

製造工程

ずっと受けたかったソフトウェアエンジニアリングの
新人研修

60206052

岡坂亮汰

目次

- プログラミングについて
- ソースコードレビューについて
- 単体テストについて
- まとめ

製造工程

- ・ プログラムを作る工程



製造工程の作業手順

プログラミング

- プログラムを書く作業
- 関数や手続きごとに作成
- 内部設計書の内容に合致するように記述

ソースコードレビュー

- プログラムのソースコードを読み, 内容を確認
- 次の2点を確認
 - 規約に沿って書いてあるかどうか
 - 正しいロジックで書かれているかどうか

コーディング規約

- プログラムの書き方を定めたもの
 - 名前の付け方や書き表し方のルール
- 記述形式が統一され、プログラムの保守性が高まる

実際のコーディング規約

- プログラムの冒頭に付けるコメントの記述ルール
- 変数名の付け方や宣言方法に関するルール
- プログラムロジックの記述ルール
- 変数の型のルール

コーディング規約の例

- Javaのコーディング規約の例

<処理に関するコメント>

○ `dataCount = 0 /* データ数カウンタを初期化する */`

✗ `dataCount = 0 /* dataCountを0にセットする */`

コーディング規約の例

<クラスに関するコメント>

Javadocが規定する書式

```
/**  
 * クラスの説明（日本語）  
 * @author 作成者（半角英字）  
 * @author 修正者（半角英字）  
 * @version バージョン番号（半角英数字）  
 */
```

コーディング規約の例

＜変数名の付け方＞

- ・ 変数名に使用する文字は、半角英字、数字、記号 `_` (アンダーバー)、記号 `-` (マイナス)とする
- ・ 変数名はその変数の持つ意味を英語で記述

○	<code>inputRecordCounter</code>
×	<code>input_record_counter</code>
×	<code>nyuuryokuRekoodoKaunta</code>

コーディング規約の例

- ・変数名の冒頭文字は英小文字にする

○ firstField

✕ 1stField

✕ _1st_field

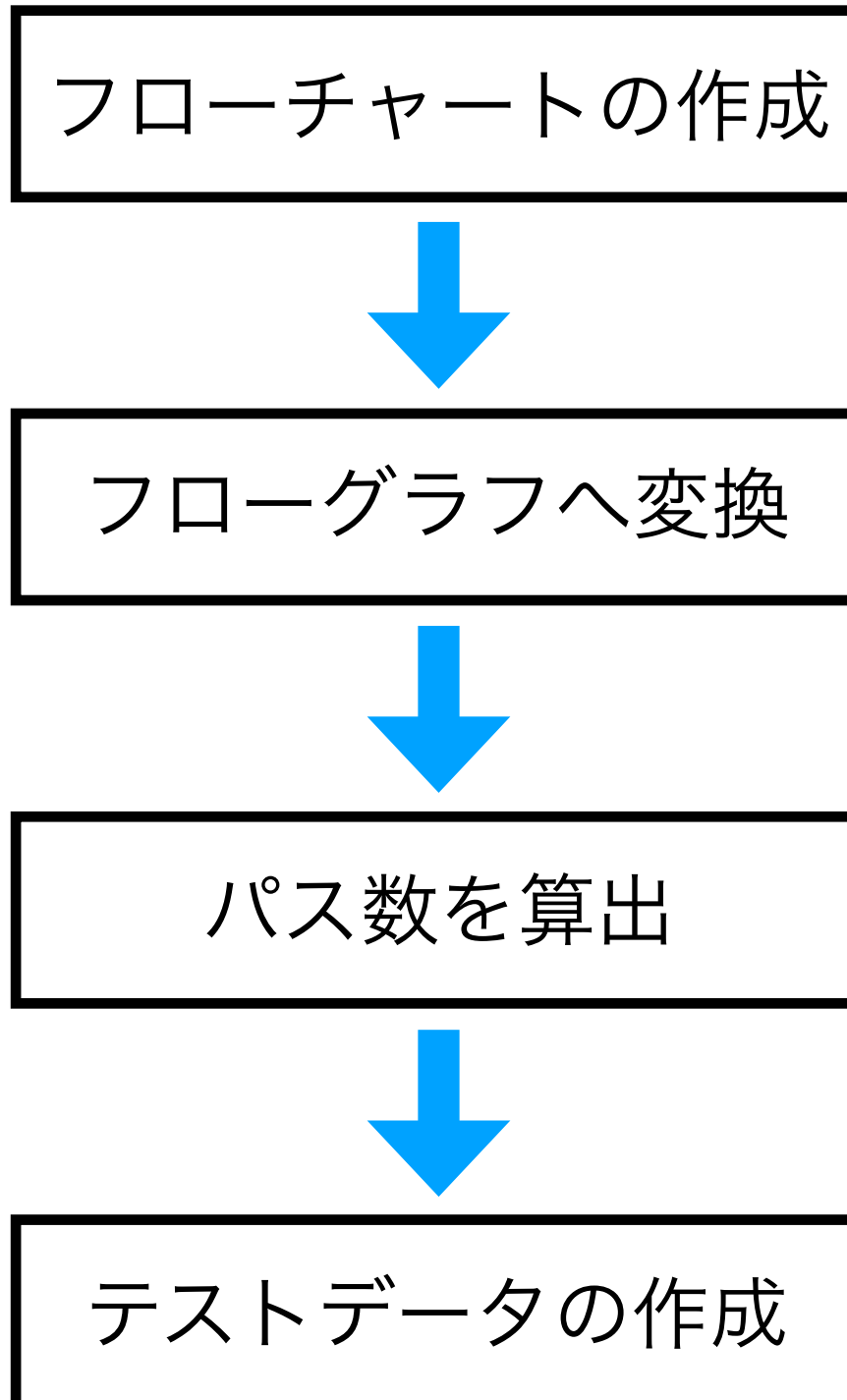
ソースコードレビューのコツ

- あらかじめコーディング規約が守られているかどうかを確認するためのチェックリストを作成
- 問題があった箇所を記録しておく

単体テスト

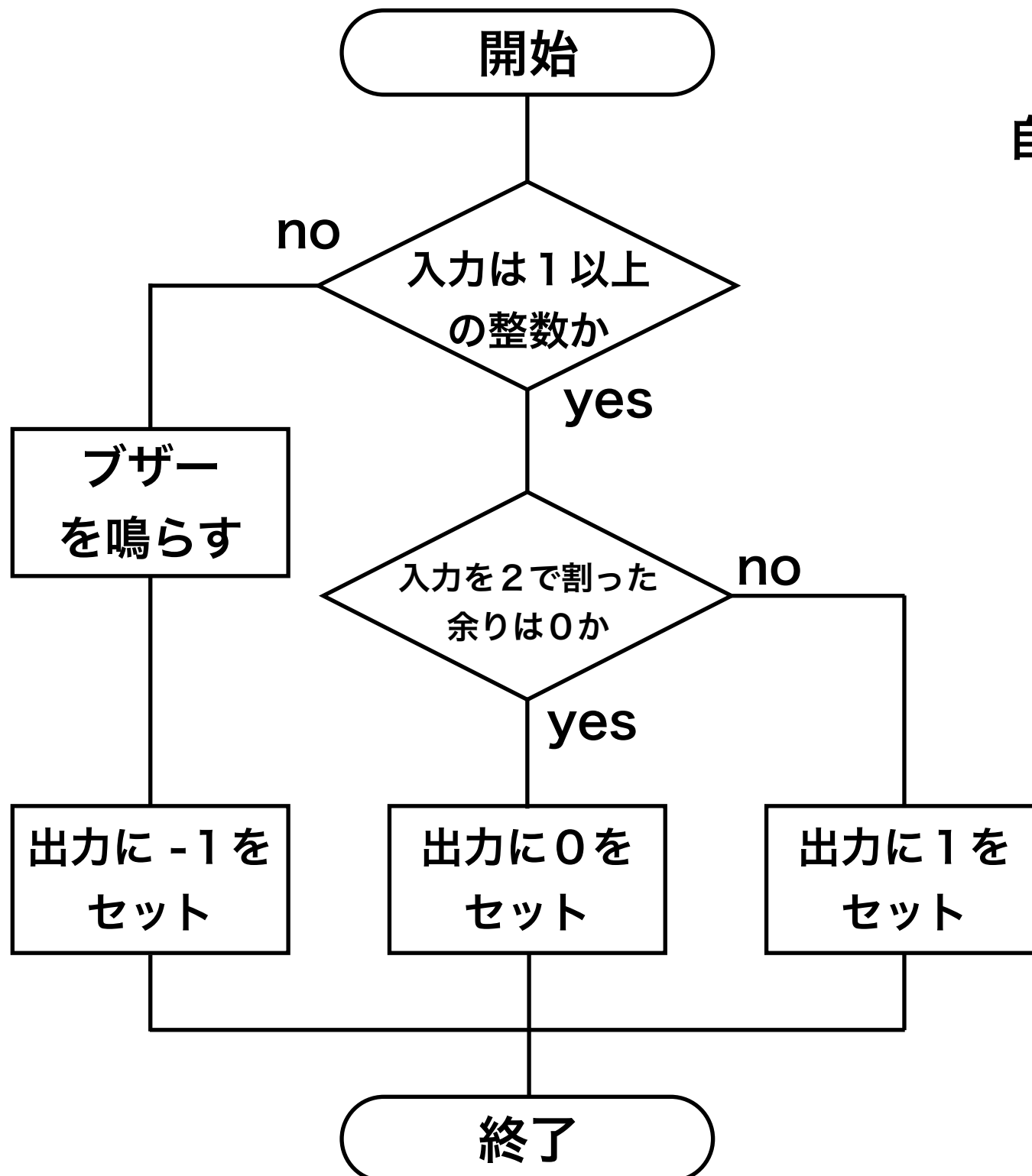
- ・ 最小単位のプログラムを単体で動かし, 内部設計書どおりに動作するか確かめる工程
 - ホワイトボックステスト
 - ブラックボックステスト

ホワイトボックステスト



フローチャート

自然数の偶数か奇数かを判別するプログラム

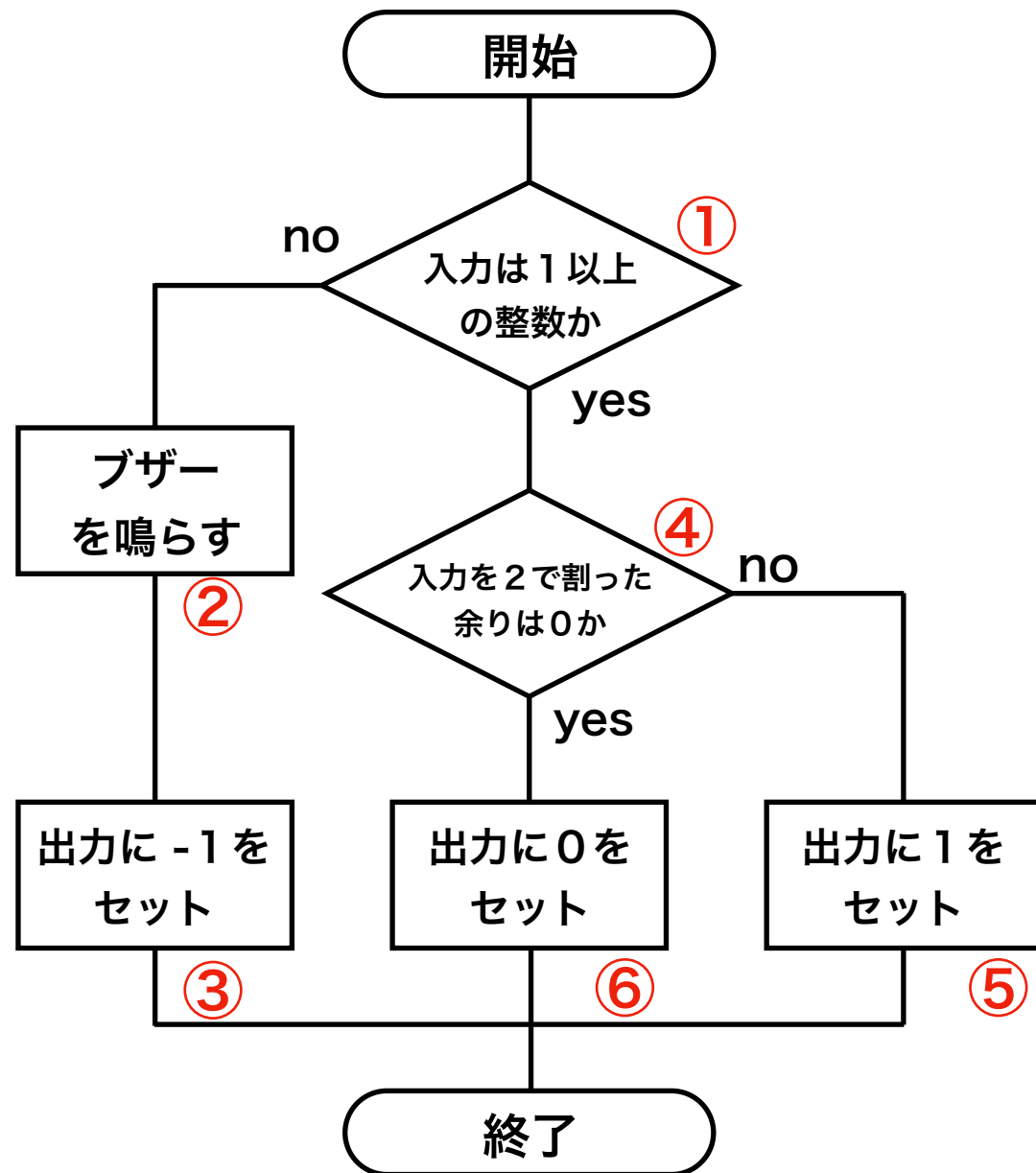


入力：1つの整数

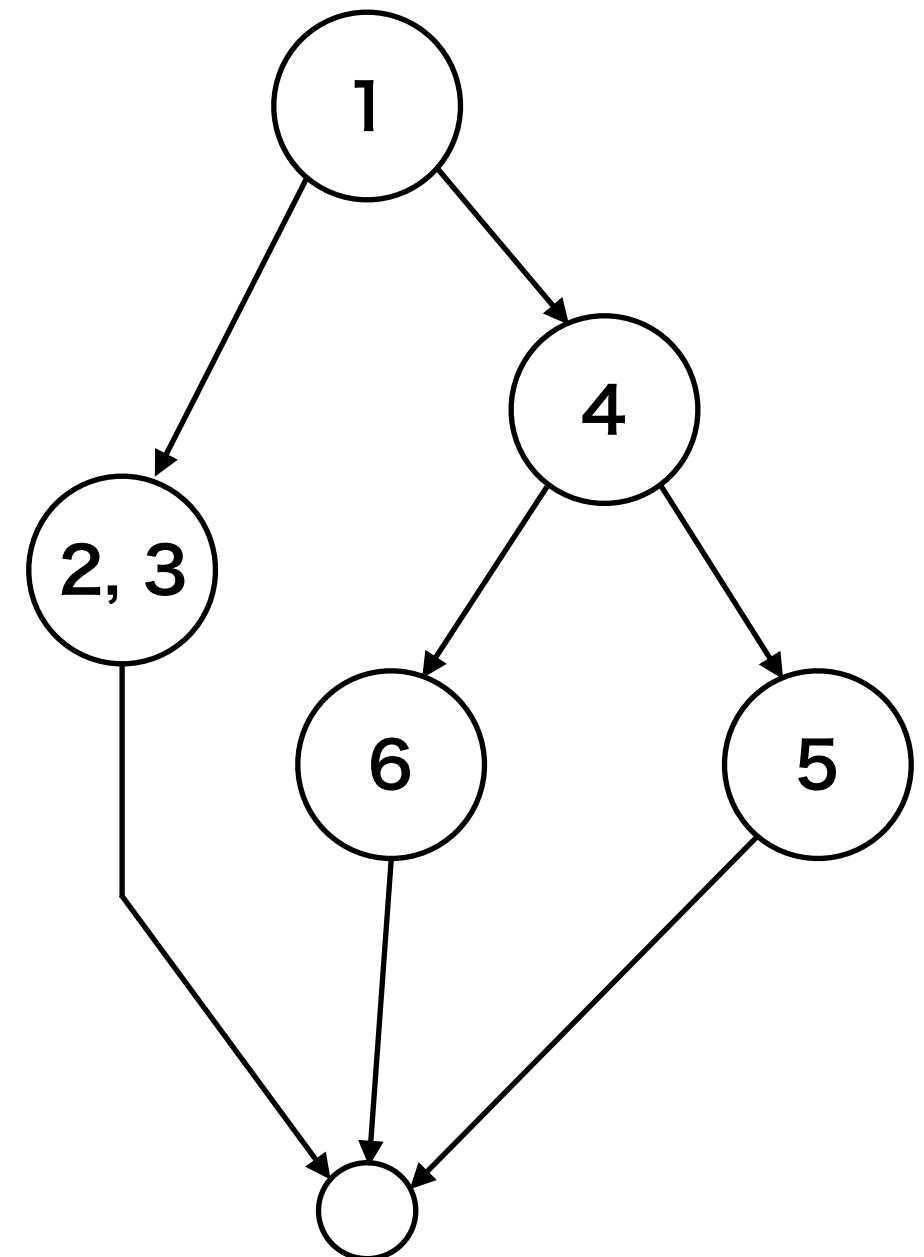
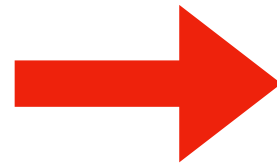
動作：自然数なら何も鳴らさない
自然数でなければブザーを鳴らす

出力：偶数なら0
奇数なら1
自然数でなければ -1

フローグラフ

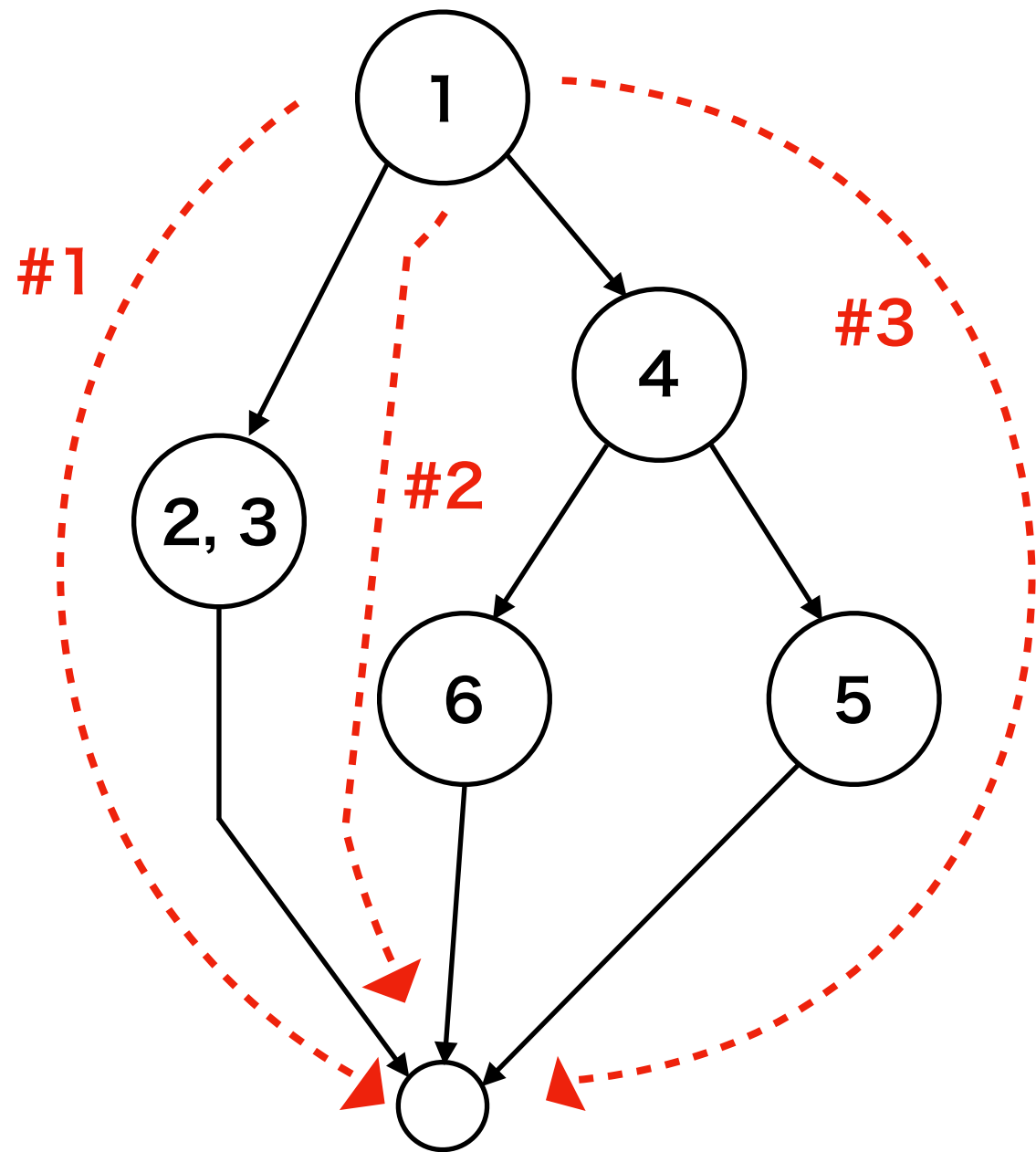


フローチャート



フローグラフ

パス数とテストデータ



フローグラフ

$$\begin{aligned}\text{パスの数} &= \text{エッジ数} - \text{ノード数} + 2 \\ &= 7 - 6 + 2 \\ &= 3\end{aligned}$$

テストデータ	通るパス
0	#1
2	#2
1	#3

テスト用の仮プログラム

- 各プログラムは最終的には他のプログラムと組み合わせて使用する

→ プログラム単体でのテストは難しい

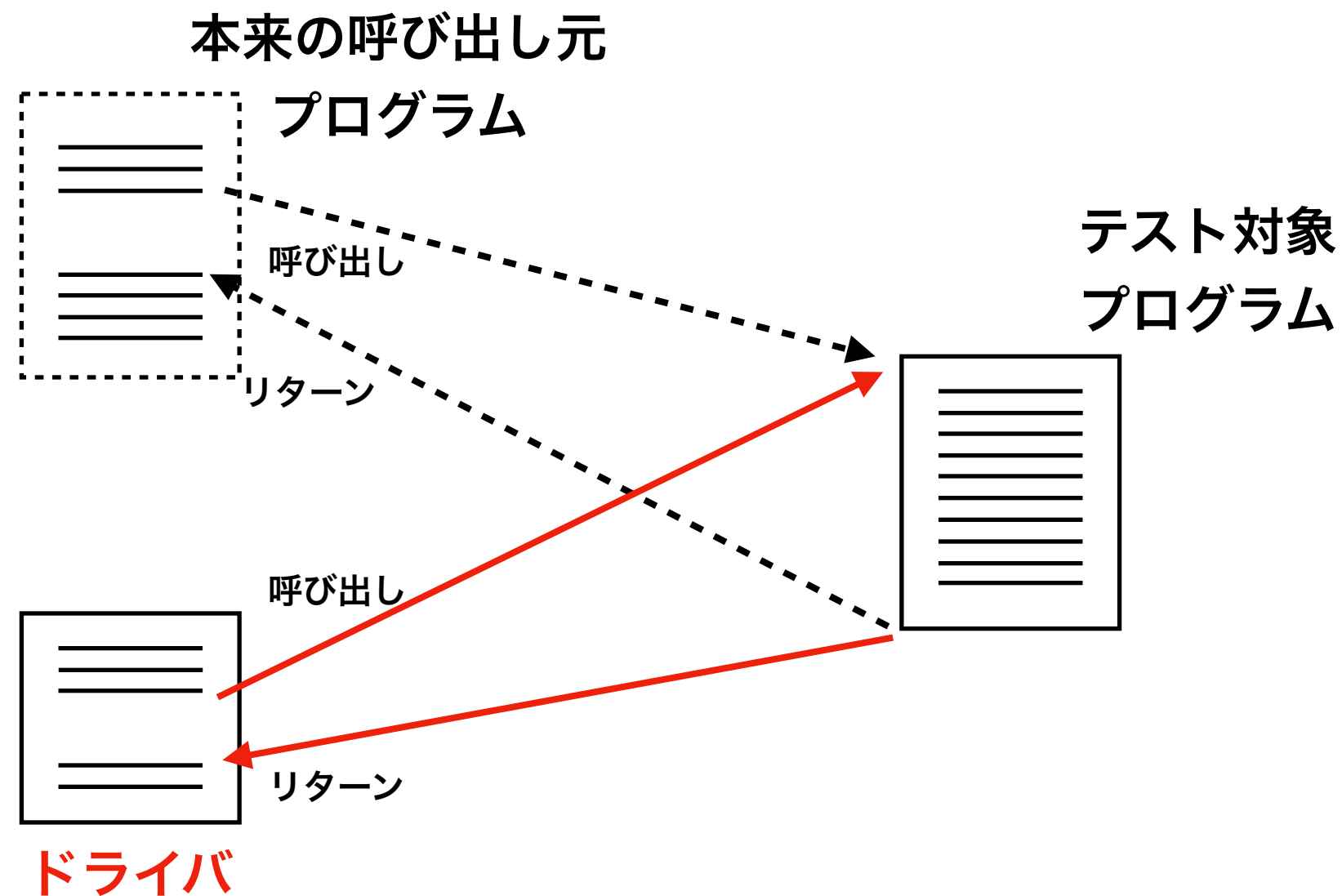
- テスト専用の仮のプログラムを使用

- ドライバ

- スタブ

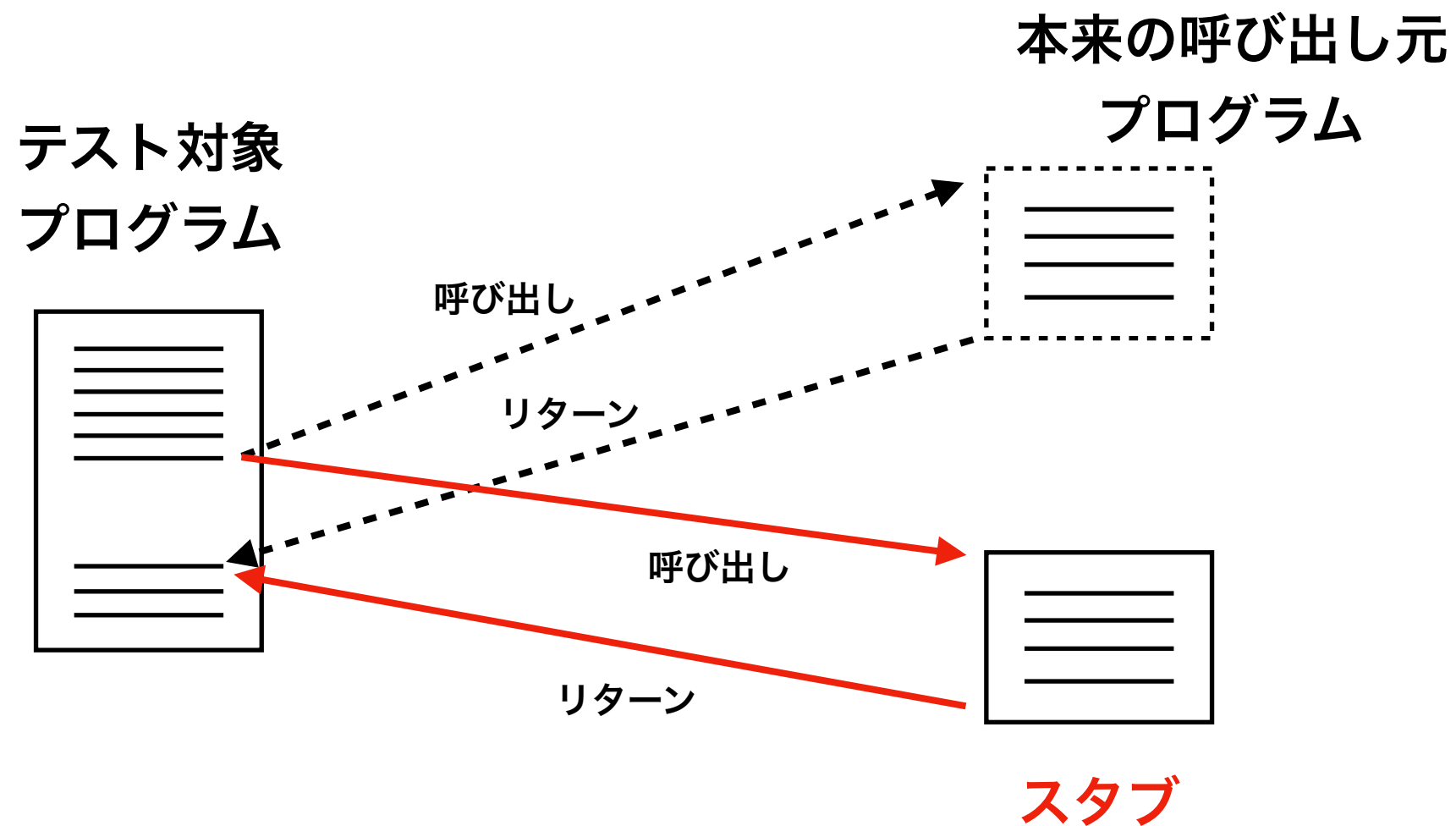
ドライバ

- ・テスト対象のプログラムが他のプログラムから呼び出される場合に使用



スタブ

- ・テスト対象のプログラムが他のプログラムを呼び出す場合に使用



まとめ

- 製造工程ではプログラミング, ソースコードレビュー, 単体テストを行う
- ソースコードレビューを行う前に, コーディング規約を定める
- 単体テストにはドライバやスタブを使用する