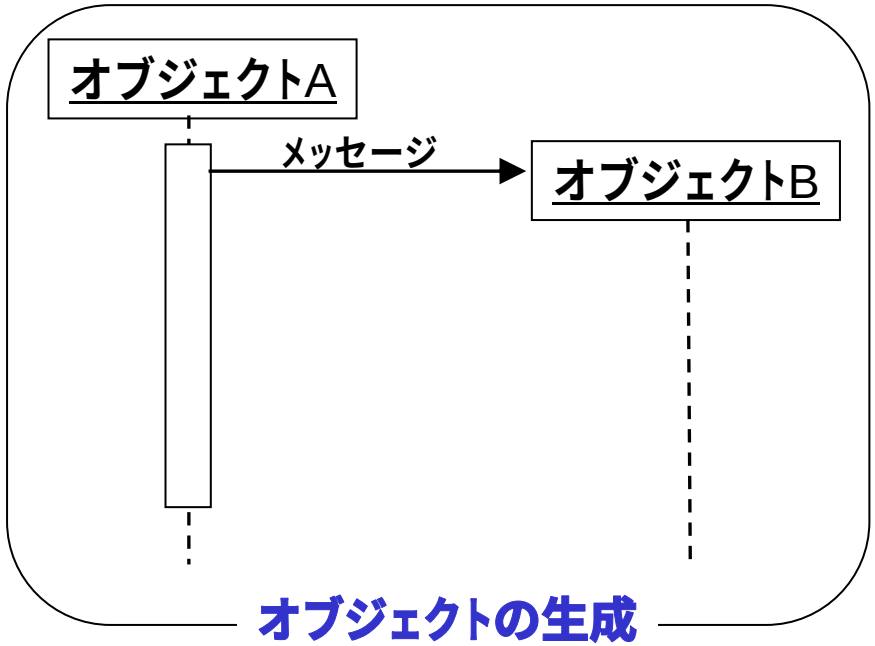




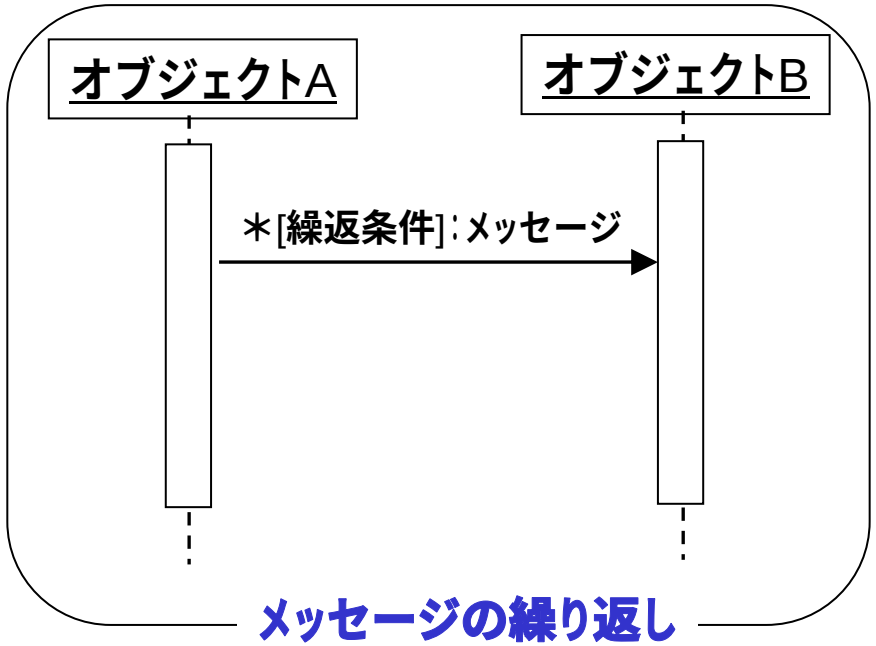


8.5 シーケンス図 (3)

□シーケンス図の要素 (2)



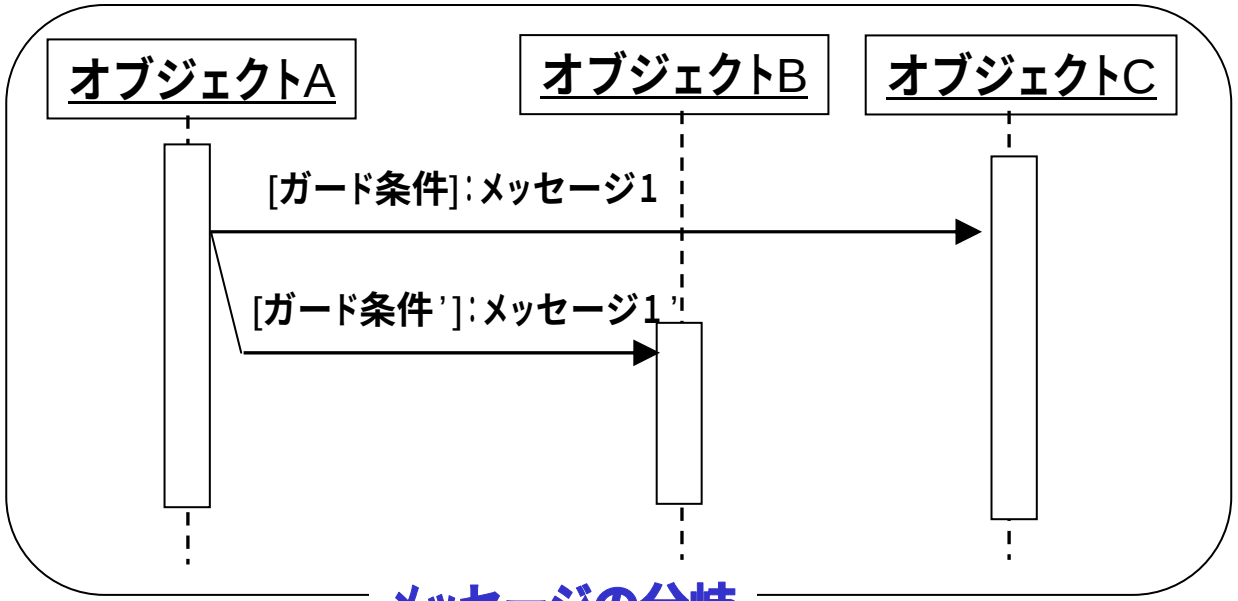
オブジェクトの生成



メッセージの繰り返し

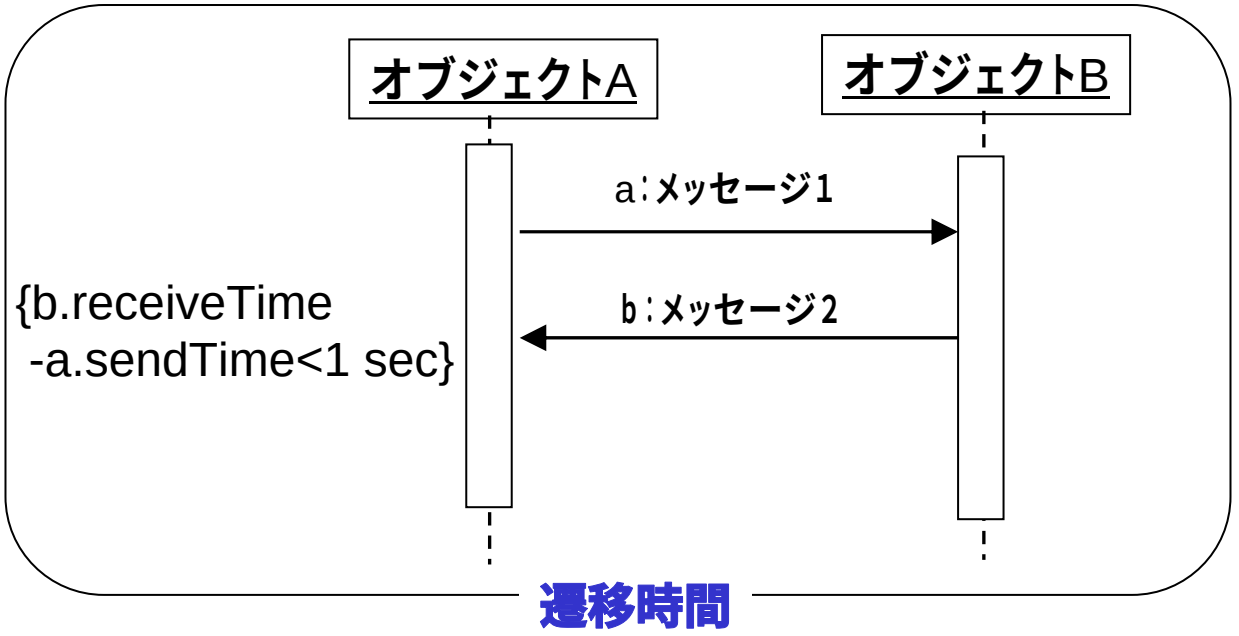
8.5 シーケンス図 (4)

□シーケンス図の要素 (3)



8.5 シーケンス図 (5)

□シーケンス図の要素 (4)



8.5 シーケンス図 (6)

□シーケンス図を使ったモデリング

「世界中の酒」の通信販売を行っている兄弟社は、
販売管理システムを導入することになった。
ここでは、受注情報登録処理を実現するオブジェクト間の
相互作用だけを考える。

8.5 シーケンス図 (7)

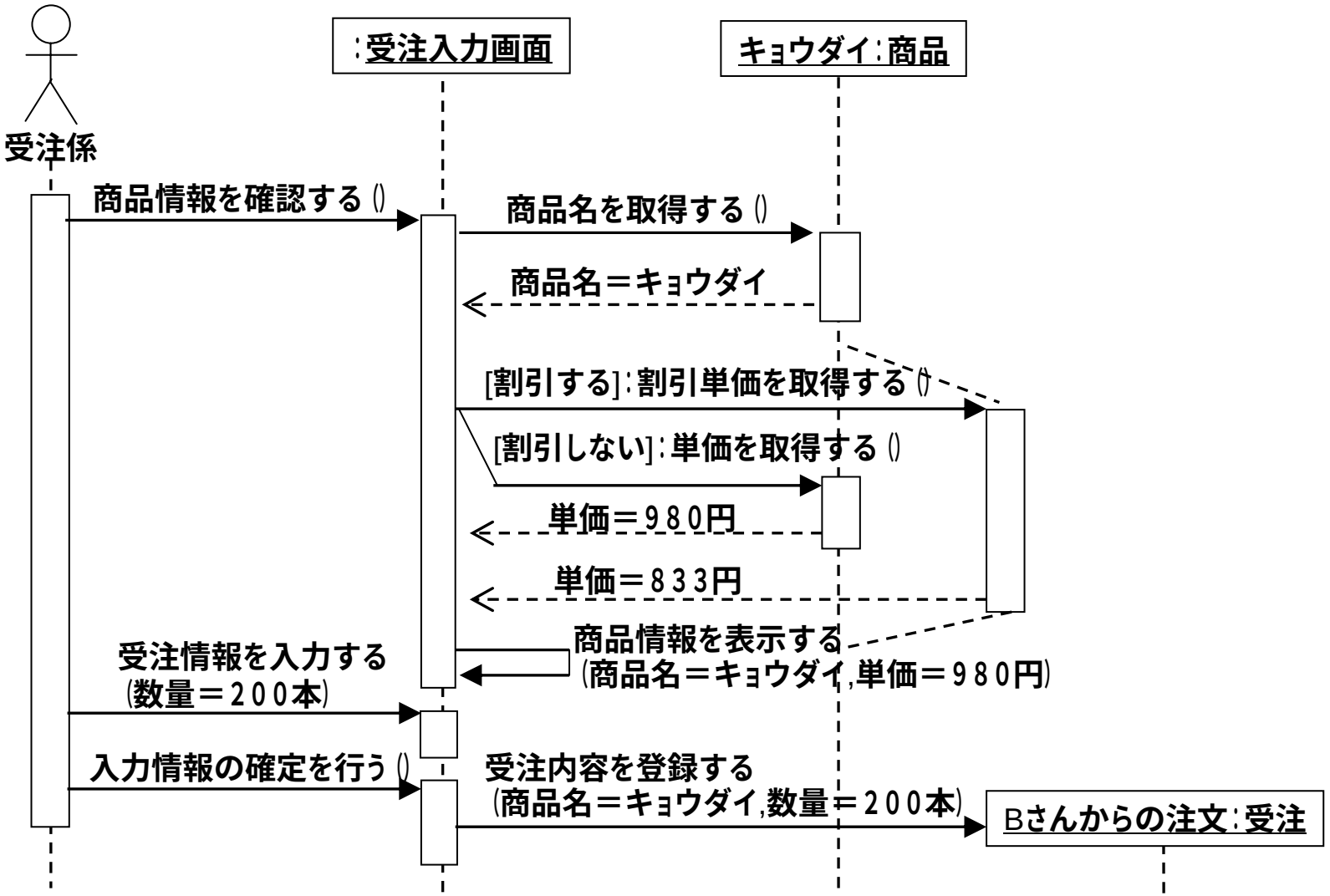
□シナリオ

Bさんから焼酎キョウダイに対して200本の注文があった
各オブジェクト間のやり取りは以下のようなものとする
オブジェクトは受注係、受注入力画面、商品情報、Bさんからの注文とする

・受注係は入力画面に対して商品名を確認する
・商品情報から商品名を取得する
・商品情報から商品名「キョウダイ」が入力画面に出される
・商品情報から割引する場合、単価は833円と入力画面に出される
・商品情報から割引しない場合、単価は980円と入力画面に出される
・受注入力画面に商品情報を出力する (商品名、単価)
・受注係は入力画面に対して受注情報 (本数) を入力する
・受注係は入力情報を確定する
・入力画面から受注内容をBさんからの注文に登録する

8.5 シーケンス図(8)

□シーケンス図を使ったモデリング

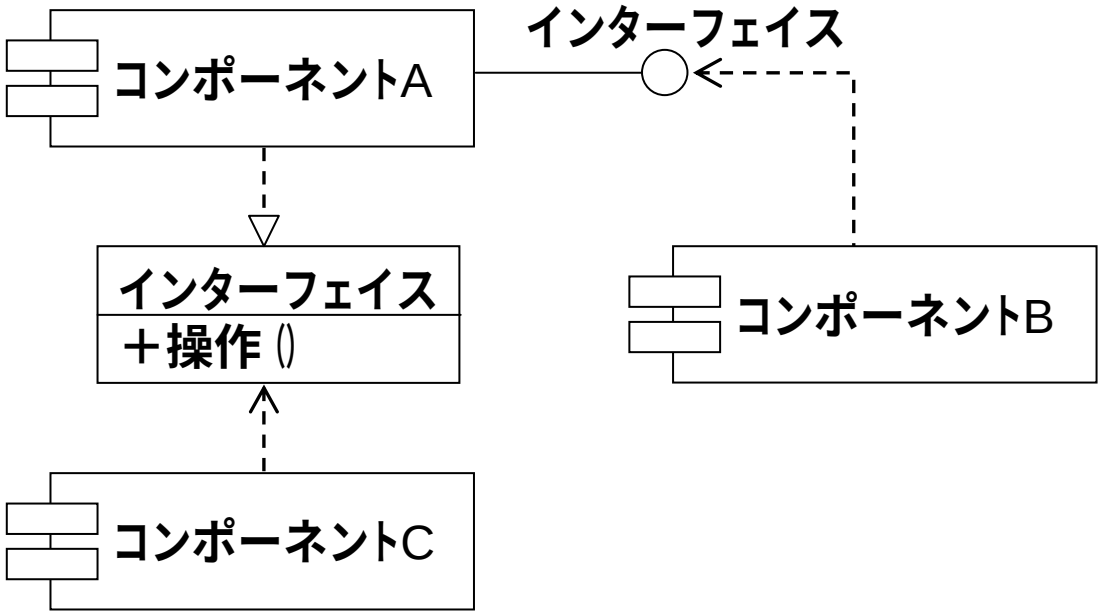


実装

8.6 コンポーネント図 (1)

□コンポーネント図

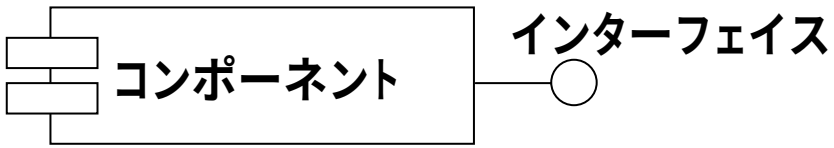
ソフトウェア内部の物理的な構成を表す
ソフトウェアのモジュール依存関係やリソースの割当などを明確化

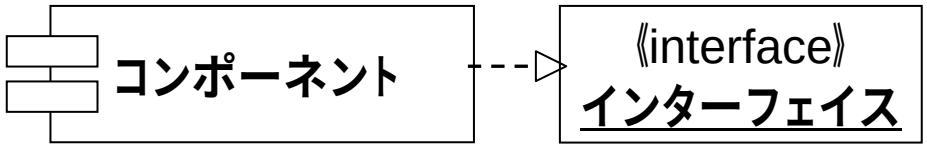


8.6 コンポーネント図 (2)

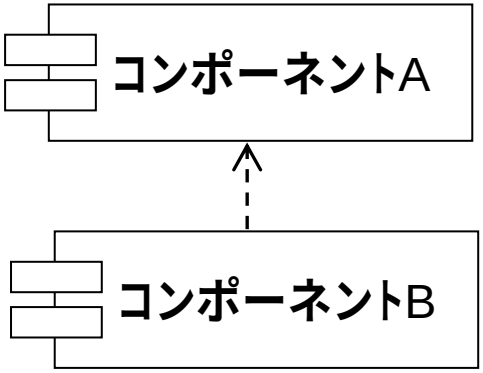
□コンポーネント図の要素

コンポーネント





インターフェース



コンポーネント間の依存関係

8.6 コンポーネント図 (3)

□コンポーネント図を使ったモデリング

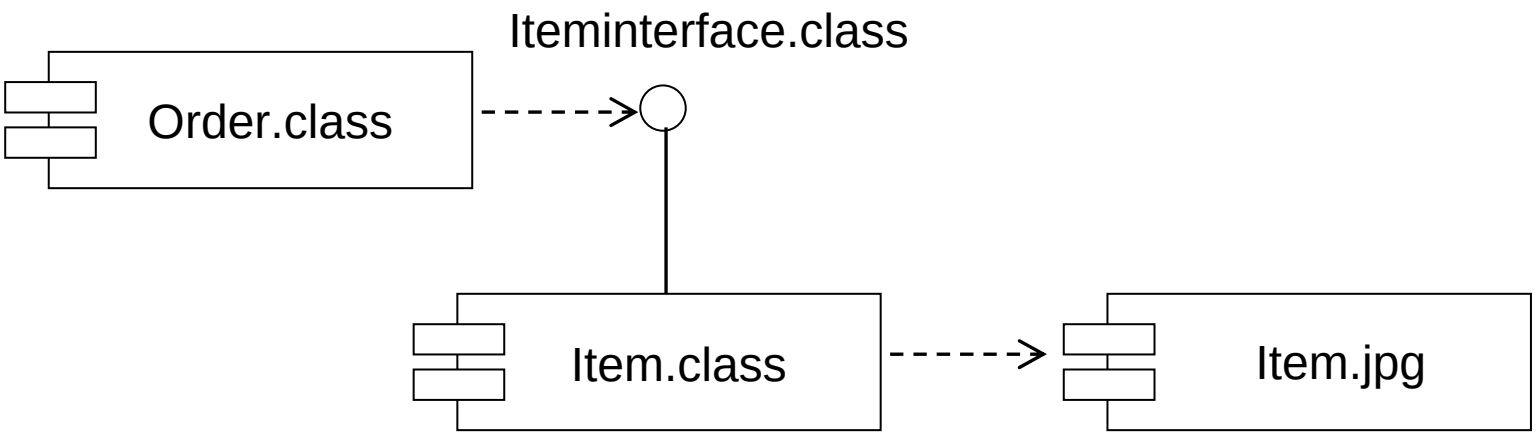
「世界中の酒」の通信販売を行っている兄弟社は、
販売管理システムを導入することになった。
ここでは、販売管理システムのうち受注に関するソフトウェア構成を考える。

□コンポーネントの関係

受注を行う (Order.class) は、商品情報を扱う (Item.class) と
中間システムである (Iteminterface.class) を介してデータのやり取りを行う。
Item.classは商品の写真を扱う (Item.jpg) と依存関係にある。
それぞれの関係を考える。

8.6 コンポーネント図 (4)

□コンポーネント図を使ったモデリング

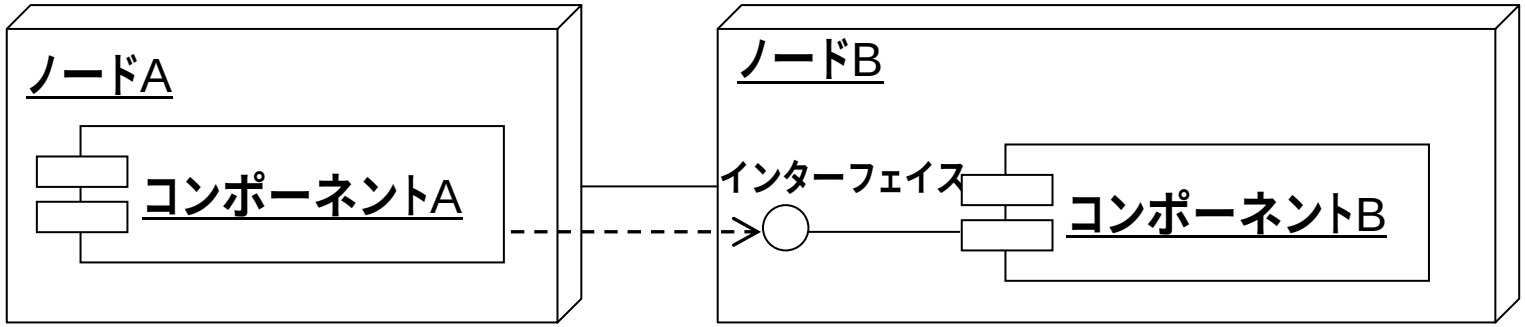


実装

8.7 配置図 (1)

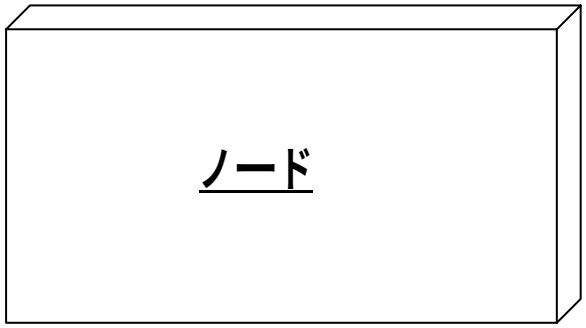
□配置図

システムの物理的な構成を表し、実行環境を考えるために
実装段階で作成する
システムのハードウェア構成やソフトウェア構成などを表現

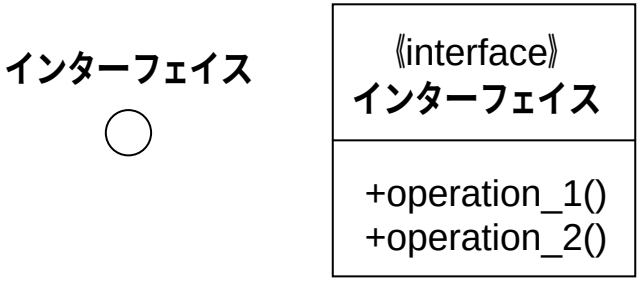


8.7 配置図 (2)

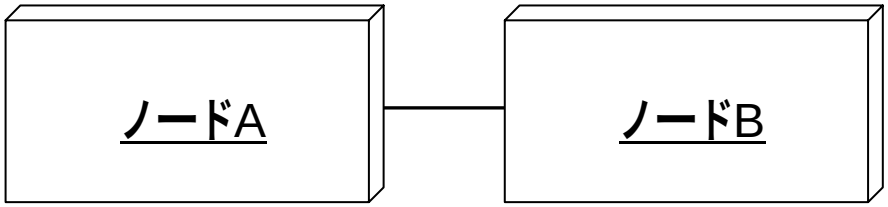
□配置図の要素



ノード



インターフェイス



関連

8.7 配置図 (3)

□配置図を使ったモデリング

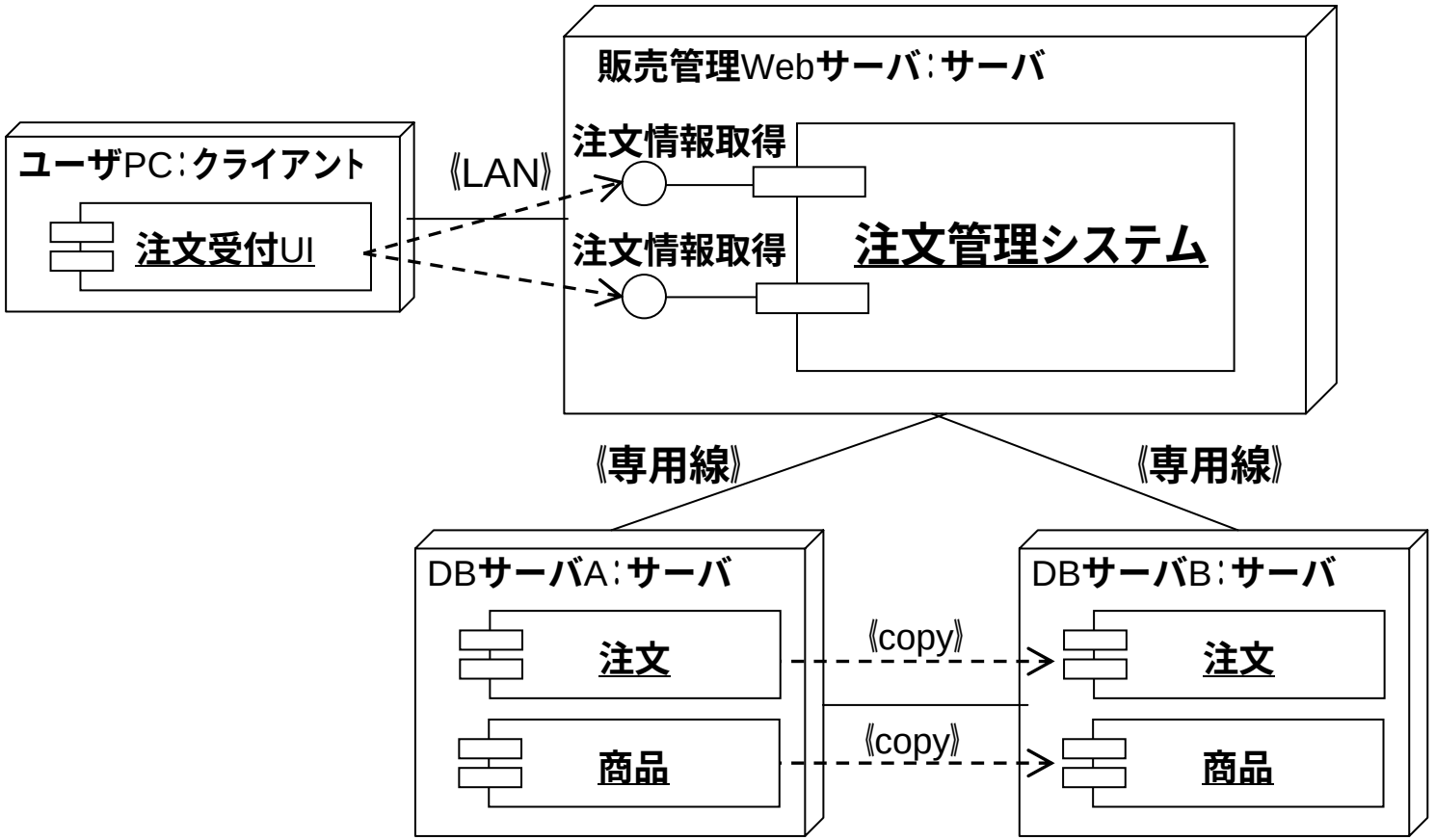
「世界中の酒」の通信販売を行っている兄弟社は、
販売管理システムを導入することになった。
ここでは、販売管理システムのうちシステム環境を考える。

□シナリオ

受注受付UI (ユーザインタフェース) は、ユーザPC (クライアント) にあり、
受注管理システムは、販売管理Webサーバにある。両者はLAN回線で
接続されている。販売管理Webサーバは、DBサーバAおよびDBサーバB
にそれぞれ専用線で接続されている。DBサーバAとDBサーバBは注文と
商品の2つデータをそれぞれミラーリング (コピー) している。

8.7 配置図 (4)

□配置図を使ったモデリング



第8回目　UML2

2004年11月25日(木)

小テスト(氏名：　　)

学籍番号：

(1) ClockTimer2クラスから作成される以下プログラムについて、空欄[①],[②]をうめよ

①

②

```
public class ClockTimer2 extends Subject implements
Runnable,Timer
{
        protected Thread timer;
        protected int time = 0;
        private String timerName = "ClockTimer2";
        public ClockTimer2 () {
            timer = new Thread(this);
            timer.start();
        }
        public int getTime() {
              return time;
        }
        public String getTimerName() {
            return timerName;
        }
        public void run() {
            While(true) {
                try {
                      [①]
                }catch(Exception e) {
                }
                          [②]
                notifyObservers();
                 }
        }
}
```

(2) 講義に関する感想等を述べよ。