TRAINING

Claude Code 活用研修 Day2

生成テストの観点とレガシーコードの整理

2026年9月9日（水）

株式会社インフォメーション・ディベロプメント 様

講師 安田 光喜（Givery）

Givery

Day2 テストで裏を取る

本資料の二次転載禁止

本資料の著作権は株式会社ギブリーに帰属します。

受講者ご本人が学習に使う範囲を超えて、複製や再配布、

社内外への二次転載を行うことはご遠慮ください。

社内での共有や引用をご希望の場合は、講師までご相談ください。

本資料に含まれる図表やスクリーンショットも同じ扱いになります。

講師紹介

人的資本インテリジェンス部門講師
生成AIエバンジェリスト

安田 光喜

Mitsuki Yasuda

プロフィール

公立はこだて未来大学 大学院（メディアデザイン領域）を卒業し、街づくり会社での複合商業ビルの立ち上げにおける情報インフラ整備やデザイン全般業務の責任として関与。その後デザイン事務所を立ち上げ、飲食店を中心としたデザイン業務や美術館展示のアートコンテンツ開発などを行う。また、地域ブランディングにも注力し、市主催の企画運営業務を行なった。2018年10月『小中学生向けのプログラミング教育』に注目し、函館市で初めての小中学生向けプログラミングスクール『D-SCHOOL北海道』を立ち上げて運営。その後北海道のドラッグストア『サッポロドラッグストアー』からスクールの買収を受け、教育を専門とするグループ会社『株式会社シーラクンス』にて技術開発責任者として運営。スクール設計・運営、20以上の自治体連携（イベントや出張教室運営）、大人向けプログラミングスクールメンターなどさまざまな『次世代IT教育』に取り組む。Dify公式資料の日本語翻訳や、SecondMeアンバサダーなど生成AI最新トレンドに注力。

得意領域

- 生成AIをはじめとした世界最先端トレンド
- 生成AIを用いた新規事業の開発、運営、補助
- アプリケーション開発/自動化プログラム環境構築
- 教育（大学講師、研修、スクール運営など）

実績

- 生成AI研修
- 大手企業、外資、教育機関（小中高大）でのITトレンド講演
- 自治体と連携した地域教育実績
- 上場企業社員対象のリスキリング支援

メッセージ

みなさんがこれから生成AIをパートナーに新しい時代を生きていけるように、3日間全力でサポートしていきます！
なにかご不明点などありましたら、講師陣になんでも気軽にご質問ください。
#ClaudeCode #Codex #Cursor #Antigravity #全国統一生成AI活用技能試験S1　#生成AIパスポート　#G検定

3日間の役割分担

Day2 は書いたコードを画面以外で確かめる日

Day1	Day2	Day3
書く	**裏を取る**	**壊れたときに効く動き**
実装を AI に任せる 画面に出たら成功	テストと段階的整理 画面に出ないものを確かめる	トレース・ログ・レビュー

> 本日は真ん中です。Day1 で書いたコードが仕様どおりかを、画面以外の手段で確かめます。

本日の到達点

達成の判定は、件数と差分で機械的に確認

	到達点	見える成功
1	生成させたテストの assert・例外型・verify・境界値を見て、採否を自分で決められる	**4件** 異常系のテストが4件以上 自分が書いたものが緑
2	バグをテストで先に赤にしてから直せる	**3件** 先に赤が出て、 直したあとに緑
3	309行のレガシーコードを、出力を変えずに段階を区切って整理できる	**309行** 出力日の行以外で差分なし

本日の流れ

前半は生成テストの観点・後半は段階を区切った整理

区分	番号	内容	時間
前半	**振り返り**	Day1 の実装が入った層	[10min]
前半	**D2-1**	OrderServiceImplTest に観点を足す	[35min]
前半	**D2-2**	BuggyOrderCalculator のバグをテストで暴く	[35min]
	休憩		[10min]
後半	**D2-3**	LegacyReportService の段階的な整理	[45min]
後半	**発展枠**	D2-2+ / D2-3+、または追いつき	[15min]
後半	**まとめ**	生成テストのレビュー観点	[15min]
	質疑	質疑応答・アンケート	[15min]
合計			**[180min]**

3つの演習と見える成功

振り返り — D2-1 — D2-2 — D2-3 — まとめ

濃い3つが本日の演習です。発展枠は D2-2+ と D2-3+ の2つです。

番号	触るファイル	やること	できたら
D2-1	src/test/java/com/example/order/service/OrderServiceImplTest.java	changeStatus の TODO 4件を埋め、抜けた観点を足す	異常系のテストが4件以上 自分が書いたテストが緑
D2-2	exercises/day2/BuggyOrderCalculator.java	テストを先に書いて赤を出してから直す	赤3件が出て、修正後に緑
D2-3	exercises/day2/LegacyReportService.java	出力を変えずに段階を区切って整理	出力比較が出力日の行以外で差分なし

補足

演習の共通フロー

任せるのは下書き・採否の判断は人の側

1	2	3	4
自分で考える	下書きを生成	差分と結果	基準で採否
何を確かめたいか、期待値をどこから取るかを先に決める	Claude Code に下書きさせる	差分と実行結果を目で見る	生成物チェックリストの基準で採否を決める

> 同じ指示でも出力は毎回ぶれます。生成させて終わりにしないところが本日の中身です。

手順とプロンプト例は教材サイト idunder02.give-app.net の Day2 ページ

01

振り返り

Day1 で書いた5つが、どの層のどこに入ったかを確かめます。
今日はそのコードを、画面ではない手段で確かめにいきます。
[10min]

振り返り | D2-1 | D2-2 | D2-3 | まとめ

Day1 の実装が入った層

Day1 の成果物は controller・service・dto の3層

- **controller**
  date-range エンドポイント
- **service**
  `findByDateRange / validateStatusTransition`
- **repository**
  変更なし
- **model・dto**
  `OrderItemDTO`

- **static**
  `dashboard.html`

- Day1 で足した箇所
- Day1 では触っていない

層の外の単体クラス

`BuggyOrderCalculator.java`
`LegacyReportService.java`

exercises/day2/ に置いてあります

本日の D2-1 は service を、D2-2 と D2-3 はその外の単体クラスを見ます

Day1 で見えた成功の形

Day1 の確認は5件とも 画面と端末の表示

演習	画面や端末に出たこと
D1-1	CLAUDE.md を読んだ状態で層の説明が返った
D1-2	コンパイルが通った
D1-3	ブラウザに受注一覧の表が出た
D1-4	受注詳細に明細行が出た
D1-5	CONFIRMED の受注がキャンセルできた

5件とも、目に見えるものが出たことで確かめています

動いたと正しいの距離

画面に出た事実と 仕様どおりか は別の確認

動いた

- 画面に出た
- コンパイルが通った
- HTTP 200 が返った

この距離を
埋めるのが今日

正しい

- 仕様どおりの値か
- 条件を外れたら止まるか
- 境界のちょうどで期待どおりか

Day1 は左だけで進みました。左が真で右が偽のコードは、配布した状態にも入っています

画面の代わりに使う 3つ の確かめ方

1 **テスト**
入力と期待値を書いて、機械に確かめさせる
D2-1・D2-2

2 **出力の固定**
整理の前後で同じ結果が出ることを比べる
D2-3

3 **レビュー観点**
生成物チェックリストの基準に照らす
全演習

本日は目でなく、基準で確かめます

クイズ

生成直後のテストの結果

Q. @OrderServiceImpl.java を読ませて、テストを生成させました。
これを実行すると、どうなる可能性が高いでしょうか。

- **A** 半分ほどのテストが落ちる
- **B** 異常系のテストだけが落ちる
- **C** ほとんどのテストが通る
- **D** コンパイルが通らない

正解は次のページで確認します。まず自分で考えてみてください。

解説

実装から書いたテストの性質

正解はC。実装の現在の振る舞いが期待値

C **正解 ほとんどのテストが通る**
実装を読んで書いたテストは、実装の現在の振る舞いをそのまま期待値にします

A 半分ほどのテストが落ちる
実装から起こした期待値なので、そのままでは落ちません

B 異常系のテストだけが落ちる
異常系はそもそも生成されにくく、落ちる以前に数がありません

D コンパイルが通らない
モックの設定ごと生成されるので、コンパイルは通ります

仕様と実装がずれている箇所では、テストも同じようにずれます
calculateDiscount(10, 50000) の期待値は、仕様なら 475,000 円、実装なら 500,000 円

出典 exercises/day2/exercise1-テスト自動生成.md

02

D2-1 生成テストへの観点追加

テストは Claude Code に書かせます。何を確かめるかは人が決めます。
抜けやすいのは null・空・境界値・不正ステータス・存在しないID の5つです。
[35min]

振り返り | **D2-1** | D2-2 | D2-3 | まとめ

テスト生成の守備範囲

埋まるのは骨組み、決めるのは観点

Claude Code が埋めるところ		人が決めるところ
@Test の骨組み	→	どの入力を試すか
モックの when 設定	→	期待値を仕様のどこから取るか
assert の書式	→	異常系で何の例外まで見るか
テスト名の付け方	→	落ちたテストを直すか実装を直すか

量は任せられます。観点は任せられません。

OrderServiceImplTest の現状

TODO 10ケースのうち、4件が今日の範囲

240行
総行数

9本
@Test の本数

10ケース
TODO の未実装ケース

ブロック	件数	内容
createOrder	3件	数量0のバリデーションエラー・単価が負・納品予定日が過去
changeStatus	4件	CONFIRMED→SHIPPED・SHIPPED→DELIVERED CANCELLED からの遷移・同一ステータスへの遷移
cancelOrder	3件	CONFIRMED のキャンセル・CANCELLED の再キャンセル・存在しないID

濃い行が D2-1 の対象です。

src/test/java/com/example/order/service/OrderServiceImplTest.java

観点マトリクス

埋まっているのは正常系の1列だけ

メソッド	正常系	null・空	境界値	不正ステータス	存在しないID
findById					
createOrder					
changeStatus					
cancelOrder					

■ 生成直後に埋まる　□ 空きマス

@Test 9本　　空きマス 16

テストは本数では測れません。空きマスの数で測ります。

抜けやすい5観点と実在の入力

仕様の期待と実装の実際がずれる5箇所

観点	この題材での入力	仕様の期待 → 配布実装の実際
null	getCustomerOrderTotal (null)	仕様に定めなし → equals が false を返して素通りする
空	findByDateRange で該当0件	空のリスト → null も例外も返さないこと
境界値	calculateDiscount (10, 50000)	475,000円（5%割引）→ 500,000円（割引なし）
不正ステータス	SHIPPED → CANCELLED	InvalidOrderStateException → HTTP 500 @RestControllerAdvice が未実装のため
存在しないID	findById(999)	OrderNotFoundException → 既存テストあり

テスト生成の実演

見るのは assert の中身と例外型

ここに実画面を差し込む: VS Code の統合ターミナル
claude 起動後に /gen-test を投げた直後の画面

ここに実画面を差し込む: 提示された差分
changeStatus の TODO ブロックに @Test が並んだところ

1. モックの when は書けている
2. assert が assertThat 1行で止まっている

差分をそのまま採る前に、assert の中身と例外型を見ます。

ワーク

D2-1 でやること

TODO 4件の下書きに、足りない観点を追加

[35min]

さわる	OrderServiceImplTest.java の changeStatus ブロック src/test/java/com/example/order/service/OrderServiceImplTest.java
やる	TODO 4件を Claude Code に下書きさせ、足りない観点を自分で指定して足します
できたら	異常系のテストが4件以上あり、自分が書いたテストが緑
考える	生成された assert は値を比べているか、存在を確かめているだけか

手順とプロンプト例は教材サイトの Day2、または exercises/day2/exercise1-テスト自動生成.md

補足

D2-1 で先に知っておくこと

赤1件は仕込み、見るのは自分が足したテスト

配布時点から赤の1件

DELIVERED の受注をキャンセルできないことを確かめるテストが、配布時点から赤のままです。
Day3 で直す既知のバグで、故障ではありません。

見るのは自分が足したテスト

全体の緑ではなく、自分が足したテストが緑になっているかを見ます。
赤が1件だけなら、そこは想定どおりです。

NullPointerException の切り分け

モックの戻り値を設定していないと NullPointerException で落ちます。
落ちた場所がテストなのか実装なのかを先に切り分けます。

解説

D2-1 の実行結果

緑は自分の追加分、赤は1件だけ

ここに実画面を差し込む: テスト実行結果
Tests run の行と Failures: 1 の行が読めること

足したテストが緑

赤は1件だけ

赤の1件の名前
異常系_DELIVEREDの受注は
キャンセルできない

赤の1件は Day3 で直します。今日は名前が一致していれば想定どおりです。

解説

採否を決める4観点

見るのは本数ではなく、assert の中身

1. assert が期待値と実値を突き合わせている（assertNotNull だけで終わっていない）
2. 異常系は assertThrows で例外の型まで確認している
3. モックは呼び出しを verify で確かめている
4. 境界値を入れている（CONFIRMED→CANCELLED と SHIPPED→CANCELLED の両側）

配布物の文言をそのまま載せています。

出典 handson/docs/生成物チェックリスト.md B-5

クイズ

モック利用時の verify の役割

**Q. orderRepository をモックにしたテストで、
verify を書かないと確かめられなくなるものはどれでしょうか。**

- **A** repository.save が呼ばれたかどうか
- **B** 戻り値の中身
- **C** スローされた例外の型
- **D** テスト名と中身の一致

正解は次のページで確認します。まず自分で考えてみてください。

解説

モック検証の対象

正解はA。モックで見えなくなるのは呼び出しの有無

A **正解 repository.save が呼ばれたかどうか**
save を呼び忘れた実装でも、戻り値だけを見るテストは通ってしまいます

B 戻り値の中身
モックの戻り値は、assert でそのまま確かめられます

C スローされた例外の型
assertThrows で例外の型まで確かめられます

D テスト名と中身の一致
名前と中身の対応は、読んで確かめる範囲です

戻り値も例外も、モックの外側で確かめられます
生成物チェックリスト B-5 の3番目が、この verify のことです

出典 handson/docs/生成物チェックリスト.md B-5

03

D2-2 テストによるバグ検出

3件のバグが仕込んであります。場所は伏せてあります。
直してからテストを書くのではなく、先に赤を出してから直します。
[35min]

振り返り | D2-1 | **D2-2** | D2-3 | まとめ

先に赤を出す順番

赤を先に出すと、その赤が修正の証拠

テストが先

赤が、直した証拠として残ります

修正が先

このテストがバグを見つけられるかは、誰も確かめていません

D2-2 では、直す前にテストを書きます。赤が出ないテストは、バグを見つけていません

実装から書いたテストが通る仕組み

期待値の出どころが、仕様か実装かの分かれ目

ずれ1箇所

仕様（Javadoc）

数量 10個以上 で 5%割引

境界は 10・20・50 の3段

10個 × 50,000円 のとき

475,000円

ここからは取っていません

実装（コード）

```
else if (quantity > 10)
```

10個ちょうどは割引の対象外

同じ入力での戻り値

500,000円

期待値

生成されたテスト

実装を読んで期待値を決めています

期待値をどこから取るかが、この演習の分かれ目です

BuggyOrderCalculator の全体像

3つのメソッドの仕様は、すべて Javadoc に記載

メソッド	仕様に書かれていること
`calculateDiscount`	数量に応じた割引後の金額
`getCustomerOrderTotal`	指定顧客の受注合計。該当なしは合計0円
`findOverdueOrders`	指定日時点で納品予定日を過ぎた受注。当日も遅延と判定

テストデータ

受注5件（東京電機工業3件・大阪精密機械1件・
名古屋システム1件）

実行の仕方

main を持つ単一ファイルとして動きます
`exercises/day2/BuggyOrderCalculator.java`

どのメソッドに何が仕込んであるかは、ここでは伏せてあります

ワーク

D2-2 でやること

テストが先で、修正はその次

[35min]

さわる	exercises/day2/BuggyOrderCalculator.java（読む） 自分で作るテストクラス
やる	Javadoc から期待値を決めてテストを書き、先に赤を出します。 赤の出方から原因の行を特定して直します
できたら	赤3件が出て、修正後にすべて緑
考える	その期待値は Javadoc から取ったか、実行結果から取ったか

手順とプロンプト例は教材サイトの Day2、または exercises/day2/exercise1-テスト自動生成.md

補足

D2-2 で先に知っておくこと

期待値は Javadoc から、境界の数字ちょうど

1. **期待値の出どころ**
   実行結果を見てから期待値を決めると、バグに合わせたテストになります

2. **境界の数字ちょうど**
   数量11個や19個で試すと、3件のうち1件は見つかりません。仕様に書かれた数字ちょうどを入れます

3. **落ち方の違い**
   同じバグでも、テストからは assertion failure、main からは NullPointerException と出ます。落ち方が違っても原因が1つのことがあります

解説

赤3件の出方

赤の文面が、原因の場所を指す手がかり

ここに実画面を差し込む:
BuggyOrderCalculator のテストで
Failures が3件並ぶ実行結果

読み方

割引

```
expected 475000 but was 500000
```

合計

```
expected 0 but was null
```

遅延

```
expected size 1 but was 0
```

ここまでが前半です。ここから原因の行を1件ずつ見ます

解説

バグ1 割引の境界

境界の 10・20・50 ちょうどだけが食い違い

実装の比較演算子

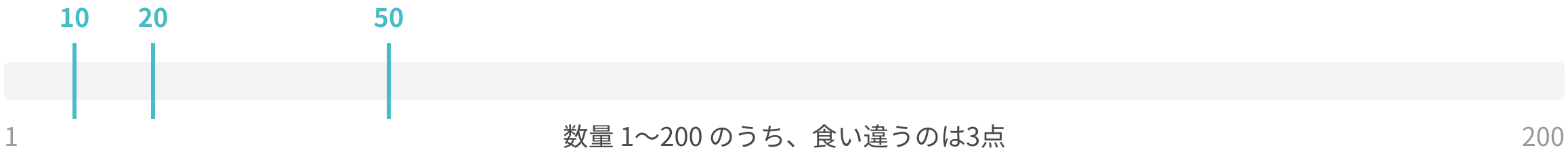
```
if (quantity > 50) { ... }
else if (quantity > 20) { ... }
else if (quantity > 10) { ... }
```

実行結果（単価 50,000円）

10個 → 500,000円

仕様の期待値 → 475,000円

差額 25,000円 の過大請求



200通り中3通り（1.5%）。11個・19個・21個・49個・51個はすべて仕様と一致します

単価 50,000円 なら、過大請求は 10個で25,000円・20個で50,000円・50個で125,000円

解説

バグ2 合計の初期値

合計0円のはずが、初期値 null で例外

合計を入れる変数の宣言

```
Integer total = null;
```

Javadoc の仕様

該当する受注がない顧客の場合は
合計0円として扱う

同じメソッドの2回の呼び出し

1回目 東京電機工業
940,000円 が返ります

2回目 存在しない会社
`NullPointerException`

テストからは expected 0 but was null の assertion failure、main からは NullPointerException。原因は同じ1行です

該当0件の顧客を1件入れるだけで、このテストは落ちます

解説

バグ3 遅延判定の当日

当日を含めない isBefore が、1件を取りこぼし

遅延の判定式

```
order.deliveryDate.isBefore(checkDate)
```

Javadoc の仕様

納品予定日当日に出荷完了していない受注も
遅延と判定する

ORD-003（納品予定日 4/12・PENDING）

チェック日 2026-04-12 で 検出0件（期待は1件）

isBefore は当日を含めません。納品予定日 4/12 の受注は、
チェック日 4/12 でも1件も出ません

仕様に当日と書かれていたら、当日ちょうどのデータを1件入れます

解説

3件に共通する形

原因は3件とも境界、見え方だけがばらばら

観点マトリクスの再掲

境界値

境界値の列は
空きマスのまま

数量の境界
10・20・50 ちょうど

→ **過大請求**
10個で25,000円 多く請求

件数0の境界
該当する受注がゼロ

→ **NullPointerException**
合計を取るだけで落ちます

日付の境界
納品予定日の当日ちょうど

→ **取りこぼし**
当日ぶんが1件も出ません

境界を狙わないテストは、この3件を1件も見つけません

クイズ

割引バグが現れる数量の数

Q. calculateDiscount に数量1から200を入れると、食い違うのは何通りでしょうか。

- **A** 190通り
- **B** 100通り
- **C** 20通り
- **D** 3通り

正解は次のページで確認します。まず自分で考えてみてください。

解説

境界値を外したテストの限界

正解はD。食い違うのは 10・20・50 の3通り

D **正解 3通り**

10・20・50 ちょうどの3点でだけ、仕様と実装が食い違います

A 190通り

残り197通りは仕様と実装が一致します。190 になる根拠はありません

B 100通り

割引は数量の全域ではなく、境界の3点でだけずれます

C 20通り

20 は境界の値であって、食い違う通り数ではありません

数量を適当に選んだテストがこのバグを踏む確率は 1.5% です

境界値は、思いついたら足すものではなく、先に決めて入れます

出典 exercises/day2/BuggyOrderCalculator.java

04

D2-3 挙動を守る段階的整理

309行のレガシーコードを、出力を変えずに読みやすくします。
一気に書き換えさせないための道具が、出力の固定と段階の区切りです。
[45min]

振り返り | D2-1 | D2-2 | **D2-3** | まとめ

LegacyReportService の実測

手を入れる前に、309行の中身を数字で把握

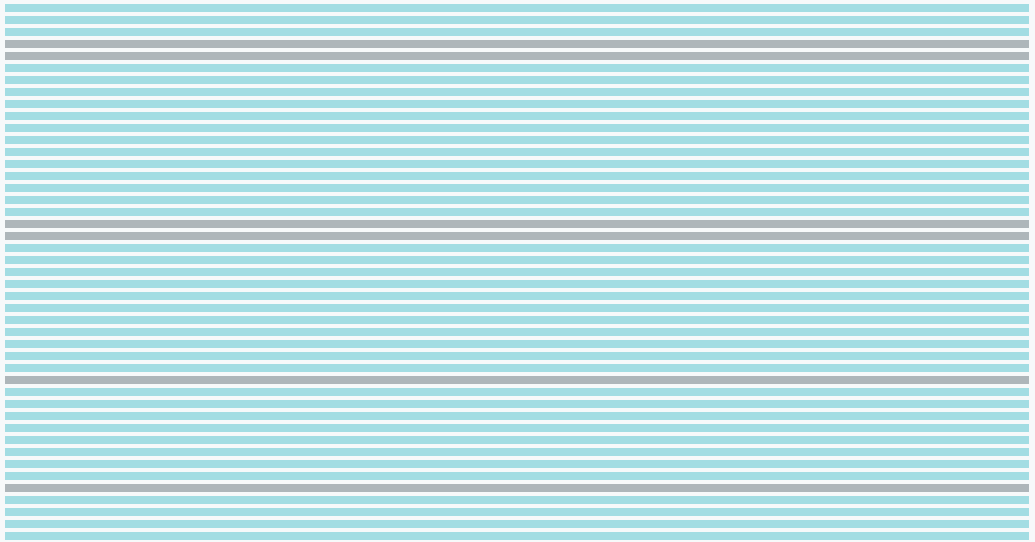
項目	実測値	内訳
総行数	309行	3メソッドで231行、ファイルの75%
最長メソッド	92行	generateMonthlyReport（57-148行）
文字列連結	87回	result = result + が61回、line = line + が26回
マジックナンバー	6箇所	1.1 が3箇所、1000000 が1箇所、2000000 が2箇所
Map<String, Object>	5箇所	d.get(23回、キャスト26回（(String) 17回、(int) 9回）
制御構文のネスト	5段	最深文のインデントは28スペース
重複	39行	42行のうち39行が一致

すべて Java 17 で実行して数えた値です。整理の指示は、この数字を添えて出します。

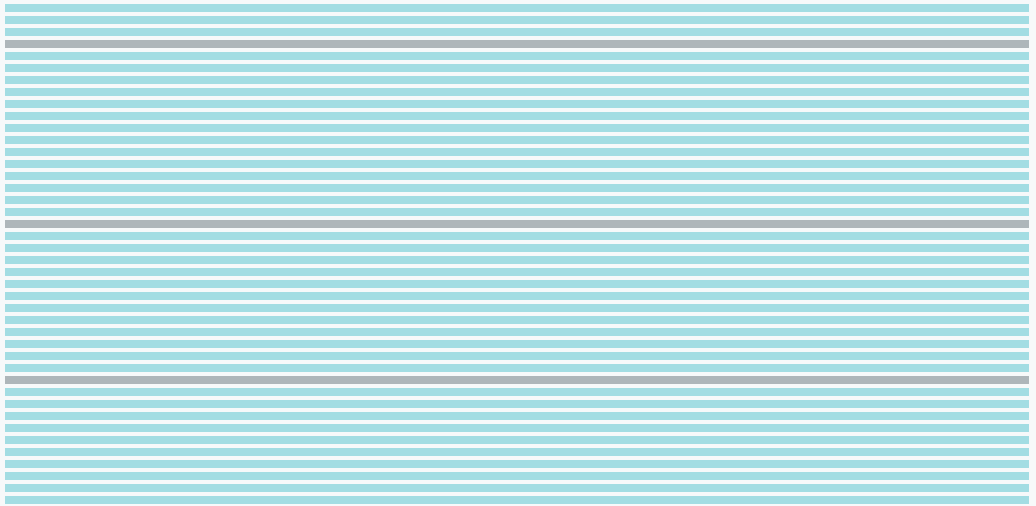
重複の正体

2つのループ本体は、42行中39行が同一

generateMonthlyReport 69-113行（45行）



generateCustomerSummary 166-207行（42行）



違う9行はフィルタ条件・明細の2列目・コメント2行の3か所です

一致した39行　違う行

42行のうち39行、93% が同じです。サマリー出力部も15行が完全一致です

先に敷く安全網

出力の固定が、段階を積むための前提

比較の対象外

出力日の行は実行日で変わるので、
比較の対象外です

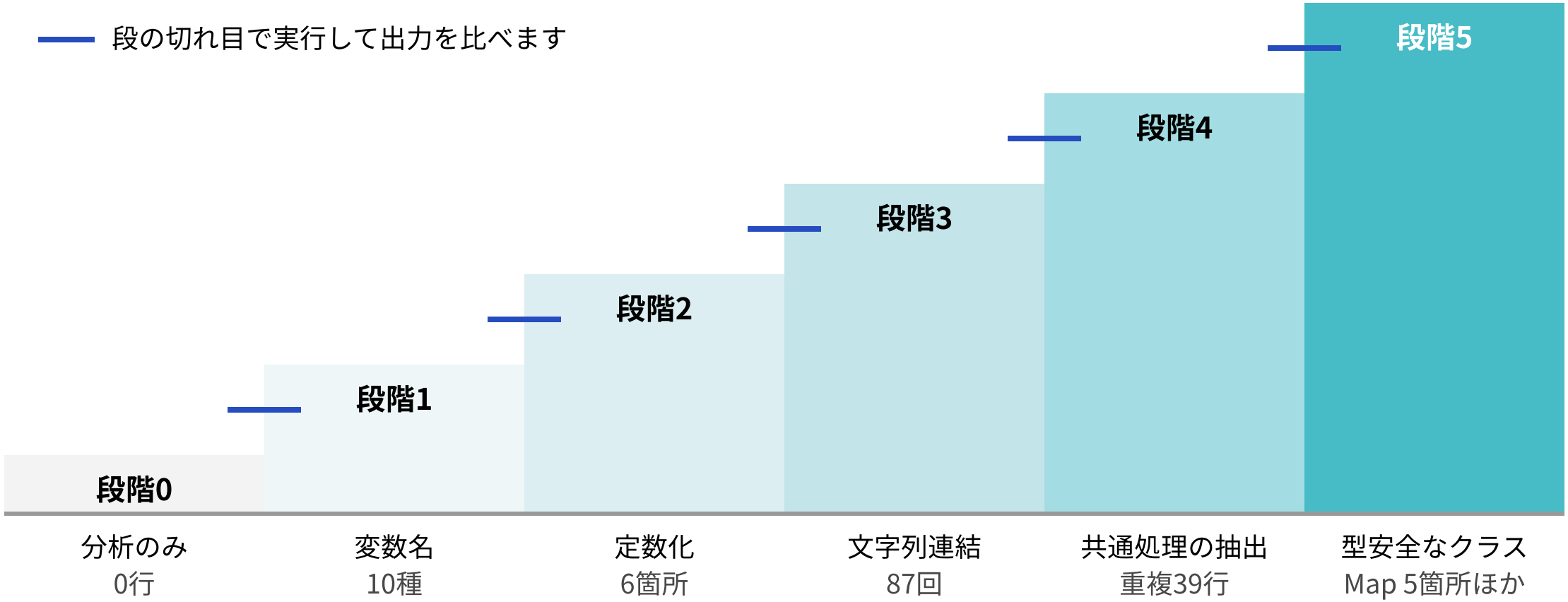
段階の区切り

1段階ずつ実行して出力を比べます。
まとめて適用しません。

先に固定していないと、差分が出たときに原因を追えません

段階0から段階5の階段

差分の小さい順に積み、型の置き換えを最後



段階5 で触るのは Map<String, Object> 5箇所・d.get(23回・キャスト26回

Planモードの実演

承認の前に、計画で変更対象を確認

ここに実画面を差し込む: Shift+Tab で Plan モードに入り、
段階1の計画が提示された画面

1. いまどのモードかが分かる表示
2. 変更対象として挙がったメソッド名
3. 承認と差し戻しの選択肢

広い範囲を触らせるときほど、計画を先に見ます

ワーク

D2-3 でやること

出力を固定してから、段階を1つずつ

[45min]

さわる	exercises/day2/LegacyReportService.java
やる	整理前の出力をファイルに固定してから、段階0・段階1・段階2と 順に進めます。1段階ごとに実行して出力を比べます
できたら	出力日の行以外で差分なし、各段階でコンパイルが通っている状態
考える	Claude Code の提案した着手順序と段階1〜5の順序が違ったとき、どちらが安全か

段階5まで到達しなくて構いません。到達した段階まで出力が一致していれば達成です。

手順とプロンプト例は教材サイトの Day2、または exercises/day2/exercise2-リファクタリング.md

補足

D2-3 で先に知っておくこと

1段階ずつ区切ることが、原因を追える条件

1. **まとめて直させない**
   差分が出たときに原因を追えません。1段階ずつ区切ります。

2. **出力日の行**
   毎回変わります。差分がその行だけなら合格です。

3. **改行の数**
   行末の改行が増減していないかを差分で見ます。

4. **戻すときの単位**
   段階を戻すときは、直前の1段階だけを戻します。

解説

出力比較の結果

差分が出力日の行だけなら合格

ここに実画面を差し込む: 整理前後の出力比較で差分なしを返した画面

ここに実画面を差し込む: 出力日の行だけ差分が出た画面

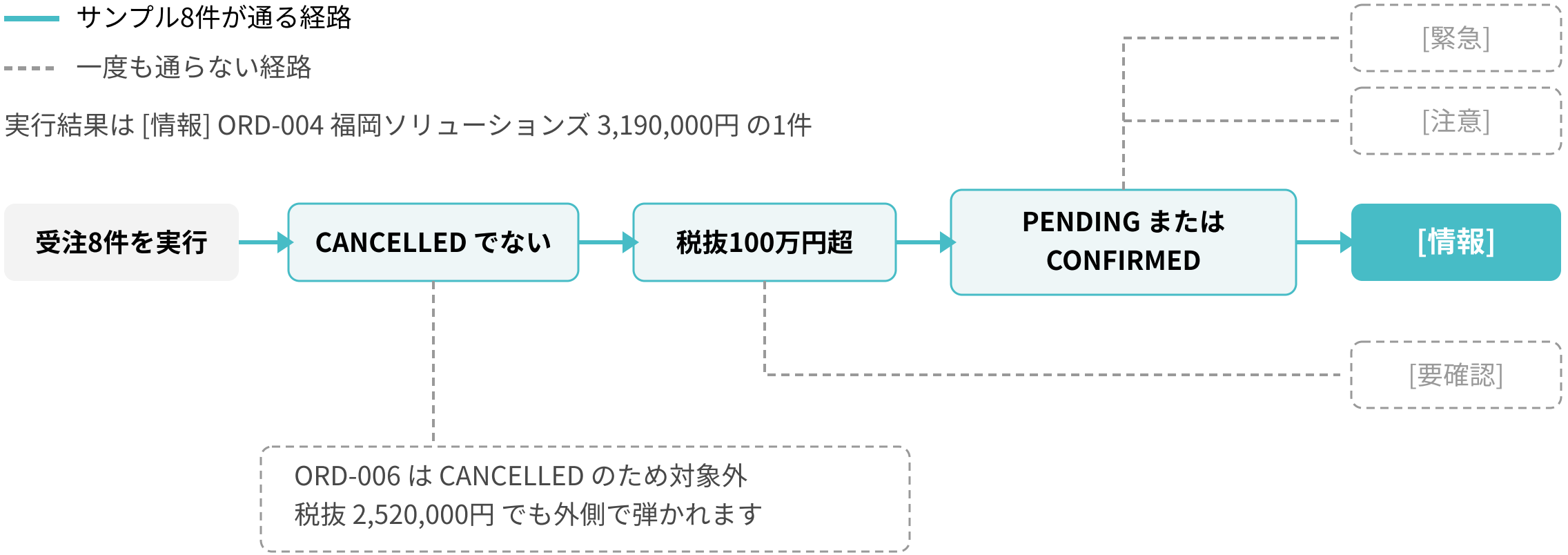
1. 出力日の行だけの差分は想定どおり
2. 明細行に差分が出たら、直前の段階だけを戻す

ここまでで、振る舞いを変えずに読みやすさだけが上がった状態になっています

解説

出力比較で守れない範囲

差分ゼロは、通った経路だけの保証



差分ゼロは、壊れていないことの証明ではありません。通っていない経路は比較されていません。

解説

リファクタ後コードの5観点

整理後に見るのは、出力・段階・名前・定数・重複

番号	観点
1	整理の前後で出力が変わらない
2	各段階の後でコンパイルが通る（まとめて適用しない）
3	短縮変数名が残っていない（d, q, tmp, cnt）
4	マジックナンバーが名前付き定数になっている（1.1, 1000000, 2000000）
5	重複が共通メソッドにまとまり、ネストが浅くなっている

整理を終えた自分のコードに、この順で当てます

出典 handson/docs/生成物チェックリスト.md B-6

クイズ

出力の差分ゼロが示す範囲

Q. 段階5まで終えて、整理前後の出力を比べたところ差分はありませんでした。
これで確かめられたことはどれでしょうか。

- **A** 整理が正しく終わったこと
- **B** サンプル8件が通る経路だけが壊れていないこと
- **C** すべての分岐が壊れていないこと
- **D** 性能が落ちていないこと

正解は次のページで確認します。まず自分で考えてみてください。

解説

出力比較の適用範囲

正解はB。比較の対象は通った経路だけ

B **正解 サンプル8件が通る経路だけが壊れていないこと**
比較できるのは、実行した入力がたどった経路に限られます

A 整理が正しく終わったこと
出力が一致しても、正しく終わった証明にはなりません

C すべての分岐が壊れていないこと
サンプル8件では、高額アラートの5分岐のうち1つしか通りません

D 性能が落ちていないこと
出力比較は結果だけを見ます。処理時間は比べていません

通らない経路まで守りたいなら、その経路を通すデータを1件足して出力を取り直します
テストが要るのはここです。出力比較は入口で、仕上げはテストになります

出典 exercises/day2/LegacyReportService.java

05

発展枠とまとめ

速く終わった方は発展課題へ、追いついている方はそのまま続けてください。
最後に、今日決めた採否の基準をもう一度そろえます。

発展枠 [15min]、まとめ [15min]

振り返り | D2-1 | D2-2 | D2-3 | **まとめ**

発展

発展課題 D2-2+ と D2-3+

次の演習に影響しない追加課題2本

番号	内容	対象ファイル
D2-2+	**再発防止テストの追加** 直したバグと同じ形の入力で、もう一度落ちないことを確かめるテストを足します。	`exercises/day2/BuggyOrderCalculator.java`
D2-3+	**段階5の続き** Map<String, Object> を型安全なクラスに置き換えます。	`exercises/day2/LegacyReportService.java`

> まず自分で方針を考えてから試します。次の演習には影響しません。
> うまくいかないときは、追加した部分を消せば元に戻せます。

今日埋めた3つの穴

今日触ったのは service と単体クラス2本

Day1 で書いた層

- controller
- **service** ●
- repository
- model・dto ●

● Day1 の実装が入った層

今日確かめたところ

service 層のテスト
changeStatus に異常系を4件以上

`BuggyOrderCalculator`
バグ3件を先に赤にしてから修正

`LegacyReportService`
309行を段階を区切って整理

Day1 で書いたものを、今日はコード以外の手段で確かめました。

生成テストのレビュー観点

本数ではなく**この4行**で採否を判断

観点	演習	具体
assert が期待値と実値を突き合わせている	D2-1・D2-2	assertNotNull だけで終わらせない
異常系は assertThrows で例外の型まで見る	D2-1	InvalidOrderStateException と OrderNotFoundException の2件
モックは呼び出しを verify で確かめている	D2-1	save が呼ばれたことを確認
境界値を入れている	D2-2	数量10・20・50、納品予定日当日

> この4行は配布物の docs/生成物チェックリスト.md B-5 と同じ文言です。
> 現場でも同じ順で見てください。

次回 Day3 の内容

次回の主題は壊れたときの動き

- D3-1 StackOverflowError の原因特定
- D3-2 遅くなったログの切り分け
- D3-3 ヘルスチェックと処理時間ログ
- D3-4 セキュリティと規約のレビュー

今日は正しいかを確かめました。次回は壊れたときに何を見るかです。
今日の環境をそのまま使います。片付けは不要です。

質問

質疑応答・アンケート

[15min]

本日の参照先は3か所

本日の参照先

教材サイト
idunder02.give-app.net の Day2 ページ

配布 ZIP
`exercises/day2/`

チェックリスト
docs/生成物チェックリスト.md の B-5 と B-6

その場で答えられないものは、次回の冒頭で回答します。

アンケート

本日の内容について、アンケートへの
ご協力をお願いします。

要記載
アンケートのURLまたはQRコードを
この位置に差し込みます。

株式会社ギブリー (Givery, Inc.)
〒150-0036 東京都渋谷区南平台町15-13 帝都渋谷ビル8F