

UML で表現する

UML とは

UML (Unified Modeling Language) は、仕様書などに記述するモデルを表現するための統一モデリング言語で「**図式化されたプログラミングの設計図**」ともいえるものです。

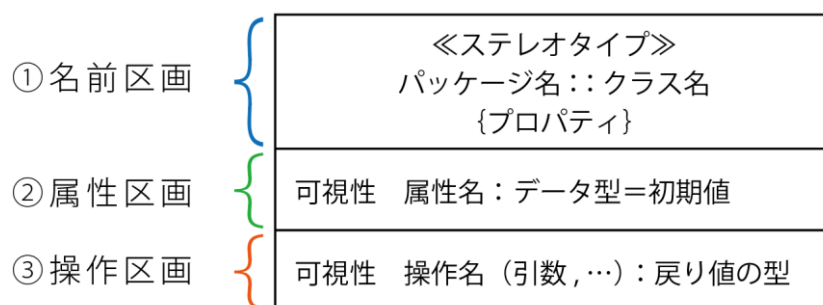
UML には構図図として「**クラス図**」「**オブジェクト図**」、振る舞い図として「**シーケンス図**」「**コミュニケーション図**」などがあります。

クラス図

- 「**クラス同士の関連性**」を示すことができます。

関連	
集約	
コンポジション	
汎化	
実現	
依存	

- 1つのクラスの図は「**名前区画**」「**属性区画**」「**操作区画**」に分かれていて、名前区画には「**クラス名**」を、属性区画には「**クラス変数（フィールド）**」を、操作区画には「**コンストラクタやメソッド**」を表記します。



① 名前区画

中央揃えで記述します。名前区画にはステレオタイプ (interface など)、パッケージ名とクラス名 (太字)、プロパティを記述します。また、抽象クラスは名前を斜体で表すか、プロパティに「{abstract}」を記述することによって表現します。

② 属性区画

可視性、属性名、データ型、初期値などを記述します。

③ 操作区画

可視性、操作名、引数、戻り値の型を記述します。

可視性

可視性とは、属性や操作を他のクラスから利用できるかどうかを示すものです。

public	どのクラスからも利用できます。「+」で表します。
package	同じパッケージ内のクラスから利用できます。「~」で表します。
protected	サブクラスから利用できます。「#」で表します。
private	そのクラス自体からのみ利用できます。「-」で表します。

【「本」クラスのクラス図】

分析クラス図

本
タイトル 内容
引数のないコンストラクタ 内容を表示する タイトルを表示する

名前区画 → クラス名

属性区画 → クラス変数

属性区画 → メソッド

設計クラス図

Book
-title : String -contents : String
+Book() +showContents() +showTitle()

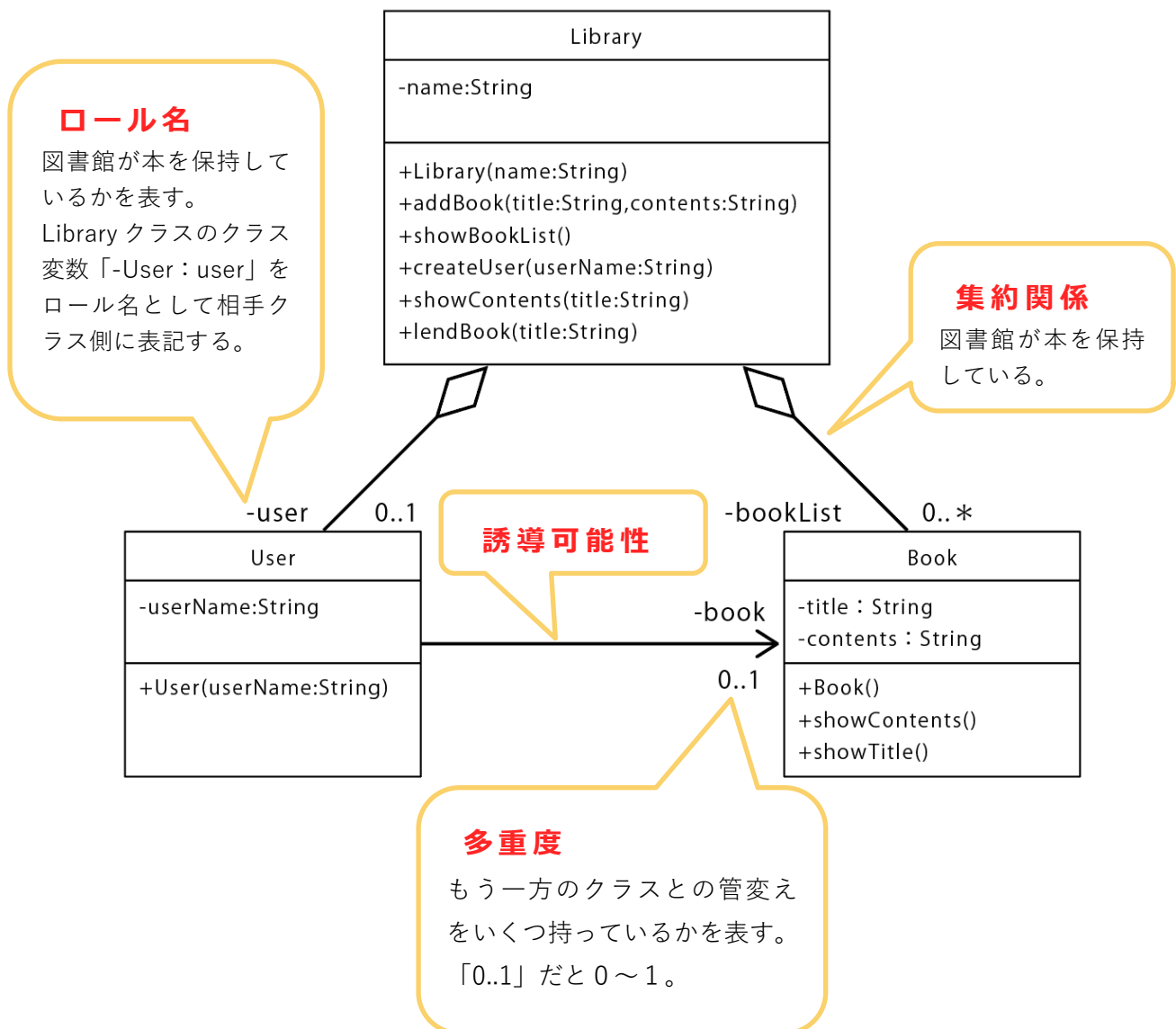
【「本」クラスのコード（具体的な処理は省略）】

```
class△Book {  
    private△String△title ;  
    private△tring△contents ;  
    public△Book() {}           //デフォルトコンストラクタ  
    public△void△showContents() { //内容表示メソッド  
        :    //省略  
    }  
    public△void△showTitle() {  //タイトル表示メソッド  
        :    //省略  
    }  
}
```

関連

- クラス間の関係性は「**関連**」という線で表現します。
- 一方のクラスがもう一方のクラスに対していくつの関係性を持つかを「**多重度**」で表します。多重度は、相手のクラスのオブジェクトに対して、そのクラスのオブジェクトがいくつ存在できるかを示します。また、1対1の関係では省略可能です。
- クラス間の知り合い関係の方向性を明確に表現するために、「**誘導可能性**」として関連に矢印を付けることができます。
- クラス同士が関連したことにより、クラスが演じる役割を「**ロール名**」として表記します。関連する相手クラスの属性名として使用し、可視性も記述します。

【例】図書館（Library）クラスと利用者（User）クラス、本（Book）クラスのクラス図。



クラス間の可視性

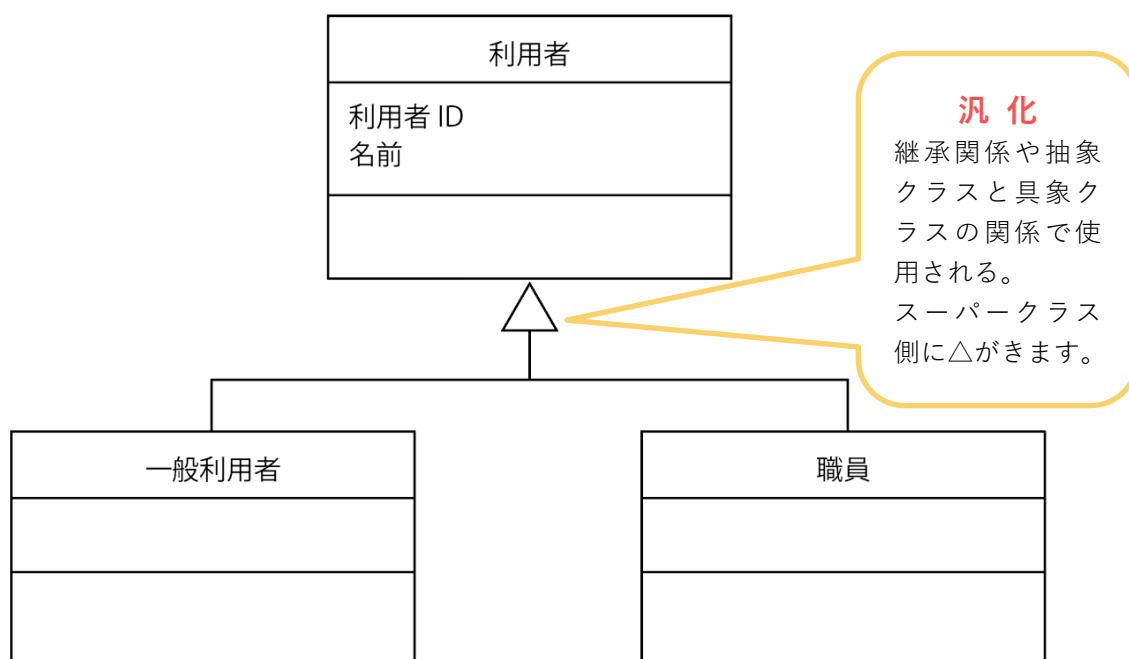
関係とは、知り合い関係です。相手を知っている、つまり、相手が見えることを「クラスが可視である」と表現します。相手を見るためには、そのオブジェクトの参照が必要です。

オブジェクトの参照を得る手段としての可視性には、主に以下の3つがあります

属性可視性	あるオブジェクトの参照を自分の属性として持つことによって、長期的な可視性が生まれます。（クラス変数として持っている＝ロール名）
パラメータ可視性	操作の引数（パラメータ）としてあるオブジェクトの参照が渡ってくると、一時的に可視性が生まれます。 この可視性は、操作の処理が終わるとなくなります。（メソッドの引数で他クラス型の変数を持っている）
ローカル可視性	操作の処理の中でオブジェクトの参照を宣言すると、可視性が生まれます。 この可視性も、操作の処理が終わるとなくなります。 （メソッド内で他クラス型の変数を利用している）

汎化

共通部分を表すクラスと詳細部分を表すクラス間における分類の関係です。汎化は、スーパークラス側に三角形をつけた実線で表します。



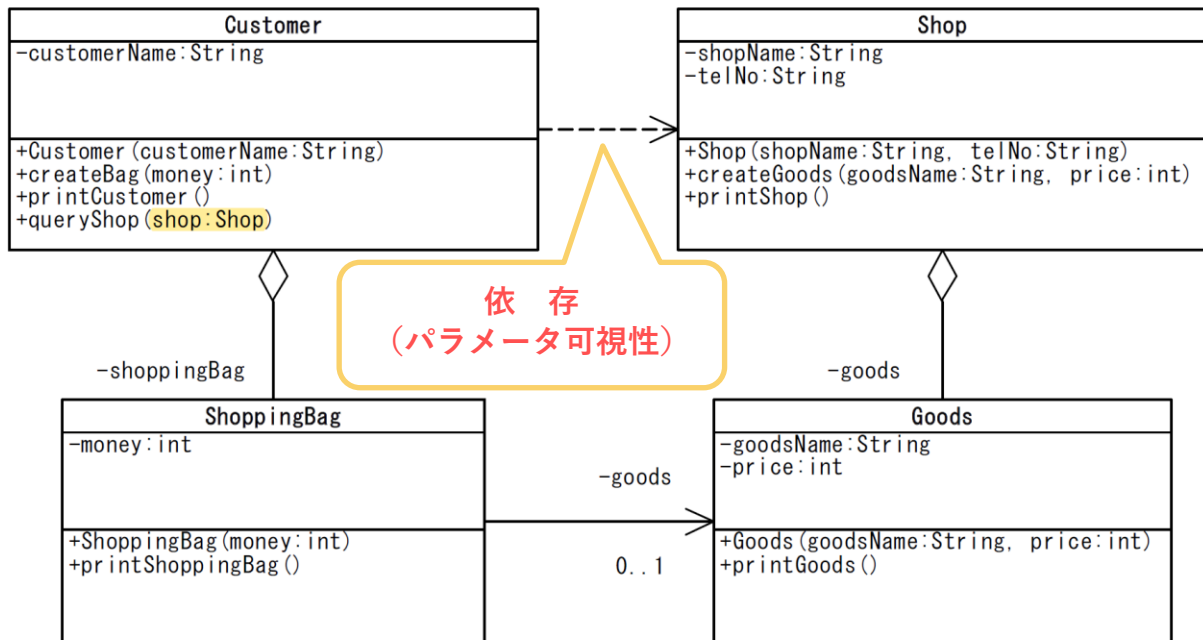
実現

実現は、シグニチャ（操作名）のみで定義された実装のない抽象操作（抽象メソッド）を持つインタフェースとそれを実装するクラスとの関係です。-----▷を表します。

依存

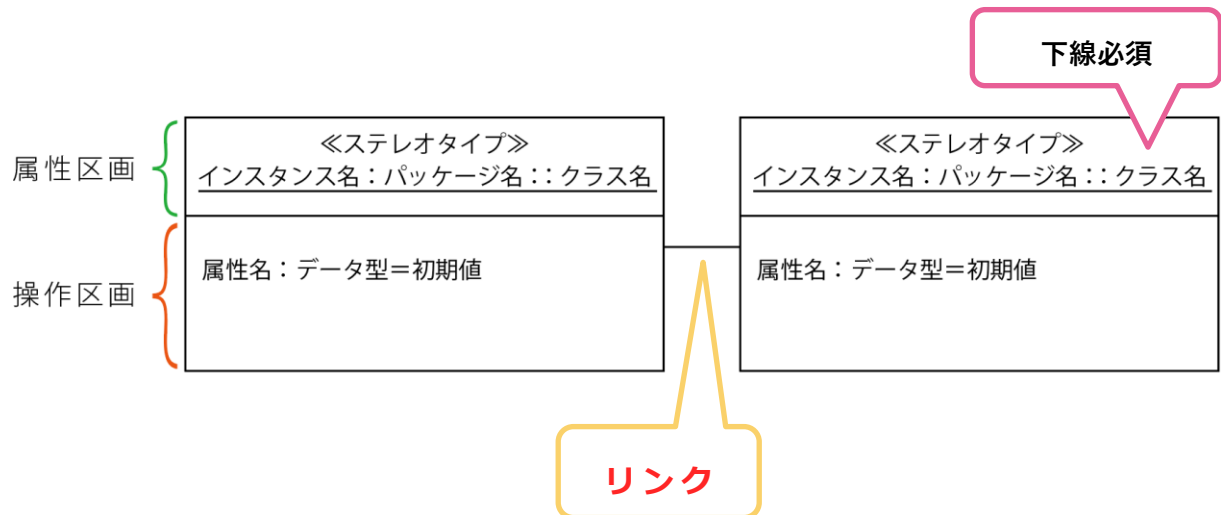
依存は一時的な関係を表すときに使用します。（FLM での）

一時的な関係とは、操作（メソッド）の引数で相手の参照を知り、メッセージパッシングを行いますが、その後は相手の存在を忘れてしまう関係です。または、関連までは必要ないが、一時的に相手を使用した関係です。依存は、破線の矢印で表現します。

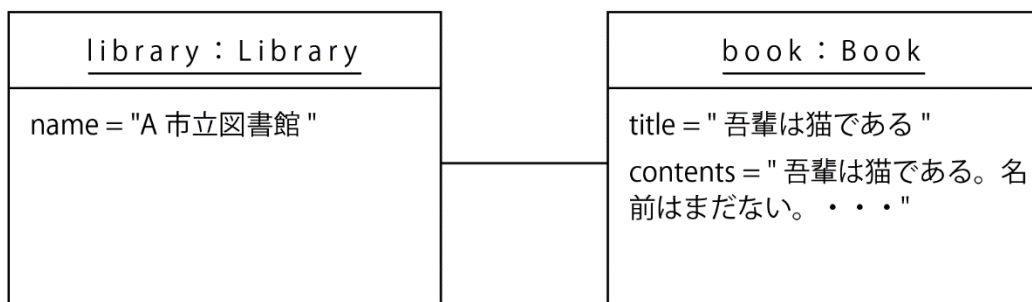


オブジェクト図

- クラスの構造に対応する**オブジェクトの静的な構造を表す**ダイアグラムです。オブジェクト図は、システムのある瞬間のオブジェクトの様子（**プログラム終了時の状態**）を表記します。
- オブジェクト仕様は、オブジェクト名を記述する上側の区画と、属性名、データ型、初期値を記述する下側の操作区画に分割した四角形で表します。



【例】図書館（Library）オブジェクトと本（book）オブジェクトのオブジェクト図。



シーケンス図

シーケンス図は、「**オブジェクト間の相互作用を時系列（縦軸は時間の流れ）で表す**」ことができるダイアグラムです。

ライフライン

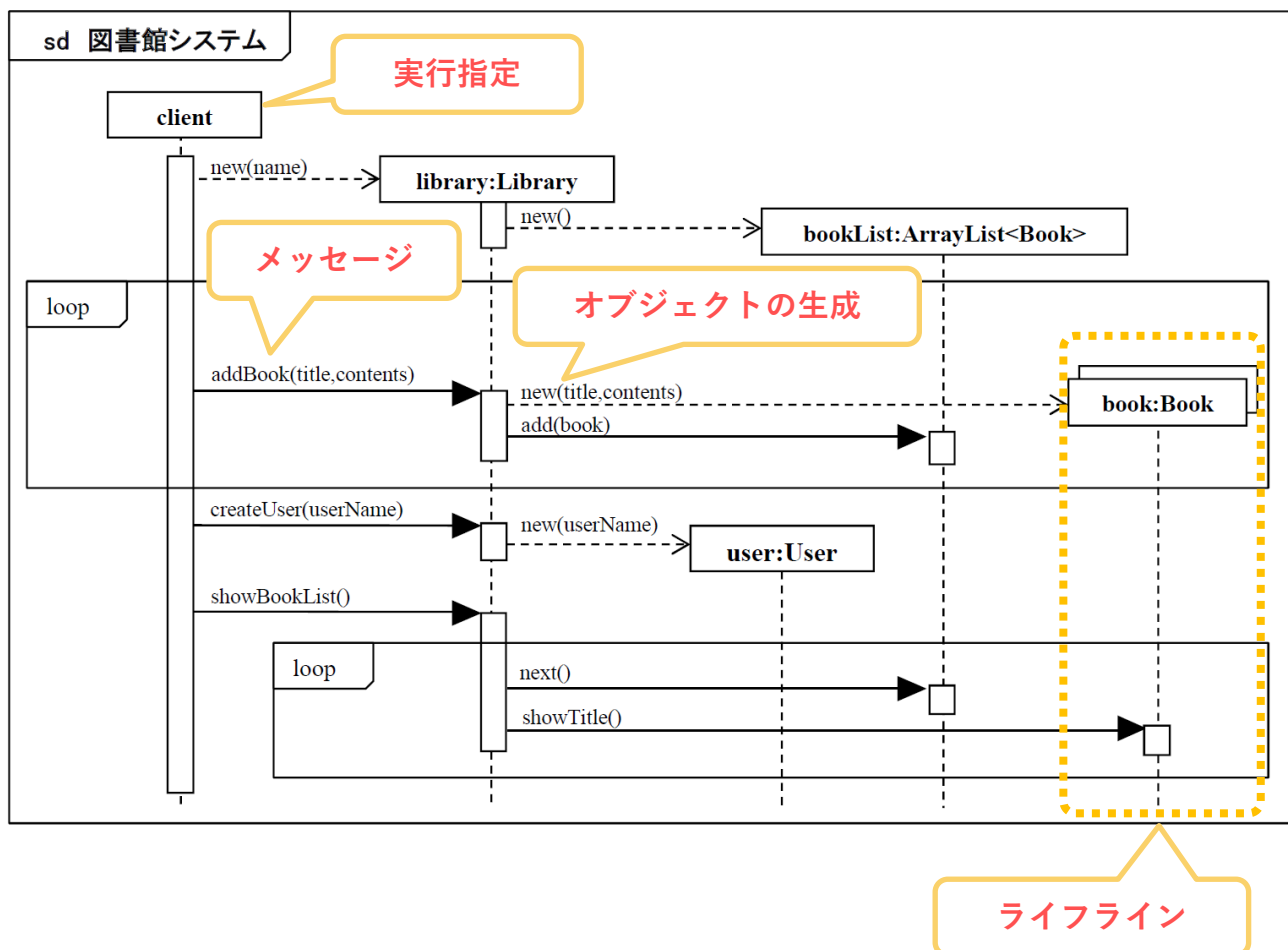
オブジェクトの存在を示します。ライフラインはオブジェクトと、そこから下に伸ばした縦の点線で構成されます。

実行指定

オブジェクトが処理を行っている期間を示します。ライフライン上の細い長方形で表します。

メッセージ

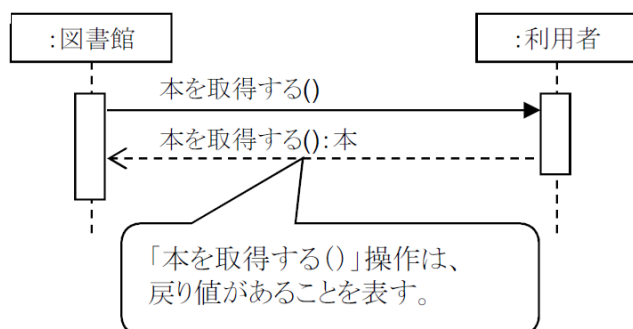
操作を示します。メッセージには種類があり、操作名をつけた矢印で表します。



処理からの戻り値

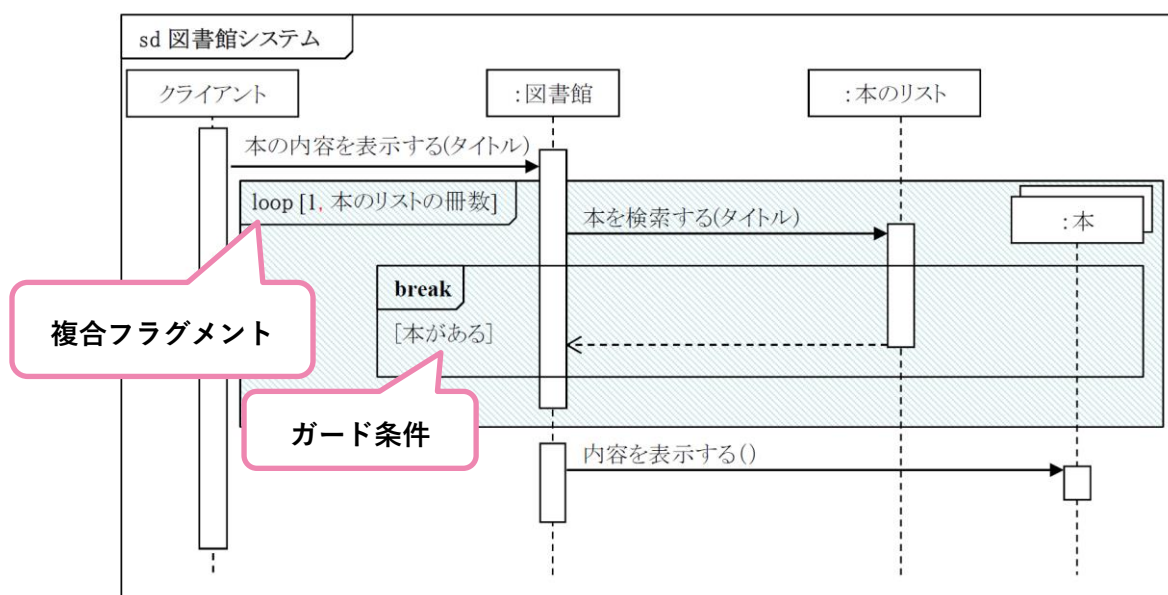
呼び出し元の実行指定に向けて点線で表現します。

【例】



複合フラグメント

制御構造を表現するために、複合フラグメントを使用することができます。複合フラグメントはフレームで表します。



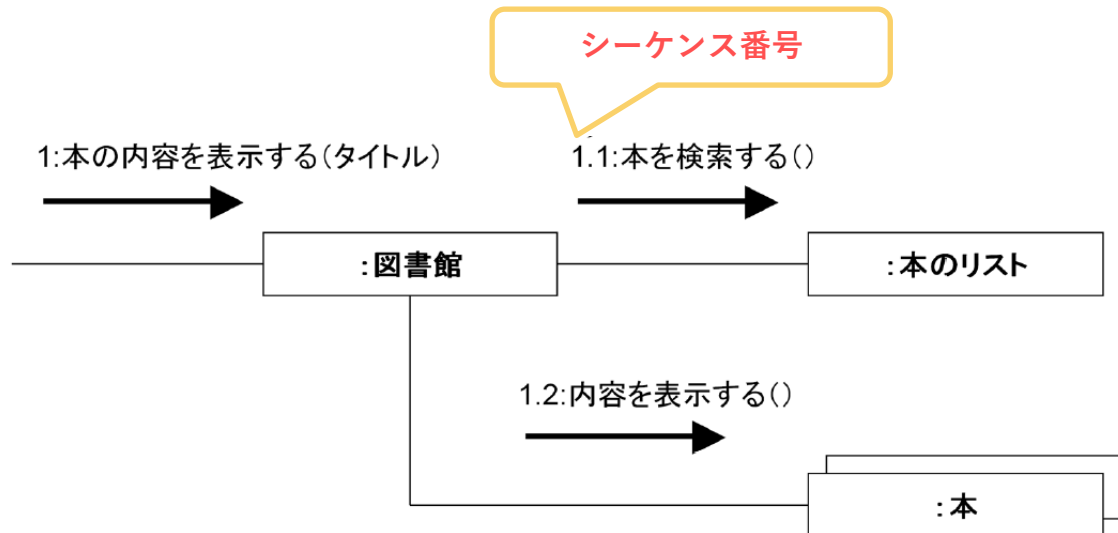
【主な複合フラグメント】

記 述 例	意 味
ref	別のシーケンス図を参照することを表します。
alt	分岐処理を表します。
opt	条件を満たした場合のみ実行される処理を表します。
loop	繰り返し処理を表します。繰り返し回数を指定する場合は「loop [開始, 終了]」と記述します。繰り返し回数は省略できます。
break	処理の中断を表します。

コミュニケーション図

オブジェクトのリンクにメッセージを記述したものです。

メッセージのラベルにはメッセージの順序を示すために、シーケンス番号を付加します。コミュニケーション図は、オブジェクト間の協調関係を明らかにします。



演習問題 Try08-01

「演習問題」フォルダ内にある「Try08」内に、以下のクラス図を元にしたひな型となるクラスを新規作成してみましょう。

ただし、現段階でコンパイルはできますが実行はできません。

【分析クラス図】

絵の具
色名 価格
引数のないコンストラクタ 引数2つのコンストラクタ 色名の表示 絵の具情報の表示

【設計クラス図】

Paint
-color : String -price : int
+Paint() +Paint(color:String,price:int) +showColor() : void +showProduct() : void

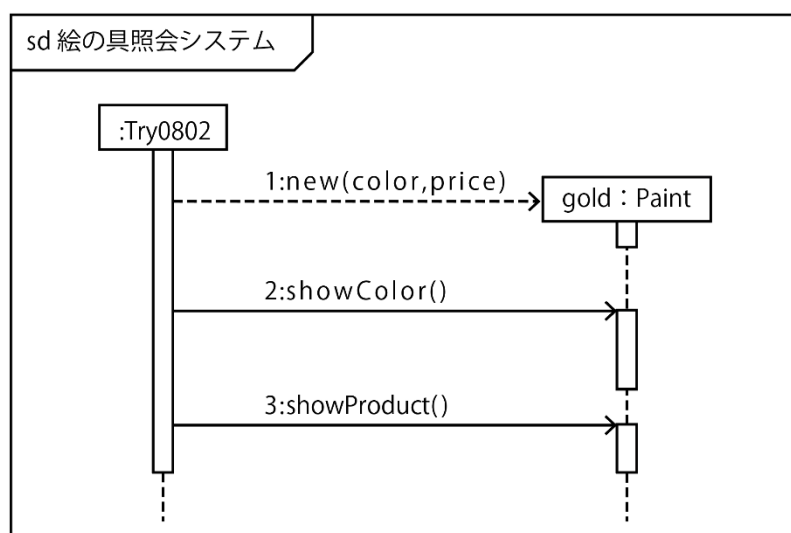
演習問題 Try08-02

「演習問題」フォルダ内にある「Try08」内に、以下のシーケンス図と実行結果を元に「Try08-01」で作成したクラスを編集しましょう。

また商品情報を表示する際、色名の表示はそのメソッドを呼び出すことで行ってください。

ただし、現段階でコンパイルはできますが実行はできません。

【シーケンス図】



【実行イメージ】

```
コマンド プロンプト
C:\¥tanoshi_2020¥03_JavaKiso¥2weeks¥演習問題¥Try08>java Try0802

【色名】   金色

【色名】   金色
【価格】   1200円
```

The screenshot shows a Windows Command Prompt window with the title "コマンド プロンプト". The command prompt shows the directory path "C:\¥tanoshi_2020¥03_JavaKiso¥2weeks¥演習問題¥Try08" and the command "java Try0802". The output of the program is displayed on a yellow background, showing two lines of formatted text: "【色名】 金色" and "【色名】 金色", followed by "【価格】 1200円".

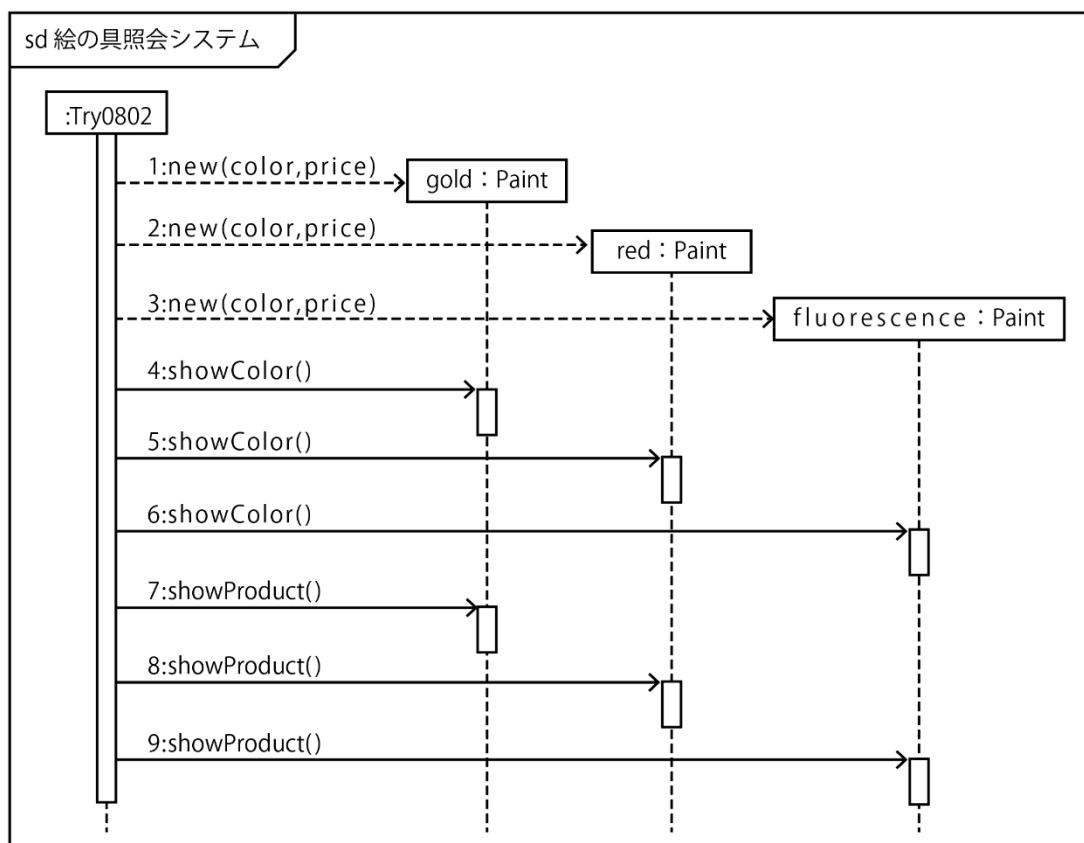
演習問題 Try08-03

「演習問題」フォルダ内にある「Try08」内に、以下のオブジェクト図・シーケンス図・実行結果を元に「Try08-02」で作成したクラスを利用して実行するためのクラス「Try0803.java」を新規作成しましょう。また商品情報を表示する際、色名の表示はそのメソッドを呼び出すことで行ってください。ただし、現段階でコンパイルはできますが実行はできません。

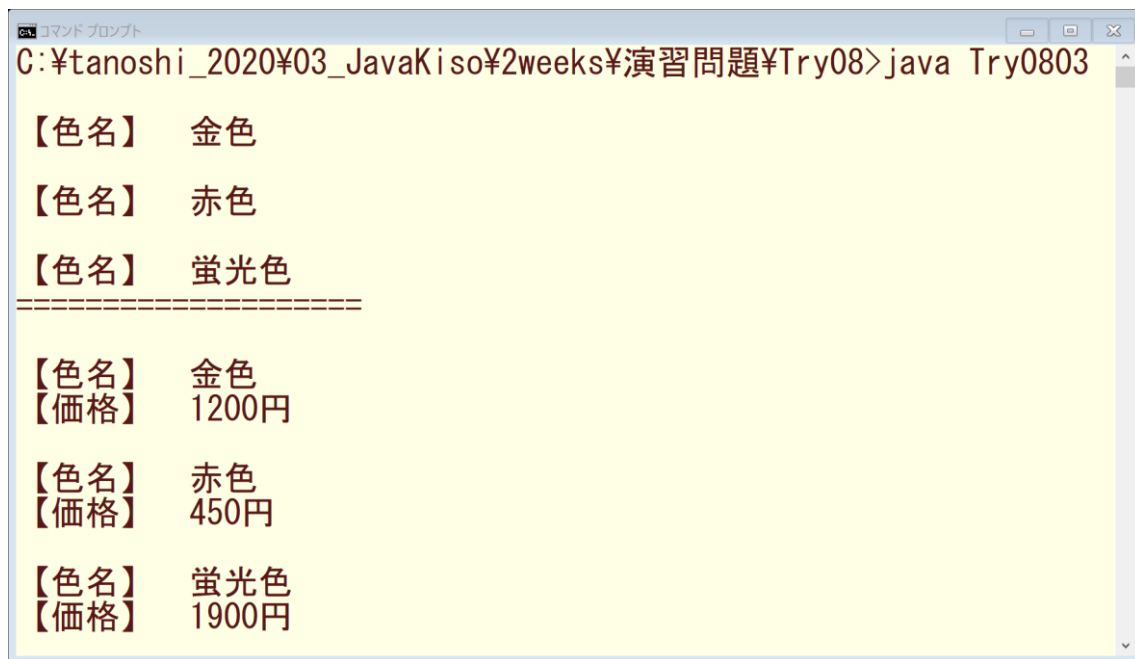
【オブジェクト図】

<u>gold : Paint</u>	<u>red : Paint</u>	<u>fluorescence : Paint</u>
color= " 金色 "	color= " 赤色 "	color= " 蛍光色 "
price= 1200	price= 450	price= 1900

【シーケンス図】



【実行イメージ】



```
コマンド プロンプト
C:\¥tanoshi_2020¥03_JavaKiso¥2weeks¥演習問題¥Try08>java Try0803

【色名】   金色
【色名】   赤色
【色名】   蛍光色
=====

【色名】   金色
【価格】   1200円
【色名】   赤色
【価格】   450円
【色名】   蛍光色
【価格】   1900円
```

「=====」は Try0803.java で表示させてください。