プログラム設計

http://bit.ly/design4d

シーケンス図（1）

後期 第8週

2019/11/18

今回と次回は

UMLのシーケンス図を学びます

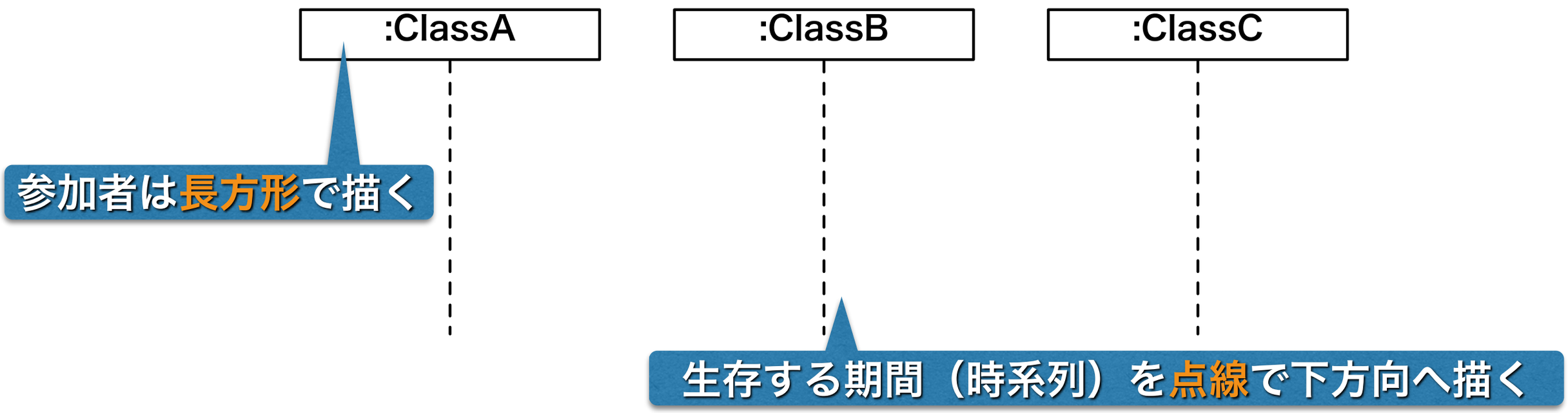
オブジェクト同士のメッセージ（操作の呼び出し）のやり取りを表現する → オブジェクト（インスタンス）の振る舞いを表現するための図

今回扱うシーケンス図の基本要素

- ライフライン
- メッセージ
- 実行指定

ライフライン

シーケンス図で表現される相互作用の参加者を表す

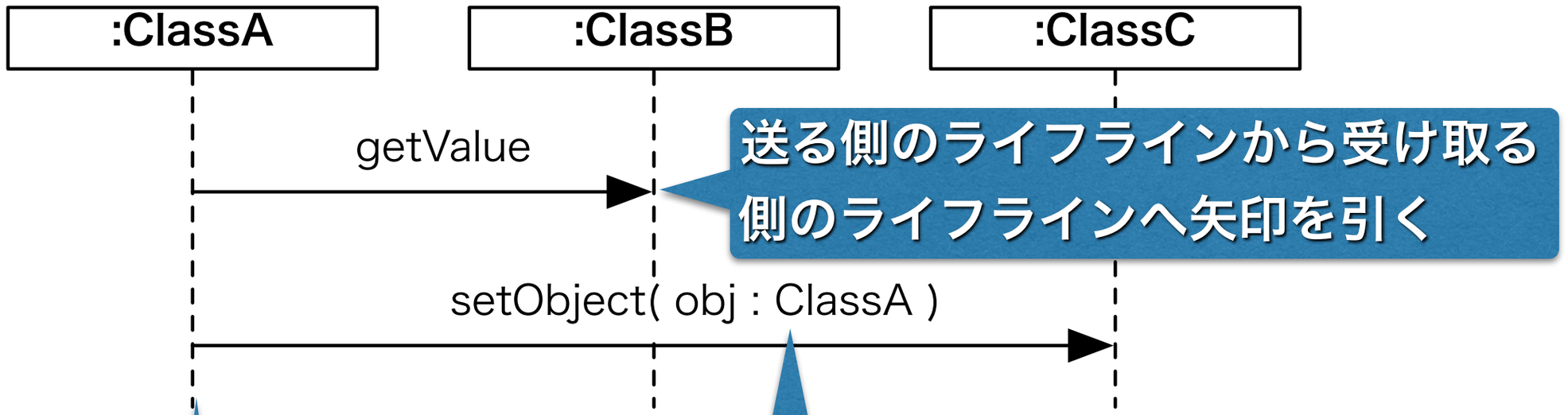


ライフラインの名前の記述方法には3種類ある

オブジェクト名のみ	オブジェクト名とクラス名	クラス名のみ
obj1	obj1 : ClassA	: ClassA

メッセージ

ライフライン間のメッセージ送信（操作呼び出し）を表す



astahの補足

ライフラインの作成について

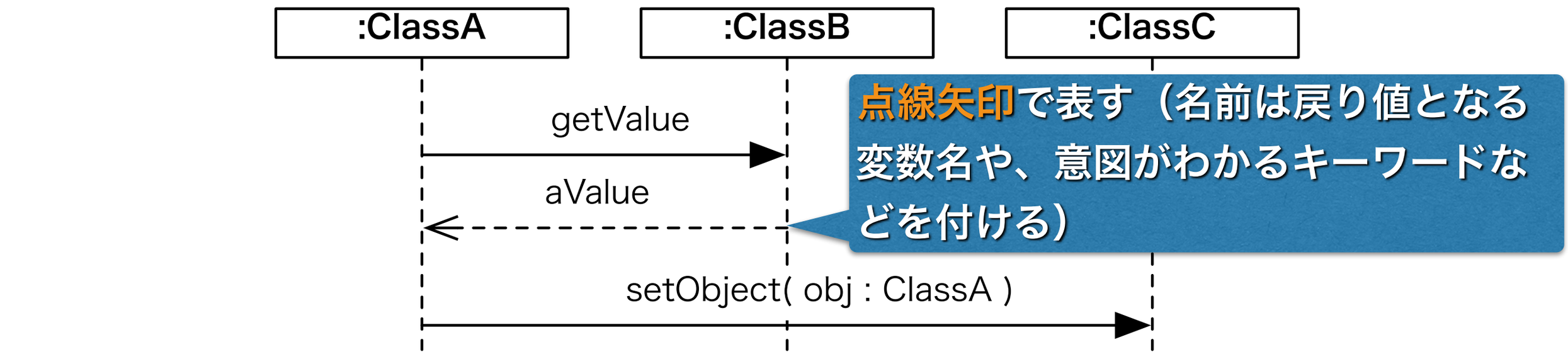
ツールバーから「ライフライン」を選択し、配置する場所でクリックする
（ドラッグすると配置順を変更することもできる）

メッセージの作成について

ツールバーから「メッセージ」（「非同期メッセージ」ではない）を選択し、
つなぐライフラインを順番にクリックする

メッセージ（リプライ）

実行した操作の戻り値がある場合リプライで表す



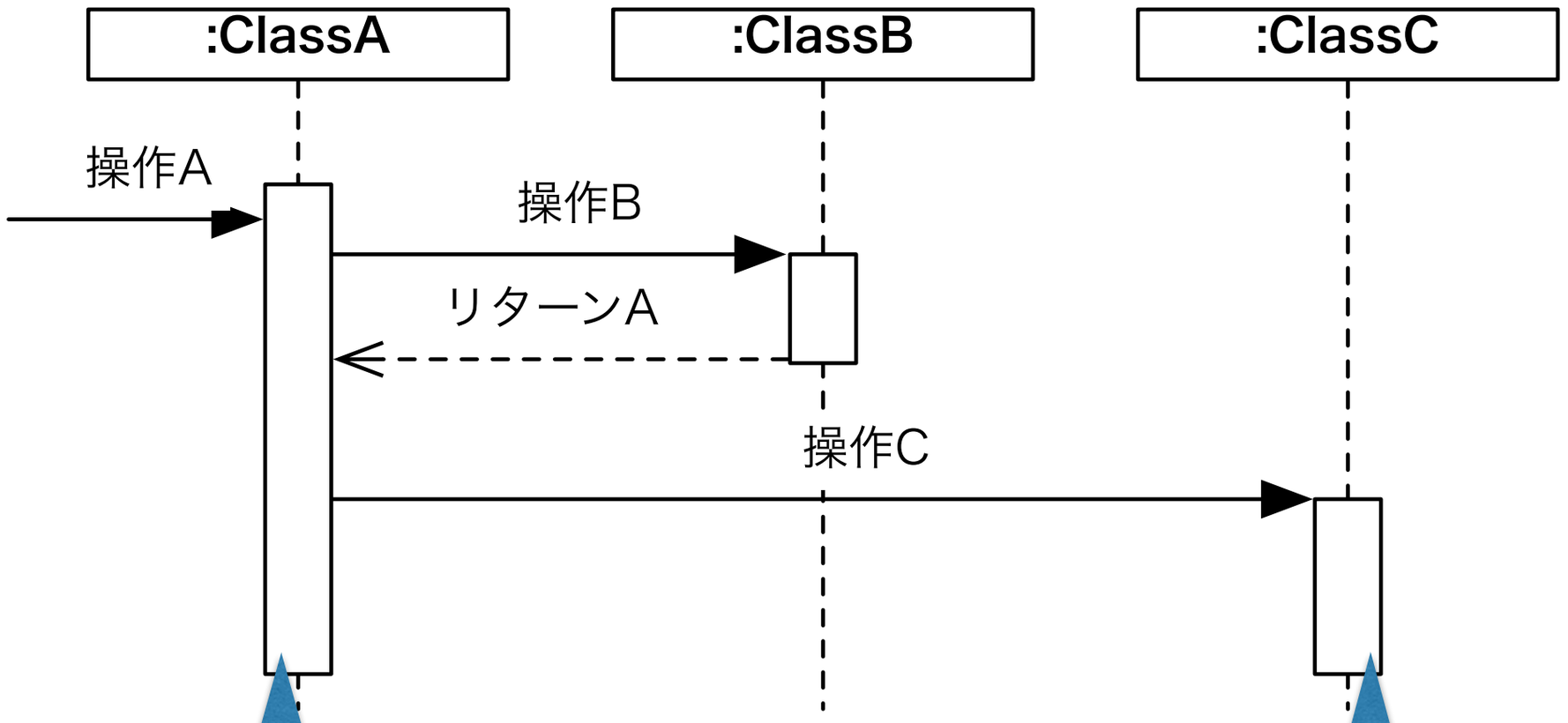
リプライの作成について

ツールバーから「Replyメッセージ」を選択し、リプライの対象となるライフラインをクリックする

実行指定（実行仕様，活性区間とも呼ぶ）

実行指定は次の2つの意味を表現する

- オブジェクトがある操作を実行している期間 → その操作の実行期間の長さ
- 「ある操作から別の操作を呼び出している」関係 → 操作の依存関係

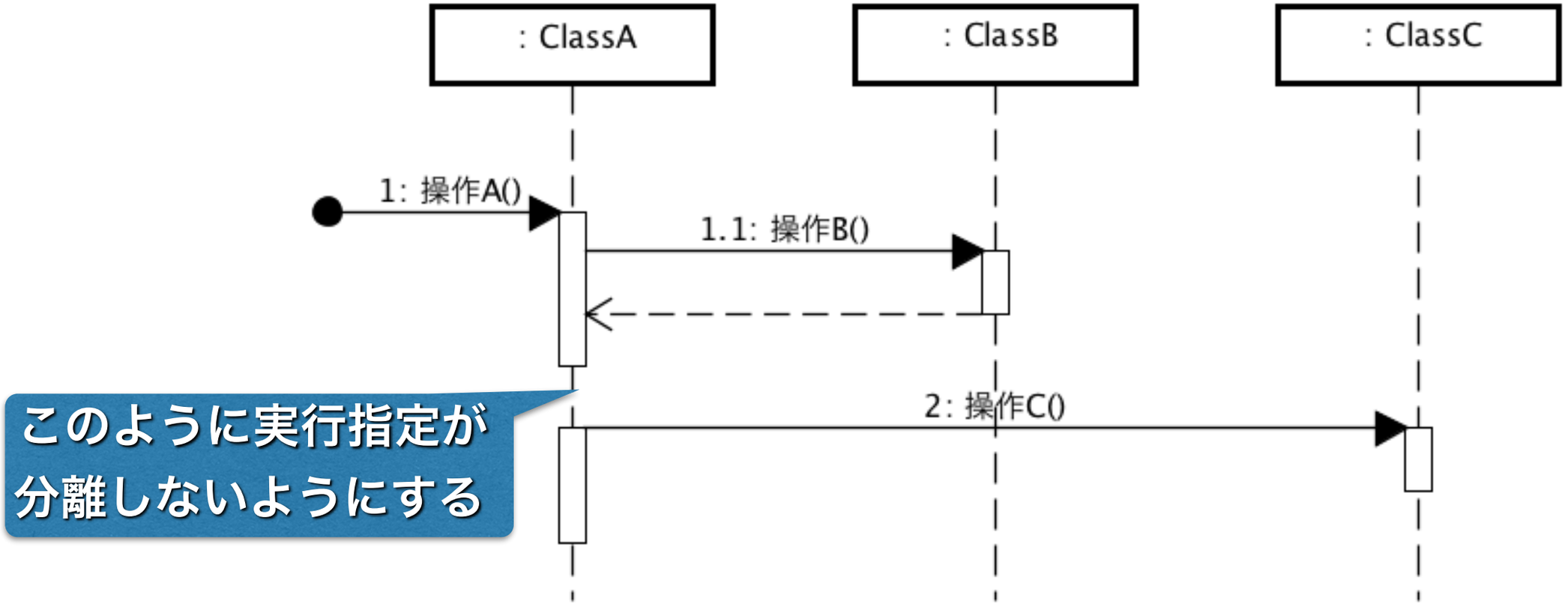


操作Aの内側に操作BとCの実行指定が描かれている → 操作BとCは、操作Aの実行している間に処理される

操作Bより操作Cの方が実行指定が長い
→ 操作Bより操作Cの方が実行時間が長い

実行指定の補足

同じ操作の中で送信されるメッセージは、同じ実行指定の中から送信されることになります。
例えば、前のスライドの場合、「操作Aから操作Bと操作Cがメッセージ送信されている」ことを表現するために、操作Bと操作Cは、操作Aの同じ実行指定から送信されています。（つまり、操作Aの実行指定が分離していない）



【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8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 week208
$ cd week208
```

※課題で作るファイル名は各自で決めて構いません。

【練習8-1】

astahを使って、「メッセージ（リプライ）」のスライドにあるシーケンス図を作ってみましょう。

【練習8-2】

astahを使って、次のJavaプログラムのmain内のメソッドshowのシーケンス図を作成しましょう。

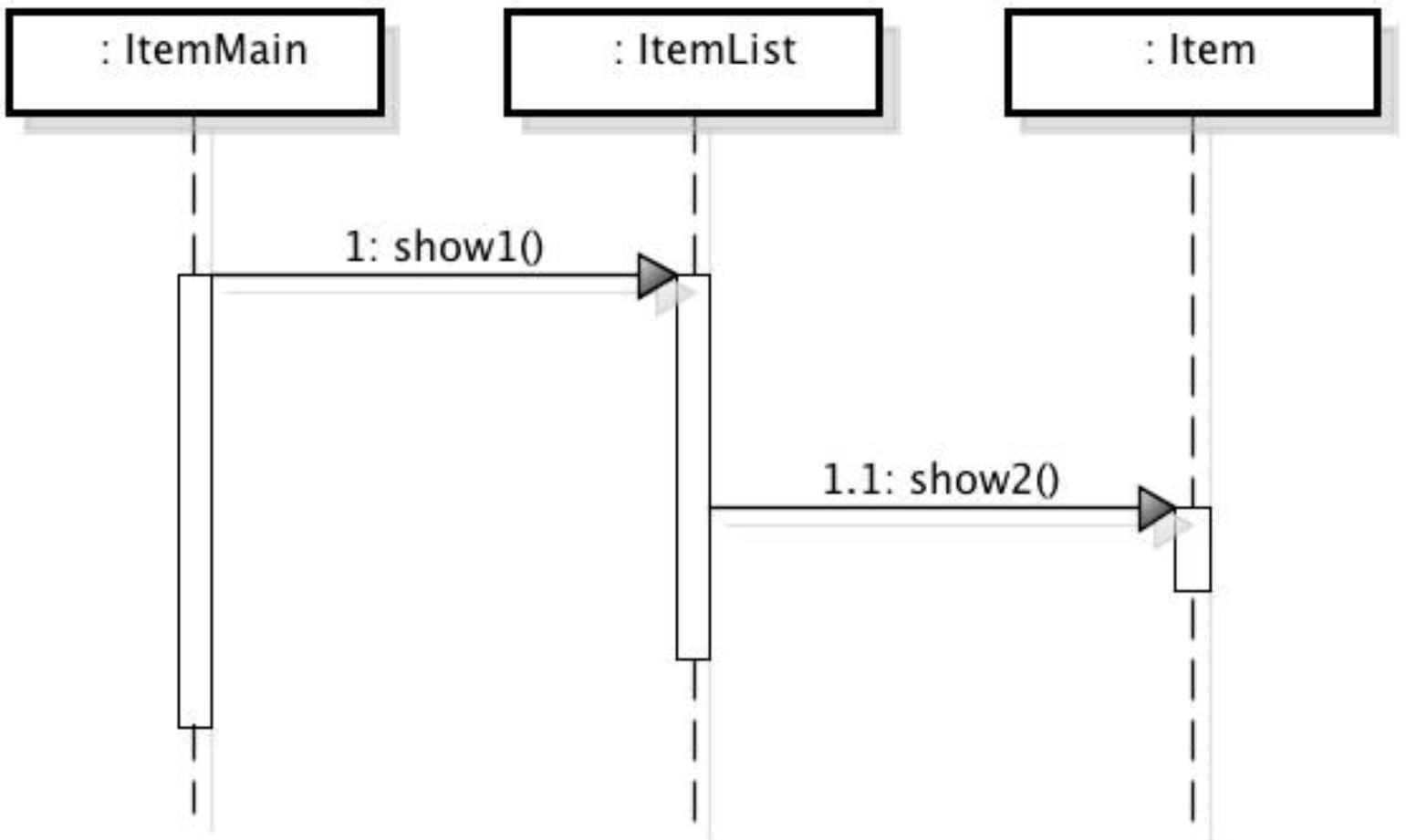
```
class Item {
  private String name;
  private int price;
  //nameを設定する
  public void setName(String n){
    name = n;
  }
  // priceを設定する
  public void setPrice(int p) {
    price = p;
  }
  // 名前と価格を表示する
  public void show2() {
    System.out.printf("name: %s\n", name);
    System.out.printf("price: %d\n", price);
  }
}
```

```
class ItemList {
  private Item list[];
  ItemList() {
    list = new Item[3];
  }
  public void show1() {
    int i;
    // list[0]～list[2]の
    // show2()を呼び出す
    for(i=0; i<3; i++) {
      list[i].show2();
    }
  }
}

class ItemMain {
  public static void main(String[]
        args) {
    ItemList il = new ItemList();
    il.show1();
  }
}
```

【練習8-2】

メッセージ送信先のクラスには、そのメッセージに対応した操作（メソッド）が存在します。



【課題8-1】

別紙に示すプログラムから、クラスNumberMainのmainメソッドより呼び出されているメソッドshowSquareXに関するシーケンス図を作成してください。なお、NumberMainもライフラインの1つとします。

【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8-1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8-2】

別紙に示すプログラムから、クラスNumberMainのmainメソッドより呼び出されているメソッドsumSquareArrayに関するシーケンス図を作成してください。なお、NumberMainもライフラインの1つとします。

【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8-2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)