プログラム設計

http://bit.ly/design4d

シーケンス図（2）

後期 第9週

2019/11/26

今回は

前回に続き、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

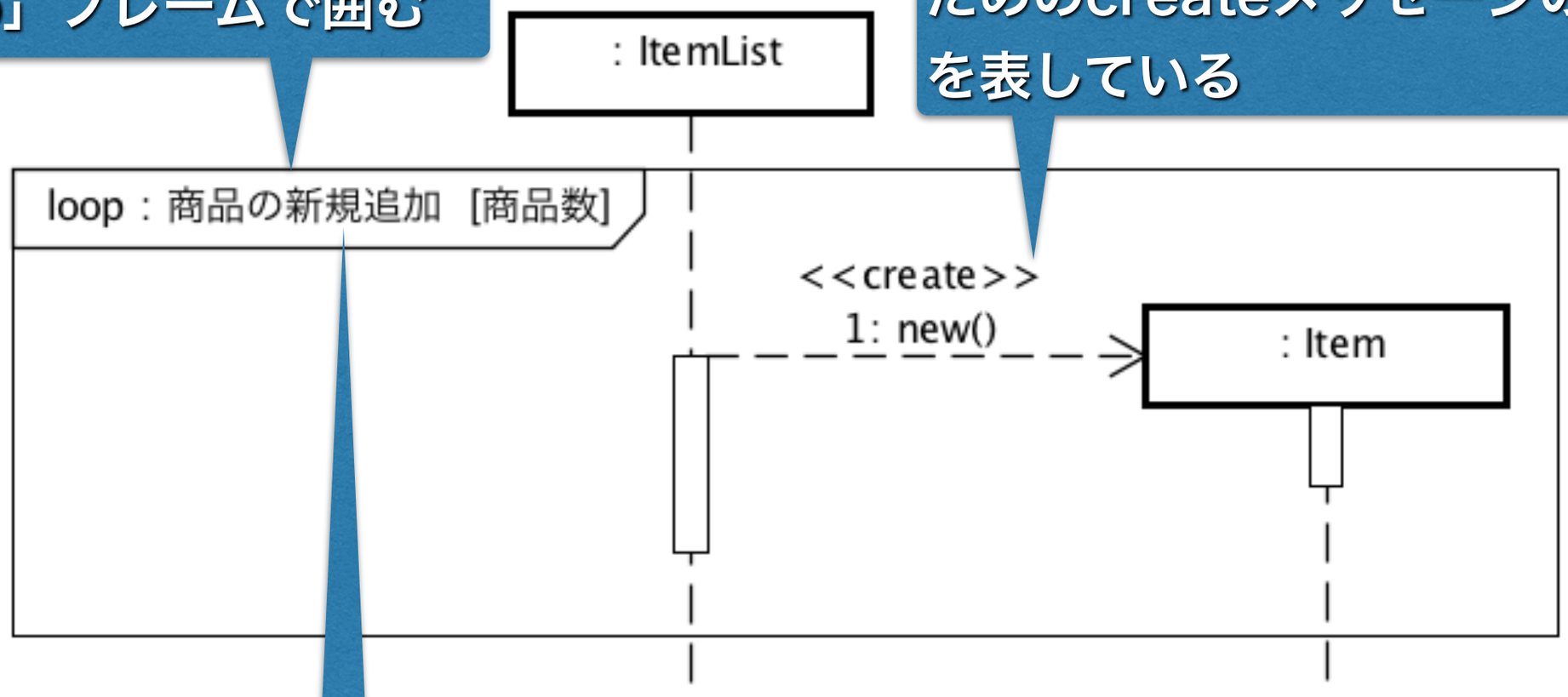
- **繰り返し**
- **分岐**
- **ライフラインの生成と消滅**

繰り返し

シーケンス図で表現されるメッセージの繰り返しを表す

ライフライン上に繰り返す部分を「loop」フレームで囲む

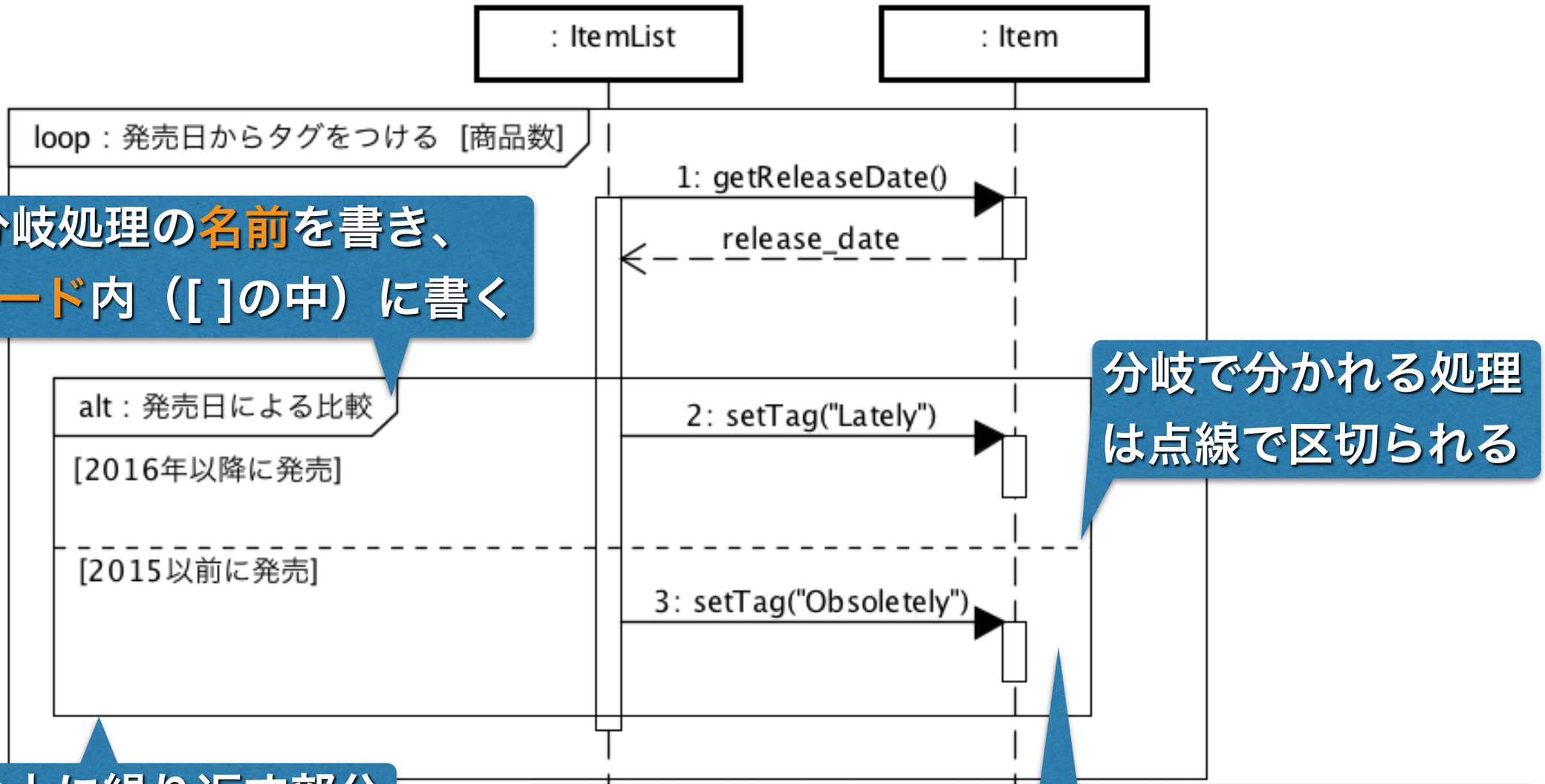
このloopは、インスタンスを生成するためのcreateメッセージの繰り返しを表している



「:」の後に繰り返し処理の名前を書き、
繰り返す回数や繰り返し条件をガード内（[ ]の中）に書く

分岐

シーケンス図で表現されるメッセージの分岐を表す



「:」の後に分岐処理の名前を書き、分岐条件をガード内（[ ]の中）に書く

分岐で分かれる処理は点線で区切られる

ライフライン上に繰り返す部分を「alt」フレームで囲む

このaltは、（商品の）発売日によって、付けるタグを変えるための分岐処理を表している

astahの補足

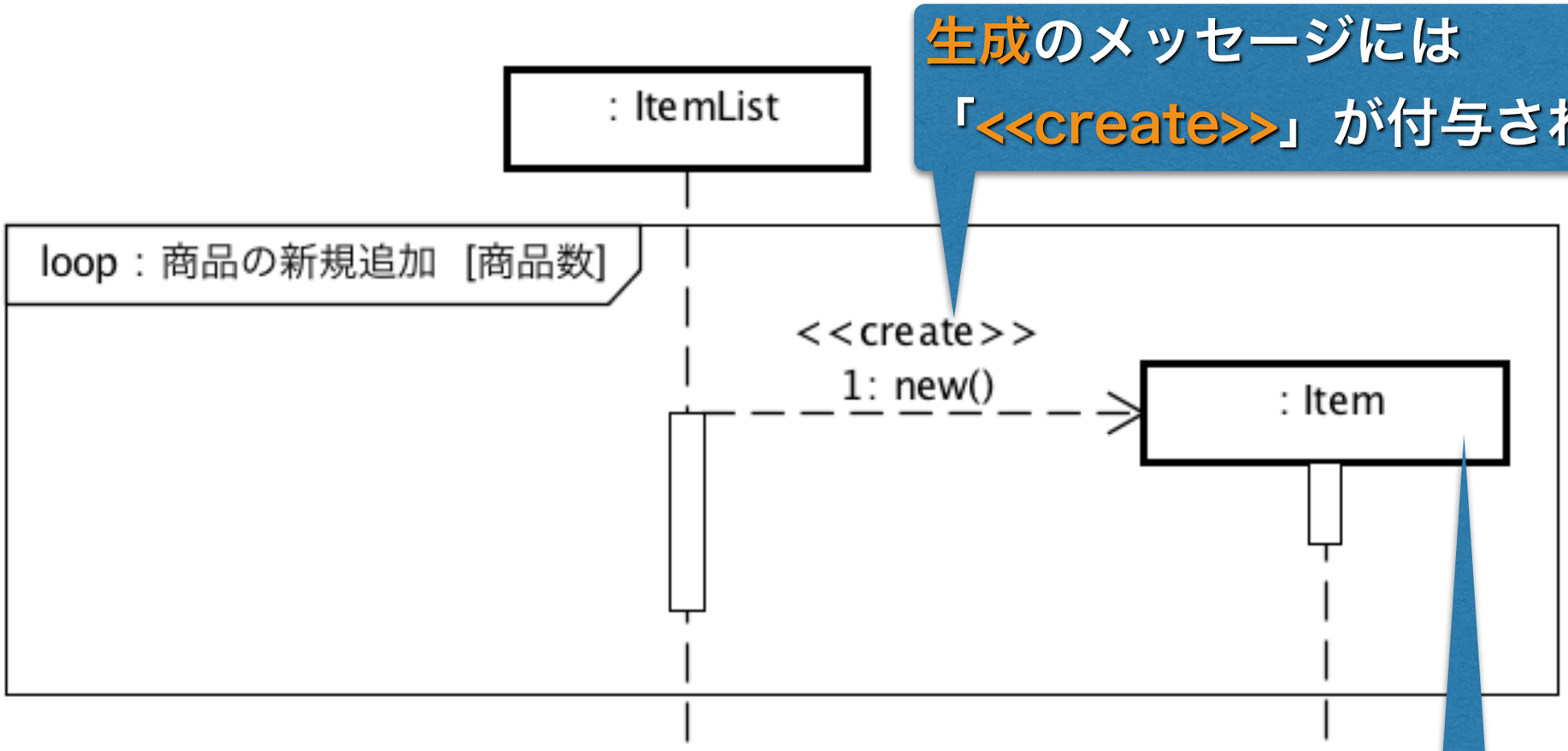
分岐と繰り返しの描き方について

- 「複合フラグメント」ボタンを選択して、ドラッグ操作で範囲を指定する

- 画面左下のプロパティで、「種類」から分岐/繰り返しなどを選択でき、
  「名前」で処理名、「オペランド」で分岐条件/繰り返し回数を入力できる

ライフラインの生成

インスタンス生成のメッセージを表す



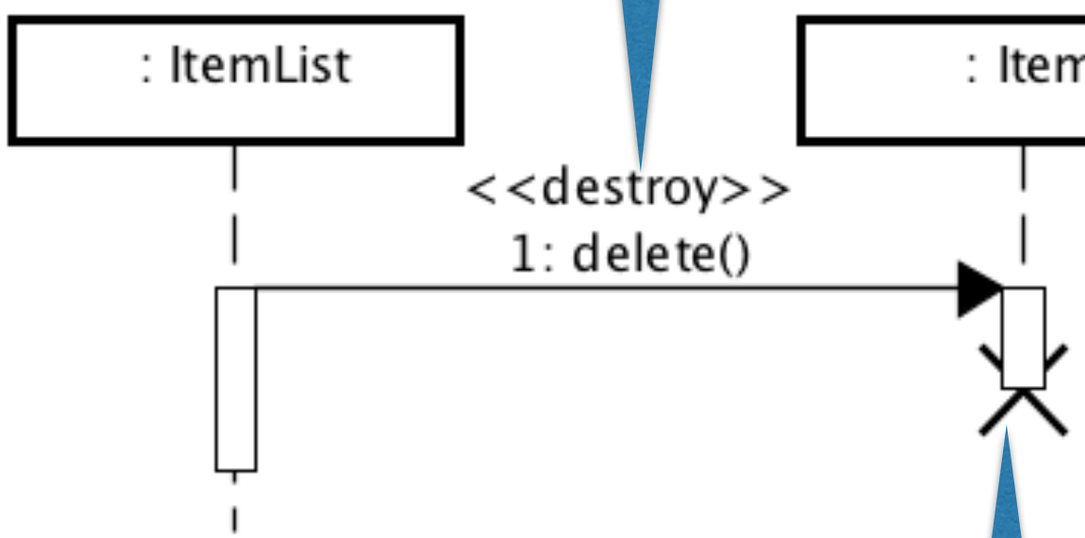
生成のメッセージには
「<<create>>」が付与される

インスタンスが生成されてからライフラインが開始するので、ライフラインの表記もそこから開始する

ライフラインの消滅

インスタンス消滅のメッセージを表す

消滅のメッセージには
「<<destroy>>」が付与される



インスタンスが消滅するとライフラインもそこで終了して、このメッセージの実行指定の終端に×印を描く

astahの補足

生成と消滅の描き方について

- 生成は「Createメッセージ」ボタンを選択して、通常のメッセージのようにライフライン同士をつなぐ（メッセージ名の上には「<<create>>」というステレオタイプが自動的に付く）

- 消滅は「Destroyメッセージ」ボタンを選択して、ライフライン同士をつなぐ（生成時と同様に「<<destrop>>」が付き、×印も自動的に付く）

- ステレオタイプは、種類を表記する際に「<<>>」で囲んで表記する（クラス図のクラスにも「<<Model>>, <<View>>, <<Controller>>」のように使うことがある）

【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9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 week209
$ cd week209
```

※課題で作るファイル名は各自で決めて構いません。

【練習9-1】

astahを使って、「繰り返し」「分岐」「生成」「消滅」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9-2】

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「繰り返し」付きで作成しましょう。

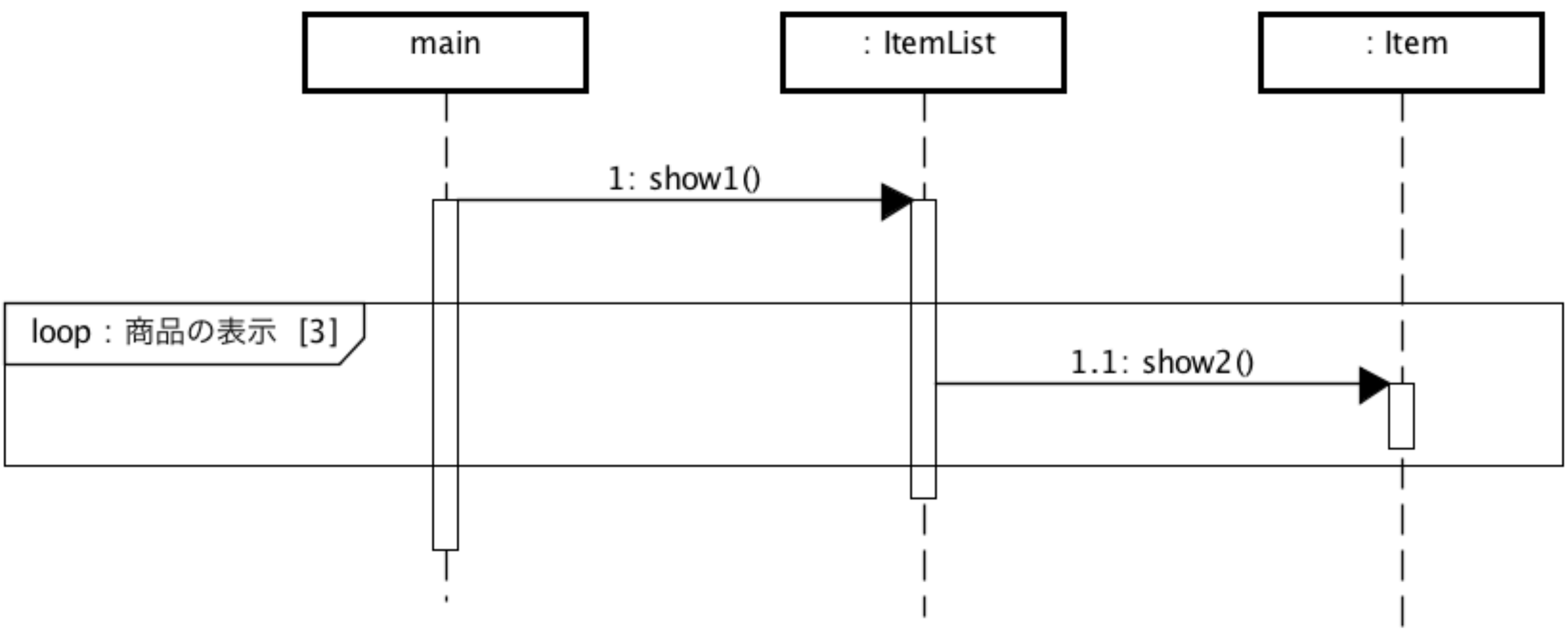
```
class Item {
  private String name;
  private int price;
  //nameを設定する
  public void setName(String n){
    name = n;
  }
  // priceを設定する
  public void setPrice(int p) {
    price = p;
  }
  // 名前と価格を表示する
  public void show2() {
    System.out.printf("name: %s\n", name);
    System.out.printf("price: %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9-2】

前回作成した練習8-2のシーケンス図に、loopを追加します。



【課題9-1】

前回課題8-2で作成したシーケンス図に、繰り返しの表記を追加して下さい。

【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9-1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)