プログラミング応用

http://bit.ly/ouyou3d

ソフトウェア開発方法論

後期 第13週

2019/1/10

本日は・・・

- ソフトウェア開発の現状と問題
- ソフトウェア開発工程・方法論
- UMLを用いたソフトウェア設計

ソフトウェア開発 ≠ プログラミング

本日は・・・

- ソフトウェア開発の現状と問題
- ソフトウェア開発工程・方法論
- UMLを用いたソフトウェア設計

ソフトウェア開発の現状

- 大規模化

  必要なソースコードの行数が増大

  → 多くの開発者が共同で開発する必要あり

- 複雑化

  社会が複雑化するのに合わせ、ソフトウェアへの要求も複雑化

ソフトウェア開発の問題

- 大規模化

  必要なソースコードの行数が増大

  → 多くの開発者が共同で開発する必要あり

  → 共同開発は難しい

  　共同開発によるバグの増大

- 複雑化

  社会が複雑化するのに合わせ、ソフトウェアへの

  要求も複雑化

  → 複雑化により設計, 実装がより難化

大規模化/複雑化したソフトウェアを完成させる方法論が必要

ソフトウェア危機

ソフトウェアへの依存度が大きくなっていることに対する危機感を以下の主要な5項目で表す

- ソフトウェアの巨大化, 複雑化に伴う開発費用の増大
- ハードウェア 対 ソフトウェアのコスト比の変化
  （「ハード ＞ ソフト」 から 「ハード ＜ ソフト」）
- ソフトウェア保守にかかる工数の増大
- ソフトウェア需要に対する供給能力の低下
- ソフトウェアトラブルの社会的問題化

ソフトウェア工学について

次のような目的をもった工学分野

- ソフトウェアの生産性，信頼性の低下を防ぐ
- 学術的(理論や方法論)および、実践的(技術や技法)な分野を目指している

ソフトウェア工学が確立した経緯

- 1968年：ソフトウェア工学とソフトウェア危機が提唱される
- 1970年代後半：ソフトウェア工学の分野が定着していく
  （ソフトウェア工学国際会議が開催される）

本日は・・・

- ソフトウェア開発の現状と問題
- ソフトウェア開発工程・方法論
- UMLを用いたソフトウェア設計

ソフトウェア開発手法

ソフトウェアを開発・運営する過程→ライフサイクル
ライフサイクルはプロセスモデルで表現される

代表的なプロセスモデル

- **ウォーターフォールモデル**
- **スパイラルモデル**
- **アジャイルソフトウェア開発**

ウォーターフォールモデル

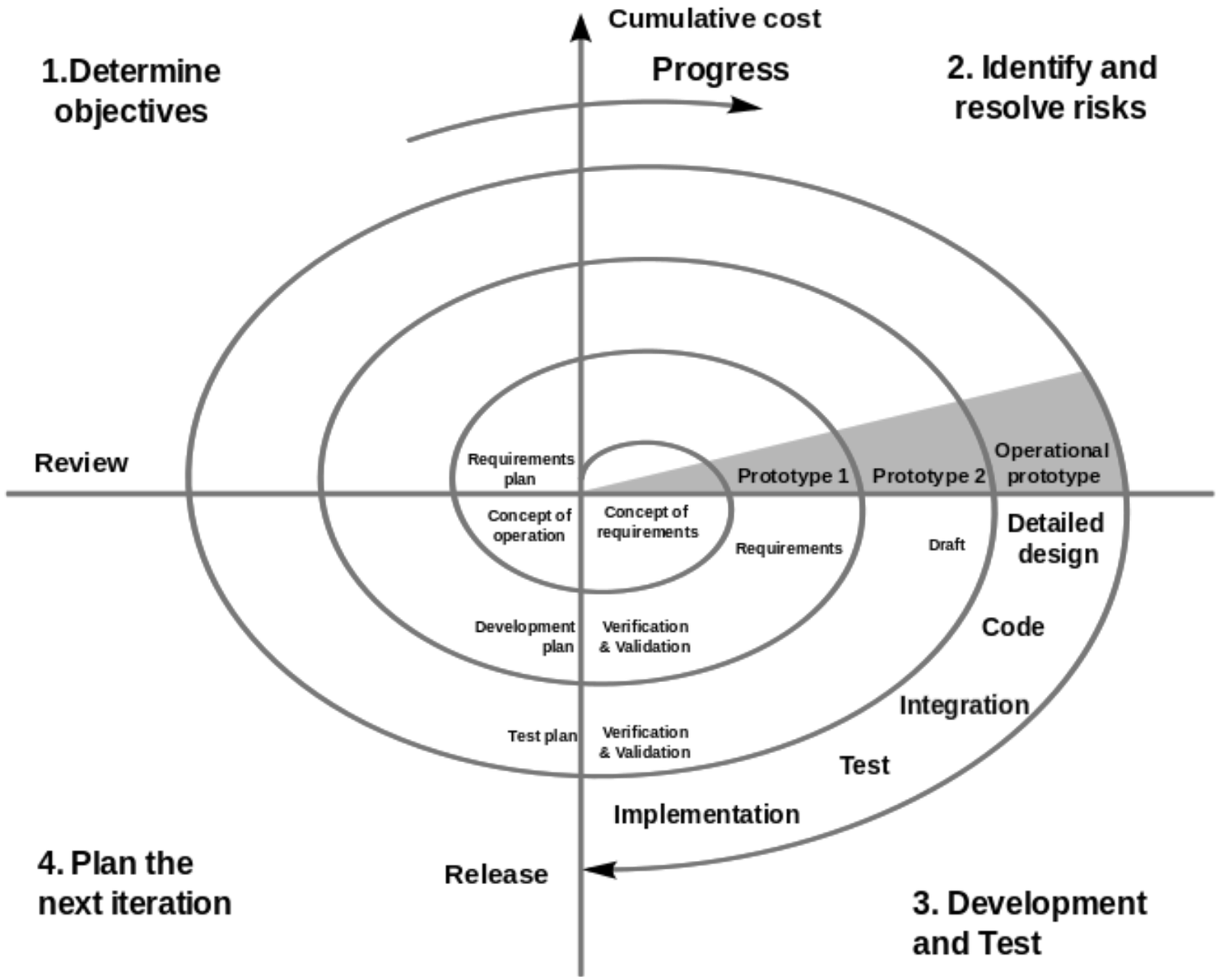
いくつかの工程に分けて開発を進める

上流工程：利用者側の視点（要求定義, 基本設計）

下流工程：開発者側の視点（詳細設計, 実装, テスト, 保守）

スパイラルモデル

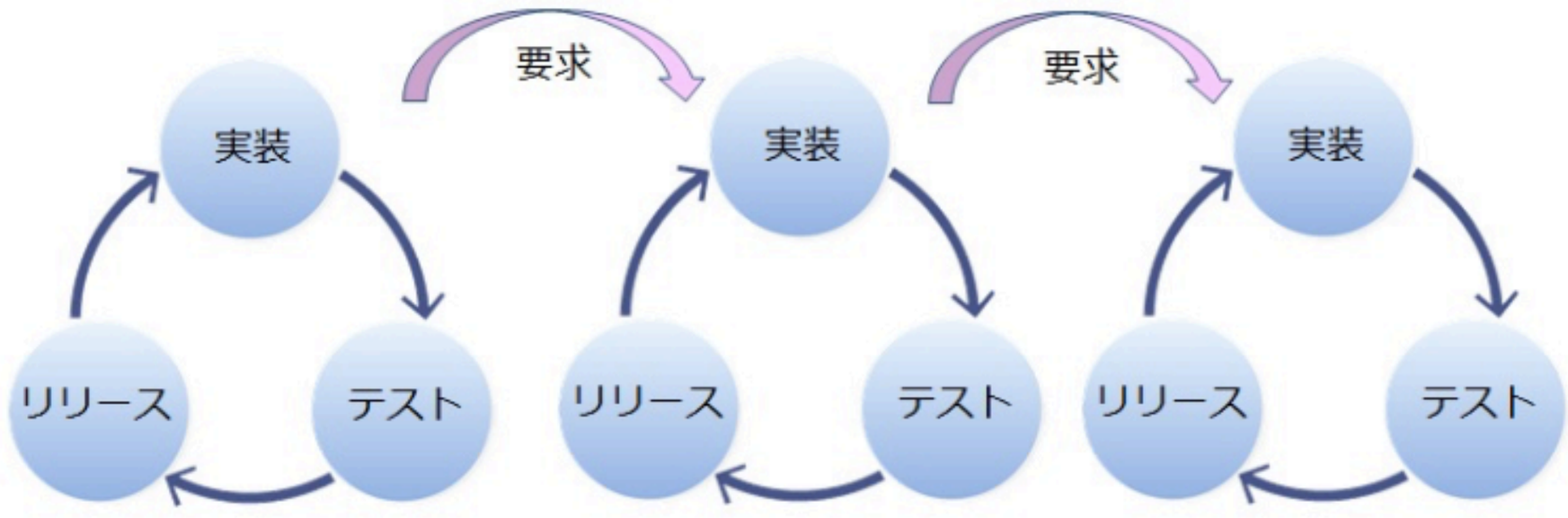
- 時間の経過とともに、ある方向へ向けて繰り返しながら成長する
- 4つのフェーズを1サイクルとし、プロトタイピングを繰り返す



※Wikipediaより

アジャイルソフトウェア開発

- Extreme Programmingが有名
- 軽量で短期間の開発に適している
- リリース（ある程度の形になって動作するソフトウェア）
  → 2〜3ヶ月間隔
- イテレーション（ソフトウェアの部分的な設計・実装・テスト）
  → 2〜3週間



いくつかのイテレーションを完了することでリリースを達成し、
リリースを繰り返すことで完成度を上げる

本日は・・・

- ソフトウェア開発の現状と問題
- ソフトウェア開発工程・方法論
- UMLを用いたソフトウェア設計

UMLとは

UML（Unified Modeling Language）は、OMGという団体により標準化された、ソフトウェアの仕様や設計を図として表現するために使用する言語です。

「クラス」の概念を用いたオブジェクト指向開発技法で利用されます。

現在はUML 2が利用されています。

ダイアグラムの種類

UML 2では、13種類の図（ダイアグラム）が定められています。

本講義では、その中でも使用頻度の高いダイアグラムを数種類を扱います。

UMLによるソフトウェア設計

主に2つの側面を図で表す

- **静的側面**：ソフトウェアを構成する要素（関数, クラスなど）の構造を表す
  → クラス図など
- **動的側面**：ソフトウェアの動きを表す
  → シーケンス図, アクティビティ図など

今回学ぶダイアグラム

クラス図

クラスの変数（属性）や関数（操作, メソッド），クラス間の関係を表現する

アクティビティ図

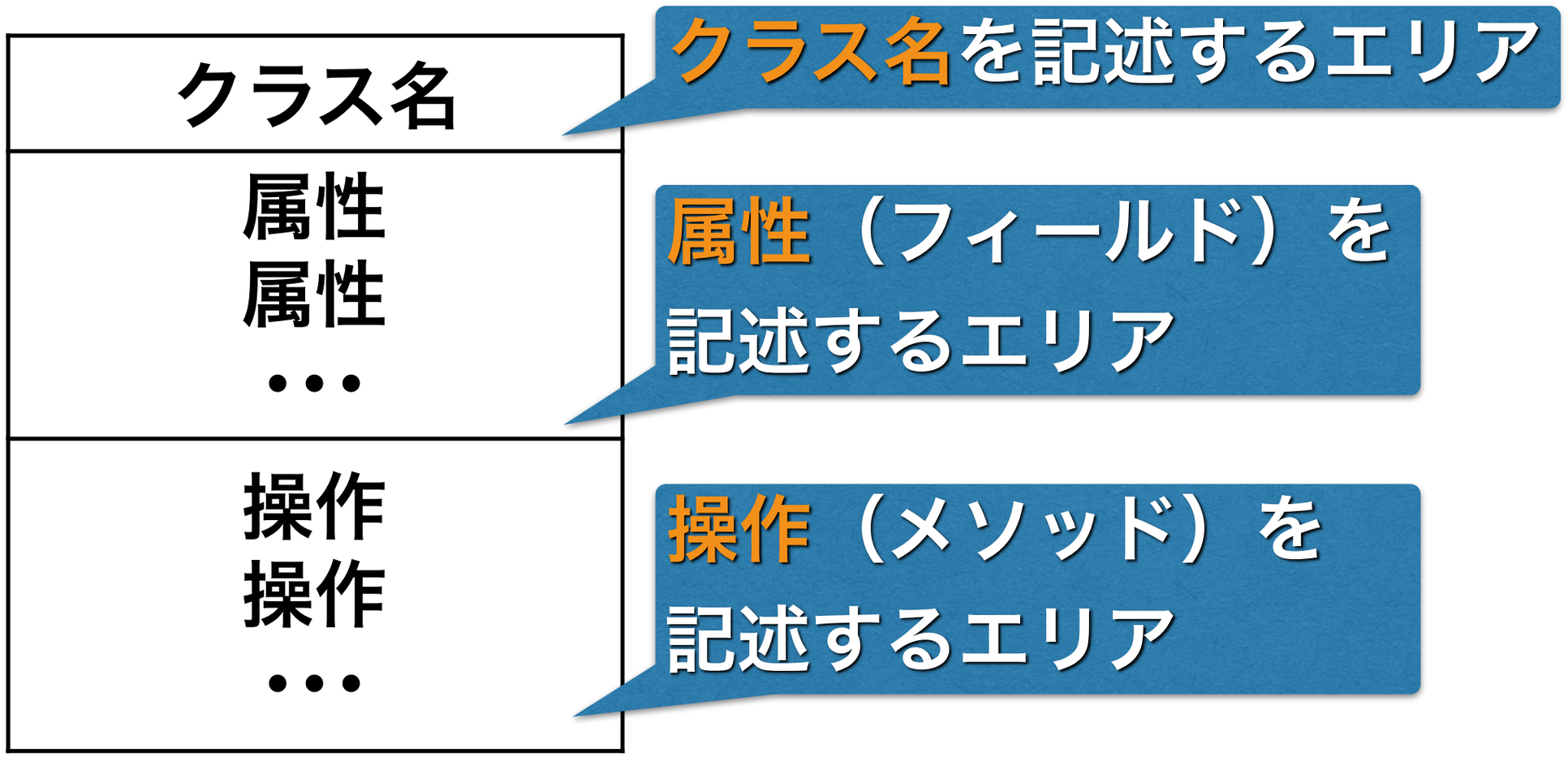
オブジェクトの操作, 手続きなどの振る舞いを表現する（フローチャートと同様）

本日は・・・

- ソフトウェア開発の現状と問題
- ソフトウェア開発工程・方法論
- UMLを用いたソフトウェア設計
  - クラス図
  - アクティビティ図

クラスの表記

クラスは、以下の3つのエリアに分けて描きます。



属性の表記

属性はクラスが持つデータを表現します。必須なのは「名前」のみで、その他は省略可能です。

【書式】

可視性　属性名: 型　＝　初期値

- **可視性**: 他のクラスから参照可能かを表す
  「+」(public),「-」(private),「#」(protected),「~」(package)
- **属性名**: 属性を表す名前
- **型**: 属性の型
- **初期値**: 属性が最初に持つ値

操作の表記

操作はクラスが持つ振る舞いを表現します。必須なのは「名前」のみで、その他は省略可能です。

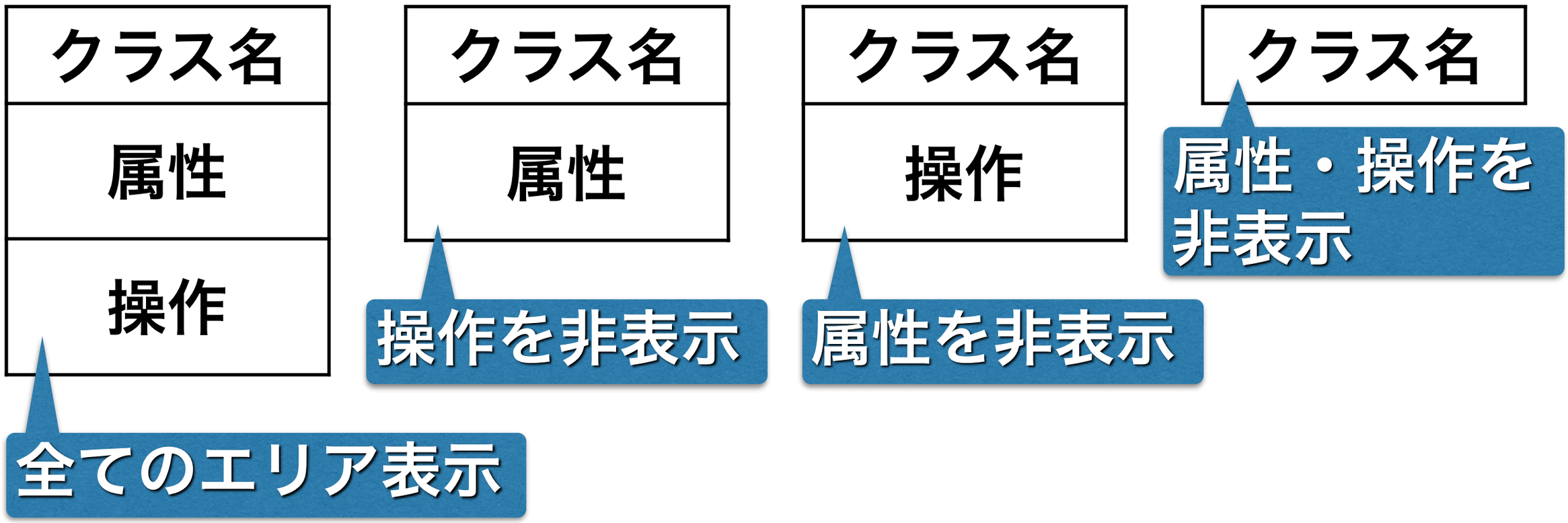
【書式】

可視性　操作名(引数名: 引数の型, ...) : 戻り値の型

- 可視性: 他のクラスから参照可能かを表す
  「+」(public),「-」(private),「#」(protected),「~」(package)
- 操作名: 操作を表す名前
- 引数名: 引数を表す名前
- 型: 引数と戻り値の型

クラス表記のバリエーション

クラスは属性および操作の表示領域のどちらか、または両方を非表示にすることができます。（つまり、非表示になっていても、属性・操作が定義されている場合があるので注意して下さい。）



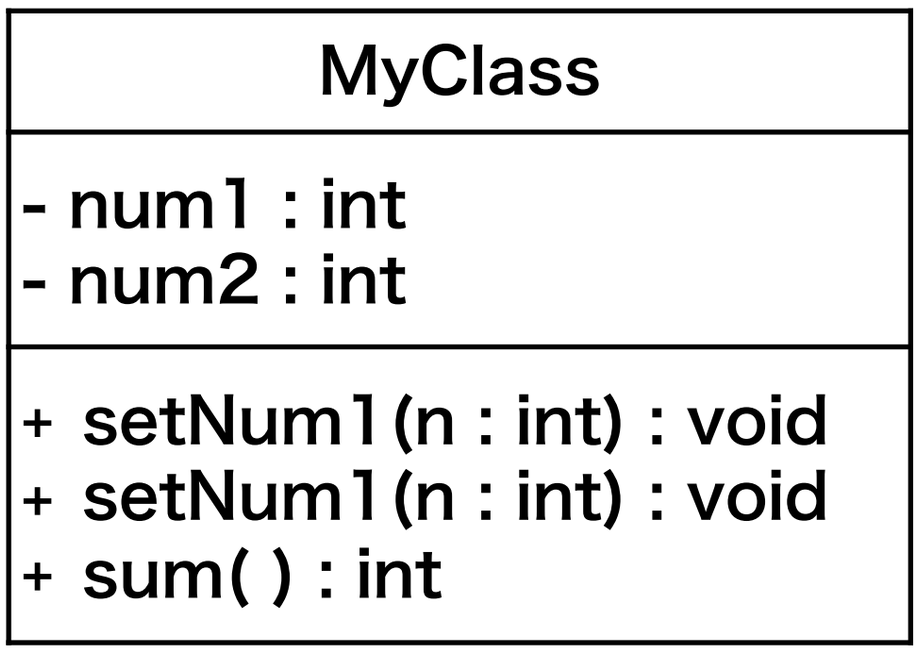
astahの参考サイト

- UML初学者向けチュートリアル
  http://astah.change-vision.com/ja/tutorial/tutorial-community.html
- 基本操作ガイド
  http://astah.change-vision.com/ja/files/astah_Basic_Operation_Guide.pdf

【練習13-1】

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 MyClass {
  private int num1;
  private int num2;
  public void setNum1(int n) {}  // num1の値を設定する
  public void setNum2(int n) {}  // num2の値を設定する
  public int sum() {}            // num1+num2を返す
}
```



【練習13-2】

練習13-1で作ったクラス図から、次の手順でJavaのスケルトンコードを生成してみましょう。

「ツール」メニュー
→ 「Java」
→ 「スケルトンコードの作成」
→ 保存先フォルダを選ぶ

【練習13-3】

資料と一緒に配付されているJavaプログラム「sample.java」を読み込んで、クラス図を作成してみましょう。

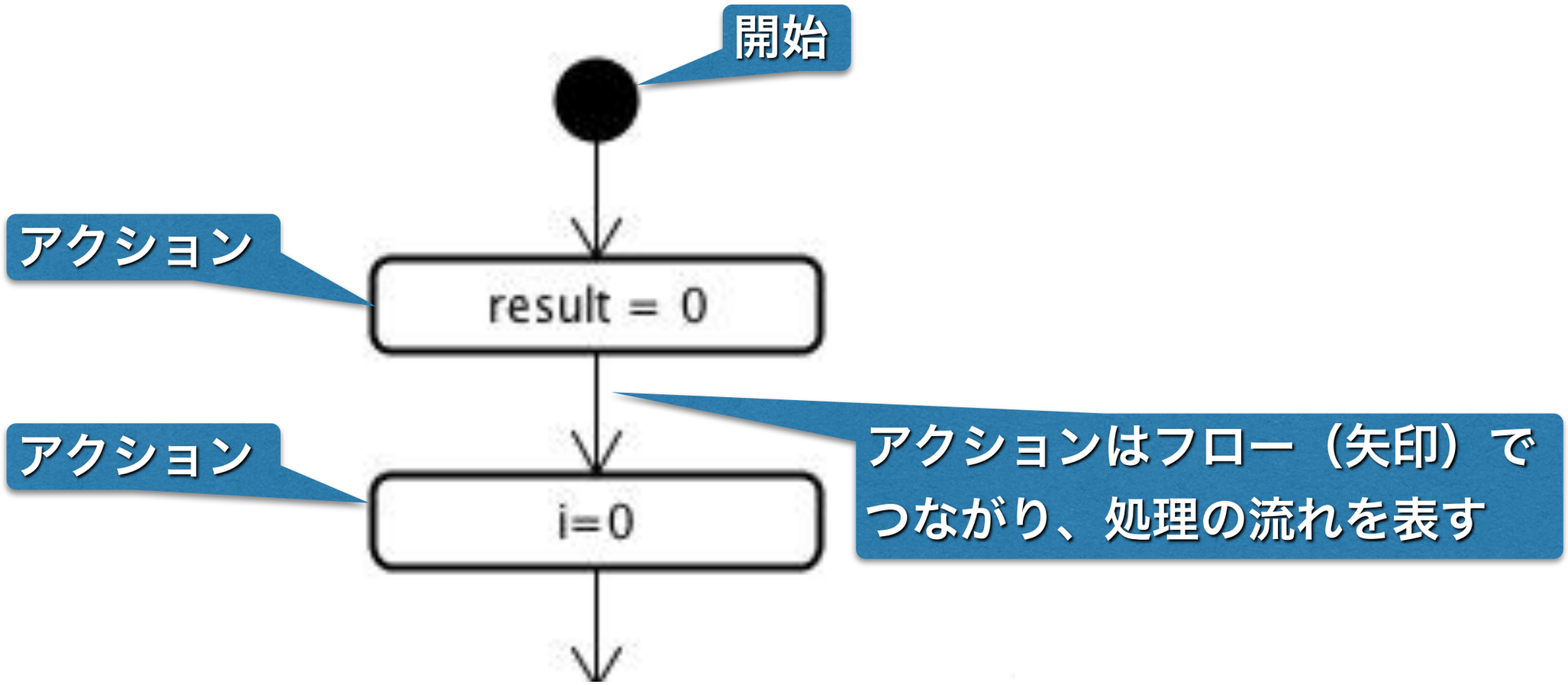
「ツール」メニュー
→ 「Java」
→ 「Javaソースコードの読み込み」
→ 読み込むJavaプログラムを選ぶ

本日は・・・

- ソフトウェア開発の現状と問題
- ソフトウェア開発工程・方法論
- UMLを用いたソフトウェア設計
  - クラス図
  - アクティビティ図

アクティビティ図：アクション

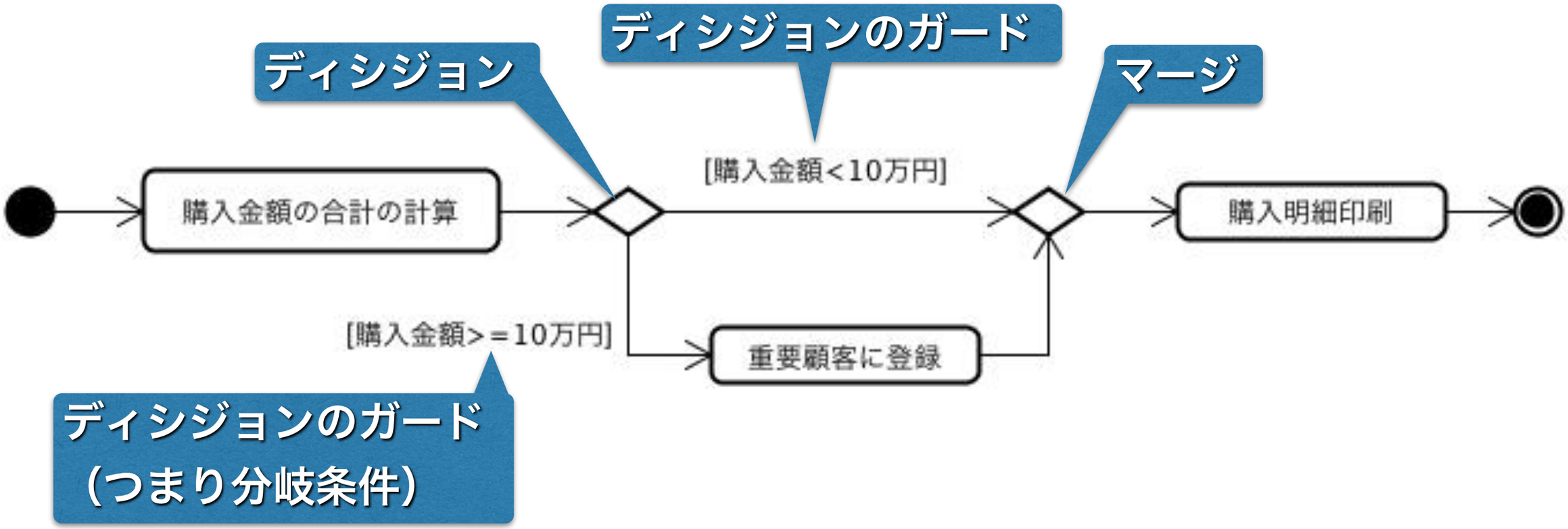
操作の呼び出しや処理をする状態を表す



アクティビティ図：ディシジョン, マージ

デシジョンはガードを付けることで処理の分岐を表す

マージは複数の処理の流れの合流を表す



【練習13-4】

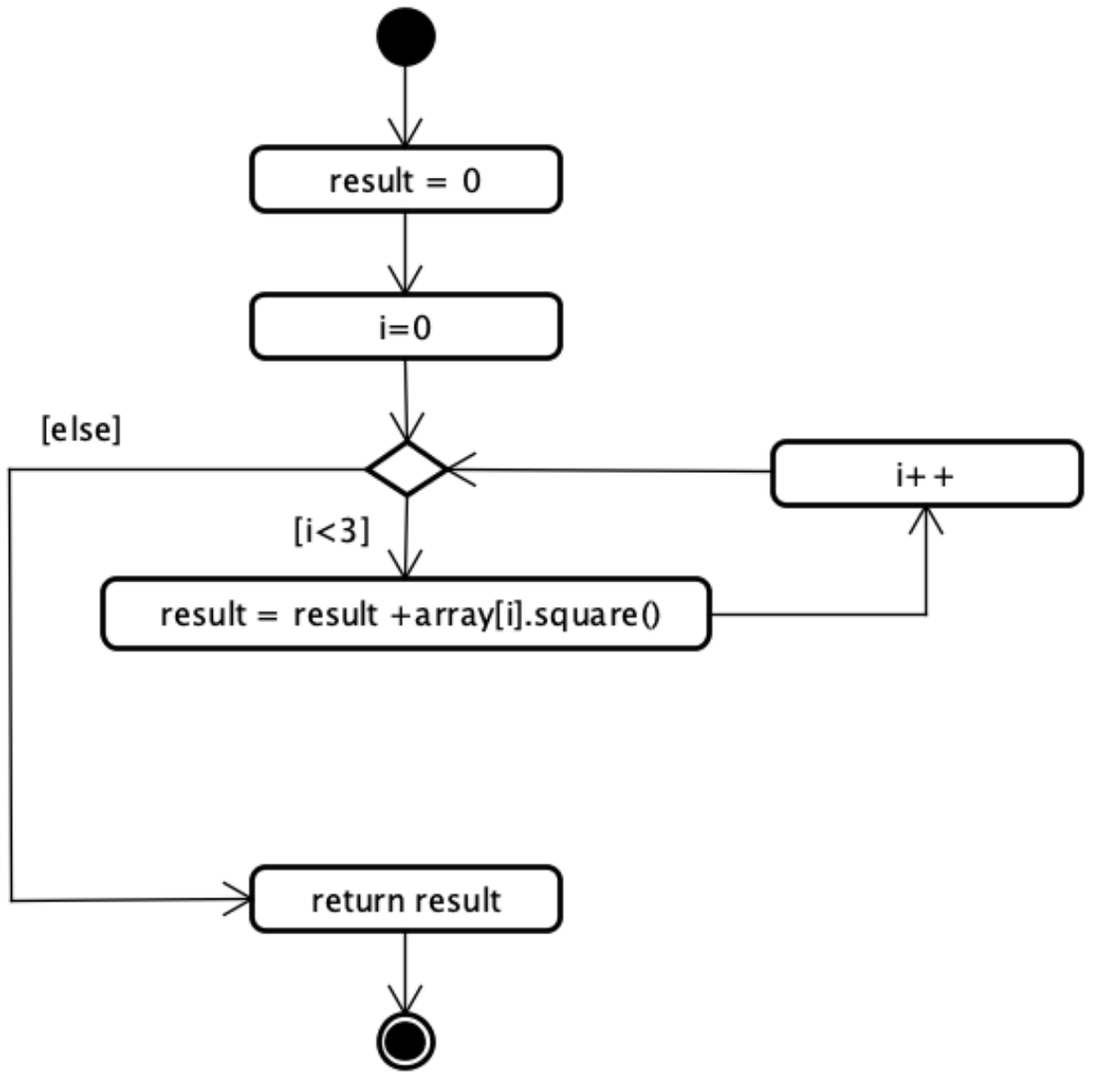
次のJavaプログラムのメソッドsumSquareArrayのアクティビティ図をastahで作成しましょう。

```
class Sum {

  //途中省略

  public int sumSquareArray() {
    int result, i;
    result = 0;
    for(i=0; i<3; i++) {
      result = result
          + array[i].square();
    }
    return result;
  }

  //途中省略
}
```



【課題13-1】

次のCプログラムの関数sum_even()のアクティビティ図をastahで作成しましょう。ただし、変数宣言はアクションとして描く必要はありません。

```
int sum_even(int n)
{
    int sum, i;
    sum = 0;
    for(i=0; i<=n; i++) {
        if(i%2==0) {
            sum += i;
        }
    }
    return sum;
}
```