第7章アクティビティ図

アクティビティ図は、システムの動きを様々な粒度で表現するダイアグラムである。

ユースケースを補う目的や、クラスの操作をビジュアル的に表現する目的で使用する。

- 必要な前処理の確認
- 並行して作業を進められるアクティビティの発見

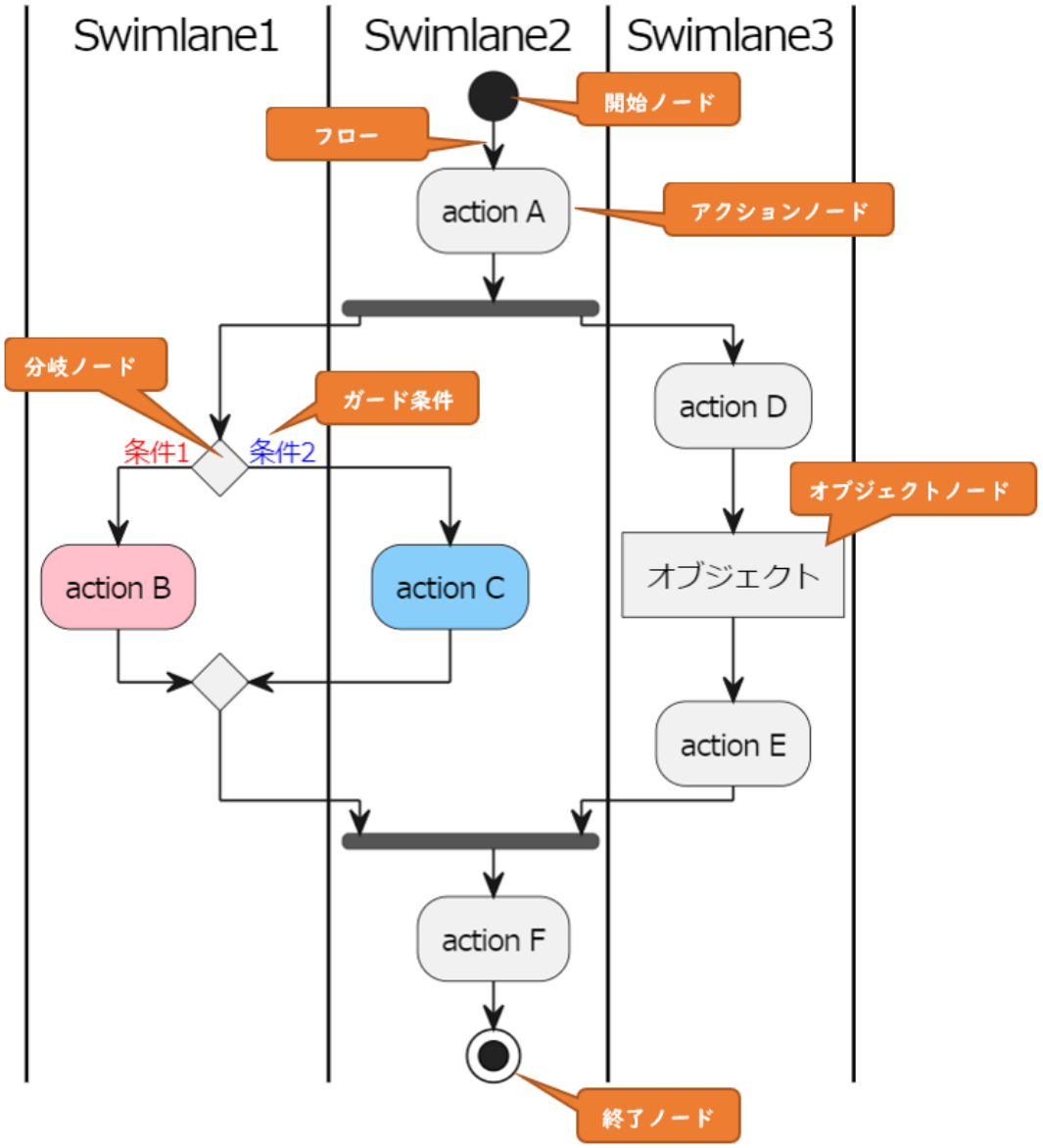


図6ー1　アクティビティ図の例

「開始ノード」から処理が始まって「action A」、「action B」(または「action C」)、「action F」、「終了ノード」へと処理が遷移する流れがコントロールフロー（処理の流れ）となる。

また、「action D」「action E」への遷移は、上記のコントロールフローと並列処理として実行され、

処理に必要なデータとして「オブジェクト」が引き渡されている。この流れはオブジェクトフローと呼ぶ。

【Point】アクティビティ図を用いるケース

- システム化の範囲を決める前に、ドメイン全体のビジネスプロセス（ビジネスフロー）を確認するとき
- ※ドメインとは問題領域や共通領域のこと
- システム化の範囲のユースケースに対する処理を表現するとき
- 並列処理表現を利用して、マルチスレッドのアプリケーションの動作を表すとき

例題）

図7－1　基本的な書き方　（図7-1_基本的な書き方.pu)

```
@startuml アクティビティ図

title 基本的な書き方

start

:アクティビティA;

:アクティビティB
(改行);

:アクティビティC;

stop

@enduml
```

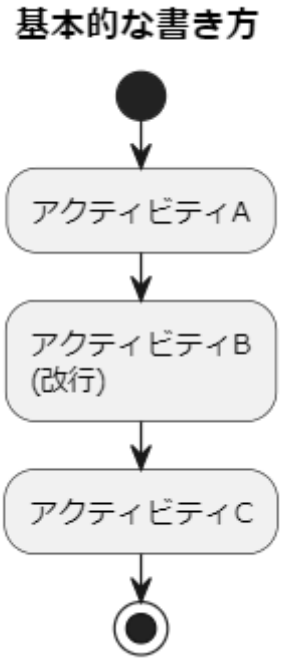
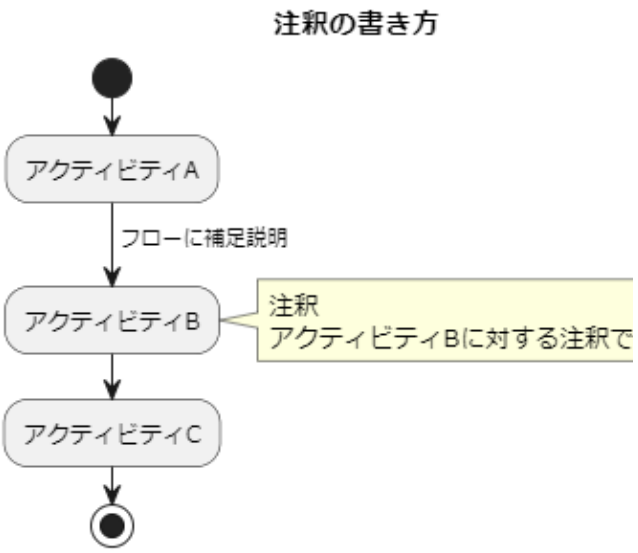


図７－２　注釈の書き方（図7-2_注釈の書き方.pu)

```
@startuml 注釈
title 注釈の書き方

start

:アクティビティA;
-> フローに補足説明;
:アクティビティB;

note right
    注釈
    アクティビティBに対する注釈です。
end note

:アクティビティC;

stop

@enduml
```



▼アクションと分岐

【アクション】

アクションは、アクティビティの最小単位となる。コンピュータや物理的にモデル化されたシステムにおいて、振る舞いの変化（遷移）やプロセスの基本単位と考えることができる。

例）

- 文字の入力やデータの検索
- 商品の発送や在庫の確認

【分岐】

あるアクションから別のアクションに移る際に、判断が必要となり、処理経路が分岐することがある。

分岐には、前の状態から入ってくる1つの遷移と、次の状態へ出ていくガード条件着きの複数の線がある。

出ていく遷移の中で1つだけが実行されるため、それぞれのガードは排他的に設定する必要がある。

図7-3　図7-3_分岐1,pu

```
@startuml 分岐
title 基本的な分岐

:ビデオを借りる;
if () is (<color:red>[会員登録している]) then
#Pink:貸出料金を支払う;
else (<color:blue>[会員登録をしていない])
#skyblue:会員登録をする;

@enduml
```

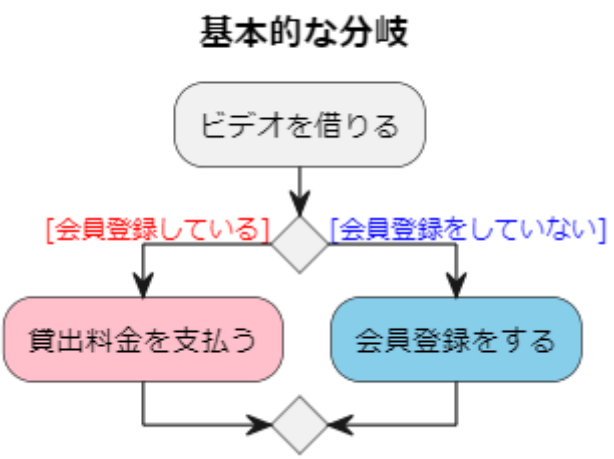


図7-4　図7-4_水平分岐.pu

```
@startuml 分岐
title 水平分岐の例

start

:アクティビティA;

if (x > 0) then (true)
    :アクティビティB;
else if (x == 0) then (true)
    :アクティビティC;
else (false)
    :アクティビティD;
endif

:アクティビティE;

stop

@enduml
```

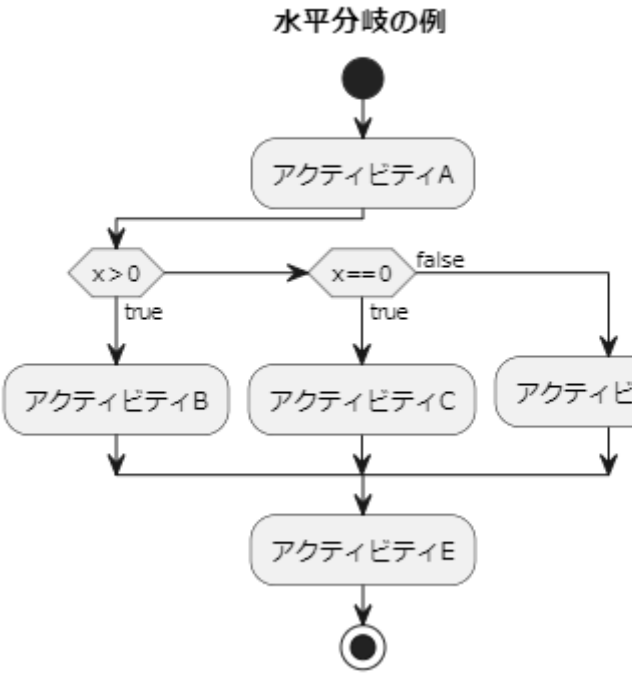
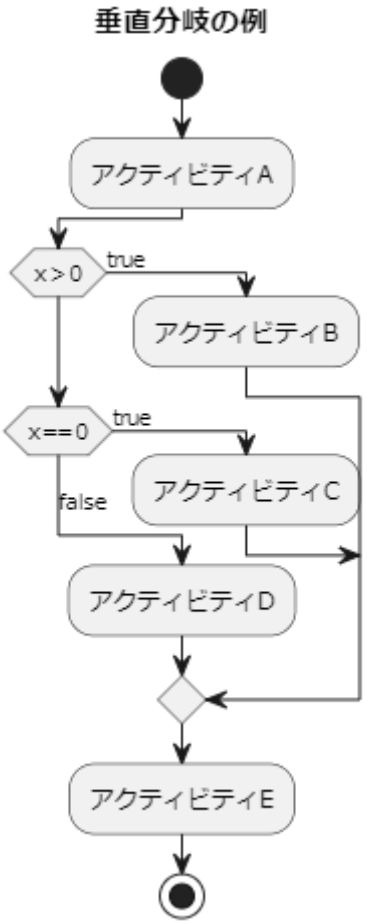


図7-5　図7-5_前判定分岐,pu

```
@startuml 分岐
title 垂直分岐の例
!pragma useVerticalIf on


start

:アクティビティA;

if (x > 0) then (true)
    :アクティビティB;
else if (x == 0) then (true)
    :アクティビティC;
else (false)
    :アクティビティD;
endif

:アクティビティE;

stop

@enduml
```



▽ループ処理

図7-6　図7-6_前判定ループ.pu

```
@startuml ループ処理
title 前判定ループ

start

:アクティビティA;

while (i < 10)
    :アクティビティB;
endwhile

:アクティビティC;

stop

@enduml
```

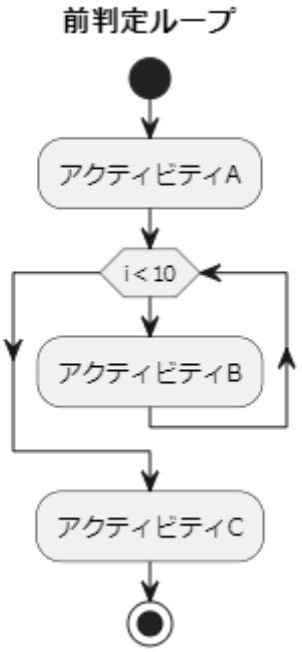
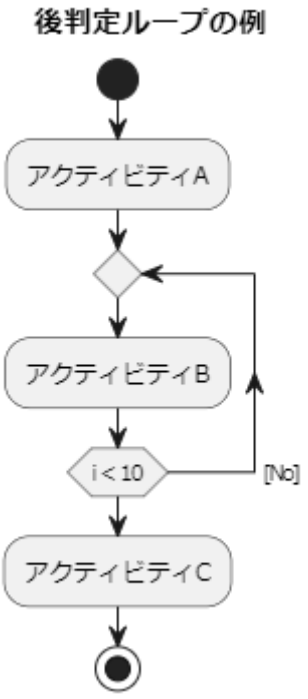


図7-7　図7-7_後判定ループ.pu

```
@startuml ループ処理
title 後判定ループの例

start

:アクティビティA;

repeat
    :アクティビティB;
repeat while (i < 10) -> [No]

:アクティビティC;

stop

@enduml
```



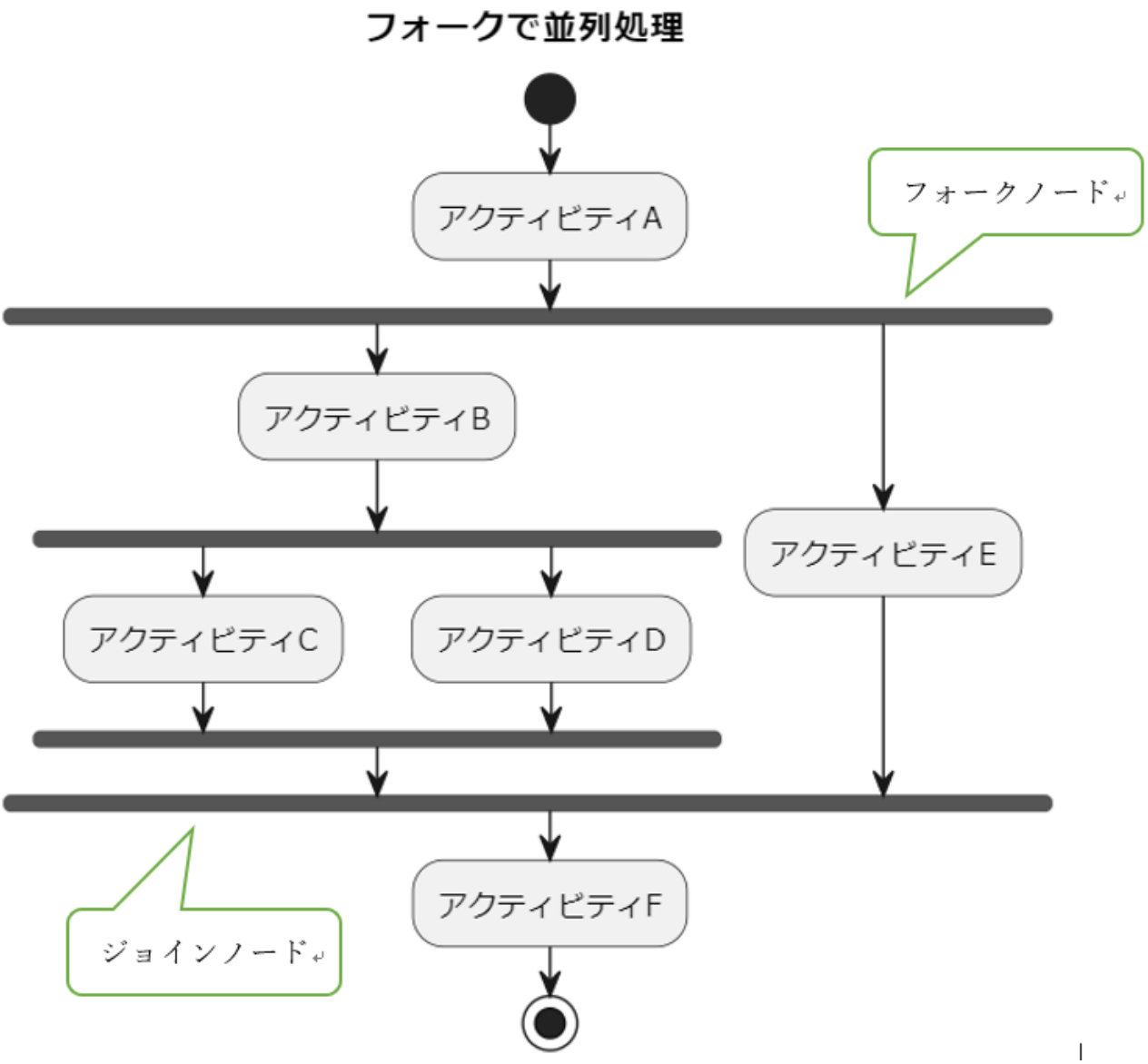
▼スイムレーンとフォークノード、ジョイントノード

並列処理を表現できる点も、アクティビティ図の最大の特徴である。マルチスレッドプログラミングを含んだシステムの動作を表現するには最適のダイアグラムである。

アクティビティ図における並列処理は、「フォークノード」もしくは「スイムレーン」を用いて表記する。（併用もある）

【フォークを利用する場合】

図7-8_フォークで並列処理.pu



```
@startuml 並列処理
title フォークで並列処理

start

:アクティビティA;

fork
    :アクティビティB;
    fork
        :アクティビティC;
    fork again
        :アクティビティD;
    end fork
fork again

    :アクティビティE;

end fork

:アクティビティF;

stop

@enduml
```

フォークノードには、1つの入力遷移と複数の出力遷移がある。入力遷移が発生すると、フォークノードの先で全ての出力遷移が並列に実行される。

フォークノードでの分岐語は、ジョインとノードで閉じるまではそれぞれのスレッドが非同期になる。

入力遷移で分岐したスレッドは、全てのアクションが実行された後に同期を行い、出力遷移が行われる。

これをジョイントノードで表現する。

【注意点】

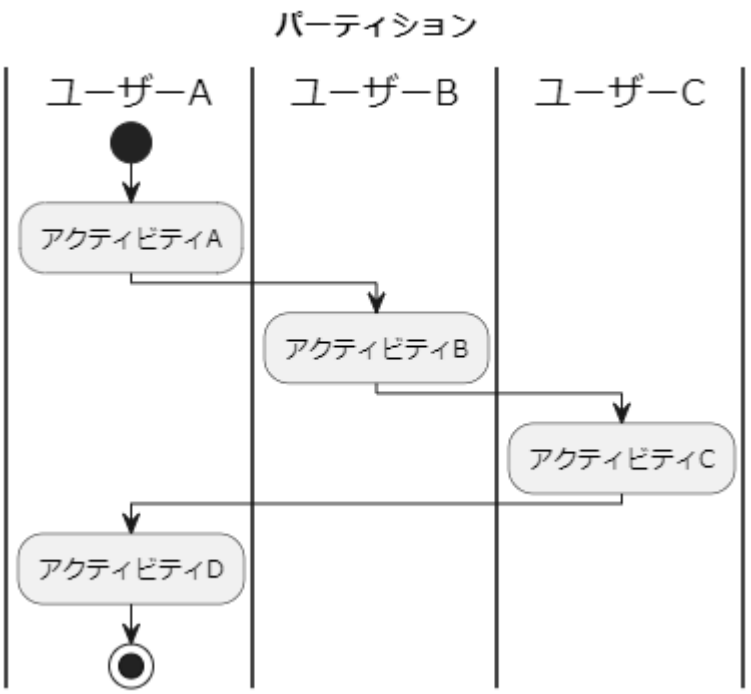
フォークとジョインは、1対1になる必要がある。フォークノードで2つのスレッドが出力遷移すると、ジョインではこの2つのスレッドが入力遷移として必ず集合しなくてはならない。

【スイムレーンを利用する場合】

スイムレーンでは役割（ロール）ごとに線で区切る。

図7-9_スイムレーンで並列処理.pu

```
@startuml
title パーティション

|ユーザーA|

start
:アクティビティA;

|ユーザーB|
:アクティビティB;

|ユーザーC|
:アクティビティC;

|ユーザーA|
:アクティビティD;

stop

@enduml
```



スイムレーンとフォークを組み合わせた例

図7-9_スイムレーンとフォークの併用.pu

```
@startuml アクティビティ図の例

|ユーザ1|
|ユーザ1|
|ユーザ2|
|ユーザ2|
|ユーザ3|
|ユーザ3|

|ユーザ2|
start
  :action A;
fork
|ユーザ2|

 |ユーザ1|
  :action B;
  :action C;
 |ユーザ1|

|ユーザ2|
  :action E;
fork again
 :action D;
|ユーザ2|

|ユーザ3|
 :action F;
 |ユーザ3|

|ユーザ2|
end fork
 :action G;
stop
|ユーザ2|

@enduml
```

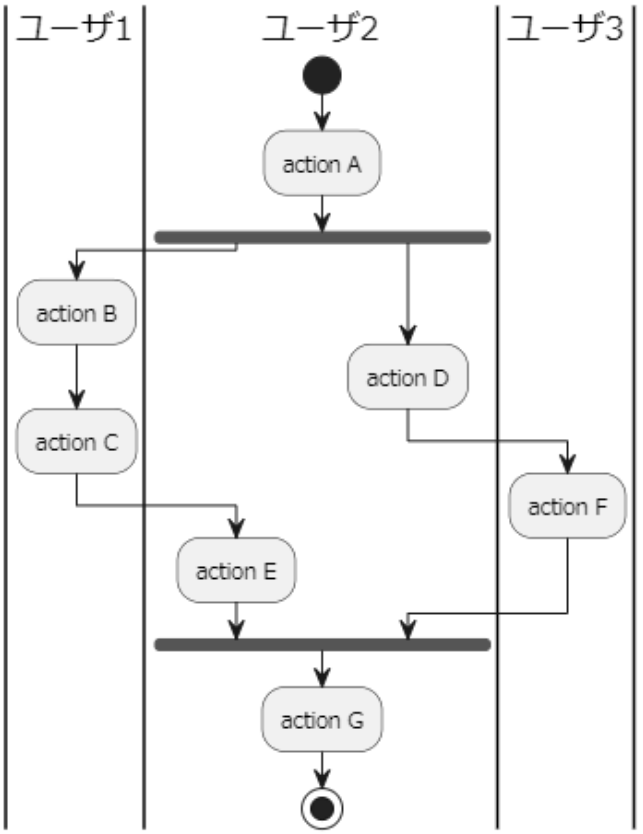
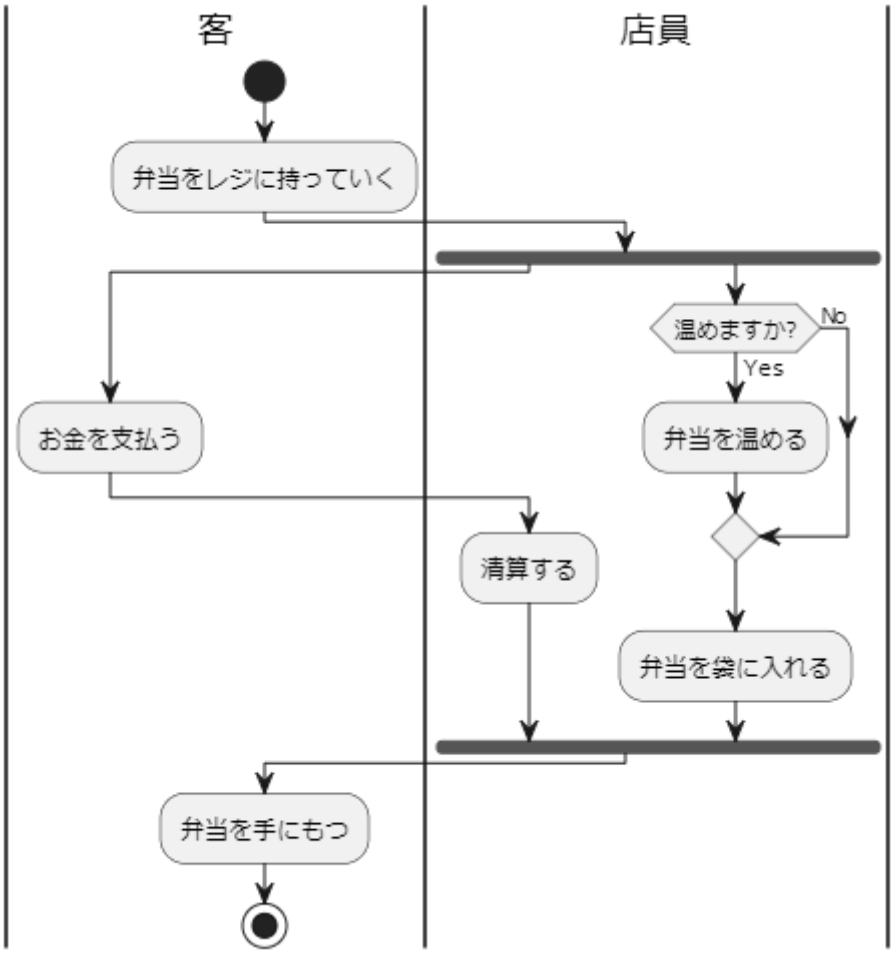


図7-10

コンビニで弁当を買う際のアクティビティ図を作成しなさい。

【提出ファイル】 図**7-10_AC_**コンビニで弁当買う**.pu**



演習7-1

次の文章は、ユーザがホテル宿泊予約システムで予約を行う際のフローを表したものである。

このフローをもとに、アクティビティ図を作成しなさい。

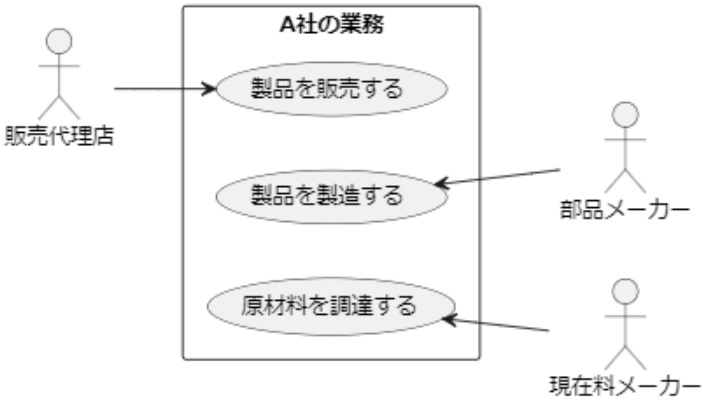
1.
2. ユーザはユーザ名を入力する
3. ユーザはパスワードを入力する
4. 認証が成功したら予約をおこなう。失敗したら、ユーザとパスワードを再入力する

【提出ファイル】 演習**7-1_AC_**ホテル宿泊予約システムで予約**.pu**

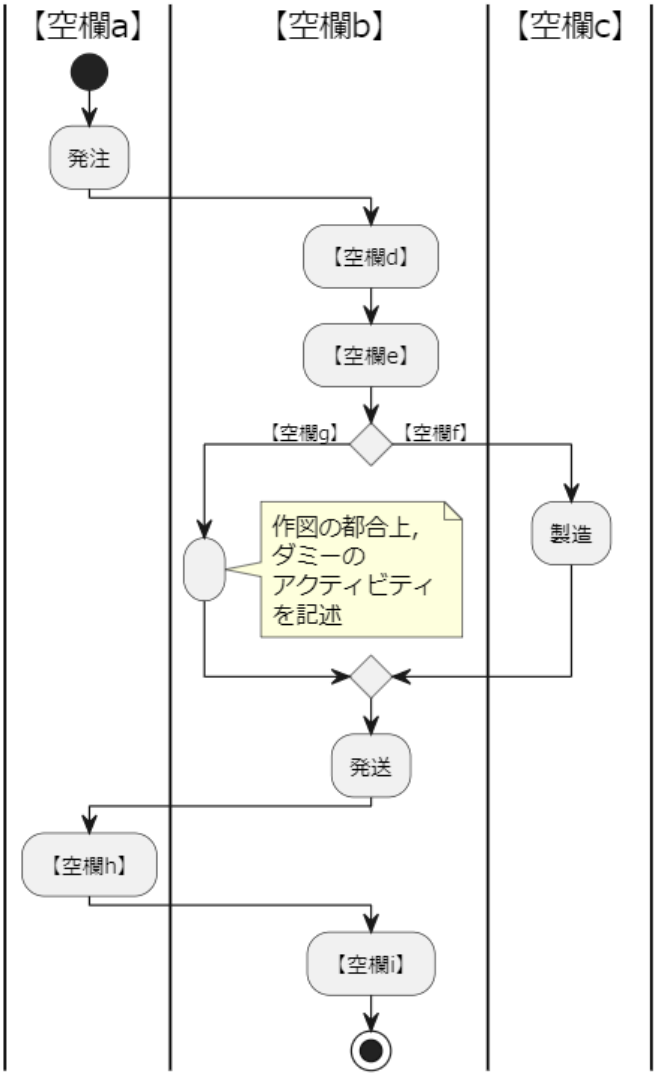
演習7-2

次の記述を読んで、A社の業務全体を表現したユースケース図を以下に示す。

（A社の業務全体）
　A社では、製品の製造を行っています。原材料メーカーから原材料を調達し、製品の製造を行い、販売代理店に対して、製品の販売を行います。
なお、製品の製造に関しては、一部部品メーカーの作成する部品を使用することもある。



「製品を販売する」ユースケースの詳細を下記のアクティビティ図で表現した。



下記の選択肢から空欄a～kに当てはまるものを選び、アクティビティ図を完成させよ。

【提出ファイル】演習**7-2_AC_**製品を製造するアクティビティ図**.pu**

＜選択肢＞

在庫あり、在庫なし、入金確認、営業部門、在庫調査、製造部門、検収/支払い、販売代理店、受注

演習7-3

あなたがレストランで食事をするとして、店に入ってから出る前までの手順を、アクティビティ図で示せ。

【提出ファイル】演習**7-3_AC_**レストランのアクティビティ図**.pu**

●スイムレーン

顧客、給仕、料理人

●アクションノード

案内を依頼する

テーブルに案内する

メニューを渡す

注文を決める

注文を受ける

伝票を書く

料理人に注文を伝える

注文を受ける

料理を作る

料理を運ぶ

伝票を渡す

料理を食べる

代金を払う

●オブジェクト

:注文　:料理　:メニュー　:伝票

※同じオブジェクトが複数回登場する場合があります。

●その他

フォークノードとジョイントノードをそれぞれ1つ作成してください。

演習7-4-1

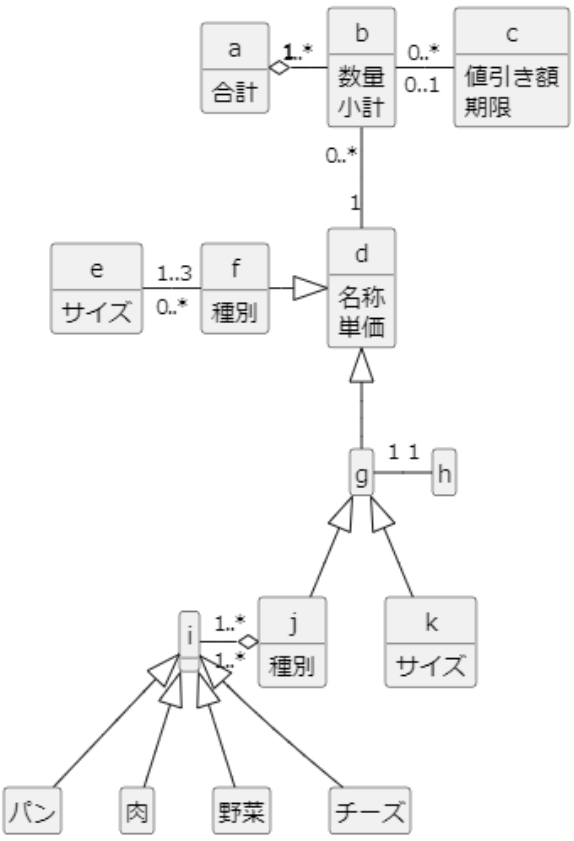
あるハンバーガーショップでは、ドリンク類（カップには1～3種類のサイズがある）、フード類（ハンバーガー、ポテトなど）の販売を行っている。今回システム化を検討するにあたり、業務内容の調査を行い、クラス図とアクティビティ図を作成した。

下記の選択肢から空欄a～kのクラス名に当てはまるものを選び、クラス図を完成させよ。

【提出ファイル】演習**7-4-1_CL_**ハンバーガーショップ**.pu**

＜選択肢＞

- ドリンク
- フード
- 商品
- 注文明細
- 注文
- カップ
- 材料
- ハンバーガー
- 割引クーポン
- レシピ
- ポテト



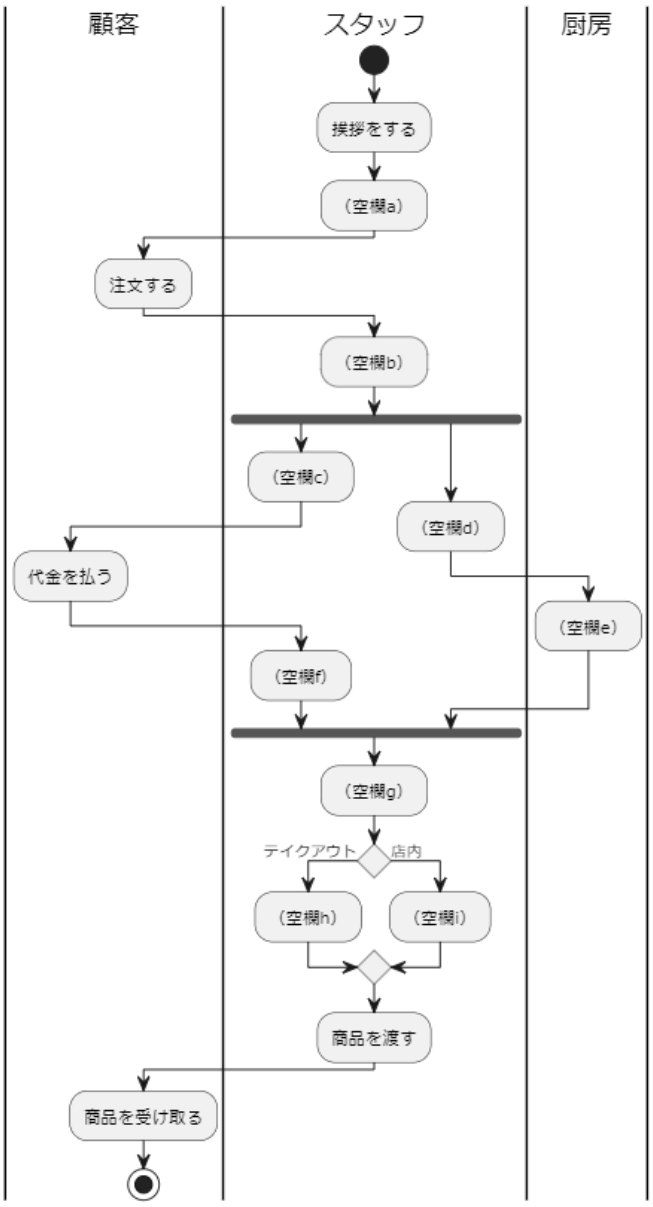
＜作成するクラス図＞

演習7-4-2

あるハンバーガーショップでは、ドリンク類（カップには1～3種類のサイズがある）、フード類（ハンバーガー、ポテトなど）の販売を行っている。今回システム化を検討するにあたり、業務内容の調査を行い、クラス図とアクティビティ図を作成した。

下記の選択肢から空欄a～iのアクションノードに当てはまるものを選び、アクティビティ図を完成させよ。

【提出ファイル】演習**7-4-2_AC_**ハンバーガーショップ**.pu**



＜作成するアクティビティ図＞

＜選択肢＞

- 商品を料理する
- 代金を請求する
- 袋に入れる
- 注文を聞く
- 注文を確認する
- テイクアウトかどうか尋ねる
- トレイにのせる
- 精算する
- 商品の調理を依頼する