プログラム設計

http://bit.ly/design4d

ステートマシン図, アクティビティ図

後期 第10週

2019/12/2

今回学ぶダイアグラム

ステートマシン図

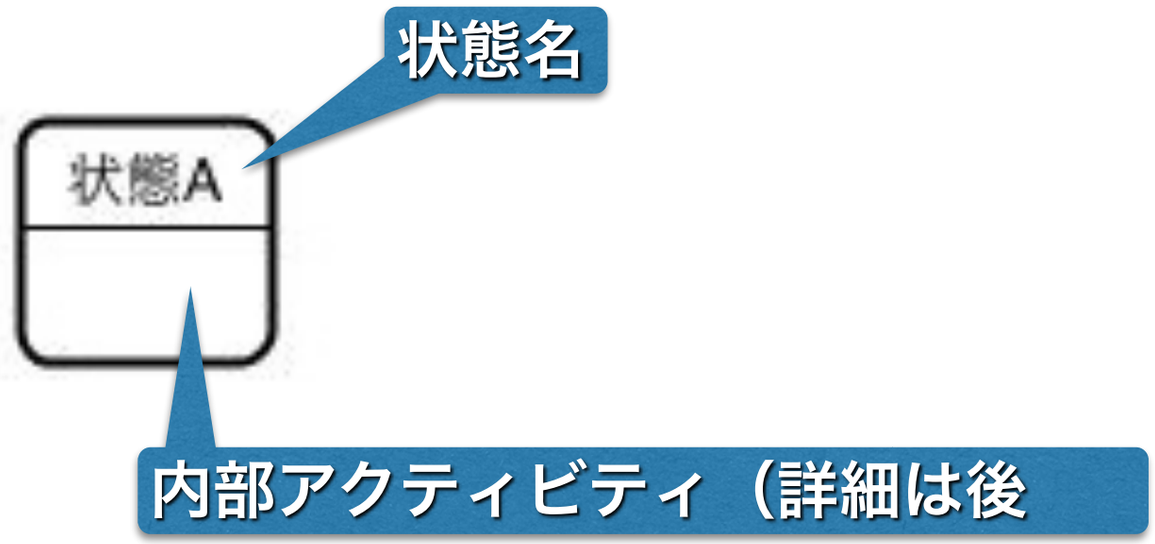
時間の経過と共に変化するオブジェクトの状態を表現する

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）

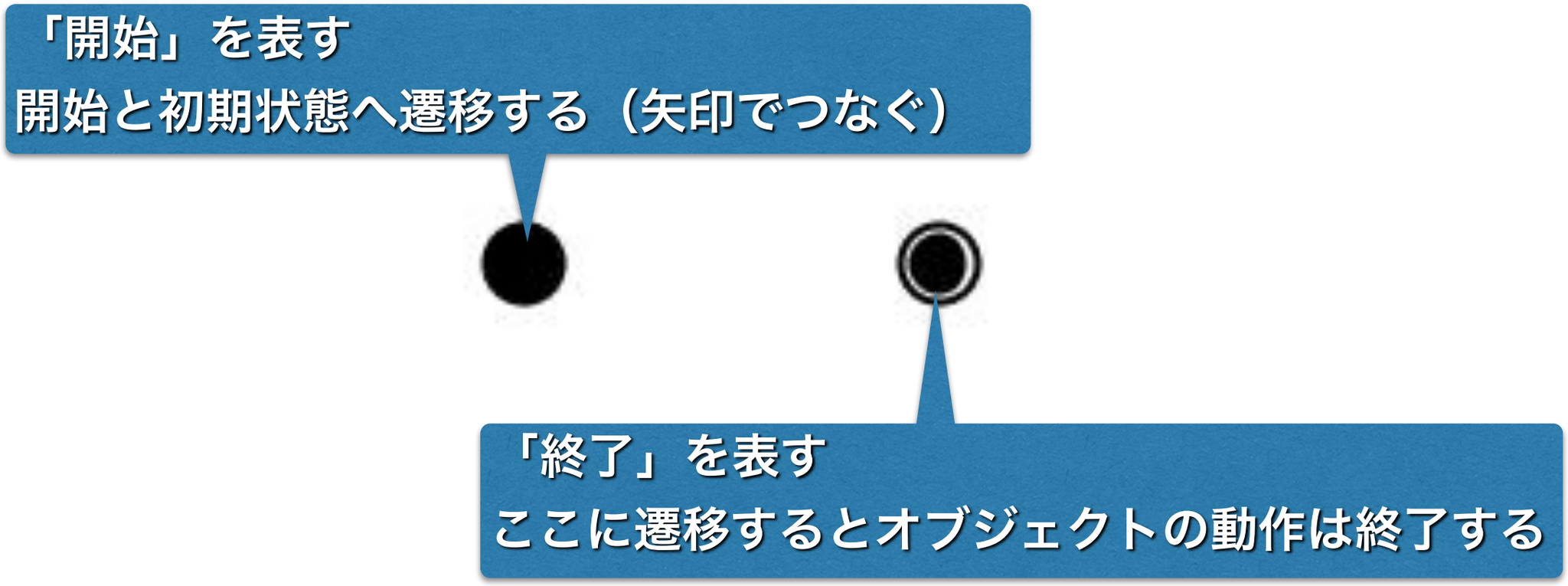
ステートマシン図：状態

オブジェクトのライフサイクル（生成から消滅まで）において、ある一定の時間とどまる状態・状況を表す



ステートマシン図：開始状態, 終了状態

オブジェクトのライフサイクルの開始と終了を表す

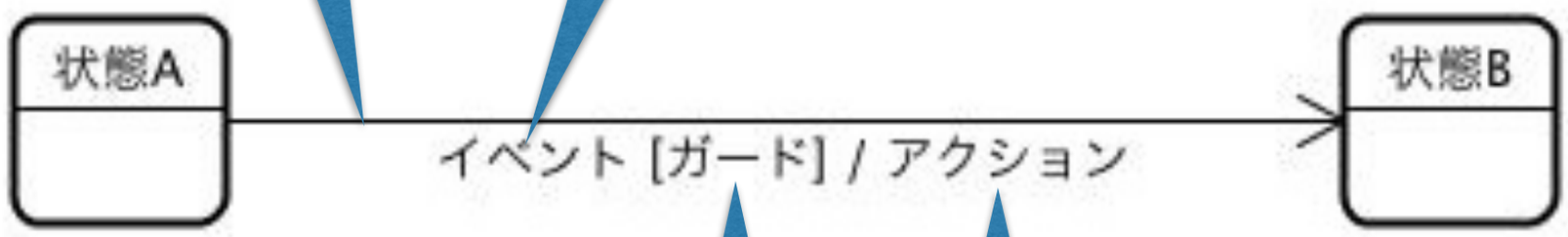


ステートマシン図：遷移

ある状態から状態への移行を表す

状態Aから状態Bへ遷移することを表す矢印

イベント（トリガシグニチャ）は、状態の変化を引き起こす可能性のあるイベントを表す



ガードは、遷移が発生するために真でなければならない真偽条件を表す

アクション（アクティビティ）は、遷移時に実行されるいくつかの振る舞いを表す

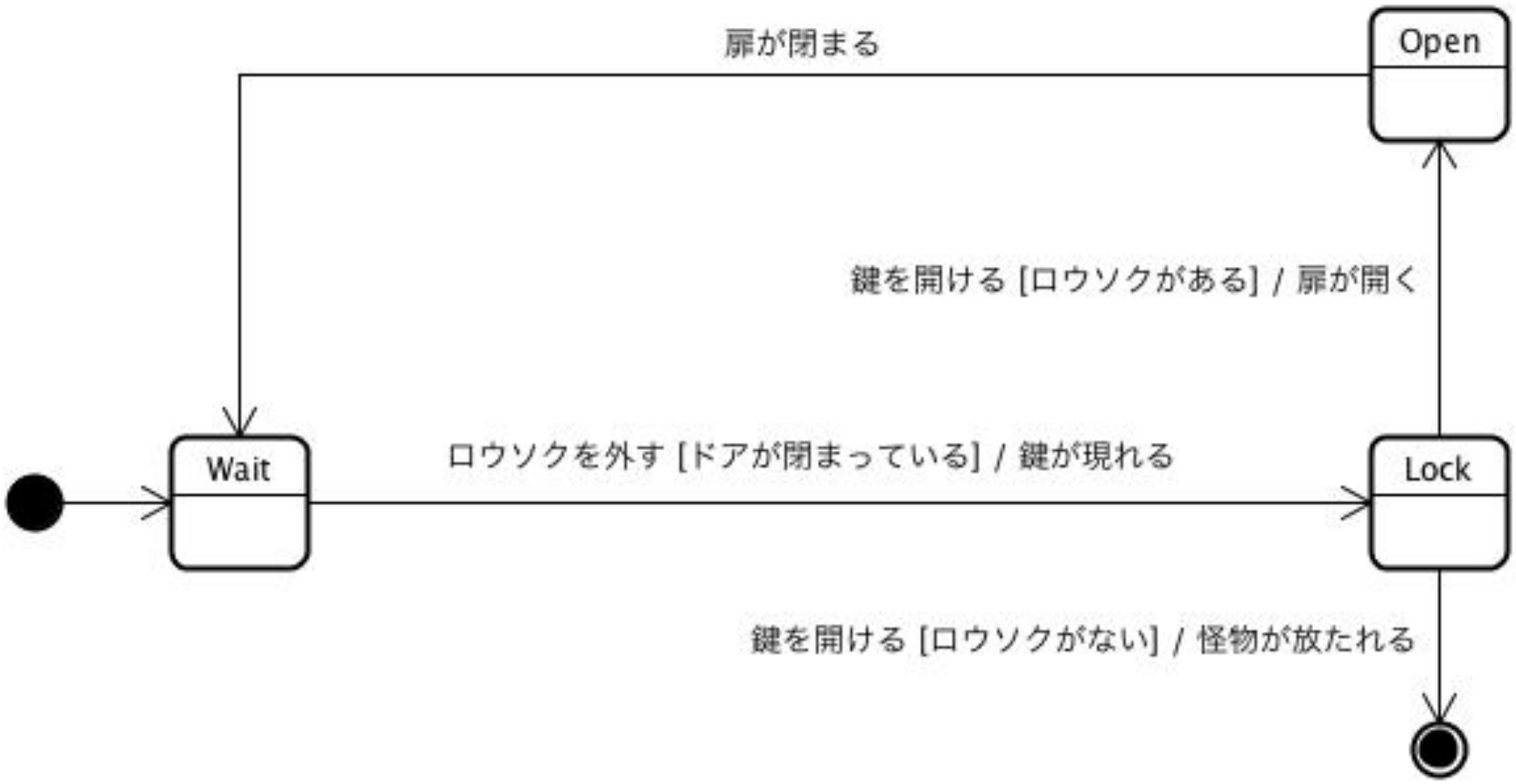
イベント, ガード, アクションの表記はどれも省略可能

ステートマシン図の例

（例1）金庫の錠は隠されており、錠を表に出すためには仕掛けになっているローソクを燭台から外す必要がある。ただし、そのときドアが閉じていることが必要。錠が見えたら、鍵を差し込んで金庫を開けることができる。より安全にするため、最初にローソクを元の位置に戻したときだけ、金庫を開けられる。そうでない場合は、怪物に食べられてしまう。

※「UMLモデリングのエッセンス」より

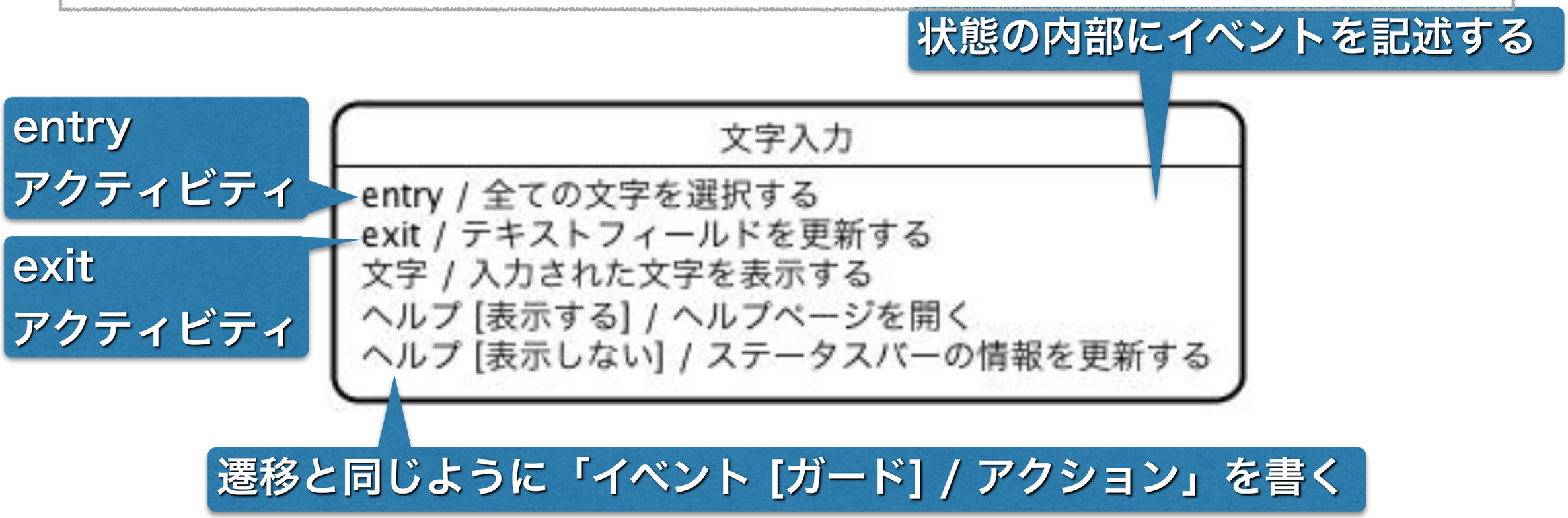
ステートマシン図の例


扉が閉まる
Open
鍵を開ける [ロウソクがある] / 扉が開く
Wait
ロウソクを外す [ドアが閉まっている] / 鍵が現れる
Lock
鍵を開ける [ロウソクがない] / 怪物が放たれる

ステートマシン図：内部アクティビティ

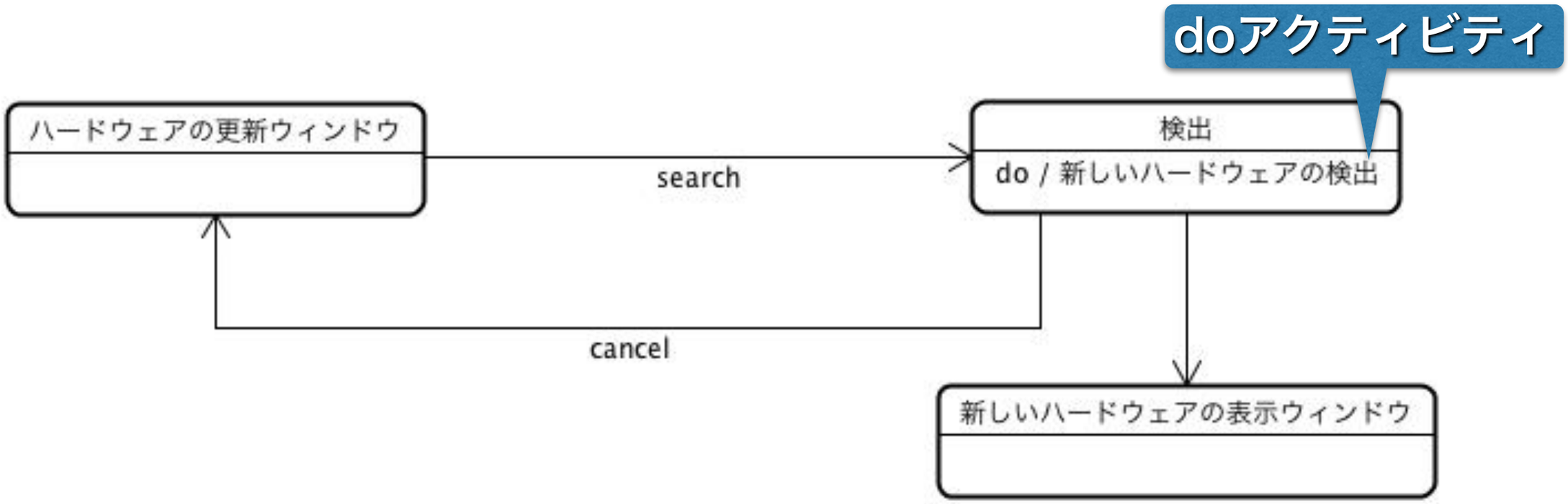
状態が遷移しない（つまり、同じ状態に留まる）場合のイベントを表現する

（例2）テキストフィールドに文字を入力する（ヘルプ機能付き）



ステートマシン図：内部アクティビティ

（例3）コンピュータに接続されるハードウェアの情報が表示されてたウィンドで、新しいハードウェアを検出して、検出された際にその情報を表示する



entryアクティビティ	・・・ この状態に入るときに必ず実行される
exitアクティビティ	・・・ この状態から出るときに必ず実行される
doアクティビティ	・・・ この状態である間は進行中である

astahの補足

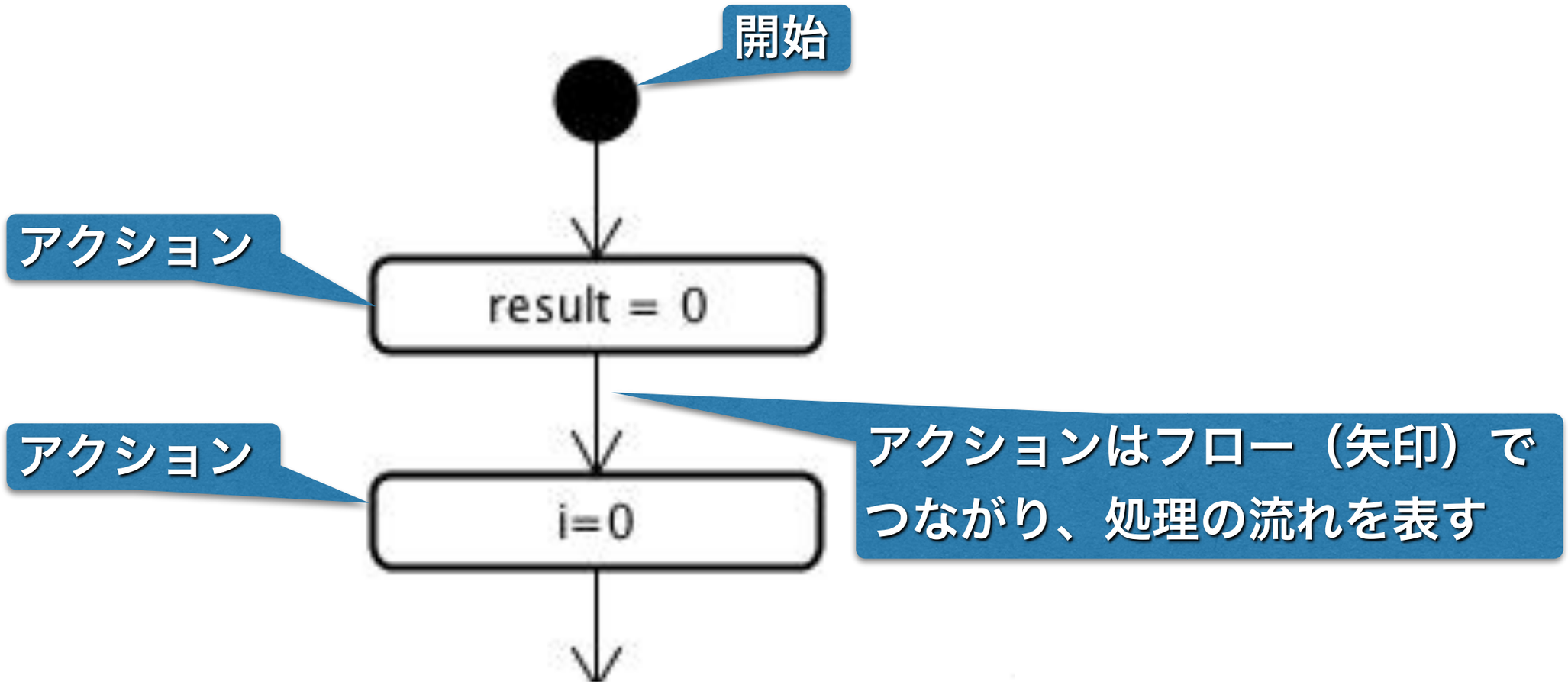
内部アクティビティの入力方法について

- entry, do, exitのアクティビティの入力は、入力する状態を選択して表示される画面左の「入場/実行/退場」で入力できる

- その他の内部アクティビティは、「内部遷移」で入力できる

アクティビティ図：アクション

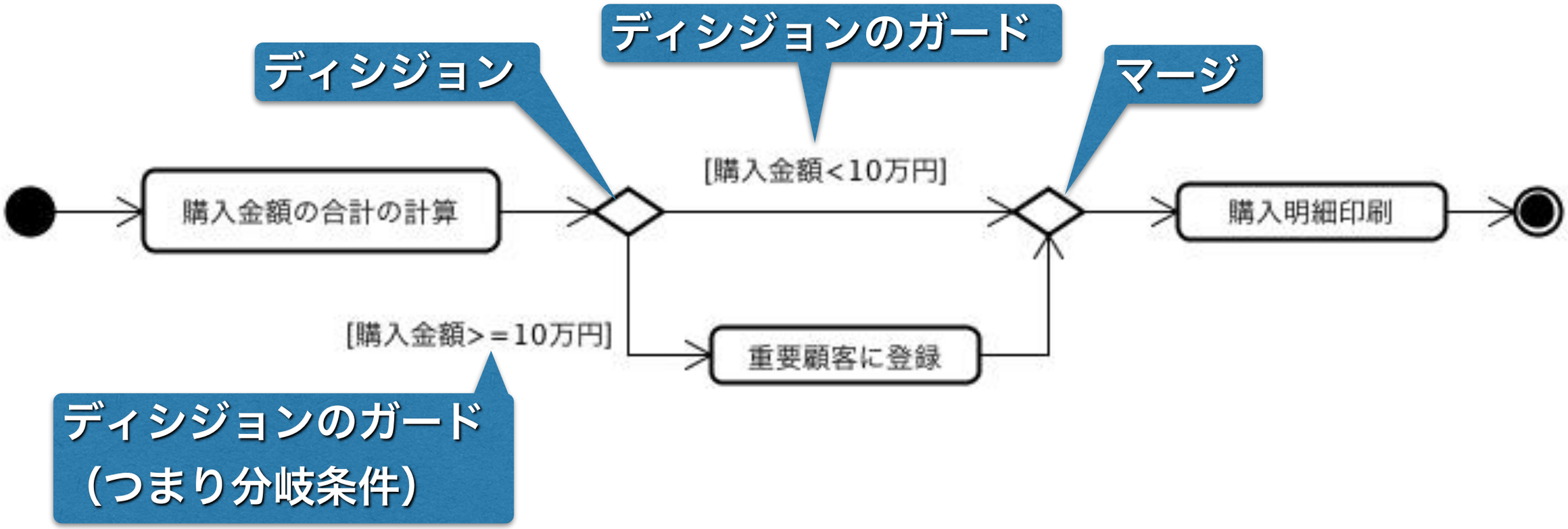
操作の呼び出しや処理をする状態を表す



アクティビティ図：ディシジョン, マージ

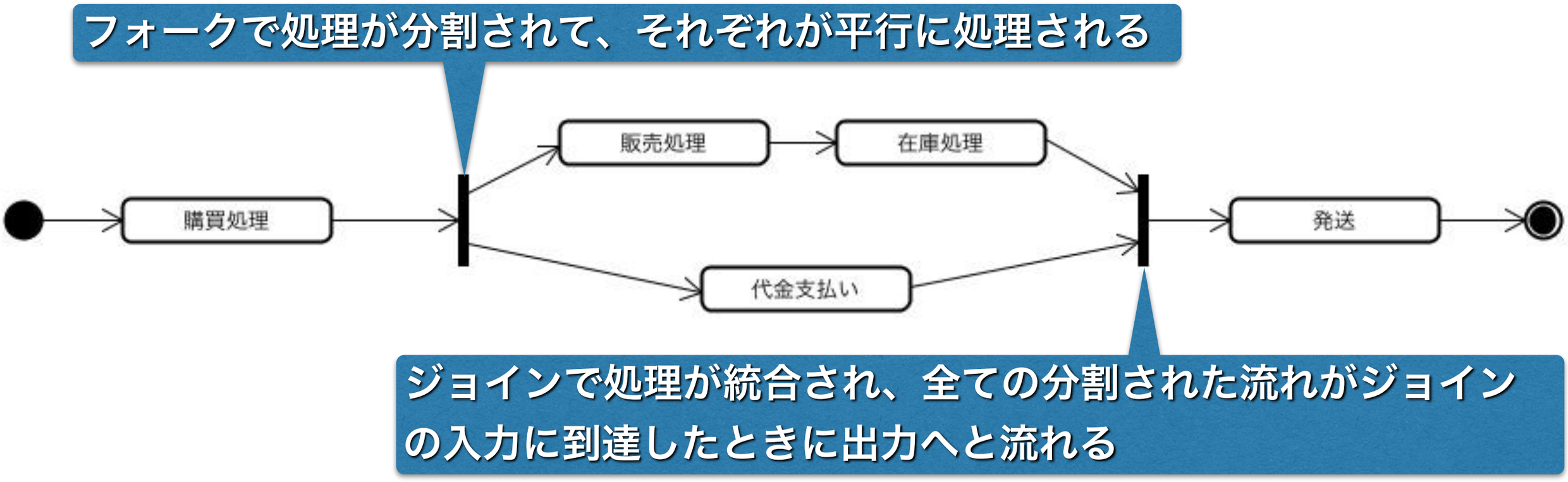
デシジョンはガードを付けることで処理の分岐を表す

マージは複数の処理の流れの合流を表す



アクティビティ図：フォーク, ジョイン

1つの処理の流れが2つ以上の制御の流れに分割されることを表す（分割された流れはそれぞれ独立して、並行に処理される）



【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10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 week210
$ cd week210
```

※課題で作るファイル名は各自で決めて構いません。

【練習10-1】

astahを使って、ステートマシン図のスライドで示した例1～例3の図を作ってみましょう。

【練習10-2】

astahを使って、アクティビティ図のスライドで示した2つの例（デシジョン・マージとフォーク・ジョイン）の図を作ってみましょう。

【課題10-1】

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

```
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10-1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10-2】

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

```
class Cell {

  //途中省略

  public void show() {
    System.out.printf("(%s)", type);
    if (type == "int") {
      System.out.printf("%d", valueI);
    } else if (type == "String") {
      System.out.printf("%s", valueS);
    } else if (type == "Date") {
      valueD.show();
    }
  }

  //途中省略

}
```

【課題10-2（補足）】

if～elseの処理は次のようなデシジョンを使ってみましょう

ガードに「else」と表記してもよいし、
trueとなる条件の否定を描いてもよい

【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10-2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)

【課題10-3】

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

```
class FunctionRange {

  //途中省略

  public void sumRange(Cell[] array) {
    int i;
    name = "sum";
    result = 0;
    for (i = start; i <= end; i++) {
      if (array[i].getType() == "int") {
        result = result + array[i].getValueI();
      }
    }
  }

  //途中省略

}
```

【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10-3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)

小テストについて

小テストの注意点

- **他人の力は借りずに、自分だけでプログラムを作成する。（つまり定期試験と同様）**

- **プログラムの提出はGitHubを使用する。**

小テストについて

小テスト中に参照できるもの

□ **教科書, 参考書, 配付資料**

□ **自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル**

□ **小テストでは紙媒体のものは参照可能**

□ **上記以外の情報を参照することは不正行為とする**

例：USBで接続された機器に保存されているファイルの参照
Webブラウザ、ネットワークを介した情報の参照
自分のPCを使用する、など