プログラム設計

http://bit.ly/design4d

クラス図（クラス間の関係）

後期 第2週

2019/10/7

クラス間の関係

クラス図におけるクラス間の関係について、以下の4つの関係を学びます。

- **集約**
- **継承（汎化）**
- **依存**
- **関連**

集約

2つのクラスが全体と部分の関係を表す。（例えば、商品リストと商品など）

【例】クラスAがクラスBを集約している

astahの補足

関係の作成について

1. ツールバーのボタンから関係の選択し、2つのクラスをつなぐ

2. 作成した関係をマウスで選択すると、画面左下のプロパティに関係の情報を確認できる
   （「関連端A」「関連端B」のタブに、接続されているクラスの情報などが表示される）

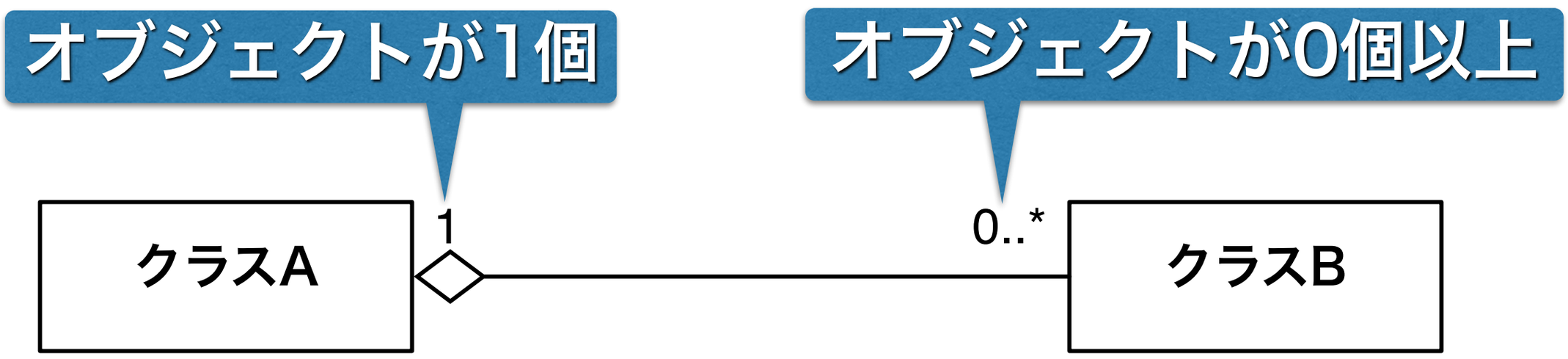
関係の削除について

削除する際は、右クリックのメニューで「モデルから削除」を実行する。
Deleteキーで削除してしまうと、見た目は消えるが、クラスには「関係でつながっている」ことが残る。その場合は、そのクラスを選択して、画面左下のプロパティから「関連」タブを開いて、関係しているクラスを削除する。

多重度

「クラスのオブジェクト（つまりインスタンス）が接続する可能性のある数」を多重度と呼び、関係の線にその情報を描く。

【例】クラスAのオブジェクト1個に対して、クラスBのオブジェクトが0個以上集約する可能性がある場合



多重度の表記

「..」は「範囲」を表す

「*」は「いくつでも」を表す

表記	意味
0..1	0または1
1	1
0..* または *	0以上
1..*	1以上
3..10	3から10

astahの補足

多重度の入力について

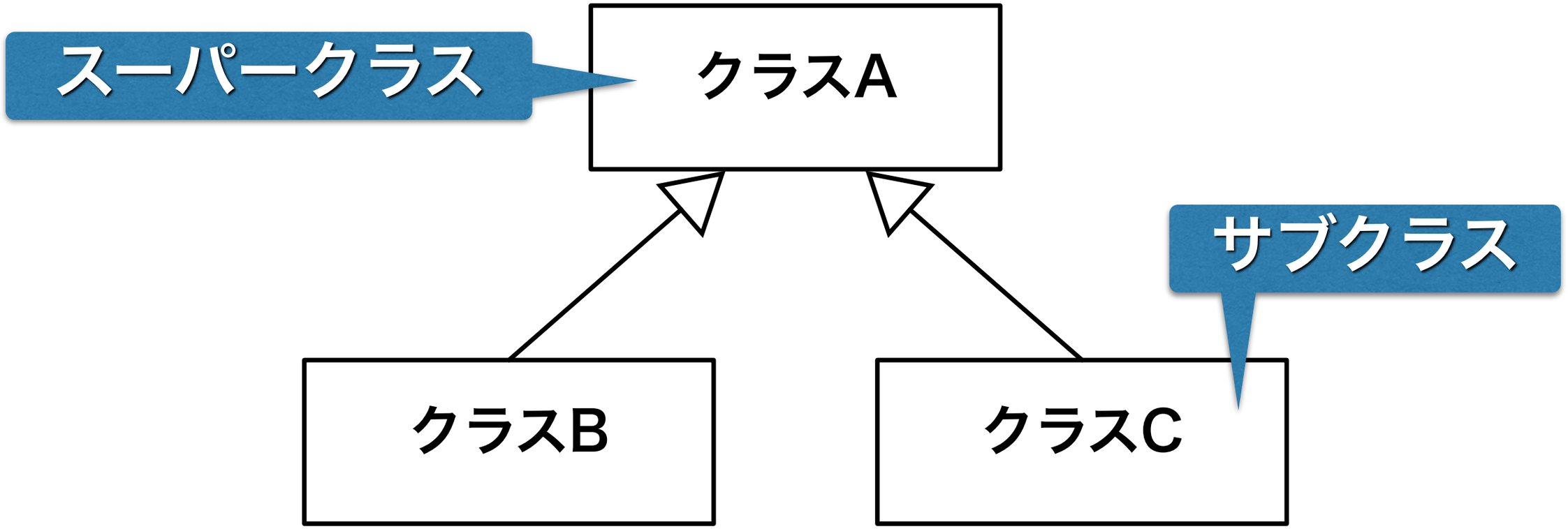
多重度は次の2つの方法で入力できる

- 関係のプロパティから「関連端」のタブで多重度を選択または入力する
- 入力する関係の関連端付近に表示される「m」をクリックする

継承（汎化）

「スーパークラス（親クラス）とサブクラス（子クラス）の継承関係」を関係線で表す。

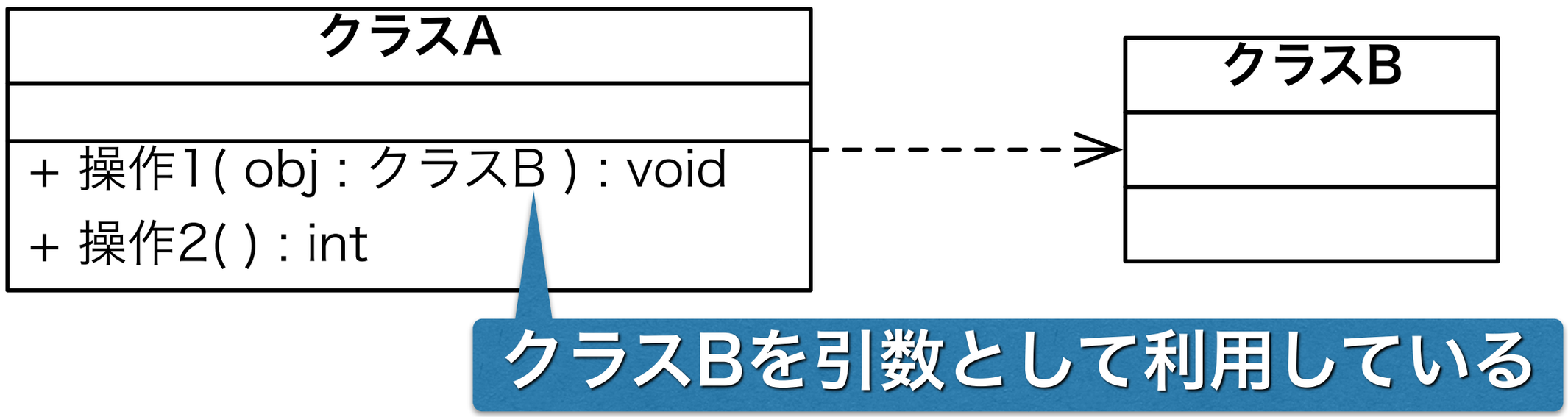
【例】クラスBとクラスCがクラスAを継承している



依存

あるクラスを別のクラスで「操作の引数として利用する場合」「操作のローカル変数として利用する場合」の関係を表す。

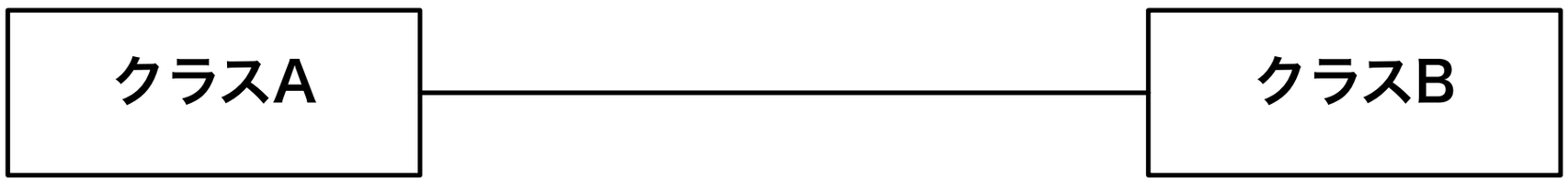
【例】クラスAがクラスBを操作1の引数として利用されている



関連

「あるクラスから他のクラスのメソッドを呼び出す、などといった、他のクラスとの構造的な関係」を関係線で表す。

【例】クラスAがクラスBに対して操作（メソッド）を呼び出す

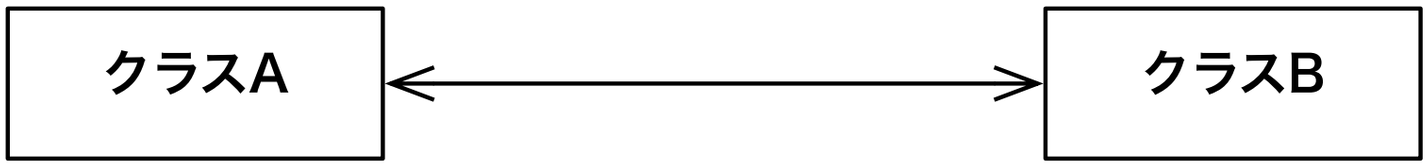


クラス間に何かしらの関係があるが、その時点では未定の場合は、「関連」で表しておき、設計が進んだ際に「集約」「依存」「汎化」などの関係に明確化することもできる

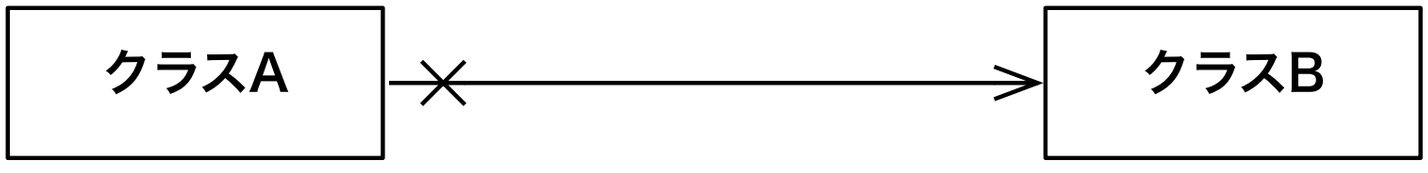
関連・集約の方向性（誘導可能性）

関連や集約に矢印を付けて参照できる方向を明示的に表すことができる。（矢印がない場合は方向性を指定しないないことを意味する）

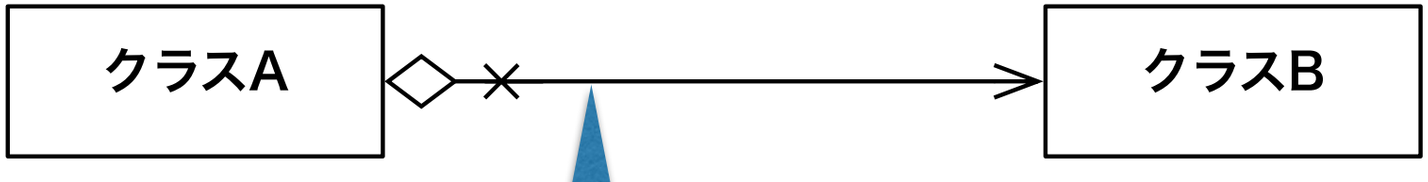
相互に参照できる場合



クラスAからのみ参照できる場合



全体を表すクラスAからのみ参照できる場合



集約の場合、全体を表すクラスから部分を表すクラスへの方向性を示すのが一般的である

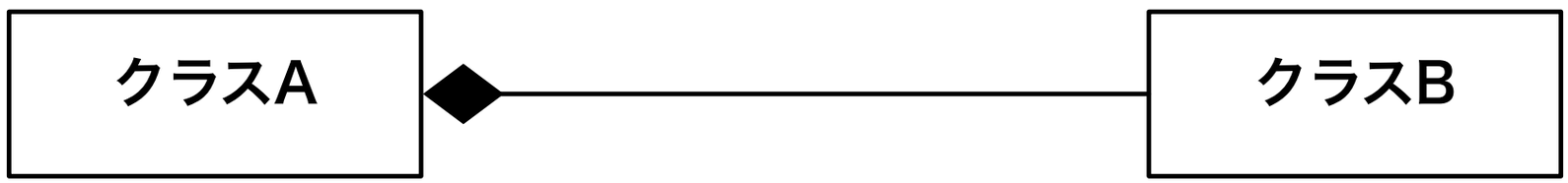
astahの補足

誘導可能性について

集約を表現する場合は、「誘導可能性」の設定を変更して、参照方向を次のようにする（このダイアグラムから生成されるスケルトンコードに影響するため）。設定方法は、プロパティ画面からの方法と、関連端で右クリックしたメニューからの方法の2つがある。

コンポジション

集約に似た関係の概念であり、表記の方法も集約と似ている。しかし、コンポジションには、以下のような「共有不可」という設計者の意図が含まれる。



- 含まれる側のクラスのインスタンスは、含む側のクラスの複数のインスタンスから同時に含まれることはない。（クラスBのあるインスタンスが、クラスAの複数のインスタンスから含まれることはない。）

- 含む側のクラスのインスタンスが削除される場合、そのインスタンスが含んでいるインスタンスも全て同時に削除される。（クラスAのあるインスタンスが削除される場合、そのインスタンスが含んでいるクラスBのインスタンスも同時に削除される。）

【課題の準備】

演習室で作業する前に、以下のコマンドを
入れるだけで準備が完了する

```
$   mygitclone4d   「自分のGitHubユーザ名」
$   cd prog4d-(ユーザ名)
$   ./myconf
```

※本体をシャットダウンするまでは、
上記「mygitclone」と「myconf」の設定は有効です

【課題の準備】

以下の流れで、課題のプログラムを作るためのフォルダを準備しましょう。

1. 端末を起動して、以下のコマンドを実行して後期第2週のフォルダを作る

```
$ cd prog4d-(ユーザ名)        (←既に移動しているなら不要)
$ mkdir week202
$ cd week202
```

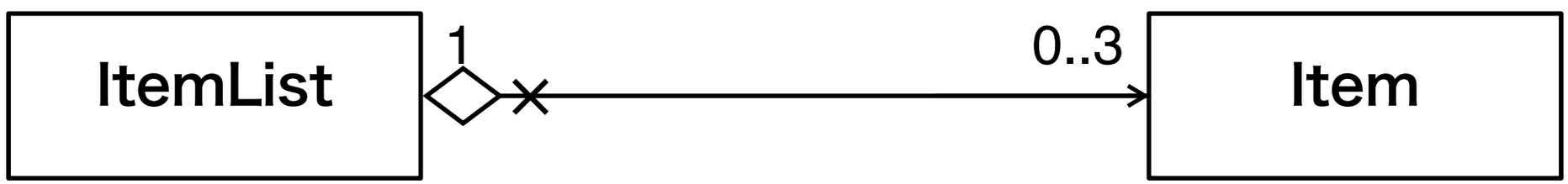
※課題で作るファイル名は各自で決めて構いません。

【練習2-1】

astahを使って、次のJavaプログラムのクラス定義から、UMLクラス図を作成しましょう。

```
class ItemList {
  private Item[] list;
  ItemList() {
    list = new Item[3];
  }
  public void show() {
    // list[0]～list[2]の
    // show()を呼び出す
  }
}
```

```
class Item {
  private String name;
  private int price;
  public void setName(String n) {}
  public String getName() {}
  public void setPrice(int p) {}
  public int getPrice() {}
  public void show() {}
}
```

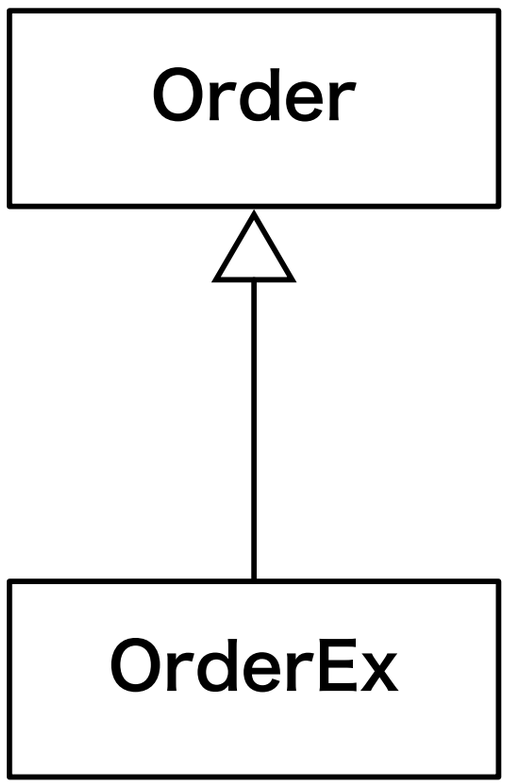


【練習2-2】

astahを使って、次のJavaプログラムのクラス定義から、UMLクラス図を作成しましょう。

```
class Order {
  protected String name;
  protected int price;
  protected int quantity;
  public void show() {}
  public int calcPrice() {}
}
```

```
class OrderEx extends Order {
  private int off;
  private String genre;
  public void show() {}
  public int calcPrice() {}
}
```



【課題2-1】

前回別紙に示したプログラム中のクラス定義から、クラス図の「集約（多重度含む）と汎化」部分を作成してください。
ただし、前回描いたクラス以外のクラスについては、クラス名のみでも構いません。

クラスの関係を表すために宣言されているフィールド変数について

クラス図ではこれを関係線として描いた場合は、意味が重複するため、フィールド変数（つまり属性）には描かない。（このダイアグラムから生成されるスケルトンコードに影響するため）

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
$ git commit -m “課題2-1提出”
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

https://github.com/nit-ibaraki-prog4d-2019/prog4d-(ユーザ名)

【課題2-2】

課題3-1で作成したクラス図に、ScheduleMainが利用しているローカル変数に関する依存関係をクラス図に追加してください。

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
$ git commit -m “課題2-2提出”
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

https://github.com/nit-ibaraki-prog4d-2019/prog4d-(ユーザ名)