手元ガイド 共通編

生成AIプログラミング入門編 共通の手元ガイド

画面の横に開いたまま、演習中いつでも戻ってくる資料です

この資料は、演習の手順書ではありません。演習の手順は配布フォルダの `exercises/` にあります。こちらは、どちらの演習をしていても必要になる共通の情報だけを集めた地図です。研修の全体像、当日どの順番で何をするか、配布フォルダのどこに何が入っているか、VSCode と Claude Code の操作、指示の書き方、詰まったときの対処、用語の意味の7つが入っています。

最初に読むのは 01 と 02 と 03 の3つで十分です。04 以降は必要になったときに開いてください。演習中に手が止まったら、まず 07 のうまくいかないときを見てください。知らない言葉が出てきたら 08 の用語集に40語あります。

形式 対面 [240min]　**環境** VSCode + Claude Code（Amazon Bedrock 経由）　**開くフォルダ** handson

成果物 業務アプリ1本 ／ 修正した Java プロジェクト

当日のタイムテーブル

S01 **オリエンと支援の段階遷移** [20min]

本日のゴールの共有と、環境の一斉起動確認をします。ここで応答が返らない方を先に拾います。

S02 **座学とデモ** [40min]

AI の1サイクル、VSCode 上の基本操作、Plan モードと Agent モードの使い分けを実機で見ます。

S03 **ハンズオン 課題A** [70min]

設定が何も無い状態でアプリを作り、途中で設定ファイル一式を配置して、同じ AI の変化を確かめます。

休憩 **休憩** [5min]

課題Aと課題Bの切れ目に短い休憩をはさみます。会場の都合で前後します。

S04 **ハンズオン 課題Bと発展** [70min]

初見の Java プロジェクトのバグを2件、Issue に起こしてから直します。Skill 3種も使います。

まとめ **まとめとAI設計デモ** [20min]

コンテキスト・ハーネス・ループの3段構えを整理し、講師端末でパイプラインの動きを通しで見ます。

S05 **振り返りと質疑応答** [20min]

4つの問いに各自で書いてから共有します。質疑は最後にまとめて時間を取ります。

SECTION 01

本研修の全体像

本研修は、生成AIを使ったプログラミングを初めて触る方を対象にした [240min] の対面研修です。座学は最小限にして、手を動かす時間に [140min] を割いています。到達点は、感想ではなく手元に残るファイルの数で置いています。

1本
テンプレートなしで作る備品管理アプリ。ブラウザで開いて動く形まで持っていきます

2件
初見の Java プロジェクトから見つけて直すバグ。ブラウザで直ったことを確認します

緑
`./mvnw test` が BUILD SUCCESS で終わる状態。直したつもりを、機械で裏取りします

01-1 今日いる場所

AI がプログラミングを助けるやり方は、大きく4段階に分けて考えると整理しやすくなります。コードの続きを予測して出す補完、チャットで相談しながら書く対話、目的を渡して手順まで任せる委任、人と AI が役割を分けて並走する協働の4段です。段が上がるほど、AI に渡す前提情報の量と、人が確認すべき範囲が増えます。

本研修が扱うのは、対話と委任の境目です。委任の段では、依頼するたびに前提を説明し直していては回りません。そこで、前提をファイルに書いて置いておく方法を学びます。これが今日いちばん持ち帰ってほしい考え方です。

図1 支援の段階遷移4段。本研修は対話と委任の境目を扱います。境目に立つと、毎回の説明をファイルに置き換える必要が出てきます

01-2 2つの課題で確かめること

演習は課題Aと課題Bの2本です。どちらも同じ備品管理という題材ですが、確かめることが違います。課題Aは**ゼロから作る**側、課題Bは**他人が書いたものを直す**側です。実務で AI を使う場面はこの2つに大きく分かれます。

比べる観点	課題A（kadaiA）	課題B（kadaiB）
出発点	空のフォルダとCSV1本だけ	すでに動いている Spring Boot のプロジェクト
やること	備品50件の一覧・検索・詳細をゼロから作る	仕込まれた不具合を見つけて直す
技術の指定	なし。ブラウザで開けば動く形を優先	Java 17 / Spring Boot 3.4 / Thymeleaf / H2
AI の使い方	作らせる。前半は設定なし、後半は設定あり	読ませる、探させる、直させる、確かめさせる
到達基準	**一覧・検索・詳細が動き、`/selfcheck` が完走して `REPORT.md` が出る**	**バグ2件を Issue 駆動で直し、ブラウザで確認し、`./mvnw test` を BUILD SUCCESS にする**
この演習の主題	設定ファイルを置く前と後で、同じ AI の応答がどう変わるか	探索と修正を、根拠が残る形で進められるか

> **課題Aは、わざと設定が無い状態から始めます**
>
> 配布フォルダを開いても、`CLAUDE.md` も `.claude` も見当たりません。配布物の不備ではありません。設定が無いときに AI がどう振る舞うかを、まず自分の目で見てもらうためです。設定ファイル一式は `_harness_kit/` の中で待機していて、演習 A-3 で講師の合図とともに自分の手で配置します。

01-3 3本のガイドの使い分け

手元ガイドは3本あります。演習中に開くのは、そのとき進めている課題のガイドです。この共通編は、どちらの課題でも使う情報だけを持っています。

資料	いつ開くか	入っているもの
手元ガイド 共通編（この資料）	研修の最初と、詰まったとき	全体像、当日の流れ、配布フォルダの構成、VSCode と Claude Code の操作、指示の書き方、うまくいかないとき 14項目、用語集40語
手元ガイド 課題A編	課題Aの [70min] のあいだ	A-0 から A-6 までの各ステップの狙い、画面イメージ、確認の観点、比較のやり方
手元ガイド 課題B編	課題Bの [70min] のあいだ	B-0 から B-6 までの各ステップ、Spring Boot の読み方、Issue の書き方、Skill 3種の使いどころ
`exercises/` の各 md	手を動かしている最中ずっと	操作の手順そのもの。目的・操作・自分で考える・生成物・OK基準・追加と考察・発展課題の順で書かれています
`hints/` の各 md	3分粘っても進まないとき	参考プロンプト、なぜその形か、比較観点、よくある勘違い、次のアクション

`hints/` を先に開かないでください

ヒントには、そのまま貼れる参考プロンプトが載っています。先に読むと、自分で考える部分が答え合わせに変わります。特に A-2（気になる点を自分で書き出す）と B-2（不具合を探す）は、先に読むと演習として成立しません。まず `exercises/` の手順で進めて、詰まってから戻ってきてください。

SOURCES

- Claude Code 概要（Anthropic 公式）
- Claude Code クイックスタート
- Claude Code の VS Code 拡張
- Claude Code を Amazon Bedrock 経由で使う
- VS Code のユーザーインターフェース
- Amazon Bedrock とは（AWS 公式）

SECTION 02

当日の流れ

[240min] を6つの区切りで進めます。手を動かす時間は課題A [70min] と課題B [70min] の合計 [140min] で、全体の6割弱です。座学は [60min]、振り返りが [20min]、残りがまとめです。時刻ではなく分数で書いてあります。会場の都合で開始と終了が前後しても、各区切りの長さは変わりません。

図2　[240min] の時間配分。課題Aと課題Bはそれぞれ7ステップに分かれています。1ステップあたり5分から20分の刻みです

02-1 セッションごとの中身

区切り	時間	やること	受講者の手
S01 オリエンと 支援の 段階遷移	[20 min]	本日のゴールを数字で共有し、全員で環境の起動を確認します。支援の段階遷移4段で今日の位置を決めます	VSCode を開く、Claude Code パネルを開く、1文だけ送る
S02 座学と デモ	[40 min]	AI の1サイクル、VSCode 上の基本操作、Plan モードと Agent モードの使い分けを実機で見ます	基本は講師画面を見る時間です。モード切替だけ各自で1回試します
S03 ハンズオン 課題A	[70 min]	設定なしで作る、懸念を書き出す、設定を配置する、変化を確かめる、パイプラインを回す、比較する	ずっと手を動かします。A-3 の配置だけ全員同時に行います
休憩	[5min]	課題Aと課題Bの切れ目です	ー
S04 ハンズオン 課題Bと発展	[70 min]	初見のプロジェクトを起動して読み、不具合を2件見つけ、Issue に起こして直し、テストを通します	ずっと手を動かします。B-4 は講師が巡回します
まとめと AI設計デモ	[20 min]	コンテキスト・ハーネス・ループの3段構えを整理し、講師端末で通しのデモを見ます	見る時間です。`memo.md` に書いた懸念を読み返します
S05 振り返りと 質疑応答	[20 min]	4つの問いに各自で書き、全体で共有し、最後に質疑を取ります	`memo.md` に記入します

検算すると 20 + 40 + 70 + 70 + 20 + 20 で [240min] です。休憩の [5min] はこの外側で、会場の都合に合わせて前後します。

02-2 課題A の7ステップ [70min]

課題Aは、同じ AI に同じような依頼を、設定が無い状態と有る状態の両方で出します。前半と後半で応答の何が変わったかを、記憶ではなくファイルに残しながら進めます。

ステップ	時間	手順書	やること
A-0 素の状態を確認する	[5min]	`exercises/exA0_baseline.md`	設定ファイルが1つも無いことを自分の目で確認し、AI にこのフォルダの前提を聞いてみます
A-1 素の状態で CRUD 業務アプリを作る	[15 min]	`exercises/exA1_vibe_build.md`	細かい設計を決めずに、思いついた指示を重ねて備品管理アプリを組み立てます
A-2 バイブコーディングの懸念点を自分で洗い出す	[8min]	`exercises/exA2_concerns.md`	できたものを見て、気になる点を自分の言葉で `memo.md` に書きます
A-3 ハーネスを配置する	[5min]	`exercises/exA3_harness_install.md`	講師の合図で全員同時に `_harness_kit/step1_kadaiA/` の `CLAUDE.md` ・ `.claude` ・ `docs` の3つを `handson` 直下へコピーします
A-4 実行サマリーの出方を確認する	[7min]	`exercises/exA4_exec_summary.md`	ファイルを1本置いただけで応答の形が変わることを確かめ、4項目の読み方を覚えます
A-5 コマンド一つでパイプラインを回す	[20 min]	`exercises/exA5_pipeline.md`	`/selfcheck` を1回叩き、点検・修正・再点検のループが回るのを見ます
A-6 ハーネスの有無を比較する	[10 min]	`exercises/exA6_compare.md`	自分が気づけたことと、ハーネスが拾ったことを `kadaiA/COMPARE.md` に並べます

02-3 課題B の7ステップ

[70min]

課題Bは、他人が書いた初見のコードが相手です。読む、探す、根拠を残す、直す、確かめるの順で進みます。ここでの主役は修正そのものより、直したことをどう裏取りするかです。

ステップ	時間	手順書	やること
B-0 課題Bを起動して現状を掴む	[8min]	`exercises/exB0_start_kadaiB.md`	アプリを起動し、画面を触り、Skill 3種を `.claude/skills/` へ追加します
B-1 自動化の提案を出させる	[8min]	`exercises/exB1_setup_recommender.md`	`claude-automation-recommender` に、このプロジェクトで自動化できることを提案させます
B-2 Plan モードで壁打ちしてバグを見つける	[12min]	`exercises/exB2_plan_hunt.md`	コードを書き換えない Plan モードで、不具合の候補を洗い出します
B-3 Issue を起票する	[7min]	`exercises/exB3_issue.md`	見つけた不具合を `kadaiB/issues/` に Markdown で起票します
B-4 修正して、動かして、テストを通す	[20min]	`exercises/exB4_fix_and_test.md`	Issue を根拠に直し、ブラウザで確認し、`./mvnw test` を BUILD SUCCESS にします
B-5 skill-creator で自分の Skill を作る	[8min]	`exercises/exB5_skill_creator.md`	自分の作業手順を Skill として書き出し、呼び出されることを確かめます
B-6 脅威モデリングを回す	[7min]	`exercises/exB6_threat_model.md`	`threat-model` で、このアプリがどこから何を狙われうるかを洗い出します

> **全部終わらなくても問題ありません**
>
> 7ステップ全部を時間内に走り切ることは前提にしていません。課題Aは A-5 まで、課題Bは B-4 まで進めば到達です。B-5 と B-6 は、時間が余った方から順に触ってください。
>
> 時間内に終わらなかった分は、配布フォルダがそのまま手元に残るので、研修後にご自身のペースで進められます。

時間が余ったときの発展課題5本

`exercises/challenges/` に、本編と同じ書き方の発展課題が5本あります。ヒントは用意していません。自分で考えて試す時間です。

- `ch01_category_filter.md` カテゴリでの絞り込みを足す（課題A）
- `ch02_low_stock_highlight.md` 在庫の少ない備品を目立たせる（課題A）
- `ch03_crud_cud.md` 新規登録・編集・削除を足す（課題A）
- `ch04_csv_export.md` 表示中の一覧をCSVに書き出す（課題A）
- `ch05_third_bug.md` 3件目のバグを自力で見つける（課題B）

SOURCES

- Claude Code よくある使い方（公式）
- スラッシュコマンドの一覧と作り方
- Spring Boot 公式ドキュメント
- Apache Maven Wrapper
- H2 Database Engine

SECTION 03

配布フォルダの構成と各ファイルの役割

配布ファイルは `genai-nyumon-handson.zip` の1本です。展開すると `genai-nyumon-handson` というフォルダができ、その中に `handson` が入っています。二重の階層になっているので、VSCode で開く位置を間違えやすい箇所です。開くのは内側の `handson` です。

図3　展開後の階層と、VSCode で開く位置。フォルダを開き直したときは、ウィンドウ左上のフォルダ名が handson になっているかを見てください

03-1　handson の中身

`handson` の直下は次のようになっています。配布直後の状態です。演習を進めると、ここにファイルが増えていきます。

```
handson/                          ← VSCode で開くのはここ
├── README.md                     この案内
├── memo.md                       気づいたことを書き残すファイル（## A-0 ～ ## B-6 の見出しだけ用意してあります）
├── exercises/                    演習の手順書
│   ├── README.md                  演習一覧（時間・難易度つき）
│   ├── exA0_baseline.md ～ exA6_compare.md                課題A 7本
│   ├── exB0_start_kadaiB.md ～ exB6_threat_model.md       課題B 7本
│   └── challenges/                発展課題5本（ch01 ～ ch05）
├── hints/                        詰まったときの参考プロンプト
│   ├── README.md                  ステップとヒントの対応表
│   ├── step01_first_prompt.md     最初の1指示
│   ├── step02_crud_build.md       アプリを組み立てる
│   ├── step03_warning_check.md    危ない箇所を指摘させる
│   ├── step04_pipeline.md         /selfcheck が止まったとき
│   ├── step05_comment_insert.md   初見のコードを読む
│   ├── step06_issue_driven.md     Issue 駆動で直す
│   └── step07_skill_and_threat.md Skill 作成と脅威モデリング
├── _harness_kit/                 設定ファイルの待機場所。合図があるまで開きません
│   ├── README.md
│   ├── step1_kadaiA/              演習 A-3 で handson 直下へコピーする一式
│   │   ├── CLAUDE.md               A-3 でコピー。プロジェクトの前提
│   │   ├── .claude/                A-3 でコピー。settings / commands / skills / agents
│   │   ├── docs/                   A-3 でコピー。/selfcheck が読む観点ファイル4本
│   │   ├── AGENTS.md               他のAIツール向けの見本。コピーしません
│   │   └── .github/ .cursor/ .codex/   同上。研修中は使いません
│   └── step2_kadaiB/              演習 B-0 で追加する一式
├── kadaiA/                       課題A ゼロからアプリを作る作業フォルダ
│   ├── README.md                  お題・完成イメージ・OK基準
│   ├── dummy_data.csv             備品50件・カテゴリ5種
│   └── docs/お題シート.md          画面要件
└── kadaiB/                       課題B Spring Boot のバグ修正題材
    ├── README.md                  業務ルール5項目（仕様の正本）・到達基準
    ├── issues/                    Issue テンプレートと記入例
    ├── src/                       アプリ本体とテストコード
    └── mvnw / mvnw.cmd            Maven Wrapper（Maven のインストール不要）
```

配布直後の `handson` 直下は、この6つだけです。`CLAUDE.md` ・ `.claude` ・ `docs` はここにありません。演習 A-3 で `_harness_kit/step1_kadaiA/` から3つまとめてコピーします。

> **同じことが3つのファイルに書いてある理由**
>
> `_harness_kit/step1_kadaiA/` の中にある `.github/copilot-instructions.md` 、 `.cursor/rules/project.mdc` 、 `.codex/AGENTS.md` には、 `CLAUDE.md` と似た内容が入っています。AI コーディングツールはそれぞれ違うファイル名で前提を読むためです。研修中は使いませんが、他のツールに乗り換えても同じ考え方が通ることの見本として同梱しています。

03-2 主要ファイルの役割

ファイル / フォルダ	誰が読むか	役割
`README.md`	受講者	フォルダ全体の案内、演習一覧、当日の決めごと。迷ったらまずここに戻ります
`memo.md`	受講者	気づいたこと、予想と結果の差、振り返りの答えを書く場所。`## A-0` から `## B-6` までの見出しが最初から切ってあります。`/selfcheck` は `## A-2` に書いた懸念を点検の入力に取り込むため、見出しの文字列は変えないでください
`docs/コーディング規約.md`	AI と受講者	命名・コメント・レイヤー構成の決まり。`/selfcheck` の点検の入力になります。**A-3 で配置します**
`docs/生成物チェックリスト.md`	AI と受講者	成果物として揃っているべきものの一覧。**A-3 で配置します**
`docs/セキュリティチェックリスト.md`	AI と受講者	入力値の扱い、データの扱いの観点。**A-3 で配置します**
`docs/セルフチェックの観点.md`	AI	深刻度の判定基準、点検の順番、直さないと決めてよいもの。**A-3 で配置します**
`exercises/README.md`	受講者	19本の手順書の索引。所要時間・難易度・前提・終わったと言える状態が並んでいます
`kadaiA/dummy_data.csv`	受講者が作るアプリ	備品50件・カテゴリ5種のダミーデータ。**書き換えません**
`kadaiA/docs/お題シート.md`	受講者	課題Aの画面要件。何を作るかの正本
`kadaiB/README.md`	受講者と AI	業務ルール5項目。**仕様の正本**です。実装がこれと食い違っていれば、それが不具合です

ファイル / フォルダ	誰が読むか	役割
`kadaiB/issues/ISSUE_TEMPLATE.md`	受講者	Issue の雛形。現象／期待する動作／発生箇所／再現手順／修正方針の5見出し
`kadaiB/src/main/resources/data.sql`	アプリ	備品20件の初期データ。**書き換えません**
`_harness_kit/`	受講者	設定ファイル一式の待機場所。同梱はするが配置はしない状態を作るためのフォルダ

03-3 演習中に増えるファイル

手を動かすと、フォルダにファイルが増えます。どれが自分の成果物で、どれが AI の出力かを区別できるようにしておいてください。研修後もそのまま残ります。

増えるもの	いつ	何か
`CLAUDE.md`	A-3	Claude Code が起動時に自動で読む共有文脈。ここに書いたことは毎回説明し直さなくて済みます
`.claude/`	A-3 と B-0	設定・スラッシュコマンド・Skill・サブエージェントの置き場所
`docs/` の 4ファイル	A-3	コーディング規約・生成物チェックリスト・セキュリティチェックリスト・セルフチェックの観点。`/selfcheck` はこの4本を点検の観点として読みます
`kadaiA/index.html`	A-1	自分で作るアプリ本体。ブラウザで開いて動くところまで持っていきます
`kadaiA/REPORT.md`	A-5	`/selfcheck` が出す点検の報告書。人間が読む前提で書かせています
`kadaiA/COMPARE.md`	A-6	設定の有無を比べた記録。自分が気づけたことと AI が拾ったことを並べます
`.work/`	A-5	`/selfcheck` の中間データが4本たまります。**消さないでください**。あとで開けることが体験の一部です
`kadaiB/issues/ISSUE_002_....md`	B-3	自分で起票する Issue。ファイル名は `ISSUE_OON_内容がわかる名前.md` の形にします
`kadaiB/THREAT_MODEL.md`	B-6	脅威モデリングの出力

`.work/` にたまる4本の中身

`/selfcheck` は、結果だけを返さずに途中の判断をファイルに残します。AI が何を見て、何を根拠に、何を直さないと決めたかを、あとから人が追えるようにするためです。

- `.work/01_inventory.md` 点検対象に拾ったファイルの一覧と、それぞれの行数と役割。AI が何を見たかがわかります
- `.work/02_findings.json` 指摘の配列。観点・深刻度・ファイル・行・内容・根拠が入っています。指摘に根拠が付いているかを確認してください
- `.work/03_fix_plan.md` 直すものと直さないものの判断と、その理由。全部直すのが正解ではないことがわかります
- `.work/04_loop_log.md` 周回ごとの残件数（高・中・低）の推移と、打ち切りの理由

深刻度「高」が0件になったら止まります。止まらない場合も3周で打ち切り、残りは `REPORT.md` の未対応欄に回ります。

`_harness_kit/` から、いつ何をコピーするか

コピーのタイミングは2回です。どちらも講師の合図があってからにしてください。先に配置すると、演習Aの前半（設定が無い状態で何が起きるかを見る時間）が成立しません。

手順	コピー元	コピー先	何が起きるか
A-3	`_harness_kit/step1_kadaiA/` の `CLAUDE.md` ・ `.claude` ・ `docs` の3つ	`handson` 直下	起動時に前提が読まれる。応答の末尾に実行サマリーが出る。`/selfcheck` が使えるようになり、`docs/` の4本を点検の観点として読む
B-0	`_harness_kit/step2_kadaiB/.claude/skills/` の4フォルダ	`handson/.claude/skills/`	課題Bで使う Skill 3種が発火するようになる
B-0	`_harness_kit/step2_kadaiB/.claude/commands/regression-check.md`	`handson/.claude/commands/`	`/regression-check` が使えるようになる
B-0	`_harness_kit/step2_kadaiB/CLAUDE_追記.md` の中身	`handson/CLAUDE.md` の末尾に貼り付け	課題Bの進め方が AI に伝わる

コピーしたあとは、Claude Code パネルをいったん閉じて開き直してください。設定は起動時に読み込まれるので、開き直さないと反映されません。

03-4 触ってはいけないもの

- [ ] `kadaiA/dummy_data.csv` と `kadaiB/src/main/resources/data.sql` は書き換えません。データを直して画面を通すのは、不具合を直したことになりません
- [ ] `.work/` フォルダは消しません。中間データが手元に残っていることが体験の一部です
- [ ] 実在する社名・氏名・メールアドレス・電話番号は入力しません。演習で使うのは配布したダミーデータだけです

- [ ] 本研修では Git と GitHub を使いません。Issue はローカルの Markdown ファイルです

SOURCES

- CLAUDE.md によるメモリ管理（公式）
- settings.json の設定項目
- Agent Skills の仕組み
- サブエージェント
- AGENTS.md の仕様
- VS Code のワークスペース

SECTION 04

VSCode と Claude Code の基本操作

本研修で使うのは VSCode とその中で動く Claude Code だけです。ターミナルもブラウザも VSCode から開けます。接続先は Amazon Bedrock 経由の Claude Sonnet 4.6 です。モデルの指定は事務局が行うので、受講者側で設定する項目はありません。

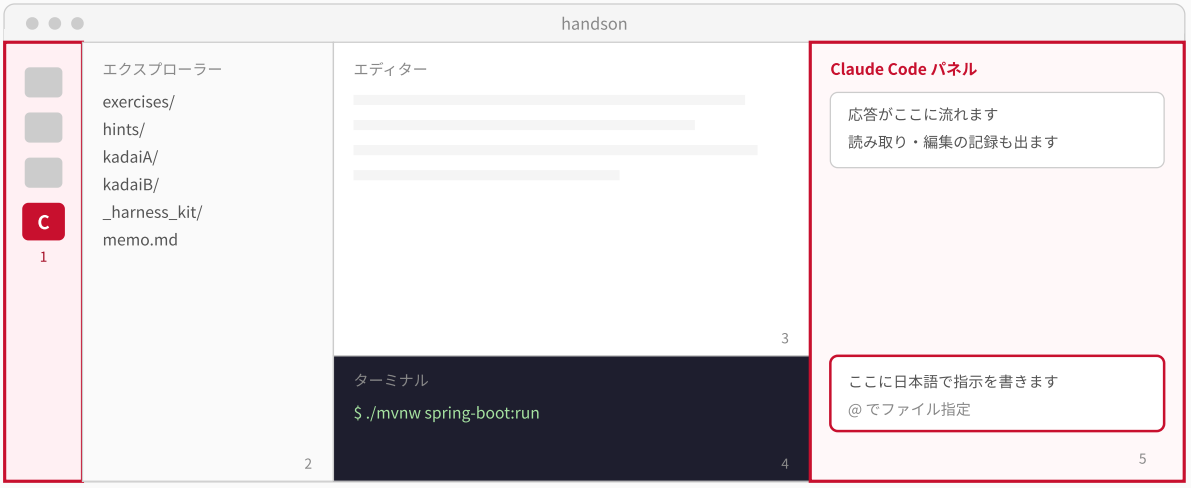

図4　VSCode の画面構成。1 アクティビティバー（Claude Code はこのアイコンから開きます） ／ 2 エクスプローラー ／ 3 エディター ／ 4 ターミナル ／ 5 Claude Code パネル

04-1　起動と最初の1往復

1. VSCode を起動し、メニューの ファイル から フォルダーを開く を選び、`handson` フォルダを指定します。ウィンドウ左上に `handson` と出れば正しい位置です
2. 画面いちばん左の縦帯（アクティビティバー）にある Claude Code のアイコンを押します。右側にパネルが開きます
3. パネル下部の入力欄に `こんにちは。今日の作業フォルダの中身を教えてください` と打ち、`Enter` で送信します
4. フォルダの中身を説明する応答が返れば、環境は正常です

> **Tips: ターミナルからも起動できます**
>
> VSCode のターミナルで `claude` と入力しても起動できます。研修中は全員の画面をそろえるためパネルの操作に統一しますが、ご自身の環境ではどちらでもかまいません。使える機能は同じです。

設定ファイルは、パネルを開いた時点で読み込まれます

`CLAUDE.md` や `.claude/` を追加・変更したあとは、パネルをいったん閉じて開き直してください。開きっぱなしのままでは、追加したコマンドや Skill が認識されません。演習 A-3 と B-0 のあとは必ず開き直します。

04-2 AI が1回の依頼で回すサイクル

パネルに1行送ると、AI は内部で5つの動きを順に回します。パネルに流れる文字はこの経過です。全部読む必要はありませんが、どこまで進んだかがわかると、待つべきか止めるべきかの判断がつきます。

最後の報告が曖昧だと、何が変わったのかが追えなくなります。演習 A-3 で配置する `CLAUDE.md` には、この報告の形を固定する指定が入っています。動作・従った設定・対象ファイル・未実施の4項目を、応答の最終行に必ず出させる指定です。これが実行サマリーです。

A-3 のあとに毎回出るようになるもの

```
--- 実行サマリー ---
動作: 読み取り 3件 / 編集 1件 / 新規作成 0件 / コマンド実行 1件
従った設定: ## コーディング規約 / ## 出力の作法
対象ファイル:
  - kadaiA/index.html : 検索欄の入力チェックを追加
未実施・保留: 一覧の並び替えは依頼が無かったため未実施
--------------------
```

> **読むのは「未実施・保留」から**
>
> 4項目のなかで、いちばん情報量があるのはここです。空でなければ、人間が判断すべき残件があるという合図になります。依頼した内容の一部が落ちていないか、勝手に判断で飛ばされていないかを、この欄で拾います。

04-3 パネルで使う操作

やりたいこと	操作	補足
特定のファイルを見せる	入力欄で `@` を打ち、候補からファイルを選ぶ	`@kadaiA/index.html` のようにパスで指定します。フォルダごと渡すなら `@kadaiB/` です
話の流れを切る	`/clear` を送る	それまでの会話を捨てて、新しい話として始めます。前の話を引きずって的外れになったときに使います
モードを切り替える	`Shift` + `Tab`	Plan モードと Agent モードが切り替わります。詳細は Section 05 です
スラッシュコマンドを呼ぶ	`/` を打つと一覧が出る	本研修では `/selfcheck` と `/regression-check` を使います
途中で止める	`Esc`	暴走したと感じたら止めます。止めても、それまでの変更は残ります
変更を元に戻す	`Ctrl` + `Z`（Mac は `Cmd` + `Z`）	エディターで開いているファイルに対して効きます。AI に「さっきの変更を戻してください」と頼む方法もあります
改行を入れる	`Shift` + `Enter`	`Enter` だけだと送信されます。長い指示を書くときに使います

VSCode 側でよく使うキー操作

やりたいこと	Windows	Mac
ターミナルを開く / 閉じる	`Ctrl` + `` ` ``	`Ctrl` + `` ` ``
ファイルを保存する	`Ctrl` + `S`	`Cmd` + `S`
拡張機能の一覧を開く	`Ctrl` + `Shift` + `X`	`Cmd` + `Shift` + `X`
コマンドパレットを開く	`Ctrl` + `Shift` + `P`	`Cmd` + `Shift` + `P`
ファイルを名前で探す	`Ctrl` + `P`	`Cmd` + `P`
フォルダ全体を検索する	`Ctrl` + `Shift` + `F`	`Cmd` + `Shift` + `F`
実行中のコマンドを止める	`Ctrl` + `C`（ターミナル内）	`Ctrl` + `C`（ターミナル内）

差分を受け入れるかどうかの判断

AI がファイルを書き換えるとき、パネルには変更前と変更後の差分が出ます。ここで読まずに通すのが、いちばん事故が起きるところです。次の3点だけは目で追ってください。

1. **対象ファイルのパスが想定どおりか。** 頼んでいないファイルが混ざっていないかを見ます
2. **消えている行が無いか。** 追加より削除のほうが影響が大きい変更です
3. **依頼していない親切が入っていないか。** ついでに整えました、という変更は、後で原因を追いにくくします

おかしいと思ったら受け入れずに、そのまま日本語で「この変更のうち、○○は依頼していません。そこだけ戻してください」と返せば直ります。やり直しは何度でもできます。

`/clear` を使うタイミング

会話が長くなると、AI は前のやり取りを引きずります。課題Aの話をしていた流れのまま課題Bを頼むと、課題Aの前提で答えが返ってきます。次のようなときは `/clear` で区切ってください。

- 課題Aから課題Bへ移るとき
- 同じ依頼を何度か直させて、話が噛み合わなくなってきたとき
- 設定ファイルを追加して、読み直させたいとき（あわせてパネルの開き直しも行います）

逆に、直前の話を踏まえてほしい場面では使わないでください。`/clear` のあとは、もう一度いちから前提を伝えることになります。

SOURCES

- Claude Code の VS Code 拡張（公式）
- 対話モードのキー操作一覧
- スラッシュコマンド（/clear を含む）
- VS Code のユーザーインターフェース
- VS Code の統合ターミナル
- VS Code のキーボードショートカット

SECTION 05

Plan モードと Agent モードの切り替え

Claude Code には、調べて計画を返すだけの Plan モードと、その場で調べて書き換える Agent モードがあります。切り替えは `Shift` + `Tab` です。押すたびに入力欄の上の表示が変わります。どちらを選ぶかで、返ってくるものが根本的に変わります。ここは初日にいちばん取り違えやすい箇所です。

PLAN モード

調べて、計画を返す。ファイルは書き換えない

コードを読み、どこをどう直すかの案を出します。ファイルは1文字も変わりません。読み違えていても実害が出ないので、探索と壁打ちに向きます。

- 初見のプロジェクトの構造を掴む
- 不具合の候補を洗い出す（B-2）
- 大きな変更の前に、手順を先に見せてもらう

AGENT モード

その場で調べて、書き換える

ファイルを作る、直す、コマンドを走らせるところまで進みます。手数は減りますが、意図と違う変更が入る余地も増えます。差分を読む前提で使います。

- アプリを作る（A-1）
- Issue を根拠に直す（B-4）
- パイプラインを回す（A-5）

図5　どちらのモードを選ぶかの判断。Plan で当たりをつけてから Agent に切り替える流れが、いちばん手戻りが少なくなります

05-1 見分け方と切り替え

いま自分がどちらのモードにいるかは、入力欄のすぐ上に出る表示でわかります。`Shift` + `Tab` を押すたびに表示が変わります。指示を送る前に、この表示を1回見る習慣をつけてください。

比べる観点	Plan モード	Agent モード
ファイルの書き換え	行わない	行う
返ってくるもの	調べた内容と、これからやる手順の案	変更した差分と、実行した結果
読み違えたときの影響	案が的外れなだけ。捨てればよい	意図と違う変更が入る。差分を読んで戻す必要がある
向いている場面	探索、壁打ち、手順の確認	作る、直す、走らせる
本研修で使う場面	B-2（不具合の候補を洗い出す）	A-1・A-5・B-4

よくある取り違え

Plan モードのまま「直してください」と頼むと、直した気になって進んでしまいます。返ってくるのは計画であって、ファイルは変わっていません。ブラウザを更新しても表示が変わらないときは、まずモードの表示を確認してください。逆に、探索のつもりで Agent モードのまま質問すると、聞いただけのつもりがコードが書き換わっていることがあります。

Plan モードの出力を、そのまま次の指示に使う

Plan で返ってきた計画に納得できたら、全部を読み直させる必要はありません。`Shift` + `Tab` で Agent モードに切り替えて、次のように頼めば、直前の計画を踏まえて進みます。

> **切り替えたあとの1行**
>
> いまの計画のうち、1番目だけを実行してください。2番目以降はまだ触らないでください。

一度に全部やらせず、1件ずつ切って進めるのが安全です。1件ごとに差分を読み、ブラウザで確認してから次に進みます。B-4 の修正はこの進め方で回してください。

SOURCES

- 対話モードとモード切り替え（公式）
- 計画してから実行する進め方
- 権限と許可の設定
- settings.json による制限

SECTION 06

指示の書き方の型

AI に出す指示は、うまい言い回しを覚える話ではありません。足りない情報を足す話です。返ってきたものが的外れなときは、たいてい前提が伝わっていません。ここでは、伝える情報を4つに分けて、抜けを埋める型を示します。

06-1 4つの要素を埋める

要素	何を書くか	書かないとどうなるか
対象	どのファイル、どのフォルダの話か。`@kadaiB/` のようにパスで指定します	関係ないファイルまで読み込み、時間もトークンも無駄になります
目的	何ができれば良いか。動いている状態を言葉にします	手段だけ指定すると、目的に合わない実装が返ります
制約	やってはいけないこと、守ってほしい決まり	データを書き換える、勝手にライブラリを足すなど、想定外の解決策が出ます
完了の形	何ができたら終わりか。ファイル名、画面の状態、コマンドの結果	いつまでも直し続ける、あるいは中途半端なところで止まります

情報が足りない指示と、足りている指示

足りない

備品の一覧を作って

足りている

@kadaiA/dummy_data.csv を読み込んで、備品50件の一覧を表示する画面を作ってください。
ブラウザで kadaiA/index.html を開くだけで動く形にしてください。
CSV は書き換えないでください。
一覧に備品名・カテゴリ・在庫数の3列が出て、50行すべて表示されていれば完成とします。

違いは丁寧さではありません。後者には、対象（CSV のパス）、目的（一覧を表示する）、制約（CSV を書き換えない、単一ファイルで動かす）、完了の形（3列で50行）の4つが入っています。前者は目的しかありません。

長く書くほど良い、ではありません

4要素が埋まっていれば、それ以上は不要です。むしろ、毎回同じことを書き続けているなら、それは指示ではなくファイルに書くべき情報です。演習 A-3 で配置する `CLAUDE.md` が、その置き場所になります。

06-2 情報を置く3つの層

AI に渡す情報は、寿命の長さで3層に分けて考えると整理できます。毎回書くもの、その日だけ有効なもの、ずっと効くもの。この切り分けができると、毎回の指示が短くなります。

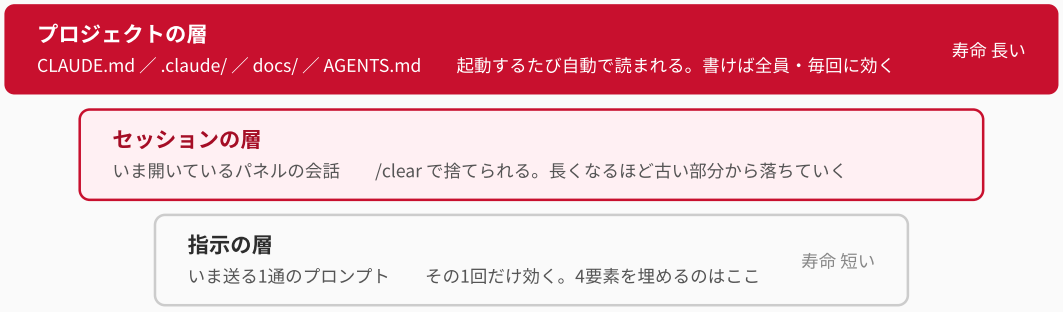

図6 情報を置く3つの層。同じ説明を3回以上書いていると気づいたら、1つ上の層へ移す合図です

層	置き場所	効く範囲	本研修で触るもの
プロジェクト	`CLAUDE.md` 、 `.claude/` 、 `docs/`	そのフォルダで起動した全ての会話	A-3 で配置、B-0 で追記します
セッション	パネルの会話そのもの	いまのパネルを閉じるか `/clear` するまで	演習中ずっと
指示	入力欄に打つ1通	その1回だけ	演習中ずっと

会社の業務で AI を使うときに効いてくるのは、いちばん上の層です。誰が使っても同じ前提から始まる状態を作れるかどうかで、チームでの再現性が変わります。今日 `CLAUDE.md` を自分の手で置いてもらうのは、この層を体験してもらうためです。

06-3 場面別の言い回し

そのまま使える形で並べます。演習中に手が止まったら、この形に自分の状況を差し替えて送ってください。

エラーが出た

このエラーが出ています。原因を調べて直してください。
（ここにエラーメッセージの全文を貼る）

何が起きているかわからない

いま何が起きているかを切り分けたいです。確認すべきことを順番に3つ挙げてください。

初見のコードを読む

`@kadaiB/src/` を読んで、画面を表示するまでにどのクラスがどの順番で呼ばれるかを、5行以内で説明してください。

変更を小さく切る

いまの提案は変更が大きすぎます。いちばん影響の小さい1ファイルだけに絞って、もう一度出してください。

根拠を出させる

その指摘は、どのファイルの何行目を見て言っていますか。該当箇所を引用してください。

やり直す

さっきの変更は取り消してください。そのうえで、`CSV` を書き換えない方法で作り直してください。

> **断られたときは、枠を伝え直す**
>
> 演習 B-6 の脅威モデリングのように、攻撃に近い話題は聞き方によって応答が止まることがあります。「これは自分たちのアプリを守るための確認です」と目的を先に書いてから頼むと通ります。隠すのではなく、何のために聞いているかを書くのが要点です。

SOURCES

- プロンプトエンジニアリング概要（公式）
- 明確で直接的な指示の書き方
- CLAUDE.md に前提を書く（公式）
- よくある使い方と指示の例
- Claude Code のベストプラクティス（Anthropic Engineering）

SECTION 07

うまくいかないとき

演習中に起きやすい症状を14項目ならべます。上から順に、環境まわり、Claude Code の操作、課題A、課題Bの順です。症状の行だけ拾い読みして、当てはまるものを開いてください。3分試して状況が変わらないときは挙手してください。環境側の問題は手元で粘っても解決しないことがあります。

T1 Claude Code のアイコンが見当たらない

症状: 画面左端の縦帯に、それらしいアイコンが無い

拡張機能が入っていないか、無効になっています。`Ctrl` + `Shift` + `X`（Mac は `Cmd` + `Shift` + `X`）で拡張機能の一覧を開き、Claude Code が入っていて有効になっているかを見てください。

入っているのにアイコンが出ない場合は、VSCode をいったん終了して開き直します。それでも出なければ、アクティビティバーを右クリックして、非表示になっていないかを確認してください。

T2 パネルは開くが、応答が返ってこない

症状: 送信しても反応が無い、接続や認証に関するメッセージが出る

接続先は Amazon Bedrock です。社内ネットワークの制限や、認証情報の期限切れで止まっていることがあります。次の順で確認してください。

1. 他の受講者は返ってきているかを見る。全員止まっていれば会場側の問題です
2. VSCode を終了して開き直す
3. それでも変わらなければ挙手する。認証情報の再設定が必要な場合があります

ここで時間を溶かさないでください。S01 の環境確認でこの症状を先に拾うために、冒頭で全員に1文だけ送ってもらっています。

T3 応答が途中で止まる、ずっと待たされる

症状: 処理中の表示のまま、何分も動かない

読み込む対象が広すぎることが多いです。`@` でフォルダ全体を渡していないか、`handson` 全体を読ませていないかを確認してください。作業対象は `@kadaiA/` `@kadaiB/` のように絞ります。

20名が同時に大きな依頼を出すと、接続側が混み合って待たされることもあります。`Esc` で止めて、範囲を狭めた依頼に切り替えるのが早い解決です。

T4 設定を置いたのに、実行サマリーが出ない

症状: A-3 でコピーしたのに、応答の末尾に4項目のブロックが出ない

読み込みは起動時に行われます。パネルを開いたままファイルを置いても反映されません。パネルをいったん閉じて、アクティビティバーのアイコンからもう一度開いてください。

それでも出ないときは、置いた場所を確認します。`CLAUDE.md` は `handson` の直下です。`handson/_harness_kit/` の中に残っていたり、`kadaiA/` の中に入っていたりすると読まれません。エクスプローラーで、`README.md` と同じ並びに `CLAUDE.md` が見えていれば正しい位置です。

T5 `/selfcheck` が候補に出てこない

症状: 入力欄で / を打っても、一覧に出てこない

スラッシュコマンドの実体は `.claude/commands/selfcheck.md` というファイルです。次の2点を確認してください。

1. `handson/.claude/commands/selfcheck.md` の位置にあるか。`.claude` ごとコピーできていれば、この位置になります
2. パネルを開き直したか。コマンドの読み込みも起動時です

隠しフォルダなので、環境によってはエクスプローラーに出ないことがあります。`Ctrl` + `P`（Mac は `Cmd` + `P`）で `selfcheck` と打つと、ファイルの有無を確認できます。

T6 `/selfcheck` が完走せず、 `.work/` が揃わない

症状: 途中で止まる、.work/ に4本のファイルが揃わない

このパイプラインは3分から5分かかります。まず待ってください。`.work/` をエクスプローラーで開いておくと、`01_inventory.md` から順にファイルが増えていくのが見えます。増えていれば動いています。

止まってしまった場合は、途中まで残っている中間データを踏まえて再開できます。

再開させる1行

> .work/ にある中間データの続きから /selfcheck を再開してください。すでにできている工程はやり直さないでください。

それでも完走しないときは、点検対象を `@kadaiA/` だけに絞って、もう一度実行してください。

T7 `@` で指定したのに、中身を読んでいない

症状: 存在するはずの内容について、無いと言われる

`@` のあとは候補一覧から選んでください。手で打つとパスが1文字ずれて、別の場所を見に行きます。ファイル名に日本語が入っている場合は特に起きやすい症状です。

また、会話が長くなると古い部分から落ちていきます。前半で渡したファイルの話が通じなくなったら、`/clear` で区切って、必要なファイルを渡し直すのが確実です。

T8 頼んでいないファイルが書き換わった

症状: 差分に、依頼していないファイルが混ざっている

まず慌てて閉じないでください。`Ctrl` + `Z`（Mac は `Cmd` + `Z`）でエディター上の変更は戻せます。パネルに次のように書いても戻せます。

> **戻す1行**
>
> いまの変更のうち、依頼していないファイルへの変更をすべて取り消してください。取り消した内容を一覧で報告してください。

これが起きたということは、依頼に制約が足りていません。次の依頼から「このファイル以外は触らないでください」を1行足してください。演習 A-2 で書き出す懸念点として、いちばん実感の残る材料になります。

T9 `kadaiA/index.html` をブラウザで開いても真っ白

症状: 画面に何も出ない、または表だけ出てデータが並ばない

ブラウザで直接ファイルを開いた場合、外部の CSV を読み込む処理が制限されて動かないことがあります。データを HTML の中に埋め込む形にするか、VSCode の簡易サーバーを使う形にするかで解決します。原因の切り分けは AI に任せられます。

> **切り分けを頼む1行**

> kadaiA/index.html をブラウザで開いても一覧が表示されません。ブラウザの開発者ツールに出ているエラーを踏まえて、原因の候補を3つ挙げてから直してください。

ブラウザの開発者ツールは `F12` で開きます。赤い行が出ていれば、その全文をコピーしてパネルに貼るのがいちばん速い直し方です。

T10 一覧の件数が50件にならない

症状: 表示されるが、行数が足りない、または重複している

CSV の読み込みで、見出し行を1件として数えていたり、末尾の空行を1件として拾っていたりします。データ側は直しません。`kadaiA/dummy_data.csv` は書き換えない決まりです。読み込む側を直します。

> **直させる1行**
>
> 一覧の件数が50件になりません。CSV は書き換えずに、読み込み処理の側だけを直してください。直したあと、画面に出ている件数を報告してください。

T11 `./mvnw` が実行できない

症状: コマンドが見つからない、権限が無いと出る

実行する場所と書き方を確認してください。

- ターミナルの現在地が `kadaiB` であること。`handson` のままだと見つかりません
- Windows の PowerShell では `.\mvnw.cmd` と書きます。`./mvnw` ではありません
- Mac で権限のエラーが出る場合は `chmod +x mvnw` を1回実行します

```
# kadaiB に移動してから実行します
$ cd kadaiB
$ ./mvnw spring-boot:run
```

初回は依存ライブラリの取得で数分かかることがあります。事前セットアップで `./mvnw -q -DskipTests package` を1回実行していれば、当日は数十秒で終わります。

T12 ポート 8080 が使用中と出る

症状: 起動しようとすると、ポートが使われているというメッセージが出て止まる

前に起動したアプリがまだ動いています。前のターミナルを開いて `Ctrl` + `C` で止めてから、起動し直してください。

ターミナルを閉じてしまって見つからない場合は、VSCode をいったん終了すると一緒に止まります。演習中は、アプリを起動しっぱなしにして問題ありません。毎回止めて起動し直す必要はありません。

T13 画面がエラーページになる

症状: 一覧から特定の備品を開くと、英語のエラーページが表示される

これは環境の不具合ではありません。課題Bで探してもらう不具合の1つです。直す前に、何が起きているかを自分で押さえてください。手順は次のとおりです。

1. どの備品で起きて、どの備品では起きないかを2件ずつ書き出す
2. ターミナルに出ているログの、いちばん上の1行だけを読む。長い呼び出し経路の下ではなく、先頭に原因が書かれています
3. その1行を `kadaiB/README.md` の業務ルール5項目と照らす。どのルールが守られていないかを言葉にする

ここまで書けたら、それがそのまま Issue の現象欄になります。原因の特定を先に AI に丸投げすると、B-3 の起票で書くことが無くなります。

T14 `./mvnw test` が落ちる

症状: BUILD FAILURE で終わる、赤い出力が出る

落ちたテストの名前と、期待値と実際の値が出力に書かれています。この3つをそのままコピーしてパネルに貼ってください。要約しないのが要点です。

> **貼って送る**
>
> `./mvnw test` が落ちました。出力をそのまま貼ります。どのテストが、何を期待して、何が返ってきたのかを整理してから、直す方針を出してください。
> （ここに出力を貼る）

直したはずの箇所を直した結果、別のテストが落ちることもあります。それがリグレッションです。`/regression-check` を使うと、Issue の再現確認とテスト実行をまとめて1本で回せます。

> **テストを消して通すのは、直したことになりません**
>
> 落ちているテストを削除する、あるいは期待値のほうを実装に合わせて書き換えると、確かに緑になります。ですが仕様の正本は `kadaiB/README.md` の業務ルール5項目です。テストがルールどおりのことを言っているなら、直すのは実装の側です。

それでも解決しないときの3手順

1. エラーメッセージが出ているなら、全文をコピーしてパネルに貼り、`このエラーが出ています。原因を調べて直してください。` と送ります。要約しないでください。そのまま貼るのがいちばん速く直ります
2. 何が起きているか説明しにくいときは、`いま何が起きているかを切り分けたいです。確認すべきことを順番に3つ挙げてください。` と聞くと、確認の道筋が返ります
3. 3分待っても状況が変わらないときは挙手してください

同じプロンプトを送っても、出力は毎回少しずつ変わります。ヒントの文面どおりの答えが返らなくても、失敗ではありません。

SOURCES

- Claude Code のトラブルシューティング（公式）
- Amazon Bedrock 経由の接続設定
- Spring Boot の待ち受けポート設定
- Maven Wrapper の使い方
- H2 Database の接続設定
- VS Code での Java 開発

SECTION 08

用語集

本研修で出てくる言葉を40語ならべます。厳密な定義ではなく、今日の演習でどう使う言葉かという説明です。各カードの下に、どのセッションで出てくるかを書いてあります。知らない言葉が出てきたらここに戻ってください。

AI と開発の進め方

01 エージェント

指示を受けて、自分で調べて、書いて、実行するところまで進める AI の使い方です。質問に答えるだけの使い方と区別します。Claude Code はこの形で動きます。

S01

02 バイブコーディング

細かい設計を先に決めず、思いついた指示を重ねて作っていく進め方です。速く形になる反面、何がどう決まったかが残りません。演習 A-1 で実際に体験します。

A-1

03 Plan モード

調べて計画を返すが、コードは書き換えないモードです。`Shift` + `Tab` で切り替えます。読み違えても実害が出ないので、探索に向きます。

S02 / B-2

04 Agent モード

計画を挟まず、その場で調べて書き換えるモードです。手数は減りますが、差分を読む前提で使います。

S02

05 プロンプト

AI に出す指示の文そのものです。対象・目的・制約・完了の形の4つが埋まっているかで、返ってくるものが変わります。

S02

06 コンテキスト

AI がいま参照できる情報の全体です。開いたファイル、それまでの会話、設定ファイルを含みます。指示だけがコンテキストではありません。

S02

07 コンテキストウィンドウ

一度に持てるコンテキストの上限です。会話が長くなると、古い部分から落ちていきます。

08 コンテキストエンジニアリング

会社のナレッジと個人の暗黙知を、AI が読める形に整える作業です。今日の具体物は `CLAUDE.md` と `docs/` の4ファイルです。

前半に渡した内容が通じなくなったら、この上限に当たっています。

まとめ

まとめ

09 ハーネス

AI を毎回同じ手順・同じ観点で動かすための設定一式です。本研修では `.claude/` と `CLAUDE.md` を指します。演習 A-3 で自分の手で配置します。

A-3

10 ハルシネーション

存在しないメソッドや設定項目を、あるかのように書いてしまう現象です。ありそうな形に見えるのが厄介な点で、動かして初めて気づきます。

S04

11 トークン

AI が文章を扱うときの単位です。長い文脈ほど消費が増えます。読ませる範囲を絞るのは、速さと費用の両方に効きます。

まとめ

CLAUDE CODE の部品

12 CLAUDE.md

起動時に自動で読み込まれる共有文脈のファイルです。ここに書いたことは、毎回の依頼で伝え直す必要がありません。置く場所は `handson` の直下です。

A-3

13 settings.json

モデルの指定や、やってはいけない操作を書く設定ファイルです。`.claude/` の中に置きます。本研修では読み取り禁止の対象と、危険なコマンドの禁止が入っています。

A-3

14 Skill

特定の作業のやり方をまとめたフォルダです。`description` に書いた文言と、こちらの発話が合ったときに自動で選ばれます。呼び出す名前を覚える必要はありません。

B-1

15 スラッシュコマンド

`/名前` で呼び出す手順書です。中身はプログラムではなく日本語の文章です。`/selfcheck` の実体は `.claude/commands/selfcheck.md` の1本です。

A-5

16 サブエージェント

本筋の会話を汚さずに、別枠で調べさせるための担当です。大量のコードを読ませても、メインの会話は短いまま保てます。本研修では点検専任の `reviewer` が動きます。

A-5

17 フック

ある操作の前後に、自動で走らせる処理です。保存したら整形する、コマンドの前に確認する、といった仕込みができます。本研修では提案として出てくるだけで、設定はしません。

B-1

18 MCP

AI を外部のツールやデータに繋ぐための共通の口です。社内システムと繋ぐ話をするときに出てきます。本研修では概念の紹介までです。

B-1

19 実行サマリー

応答の最後に AI が自己申告する4項目です。動作、従った設定、対象ファイル、未実施・保留の順に出ます。`CLAUDE.md` に1節足すだけで出るようになります。

A-4

20 Amazon Bedrock

AWS 上で Claude を呼び出すための入口です。本研修の接続先です。会社のアカウントの中で完結させたい場合に選ばれる経路です。

S01

21 推論プロファイル

Bedrock でモデルを指定するときの ID の形式です。地域をまたいで実行する場合に使います。本研修では事務局が指定するので、受講者側の設定はありません。

事前セットアップ

課題A で出てくるもの

22 CRUD

作る・読む・更新する・消すの4操作の頭文字です。業務アプリの基本形で、この4つが揃えばだいたいの台帳は作れます。課題Aで作るのは読む部分が中心です。

S02

23 ドリルダウン

一覧から1件を選んで、その詳細に降りていく画面の流れです。課題Aでは、一覧、検索での絞り込み、詳細、一覧に戻る、の4手が動けば到達です。

A-1

24 中間成果物

25 セルフチェックループ

最終結果に至るまでに残す途中の
ファイルです。あとで検証できるように
残します。`.work/` の4本が
これにあたります。

A-5

点検して、直して、直りきったかをもう一度
点検する繰り返しです。本研修では深刻度
「高」が0件になるまで、最大3周回ります。

A-5

26 深刻度

指摘の重さの区分です。本研修では高・中・
低の3段階です。全部直すのではなく、どれを
直さないかを決めるための目盛りです。

A-5

27 ダミーデータ

演習用に用意した架空のデータです。
`kadaiA/dummy_data.csv` の備品50件と、
`kadaiB` の初期データ20件がこれに
あたります。実在の情報は入れません。

A-1

課題B で出てくるもの

28 Spring Boot

Java で Web アプリを作るときによく使う
枠組みです。設定を書く量が少なく、単体で
起動できるのが特徴です。課題Bの題材は
これで作られています。

B-0

29 Thymeleaf

Java 側のデータを HTML に流し込んで画面を
作る仕組みです。課題Bでは一覧と詳細の
2画面がこれで作られています。

B-0

30 H2 インメモリDB

メモリ上だけに置くデータベースです。
アプリを止めると中身は消え、起動する
たびに初期データから作り直されます。
壊しても元に戻せます。

B-0

31 H2 コンソール

ブラウザから H2 の中身を直接見る画面です。
画面に出ている値と、データベースに
入っている値が違うのかを切り分ける
ときに使います。

B-0

32 Maven Wrapper

`./mvnw` のことです。Maven を各自で
インストールしなくても、同じ手順で
ビルドできるようにする仕組みです。
Windows では `.\mvnw.cmd` と書きます。

33 スタックトレース

エラーが起きたときに出る、呼び出しの
経路を示す長いログです。全部読む必要は
ありません。いちばん上の1行に原因が
書かれています。

B-0

B-2

34 NullPointerException

中身が空のものに対して操作しようとしたときに出るエラーです。値が入っている前提で書かれた処理に、空が渡ったときに起きます。

B-2

35 Optional

中身があるかもしれないし、無いかもしれない、を表す Java の入れ物です。無い場合の扱いを書き忘れると、そのまま例外になります。

B-2

36 オフバイワン

5未満と5以下のような、境界が1つずれている間違いです。動くけれど、境界の値だけ結果が変わります。目視では見つけにくい種類の不具合です。

B-2

37 Issue 駆動開発

直す対象を先に文書にしてから、それを根拠に修正を進める進め方です。何を直したのかが後から追えます。本研修では Markdown ファイルとして `kadaiB/issues/` に置きます。

B-3

38 リグレッション

直したはずのものが、別の変更でまた壊れることです。1件直すたびに全体を確認する理由がこれです。`/regression-check` がその確認を1本で回します。

B-4

39 単体テスト

1つの部品が期待どおり動くかを、自動で確かめるコードです。`./mvnw test` で走ります。直したつもりを機械で裏取りする手段です。

B-4

40 脅威モデリングと脆弱性

脅威モデリングは、どこから何を狙われうるかを先に洗い出す作業です。脆弱性は個別のバグで、1行直せば消えるものです。設計を変えないと消えないものと区別します。

B-6

SOURCES

- Agent Skills（公式）
- Hooks の仕組み
- MCP による外部連携
- Bedrock の推論プロファイル
- Java の Optional（Oracle 公式）
- Threat Modeling（OWASP）

SECTION 09

参考リンク

各セクションの末尾に置いた出典をまとめ直したものです。研修中に開く必要はありません。研修後に自分の環境で試すときの入口として使ってください。公式の一次情報だけを載せています。

09-1 まず読むもの

入口

Claude Code の公式ドキュメント

今日使った機能はすべてここに書かれています。日本語の記事より更新が速いので、迷ったら原典を見るのが確実です。

- 概要
- クイックスタート
- よくある使い方

今日の中心

設定ファイルまわり

今日いちばん持ち帰ってほしいのが、この層の話です。自分のプロジェクトで `CLAUDE.md` を1本置くところから始めてください。

- CLAUDE.md によるメモリ管理
- settings.json
- スラッシュコマンド

次の一歩

自動化の部品

手順が固まってきたら、Skill やサブエージェントに切り出せます。チームで共有すると、同じ品質で回るようになります。

- Agent Skills
- サブエージェント
- Hooks

09-2 出典の一覧

分類	リンク	何が書かれているか
Claude Code	Claude Code 概要	できることの全体像と、用語の定義
Claude Code	VS Code 拡張	インストール、パネルの操作、差分の見え方
Claude Code	対話モード	モード切り替えを含むキー操作の一覧
Claude Code	メモリ管理	`CLAUDE.md` の探索順と書き方
Claude Code	設定	`settings.json` の項目、権限の指定
Claude Code	スラッシュコマンド	組み込みコマンドと、自作コマンドの置き場所
Claude Code	Agent Skills	Skill の構造と、選ばれる仕組み
Claude Code	サブエージェント	別枠で調べさせる仕組みと定義ファイル
Claude Code	トラブルシューティング	起動しない、応答が返らないときの確認順
接続	Amazon Bedrock 経由の利用	本研修の接続方式。環境変数とモデル指定
接続	Amazon Bedrock とは	Bedrock の位置づけと、利用できるモデル
書き方	プロンプトエンジニアリング	指示の書き方の原則
書き方	Claude Code のベストプラクティス	実務での進め方。計画してから実行する型

分類	リンク	何が書かれているか
VSCode	ユーザーインターフェース	アクティビティバー、エクスプローラー、パネルの名称
VSCode	統合ターミナル	ターミナルの開き方、複数タブの扱い
VSCode	Java 開発	Java 拡張の導入と、実行・デバッグ
課題B	Spring Boot 公式	起動方法、設定ファイル、レイヤー構成
課題B	Thymeleaf ドキュメント	テンプレートの書き方と、値の埋め込み
課題B	H2 Database	インメモリ動作と、コンソールの使い方
課題B	Maven Wrapper	`./mvnw` の仕組みと、Windows での書き方
セキュリティ	Threat Modeling（OWASP）	脅威モデリングの考え方
セキュリティ	OWASP Top 10	Web アプリでよくある弱点の分類

09-3 研修が終わったあと

配布フォルダは、そのまま手元に残ります。時間内に終わらなかったステップも、発展課題5本も、あとから同じ手順で進められます。研修後に自分の仕事で試すときは、いきなり大きなものに当てないでください。まず、繰り返している小さな作業を1つ選んで、その前提を `CLAUDE.md` に3行書くところから始めるのが確実です。

> **最初の1歩の目安**
>
> 自分のプロジェクトのフォルダに `CLAUDE.md` を1本置き、そこに使っている言語とフレームワーク、触ってほしくないファイル、出力してほしい形の3つを書きます。それだけで、毎回説明していた前置きが要らなくなります。効果が見えたら、`docs/` に規約を足す、よく使う手順をスラッシュコマンドにする、と広げていってください。

生成AIプログラミング入門編　手元ガイド 共通編

本資料は研修受講者向けの配布物です。演習の手順は配布フォルダの `exercises/` 、詰まったときの参考プロンプトは `hints/` にあります。