TRAINING

下流工程担当向け Claude Code 活用研修

Day1 題材を掴む ＋ Claude Code の3操作

Givery

2026年9月3日

株式会社インフォメーション・ディベロプメント 御中

本資料の二次転載禁止について

受講者ご本人の学習目的に限った利用

著作権の帰属

本資料の著作権は株式会社ギブリーに帰属します。
当社の書面による事前の許可なく、全部または一部を
複製・転載・改変・再配布することを禁止します。

利用の範囲

受講者ご本人が本研修の学習目的で利用する範囲に限り、閲覧と印刷ができます。
社外への共有、SNS・ブログ等への掲載、他の研修や資料への流用は
ご遠慮ください。

引用や社内共有の可否について迷ったら、講師にご相談ください。

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

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日間の到達点

Day1 は題材の把握と3操作の実行

Claude Code

Day1
2026年9月3日

題材を掴む ＋ 3操作

`service / dto / static`

[180min]

Day2
2026年9月9日

テストで裏を取る

`src/test と exercises/day2`

[180min]

Day3
2026年9月17日

壊れたときに効く動き

`exercises/day3 と config`

[180min]

3日とも同じ受注管理システムを触ります。Day1 で掴んだ構造が、Day2 のテストと Day3 の障害対応でそのまま効いてきます。

※ 各ロゴは各社の商標です。

本日のゴール

システムの全体像と、Claude Code の3操作

到達状態

・受注管理システムが何をするか、
業務の言葉で説明できる

・ターミナル対話・Planモード・
スラッシュコマンドの3操作を、
全員が1回以上動かしている

番号	見える成功
D1-1	層の説明が返り、CLAUDE.md の読込を確認
D1-2	ブラウザで叩くと JSON が返り、両端を含む
D1-3	http://localhost:8080/dashboard.html に 受注が表で並ぶ
D1-4	明細行が出て、明細の税込合計が ヘッダーの totalAmount と一致
D1-5	id=2 への {"status":"CANCELLED"} が 200

コンパイルが通ったかではなく、画面に出たかで達成を判定します。
D1-3 で作るダッシュボードは Day2 と Day3 でも使います。消さないでください。

本日の流れ

区分	番号	内容	時間
前半	**導入**	受注管理システムの歩き方	[20min]
前半	**D1-1**	claude を起動しシステムの構造を説明させる	[15min]
前半	**D1-2**	findByDateRange とエンドポイントを実装	[20min]
前半	**D1-3**	受注ダッシュボードをゼロから作る	[25min]
休憩		アプリは起動したままにしておきます	[10min]
後半	**D1-4**	OrderItemDTO をゼロ生成し明細をレスポンスに載せる	[25min]
後半	**D1-5**	ステータス遷移ルールを実装し、遷移を叩いて確かめる	[25min]
後半	**発展枠**	D1-1+ 〜 D1-5+ から1本選ぶ	[10min]
後半	**まとめ**	3操作の使い分け	[15min]
後半	**質疑**	質疑応答・アンケート	[15min]
合計			**[180min]**

演習は5本あります。時間内に終わらなくても、次に進んで構いません。

補足

進め方の約束

同じ依頼でも出力は毎回ばらつく前提

1 出力は毎回ばらつく

同じ依頼でも違う形が返ります。形ではなく、
何を呼んで何を返しているかで判断します。
隣の人と違う形でも問題ありません。

2 全員そろって次へ

全員が終わるのを待ってから、
次の演習に進みます。手が止まったら、
手を挙げて教えてください。

3 壊しても戻ります

データはメモリ上の H2 で動きます。アプリを
再起動すると受注10件の初期状態に戻るので、
気にせず触ってください。

4 手順は教材サイト

手順の逐語とプロンプト全文は
教材サイトにあります。このスライドは、
全体の地図として見てください。

演習の共通フローと進捗マップ

頼む前に予想し、返ってきたら画面で確認

自分で予想する	一言で頼む	計画と差分を読む	画面で確かめる
返ってくる形を先に思い描く	長い指示は書かない	承認する前に中身を見る	ブラウザを再読み込みする

進捗マップの読み方

導入 | D1-1 | D1-2 | D1-3 | D1-4 | D1-5 | まとめ

D1-2 濃いコマがいまの現在地

D1-3 薄いコマはこれから進む演習

章の変わり目に同じ帯が出ます。
いま何番目かは、ここで確認してください。

SESSION

01

受注管理システムの歩き方

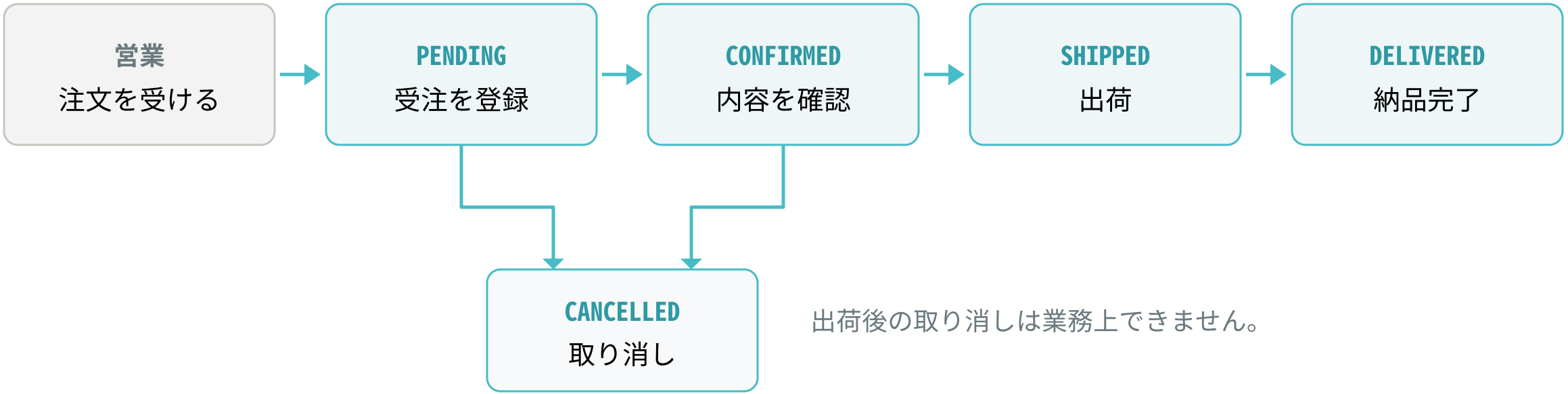
3日間ずっと触るシステムです。
業務・データ・ファイル配置をここで押さえます。
ここを飛ばすと、自分がどこを直しているのか分からないまま実装に入ります。

[20min]

受注から納品までの流れ

取り消せるのは PENDING と CONFIRMED の2状態

法人向けにIT機器を売っている会社の受注を管理します。
画面はありません。他のシステムから呼ばれる API だけの構成です。



出荷後の取り消しは業務上できません。

配布データは、この流れの途中の状態を10件ぶん持っています。

扱うデータ3テーブルと金額の決まり

受注1件に明細が複数ぶら下がる1対多の構造

customers 5件

顧客コード・会社名・住所・電話・メール

↓ 1対多

orders 10件

受注番号・顧客名・商品名・数量・単価
合計金額・ステータス・受注日・納品日

↓ 1対多

order_items 13件

受注ID・商品コード・商品名・数量・単価・小計

金額の決まり

税込 = 数量 × 単価 × 1.10

端数は int で切り捨てます。

受注番号の形

`ORD-yyyyMMdd-XXX`

ORD	20260401	001
固定	受注日	連番

配布データの受注10件

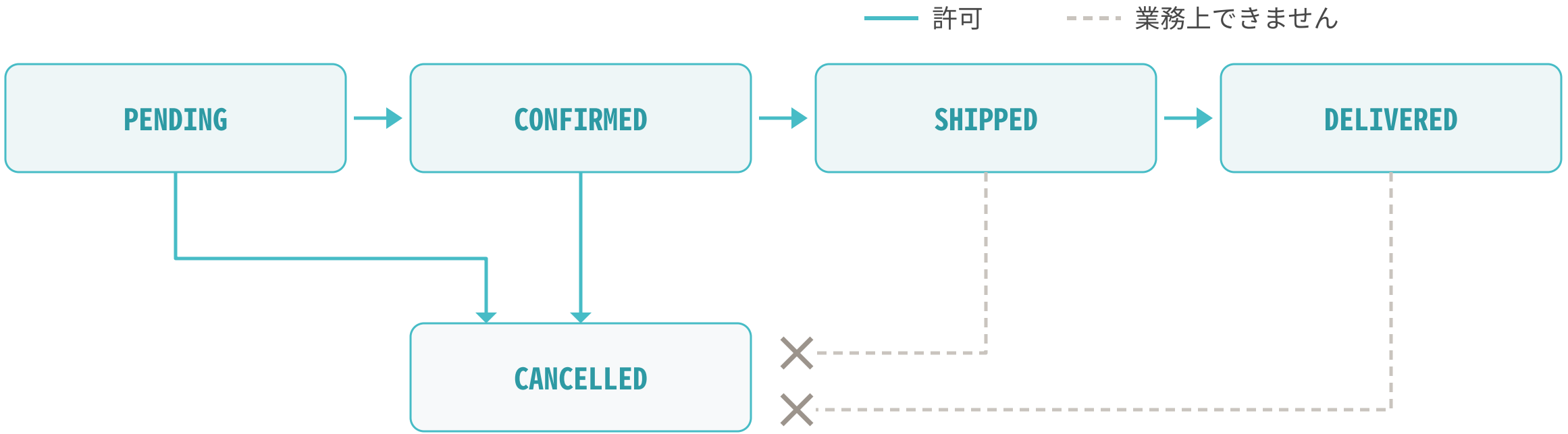
演習の期待値の出どころになる受注10件

ID	受注番号	顧客	数量	合計（税込）	ステータス	受注日
1	ORD-20260401-001	東京電機工業	3	594,000	PENDING	04-01
2	ORD-20260401-002	大阪精密機械	10	935,000	CONFIRMED	04-01
3	ORD-20260402-003	名古屋システムサービス	5	660,000	CONFIRMED	04-02
4	ORD-20260403-004	福岡ソリューションズ	20	3,190,000	SHIPPED	04-03
5	ORD-20260403-005	東京電機工業	20	990,000	PENDING	04-03
6	ORD-20260404-006	北海道テクノロジー	15	2,772,000	DELIVERED	04-04
7	ORD-20260405-007	大阪精密機械	2	770,000	PENDING	04-05
8	ORD-20260406-008	名古屋システムサービス	50	660,000	CONFIRMED	04-06
9	ORD-20260407-009	福岡ソリューションズ	30	280,500	CANCELLED	04-07
10	ORD-20260408-010	東京電機工業	25	187,000	PENDING	04-08

受注日は 2026-04-01 から 2026-04-08 に分布します。
内訳は PENDING 4件・CONFIRMED 3件・SHIPPED / DELIVERED / CANCELLED 各1件です。
SQL で直接のぞくときは http://localhost:8080/h2-console を開きます。
JDBC URL は jdbc:h2:mem:orderdb、ユーザー sa、パスワードなし。

ステータス遷移の許可と禁止

出荷後の取り消しを塞ぐ理由は帳簿の整合



出荷済みを取り消せると、モノは動いたのに帳簿だけ消える状態になります。

```
Order.STATUS_PENDING / STATUS_CONFIRMED / STATUS_SHIPPED
STATUS_DELIVERED / STATUS_CANCELLED
```

クイズ

ステータス遷移の可否

Q. 次のうち、業務上認められていないステータス変更はどれでしょうか

A　PENDING から CANCELLED

B　CONFIRMED から SHIPPED

C　SHIPPED から CANCELLED

D　PENDING から CONFIRMED

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

遷移可否の答え合わせ

正解は C、出荷後の取り消しは業務上不可

C **SHIPPED から CANCELLED**
出荷したものを取り消せると、モノは動いたのに帳簿だけ消えます。

A PENDING から CANCELLED
まだ出荷していないので、取り消せます。

B CONFIRMED から SHIPPED
確認が済んだものを出荷する、通常の流れです。

D PENDING から CONFIRMED
登録した受注を社内で確認する、最初の一歩です。

配布コードの状態
CONFIRMED から CANCELLED はまだ通りません。D1-5 で埋めます。

4層の役割分担

上から下へ呼び出し、下は上を知らない構造

controller
受け口

外から来た依頼を受けて下へ渡し、結果を返します。
業務の判断はここでは書きません。
`OrderController.java / CustomerController.java`

service
判断

ステータスを変えてよいか、金額をどう計算するか。
業務の決まりごとは、すべてここに集まります。
`OrderService.java / OrderServiceImpl.java`

よく触る

repository
出し入れ

メソッド名を決めるだけで、問い合わせが組み立てられます。
SQL を自分で書く場面はほとんどありません。
`OrderRepository.java / CustomerRepository.java`

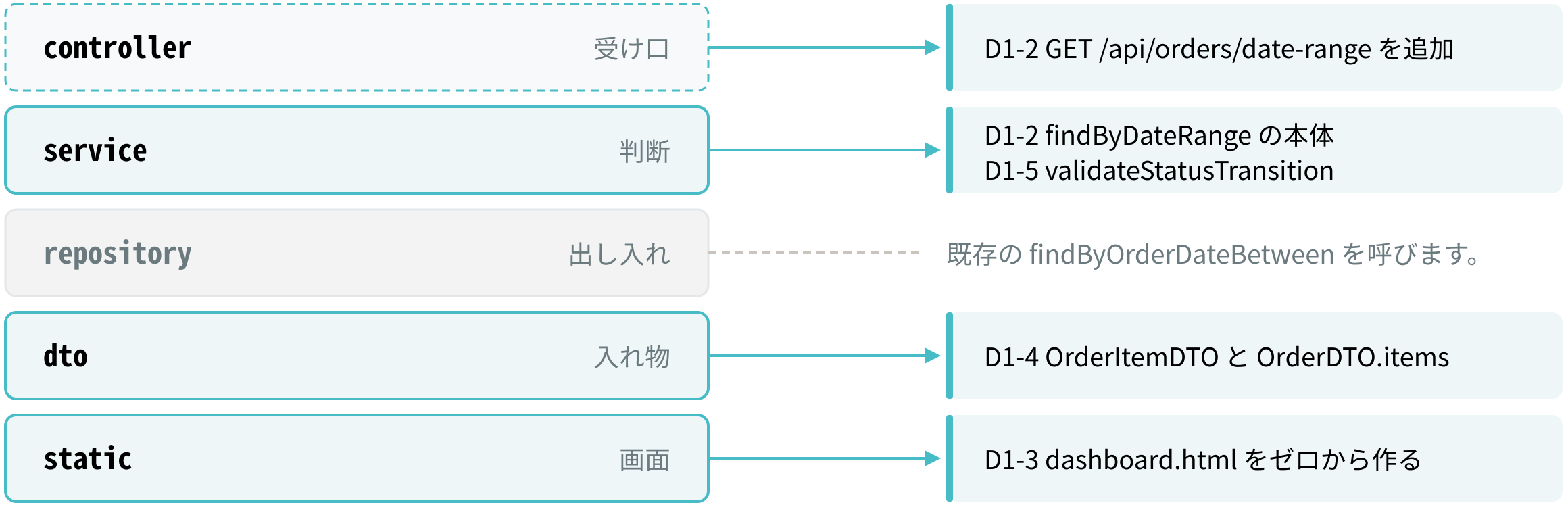
model / dto
入れ物

model はテーブルの1行、dto は外へ返すための入れ物です。
内部の持ち方をそのまま外に出さないために分けてあります。
`Order.java / OrderItem.java / OrderDTO.java`

ほかに exception/ と config/ があります。Java のファイルは全部で15本です。

本日触る3か所

主に触るのは service・dto・static の3か所



Java を書くのは D1-2 / D1-4 / D1-5 の3本。D1-3 は HTML です。

実演

ブラウザに出る生のJSON

受注10件でこの読みにくさ

ここに実画面を差し込む
ブラウザで開いた http://localhost:8080/api/orders の生 JSON
受注10件が1かたまりで折り返している状態

手が止まる問い

- どのステータスが何件か
- 合計はいくらか
- 4月1日から3日までは何件か

この読みにくさは D1-3 で消します。

講師の画面を見ながら、同じURLを自分のブラウザでも開いてください。

配布コードに空けてある5つの穴

壊れているのではなく、仕様として空けた5つの穴

OrderServiceImpl#findByDateRange UnsupportedOperationException のまま	→	**D1-2** 本体を実装
GET /api/orders/date-range 窓口そのものが無く、叩くと HTTP 400	→	**D1-2** 窓口を追加
src/main/resources/static/ 空のまま。画面が1枚も無い	→	**D1-3** 1枚つくる
OrderDTO 明細 items を持っていない	→	**D1-4** DTO を新規
validateStatusTransition CONFIRMED から CANCELLED が無い	→	**D1-5** 1マス足す

穴は配布物では直していません。5つとも今日の演習で埋めます。

SESSION

02

Claude Code

[20min]

Claude Code の起動と3操作

ターミナルで claude を起動し、計画を先に出させてから承認する流れを手に入れます。
つまずきの大半は、Planモードに入ったつもりで入っていないことです。
この章で現在地の見え方まで確認します。

3操作の使いどころ

依頼の大きさで決まる3操作の選び分け

場面	使う操作
構造を知りたい・質問だけしたい	**ターミナル対話** 質問と小さな変更はここで
小さな変更を1か所に入れたい	
破壊的な変更、複数ファイルにまたがる依頼	**Planモード（Shift+Tab）**
決まった手順を毎回同じ形で流したい	**スラッシュコマンド** `/review /gen-test /clear /compact /init`

@ でファイルを指定すると、Claude Code の探索が減ります。
この図はまとめでもう一度出します。

起動と同時に読まれる CLAUDE.md と .claude 一式

ここに実画面を差し込む:
VSCode のターミナルで claude を起動し、
CLAUDE.md を読み込んだ表示が出ている画面

CLAUDE.md
技術スタック・コーディング規約・受注の業務ルール

.claude/

settings.json モデルは sonnet、.env は読み取り拒否

commands/ /review と /gen-test の2本

agents/ コードレビュー役の code-reviewer

skills/ 受注ドメインの order-domain

VS Code　Claude Code

前提を毎回伝え直す必要はありません。

補足

Planモードの往復と現在地

計画を先に出させる4段の往復

Shift+Tab で切り替え → 計画の提示 → 承認と差し戻し → 差分を読んで採否

ここに実画面を差し込む:
通常モードの入力欄

通常モード

ここに実画面を差し込む:
Planモード切り替え後の入力欄

Planモード（Shift+Tab）

入ったつもりで入っていない状態がいちばん多いので、依頼の前に入力欄を1回見てください。

起動から計画提示までの実演

承認と差し戻しを両方見せる実演

[3min]

1. claude 起動と CLAUDE.md 読込の確認
2. / と @ を打ったときの候補表示
3. Shift+Tab で Planモードへ切り替え
4. 計画の提示で止め、承認と差し戻しの両方

ここに実画面を差し込む:
計画が提示され、承認と差し戻しが
選べている画面

> ここは手を止めて見てください。打鍵は次の D1-1 からです。

ワーク

D1-1 コードベースの説明と計画確認

[15min]

起動・Planモード・差し戻しを1周

やること	claude を起動してコードベースの構造を説明させ、 findById の直下に受注番号で検索するメソッドを1本足す
使う操作	ターミナル対話から Shift+Tab で Planモードへ、承認と差し戻しを1回ずつ
できたら	CLAUDE.md 読込が確認でき、計画を1回以上表示できた 試したコードを消して、配布時の状態に戻す .\mvnw.cmd compile が BUILD SUCCESS
詳細	手順の逐語と参考プロンプトは教材サイトの D1-1 カード

予想を先に一言だけ書いてから、Claude Code に頼んでみてください。

解説

D1-1 の確認と後片づけ

配布時の状態に戻すところまでが D1-1

戻ってきたら見る4点

1. CLAUDE.md が読み込まれた状態で対話が始まった
2. 計画を承認した回と差し戻した回の両方をやった
3. 提案されたメソッド名が既存の並びに沿っていた
   `findById findByOrderNumber findByStatus`
4. お試しコードを残さない

後片づけと次への前提

追加した部分を消すか、ファイルを配布時の内容に戻します。この研修では Git 操作を使いません。

最後に .\mvnw.cmd compile で BUILD SUCCESS を確認します。

次の D1-2 は、このファイルが配布時のままであることを前提にしています。

SESSION

03

日付範囲検索の実装

[20min]

空けてある穴を1つ埋めて、ブラウザに JSON が返るところまで通します。
手で書き始めず、やりたいことを言葉にして渡します。
新しいメソッドは作らず、既にある本体を埋めます。

埋める穴と2つの手がかり

新しいメソッドは作らず、既にある本体を埋める作業

配布時の OrderServiceImpl

```
@Override
@Transactional(readOnly = true)
public List<OrderDTO> findByDateRange(LocalDate from, LocalDate to) {
    // TODO: このメソッドを実装すること
    throw new UnsupportedOperationException("未実装: findByDateRange");
}
```

手がかり1 問い合わせの本体

OrderRepository に findByOrderDateBetween が
すでに用意されています。JPQL を書く必要はなく、
メソッドを呼ぶだけで条件が組み立てられます。

手がかり2 詰め替えの手本

同じファイルの findByStatus と
findByCustomerName が、OrderDTO へ
詰め替える書き方の手本になります。

メソッド名から組み立てる問い合わせ

BETWEEN は両端を含む区間指定

メソッド名の3分割

findBy	OrderDate	Between
検索の宣言	orderDate 列	範囲の指定

組み立てられる条件

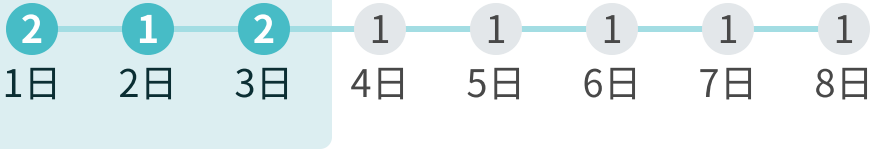
```
order_date BETWEEN ? AND ?
```

メソッド名の綴りが、そのまま問い合わせになります。
repository は今回は触りません。呼ぶだけです。

2026年4月の受注日

丸の中の数字は、その日の受注件数です。

from 2026-04-01 から to 2026-04-03



範囲に入るのは5件

1日だけを指定しても、その日の2件が返ります。

両端を含むので、from と to に同じ日を入れても、その日の受注が返ります。
日付は 2026-04-01 の形で渡します。順序が逆なら空で返ります。

補足

実装前に 400 が返る理由

まだ窓口が無いだけの HTTP 400

ここに実画面を差し込む: 実装前のエラー画面

/api/orders/date-range を叩いて HTTP 400 が返る画面

画面: 実装前のブラウザ表示

`GET /api/orders/date-range`
受講者が叩く URL

↓

`GET /api/orders/{id}`
date-range が {id} に入ります

↓

`HTTP 400 Bad Request`
数値に変換できず、ここで止まります

壊れているのではなく、まだ窓口が無いだけです。実装後は 200 が返ります。
`http://localhost:8080/api/orders/date-range?from=2026-04-01&to=2026-04-03`
この 400 は、配布物で実機確認した挙動です。

ワーク

D1-2 日付範囲検索とエンドポイント追加

本体の実装とエンドポイントの追加

[20min]

1 **やること**
OrderServiceImpl#findByDateRange の本体を実装します。
OrderController に GET /api/orders/date-range を足します。

2 **使う操作**
@ で対象ファイルを指定して、ターミナル対話で意図を渡します。
広く触るときは Shift+Tab で Planモードに入ります。

3 **できたら**
from=2026-04-01&to=2026-04-03 で5件。両方 2026-04-01 で2件。
2026-05-01〜2026-05-31 は空配列。確認はブラウザか curl.exe。

4 **詳細**
手順の逐語と参考プロンプトは D1-2 カードにあります。
教材サイトは idunder02.give-app.net です。

解説

D1-2 で確かめる3つの境界

境界は両端・単日・該当なしの3つ

確認する境界	期待する結果
両端を含むか	2026-04-01 から 2026-04-03 で5件
単日の指定	2026-04-01 だけで2件。id=1 と id=2
該当なし	2026-05-01 から 2026-05-31 で空配列 []

採用前に見る2点

変換の道すじ
リポジトリを呼んで OrderDTO に変換していれば、
for でも stream でも正解の範囲です。

読み取り専用の指定
@Transactional(readOnly = true) が
付いているかを確かめます。

from に新しい日付、to に古い日付を渡すと空で返ります。
空でよいのか弾くべきかは、設計として決める話です。

SESSION

04

[25min]

受注ダッシュボードの生成

Java は書きません。HTML 1枚を起こして、生の JSON を読める画面に変えます。
表が出れば達成です。見た目の完成度は問いません。
手が止まる人は、たいていプロンプトが長すぎます。

消す対象としての生JSON

受注10件で、もう読み取れない画面

ここに実画面を差し込む: 生JSONのブラウザ表示
http://localhost:8080/api/orders を開いた状態
受注10件が1かたまりで折り返している画面

最初に見たこの画面を、これから消します。

手が止まる3つの問い

- どのステータスが何件か
- 合計はいくらか
- 4月1日から3日までは何件か

この3つを読み取る道具を、いまから
Claude Code に作らせます。

使い捨ての道具を起こす場面

Claude Code がいちばん速いのは道具のゼロ生成

想像しがちな使い方

既存クラスの続きを埋める。
メソッドの中身を書かせる。
型がすでに決まっている場所を、
決まった形で埋めていく使い方です。

いちばん速い使い方

使い捨てだけど今すぐ欲しい道具を、
ゼロから1枚起こす使い方です。

- 確認用の画面
- ログの集計スクリプト
- データ投入用の小さなツール

Java 以外を書かせる経験そのものに意味があります。
この感覚が、現場に戻ってからの応用範囲を決めます。

Planモードに渡すのは、この3行

```
@src/main/java/com/example/order/controller/OrderController.java

この API を叩いて受注一覧を表で見られる画面を作りたいです。

src/main/resources/static/dashboard.html に単一の HTML で作ってください。
```

Planモード（Shift+Tab）で頼みます。
長い指示は要りません。
手が止まる人は、プロンプトが長すぎることが多いです。

> 依頼の前に、自分が見たいものを3つ決めます。
> 並べる列・金額の見せ方・日付検索の操作、この3つです。

実演

一言から表が出るまでの実演

[3min]

3行送って承認、そのまま**ブラウザ再読み込み**

見せる順番

1. Planモードで3行を送る
2. 提示された計画を読む
3. 承認する
4. ブラウザで dashboard.html を再読み込み

ここに実画面を差し込む: 表が出た直後の画面
ステータスラベルと3桁区切りまでの水準
http://localhost:8080/dashboard.html

打鍵に入る前に、自分が見たい3つを決める時間を取ります。

育っていくダッシュボード

dashboard.html は、以降の演習を確かめる計器

Day3	D3-3 ヘルスチェックの表示が増える
D1-5	ステータス変更ボタンが増える
D1-4	明細欄が増える
D1-3	**受注一覧の表が出る**

この1枚は消さないでください。
以降の演習で、自分の実装が
効いたかを確かめる場所です。

消しても作り直せます。ただし
自分で足した工夫は戻りません。

Day2 と Day3 でも、この1枚を
そのまま使い続けます。

■ 今日つくる　□ これから積む

ワーク

D1-3 受注ダッシュボードのゼロ生成

[25min]

表が出たら達成

1	**やること**	src/main/resources/static/dashboard.html を Claude Code に1枚作らせ、出てきた画面を見ながら1回に1つずつ直します。
2	**使う操作**	Planモードで一言を送り、承認します。 ブラウザを再読み込みして、短い言葉で1つずつ足します。
3	**できたら**	http://localhost:8080/dashboard.html に受注が表で並びます。 自分で決めた3つのうち2つが画面に出ています。 追加の依頼を1行出して、画面が変わるところまで確認しました。
4	**詳細**	詳細と参考プロンプトは教材サイトの D1-3 カードにあります。

補足

手が止まったときの言い換え

症状	投げ直す言葉
計画が返ってこない	作りたい画面を1文に縮めて言い直します。 長い指示ほど計画が返りにくくなります。
表は出るが中身が空	「データが表示されません。fetch のパスと、 レスポンスの構造を確認して直してください」
見た目が派手すぎる	「装飾を減らして、白背景に細い罫線だけの 落ち着いた見た目にしてください」
一度に直して壊れた	「さきほどの変更を取り消して、金額の3桁区切り だけを適用してください」
画面が真っ白	F12 の Console に出た赤い文言をそのまま貼り、 「これが出ています」と伝えます。

このページは演習中も出したままにします。

解説

一発指示と往復の差

一発で当てにいかず、1回1つ直す往復

一発で当てにいく

仕様を全部書いてから投げます。
返ってきたものが違うと、どこが
ズレたのか分からず、もう一度
全部書き直すことになります。

手戻り ← ← ←

1回1つ直す

一言で出して、画面を見てから
1つだけ直します。ズレても、
直前の1つを戻すだけで済みます。
手戻りが小さいまま進みます。

手戻り ←

生成された JavaScript が fetch の失敗を
握りつぶしていないかだけ、一度目を通してください。
使い捨てでも、黙って失敗する画面は困ります。

ここに実画面を差し込む: 完成見本
dashboard_完成見本.html の画面

目指さなくてよい見本です。

休憩

[10min]

アプリは起動したままで構いません。
dashboard.html は消さないでください。
再開後は、受注明細を返せるようにします。

SESSION

05

[25min]

受注明細DTOの新規生成

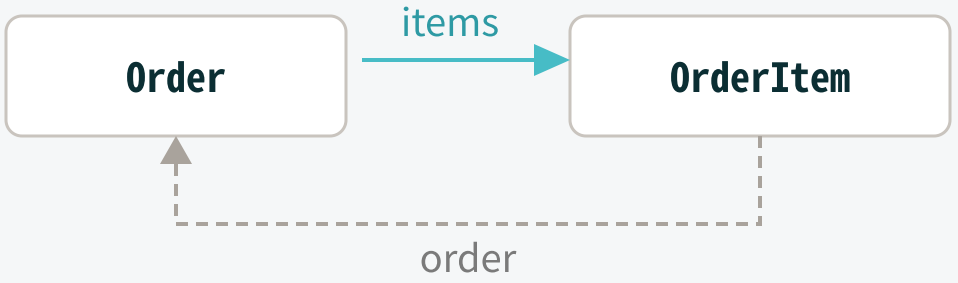
受注のヘッダーは返るのに明細だけ返らない、という抜けを埋めます。
既存メソッドの穴埋めではなく、クラスを1枚まるごと新しく起こします。
複数ファイルにまたがるので、計画を先に確認してから承認します。

導入 | D1-1 | D1-2 | D1-3 | **D1-4** | D1-5 | まとめ

エンティティを直で返さない理由

先に決めるのは、外に出さない項目

エンティティの直返し



親から子・子から親と参照が戻るため、
JSON にするたびに止め方の判断が要ります。
この題材では @JsonIgnore で片方を止めています。
列を足すと、API の出力にも黙って増えます。

OrderItemDTO への詰め替え

id	外す
order	外す
productCode	移す
productName	移す
quantity	移す
unitPrice	移す
subtotal	移す

id と order を外した5項目が OrderItemDTO になります。

items 配列が出たレスポンス

受注1件の JSON に、明細が並ぶ状態

`GET /api/orders/1`　　※ 既存の項目は一部省略

```
{
  "id": 1,
  "orderNumber": "ORD-20260401-001",
  "customerName": "東京電機工業株式会社",
  "totalAmount": 594000,
  "status": "PENDING",
  "items": [
    { "productCode": "RACK-42U-BK",
      "productName": "サーバーラック 42U ブラック",
      "quantity": 3, "unitPrice": 180000,
      "subtotal": 540000 }
  ]
}
```

ダッシュボードに明細欄を足すと、この一致を画面で確かめられます。

確認する2点

id=2 は明細が2件

SW-48P-L3 と
SW-48P-L2 が並びます。

明細が無い受注

null ではなく空の配列
[] を返します。

小計 540000 を1.10倍すると、
ヘッダーの 594000 と一致します。

ワーク

D1-4 OrderItemDTO の新規作成

[25min]

items に明細が出たら達成

1	**やること**	dto に OrderItemDTO を新規作成し、OrderDTO に List<OrderItemDTO> items を足して fromEntity で変換します。
2	**使う操作**	@ で OrderItem.java と OrderDTO.java を指定します。 複数ファイルに入るので、Planモードで計画を見てから承認します。
3	**できたら**	GET /api/orders/1 の items に明細が出ます。 明細が無い受注は、空の配列 [] で返ります。
4	**詳細**	詳細と参考プロンプトは教材サイトの D1-4 カードにあります。

新しいファイルが1枚増えます。作られた場所が dto かどうかも見てください。

補足

差分を読むときの3点

承認する前に、差分を読む一手間

見るところ	ずれていたら
親への参照 order	order が入っていたら、外すよう指摘して投げ直します。
フィールド名	productCode / productName / quantity / unitPrice / subtotal の5つと一致させます。
明細が空のとき	null ではなく、空の配列 [] を返します。

気になる箇所があれば承認せず、指摘を足して投げ直します。
読まずに取り込まない癖を、ここでつけます。

解説

明細が出たあとに残る論点

明細を返せた次は、件数と計算の置き場所

1 受注10件を返すときの問い合わせ回数

GET /api/orders も同じ OrderDTO.fromEntity を通ります。
Order.items は遅延読み込みなので、受注1件ごとに
明細の SELECT が1回ずつ走ります。配布の10件では体感できません。
件数が増えると効いてきます。この形は Day3 の性能ログ分析で扱います。

2 subtotal をどこで担保するかの判断

明細の subtotal をそのまま返すか、
DTO の中で quantity × unitPrice を計算し直すかの判断です。
配布データはどちらで出しても同じ値なので、方針だけ決めます。

06

ステータス遷移ルールの実装

[25min]

状態に応じた可否判定は、業務システムの随所に出てきます
メソッドを丸ごと書き直させず、直したい数行を指定して差分だけを読みます
読む範囲が差分に絞られると、業務ルールが正しいかの確認に集中できます

3か所の穴と遷移の行列

更新・削除・遷移の3か所に穴、足すのは1マス

手を入れる場所

1. updateOrder に PENDING 以外を拒否するチェックが無い
2. deleteOrder にも同じチェックが無い
3. validateStatusTransition の case CONFIRMED が SHIPPED しか許可していない

許可　禁止　今回埋めるマス

遷移元＼遷移先	PENDING	CONFIRMED	SHIPPED	DELIVERED	CANCELLED
PENDING		■			■
CONFIRMED			■		□
SHIPPED				■	
DELIVERED					
CANCELLED					

case CONFIRMED の SHIPPED を消さずに、CANCELLED を足します。

クイズ

遷移ルールを確かめる入口

Q. ステータス遷移のルールが守られているかを確かめるとき、叩くのはどれでしょうか

- **A** PATCH /api/orders/{id}/cancel
- **B** PATCH /api/orders/{id}/status
- **C** PUT /api/orders/{id}
- **D** DELETE /api/orders/{id}

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

入口が2つある理由

正解はB、遷移ルールを通る入口は /status

遷移ルールを確かめるときは、/status に {"status":"CANCELLED"} を送ります。
/cancel ではありません。入口が違うと、通る検証も変わります。

ワーク

D1-5 CONFIRMED からのキャンセル許可

3か所を埋め、id=2 と id=4 で確認

[25min]

やること

OrderServiceImpl の3か所を埋めます。updateOrder と deleteOrder に
PENDING 以外を拒否するチェックを足します。
validateStatusTransition に CONFIRMED → CANCELLED を足します。

使う操作

直したい数行を言葉で指定して、ターミナル対話で頼みます。
Planモード（Shift+Tab）で計画を確認してから承認します。

できたら

id=2（CONFIRMED）に {"status":"CANCELLED"} を送ると 200。
id=4（SHIPPED）への PUT ははじかれ、customerName は変わりません。

詳細

教材サイトの D1-5 カードに、手順とプロンプトの全文があります。

補足

想定どおりの HTTP 500

D1-1 | D1-2 | D1-3 | 休憩 | D1-4 | **D1-5**

受け皿が無いので 500、はじかれた事実が達成

service が投げる
`InvalidOrderStateException`

→

受け皿がまだ無い
`@RestControllerAdvice`

→

既定の動きで返る
HTTP 500

判定に使う2点

id=4 の受注が書き換わっていないこと

起動ログに InvalidOrderStateException が出ていること

ここに実画面を差し込む:

起動ログに InvalidOrderStateException が出ている画面

400 への翻訳は、発展課題の D1-5+ で足します。

id=2 を一度 CANCELLED にすると、遷移元が変わります。やり直すときは
アプリを再起動すると、受注10件の初期状態に戻ります。

解説

埋めたマスと採用前の3点

D1-1 | D1-2 | D1-3 | 休憩 | D1-4 | **D1-5**

1マスが埋まり、残るのは default の扱い

遷移元＼遷移先	PENDING	CONFIRMED	SHIPPED	DELIVERED	CANCELLED
PENDING		■			■
CONFIRMED			■		■
SHIPPED				■	
DELIVERED					
CANCELLED					

■ 許可　□ 禁止　■ 今回埋めたマス

採用前に見る3点

1. 比較は Order.STATUS_PENDING の定数か、"PENDING" の直書きか
2. 例外は InvalidOrderStateException か
3. case CONFIRMED で SHIPPED への遷移を消していないか

同一ステータスへの遷移と、存在しないステータス値の扱いは明示的に書かれていません。
switch に default が無い点に着目すると、埋め残しが見えます。

発展枠の5本

先に自分の方針を決めてから試す枠

[10min]

番号	内容
D1-1+	フィールドの Javadoc 補完
D1-2+	既存シグネチャに合わせたラッパー実装
D1-3+	ステータス別の件数と合計金額をダッシュボードの上に出す
D1-4+	新規エンティティの生成
D1-5+	@RestControllerAdvice を足して 500 を 400 と 404 に変える

追いつきに使っても構いません。次の演習には影響しません。

3操作の分かれ目は変更の範囲

場面	使う操作	現在地の見え方
構造を知りたい・質問だけ	**ターミナル対話**	通常の入力欄
小さな変更を1か所	**ターミナル対話**	通常の入力欄
破壊的な変更・複数ファイル	**Planモード（Shift+Tab）**	入力欄の表示が変わる
決まった手順を毎回同じ形で	**スラッシュコマンド**	/ を打つと候補が出る

スラッシュコマンドは /review と /gen-test の2本。文脈が長くなったら /clear と /compact を使います。

> Planモードは切り替え忘れがいちばん多いので、依頼の前に入力欄を1回見てください。

今日埋めた穴と作った1枚

空けてあった5つの穴が全部埋まった状態

controller	GET /api/orders/date-range を追加	D1-2
service	findByDateRange の本体を実装	D1-2
	validateStatusTransition に CONFIRMED から CANCELLED を追加	D1-5
repository	触っていません。findByOrderDateBetween を呼んだだけです	
dto	OrderItemDTO を新規作成、OrderDTO に items を追加	D1-4
static	dashboard.html をゼロから生成	D1-3

ここまでは、動いたかを画面で確かめてきました。
Day2 は、動いたものが正しいかをテストで確かめます。

次回までの持ち帰り

dashboard.html は消さずに残す1枚

1 **消さないファイル**

Day2 と Day3 でも育てます。
自分の実装が効いたかを、この1枚で確かめます。

2 **最初に試す操作**

3操作のうち、自分の担当コードで最初に試すなら
Planモードです。破壊的な変更や複数ファイルに
またがる依頼の前に、計画を出させて使います。

3 **詳細の置き場所**

演習カードと手順の逐語、参考プロンプトは
教材サイトにあります。
idunder02.give-app.net

3日間の日程

- **Day1　2026年9月3日**
  題材を掴む ＋ 3操作
- **Day2　2026年9月9日**
  テストで裏を取る
- **Day3　2026年9月17日**
  壊れたときに効く動き

質問

質疑応答とアンケート

粒度を問わない質問の時間

[15min]

拾いたい観点

- 自分の現場のコードで同じことをやるときに
  引っかかりそうな点
- 出力がズレたとき、何を渡し忘れたと考えるか
- Planモードと通常対話の切り替えで迷った場面

アンケート

配布されたフォームから、
この後ご回答をお願いします。

質問はどんな粒度でも構いません。

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