S03 ハンズオン 課題A / 70分

手元ガイド 課題A編

設定ゼロの状態から始めて、コマンド1本が回るところまで

課題Aの70分は、同じ AI を「設定が何も無い状態」と「設定ファイル一式を置いた状態」の両方で使い、その差を自分の手元のファイルとして残すために使います。前半の A-0 から A-2 では、テンプレートも規約も無いところから備品管理アプリを立ち上げ、できたものへの不安を自分の言葉で書き出します。後半の A-3 から A-6 では、配布フォルダの中で待機している設定ファイルを配置し、応答の形が変わることを確かめ、最後にコマンド1本で点検から報告書までを回します。

このページは演習中に手元で開いておく資料です。手順の本体は配布フォルダの `exercises/exA0_baseline.md` から `exA6_compare.md` にあり、内容は同じです。画面で図を見ながら進めたいときはこのページを、ファイルに書き込みながら進めたいときは `exercises/` を使ってください。

時間 70分（A-0 から A-6 の7ステップ）　**開くフォルダ** handson（最初から最後まで固定）

環境 VSCode × Claude Code（Amazon Bedrock 経由 / Sonnet 4.6）

主な生成物 `kadaiA/index.html` / `REPORT.md` / `COMPARE.md`

SECTION 01

70分の地図

課題Aは7つのステップに分かれています。前半3つは設定ファイルが1つも無い状態で進み、A-3で設定ファイル一式を配置してから、後半3つが動きます。折り返し地点がどこにあるかを最初に頭に入れておくと、いま自分がどちら側にいるかを見失いません。

図8 演習A 7ステップのロードマップ。帯の幅が所要時間の比率です。A-1 と A-5 の2つで70分の半分を使います。この2つが山場です。

前半と後半で何が変わるのか

A-3 の前後で変わるのは AI の性能ではありません。同じ Sonnet 4.6 が同じフォルダを見ています。変わるのは、AI が起動時に読み込める材料の量です。素の状態では、AI が知っているのはあなたが送った文と、AI が自分で開いたファイルの中身だけです。
設定ファイルを置くと、そこに書いた前提が毎回の会話の出発点になります。

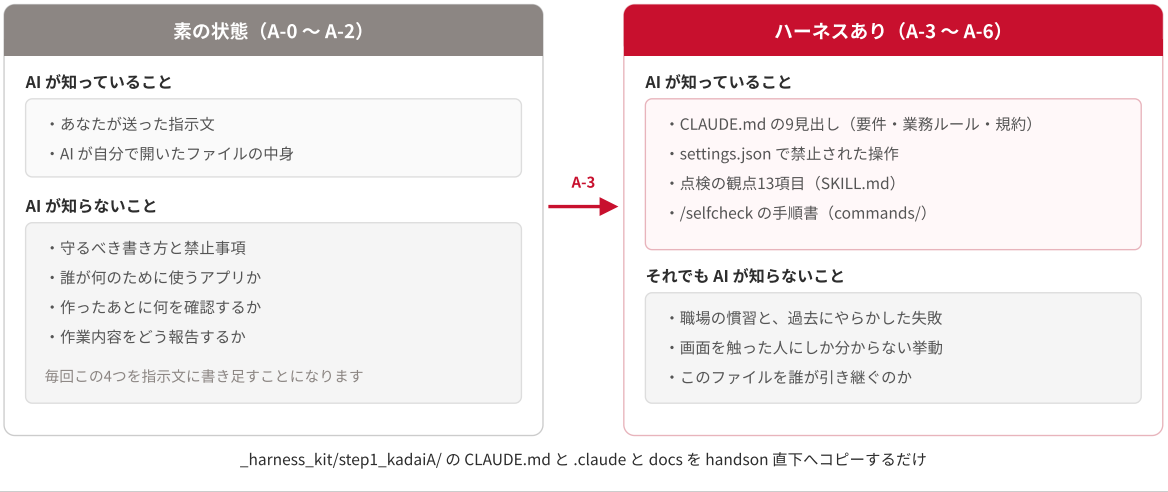

図9 素の状態とハーネスありの対比。右側にも「知らないこと」の枠が残っている点が要点です。設定ファイルで渡せる情報と、渡しても伝わらない情報の境目が A-6 の比較表の3列目になります。

判定に使う数字

課題Aでは「なんとなく動いている」で先へ進まないよう、実データの件数で合否を判定します。次の4つの数字は `kadaiA/dummy_data.csv` の実物から数えたものです。演習中は何度も参照するので、いま覚えておいてください。

50件
備品データの総件数。一覧に全件出ているかの判定に使います（A-1・A-5）

3件
検索欄に「ノート」と入力したときの絞り込み結果（A-1）

4件
在庫が5個以下の備品。5個ちょうどを含みます（A-4）

22件
備考が空欄の備品。詳細表示が崩れないかの確認に使います（A-1・A-2）

カテゴリは PC周辺機器・AV機器・家具・事務用品・その他の5種類です。備品1件あたりの項目は、備品コード・備品名・カテゴリ・在庫数・保管場所・購入日・備考の7つです。詳細表示で7項目すべてが出ているかを A-1 の OK基準で確認します。

各ステップの読み方

A-0 から A-6 まで、すべて同じ7つの見出しで書かれています。この順番には意味があります。手を動かす前に目的を読み、手を動かしたあとに自分で考える欄で立ち止まる。生成物とOK基準で「終わった」を自分で判定できるようにし、追加と考察で仕組みの説明を受け取る。発展課題は時間が余った人向けです。

見出し	そこで何をするか	飛ばしてよいか
目的	このステップで何を体験するのかを先に把握します	読んでください。時間は30秒です
操作	実際に手を動かす手順です。番号順に進めます	本体です
自分で考える	手を止めて考える問いです。答えは `memo.md` に書きます	ここを飛ばすと A-6 の比較が書けなくなります
生成物の名前と場所	何が、どのパスに残るかを確認します	ファイルを取り違えたときの復帰点になります
OK基準	終わったと言える条件です。件数など数字で判定します	次へ進む前に必ず確認します
追加と考察	仕組みの説明です。なぜそうなるのかを補います	時間が無ければ休憩中に読んでも構いません
発展課題	余った時間の使い道です。全員がやる想定ではありません	飛ばして構いません

> **memo.md はこの70分の背骨です**
>
> `handson/memo.md` には A-0 から A-6 までの見出しが空のまま並んでいます。各ステップの「自分で考える」で出した答えは、対応する見出しの下に書き足してください。見出しの文字列は変えないでください。A-5 で `/selfcheck` が `## A-2` の位置を探して読み込むため、見出しがそのまま目印になっています。

SECTION 02

A-0 素の状態を確認する

最初の5分は何も作りません。作業するフォルダに AI へ渡す設定が1つも置かれていないことを、自分の目で確かめることに使います。基準点を取らずに変化を語ると、印象の話で終わります。

A-0 素の状態を確認する

触るフォルダ	`handson`（VSCode で開いているフォルダ全体）
生成物	`handson/memo.md` の `## A-0` に3行以上
終わったと言える状態	`CLAUDE.md` ・ `.claude` ・ `docs` が無いことを確認し、AI の回答を1つ受け取っている
難易度	1 / 3（操作の練習を兼ねています）

目的 あとで差分を測るための基準点を作る

今から作業するフォルダに、AI に渡す設定が1つも置かれていないことを確認します。このステップ単体では何も作りません。それでも最初に置いているのは、あとで設定ファイルを配置したときに「何が変わったか」を言えるようにするためです。

A-3 以降で「前はこうだった」と言うためには、いまの状態を記録に残しておく必要があります。研修が終わって社内で説明するときも、前後の差が手元のファイルとして残っているかどうかで説得力が変わります。

操作 5つの手順

1. VSCode で `handson` フォルダを開く

メニューから「ファイル」→「フォルダーを開く」を選び、配布フォルダの中の `handson` を選びます。キーボードなら Windows は `Ctrl` + `K` のあと `Ctrl` + `O`、Mac は `Cmd` + `K` のあと `Cmd` + `O` です。

> **開くフォルダは最初から最後まで handson です**
>
> `kadaiA` や `kadaiB` を単体で開かないでください。単体で開くと、A-3 で配置する設定ファイルが読み込まれる範囲の外に出てしまい、以降のステップが動きません。作業対象は `@kadaiA/` のようにパスで指定して切り替えます。

2. 直下に何があるかを見る

画面左のエクスプローラで `handson` の直下を上から下まで眺めます。次の2つが**無い**ことを確認してください。

- `CLAUDE.md` というファイル
- `.claude` というフォルダ

`.claude` は先頭にドットが付く隠しフォルダです。VSCode のエクスプローラでは既定で表示されます。見当たらなければ、それは「無い」という意味です。

代わりに `_harness_kit` というフォルダがあります。この中に設定ファイル一式が入っていますが、いまは開かないでください。中身を先に読むと、このあと5分かけて確かめることの答えを読んでしまうことになります。

3. Claude Code パネルを開く

VSCode の左端に縦に並んでいるアイコン列（アクティビティバー）から、Claude Code のアイコンをクリックします。パネルが開き、入力欄が表示されます。

> **Tips: ターミナルからも起動できます**
>
> ターミナルで `claude` と入力する起動方法もあります。本研修ではパネルからの起動に統一します。操作の説明が1本になるほうが、詰まったときに原因を切り分けやすいためです。ご自身の環境ではどちらでも構いません。

4. AI に、いま何を参照しているかを尋ねる

入力欄に、次の2点を尋ねる1文を**自分の言葉で**書いて送ります。短くて構いません。

- いま参照している設定ファイルがあるか

- あるとしたら、どのファイルか

ここで書いた文面は、A-3 でもう一度そのまま使います。送ったあと、自分が何と書いたかを `memo.md` に控えておいてください。同じ問いを2回送るからこそ、答えの違いが設定ファイルによるものだと言い切れます。問いが違えば、答えが変わった理由を問いの側に持っていかれます。

> **HINTS** 参考プロンプトは `hints/step01_first_prompt.md` にあります。自分で1文書けたなら、開かずに進んでください。

5. 課題Aの材料を眺める

次の2つを開いて中身を見ます。読むだけで、まだ何も作りません。

- `kadaiA/docs/お題シート.md`：これから作る画面の要件が書かれています
- `kadaiA/dummy_data.csv`：備品データの実物です

CSV は1行目が見出し行で、そのあとにデータが50行続きます。カテゴリは PC周辺機器・AV機器・家具・事務用品・その他の5種類です。備考の列は空欄の行が22件あります。この数字はあとで OK基準の判定に使うので、いま見ておいてください。

自分で考える AI は何を知っていて、何を知らないのか

設定ファイルが1つも無い状態で、AI がこのプロジェクトについて知っていることは、あなたが送った文と、AI が自分で読みに行ったファイルの中身だけです。では知らないことは何でしょうか。次の3つの視点で考えると出しやすくなります。

視点	自分に問いかける形
決まりごと	このプロジェクトで守るべき書き方や禁止事項を、AI はどこから知るのか
前提	誰が何のために使うアプリなのか。どこまで作れば完成なのか
手順	作ったあとに何を確認すればよいのか。確認結果はどう報告されるのか

思いついたものを `memo.md` の `## A-0` の下に3つ以上書き出します。正解はありません。A-3 で答え合わせをするので、いま思ったことをそのまま書いてください。

生成物 memo.md の ## A-0 に3行以上

`handson/memo.md` は最初から `handson` 直下にあり、A-0 から A-6 までの見出しが空のまま並んでいます。該当する見出しを探して、その下に書き足してください。見出しの順番は入れ替えないでください。

OK基準 この3つが満たせたら A-0 は終わり

① `handson` 直下に `CLAUDE.md` ・ `.claude` ・ `docs` が**無い**ことを、
エクスプローラで目視確認した

② AI から「参照している設定ファイルはありません」に相当する回答が返ってきた

③ `memo.md` の `## A-0` に、AI が知らないと思うことを**3項目以上**書いた

2番については、AI が「このフォルダには設定ファイルが見当たりません」のように答えれば同じ意味です。言い回しの一致は求めません。逆に、AI が特定のファイル名を挙げて「これを読みました」と答えた場合は、開いているフォルダが `handson` ではない可能性があります。エクスプローラの一番上に表示されているフォルダ名を確認してください。

追加と考察 お題シートに技術スタックが書かれていない理由

`kadaiA/docs/お題シート.md` には、技術スタックの指定がありません。使う言語も、ライブラリも書かれていません。これは書き忘れではなく、意図的に空けてあります。

指定が無いとき、AI は自分で選びます。選んだ結果が自分の環境で動くかどうかは、選ばれてみるまでわかりません。A-1 では、この空白を自分の指示でどこまで埋めるかを試します。埋めなかった分だけ、AI の判断が入ります。

発展課題 読ませずに推測させる

`dummy_data.csv` を AI に読ませずに、「このフォルダにはどんなデータが入っていそうか推測してください」と聞いてみてください。フォルダ名とファイル名だけから、AI が何をどこまで言い当てるかを見ます。

推測と実物のずれが、そのまま「文脈を渡していない状態のずれ幅」です。ずれた項目を1つ `memo.md` に書いておくと、A-3 で設定ファイルを置いたあとの比較材料になります。

うまくいかないとき（A-0）

Claude Code のアイコンがアクティビティバーに見当たらない
拡張機能がインストールされていない可能性があります。VSCode の拡張機能ビュー（四角が4つ並んだアイコン）で「Claude Code」を検索し、インストール済みかを確認してください。
インストール済みでも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。

パネルは開くが、送信しても応答が返ってこない
初回はサインインや接続先の確認が入ることがあります。パネル内に表示されている案内に従ってください。案内が出ないまま無反応の場合は、事前セットアップガイドの接続確認の手順に戻ります。それでも変わらなければ挙手してください。ここで止まったまま先へ進むと、以降のステップがすべて動きません。

`.claude` フォルダが見えている
すでに誰かがコピー済みか、配布フォルダを取り違えています。エクスプローラの一番上のフォルダ名が `handson` であることを確認してください。それでも `.claude` がある場合は、`_harness_kit` からコピーされた状態です。演習の出発点としては不都合なので、講師に声をかけてください。

日本語のファイル名が文字化けして見える
ZIP の展開に Windows 標準以外の解凍ソフトを使うと起きることがあります。
配布フォルダをいったん削除し、`genai-nyumon-handson.zip` を右クリックして「すべて展開」から展開し直してください。

英語で応答が返ってくる
「日本語で答えてください」と一言送れば切り替わります。A-3 で設定ファイルを置いたあとは、この指定を毎回書かなくてもよくなります。

SECTION 03

A-1 素の状態で CRUD 業務アプリを作る

設定ファイルが1つも無いまま、自然言語の指示だけで動くアプリを立ち上げます。速さと手軽さを体で覚えるのがこのステップです。この15分は講師が巡回しません。全員が自力で動く時間として意図的に空けています。

A-1 素の状態で CRUD 業務アプリを作る

触るフォルダ	`kadaiA/`（VSCode で開いているのは `handson`）
生成物	`handson/kadaiA/index.html`
終わったと言える状態	ブラウザで一覧50件が出て、検索と詳細表示が動く
難易度	2 / 3（指示の分け方が主題です）

目的 指示を分けて出すことの効果を確かめる

テンプレートも雛形も無いところから、指示だけでアプリが立ち上がることを体験します。

もう1つ、指示を分けて出すことの効果を確かめます。1回で全部を頼むと、返ってきたものが自分の想像と違ったときに、どこから直せばよいかがわかりません。3回か4回に分けると、1回ごとに画面を見て、次の指示で方向を足せます。この差は A-6 の比較にも効いてきます。

操作 3〜4回に分けて指示を出す

前提の確認

- VSCode で開いているのは `handson` です。`kadaiA` を単体で開いていないことを確認してください
- 作業対象は `@kadaiA/` のようにパスで指定します。ファイルを名指しするときは `@kadaiA/dummy_data.csv` のように書くと、AI がそのファイルを読みに行きます

1回目：一覧を出す

最初の指示で伝える内容は次の4点です。文面は自分で組み立ててください。

伝えること	具体的に	落とすとどうなるか
読ませるファイル	`@kadaiA/dummy_data.csv`	AI が架空のデータを埋め込みます
作るもの	備品の一覧を表で表示する画面	要件が発散します
置き場所とファイル名	`kadaiA/index.html`	別の場所に置かれ、A-5 の点検対象から外れます
動かし方の条件	**ブラウザで開くだけで動く形**にすること	サーバー起動が必要な作りになり、確認のたびにコマンドが要ります

4点目を落とさないでください。ここを書くか書かないかで、このあと10分間の作業内容が変わります。理由は「自分で考える」で扱います。

> **HINTS** 参考プロンプトは `hints/step02_crud_build.md` にあります。まず自分で書いてみて、指示が通らなかったときに開いてください。

2. ブラウザで開いて確かめる

VSCode のエクスプローラで `kadaiA/index.html` を右クリックし、「エクスプローラーで表示」（Mac は「Finder で表示」）を選びます。表示されたファイルをダブルクリックすると、既定のブラウザで開きます。

一覧が表示されたら、件数を数えます。**50件**あるはずです。ここで件数が足りない場合は、先へ進む前に直してください。データの読み込みが途中で切れている状態のまま機能を足すと、原因の切り分けが難しくなります。

3. 2回目：検索を足す

備品名の一部を入れると一覧が絞り込まれる検索を足します。伝えるのは「何が
できるようになってほしいか」です。実装方法は指定しません。

ブラウザを再読み込み（`F5`、Mac は `Cmd` + `R`）して、動きを確認します。試しに「ノート」と入力してください。**3件**に絞り込まれます。

4. 3回目：詳細表示を足す

一覧の1件をクリックすると、その備品の全項目が見える表示に切り替わり、一覧に
戻れるようにします。全項目とは、備品コード・備品名・カテゴリ・在庫数・保管場所・購入日・備考の7つです。

ここでも実装方法は指定しません。別ページに遷移するか、同じページ内で表示を切り替えるかは AI が選びます。選んだ結果を見てから、気に入らなければ次の指示で変えれば十分です。

5. 4回目（任意）：見た目を整える

表示が崩れている、文字が詰まって読みにくいといった不満があれば、そこだけを差分で
伝えます。「作り直してください」ではなく「ここをこう変えてください」と伝えるほうが速く、いま動いている部分が壊れにくくなります。

残り時間が3分を切ったら、見た目には手を付けずに次のステップへ進んでください。A-1 の目的は完成度ではありません。

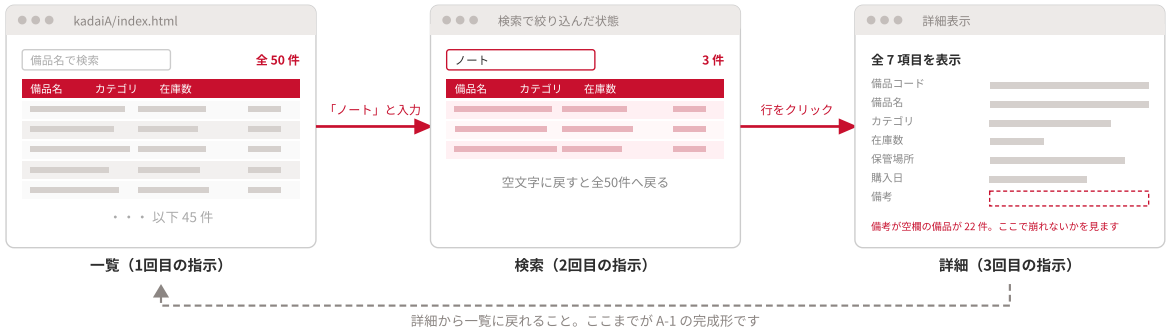

図10 課題Aの画面遷移。3回の指示がそのまま3つの画面状態に対応します。1回目で一覧、2回目で絞り込み、3回目で詳細と復路。指示を分けると、どの指示がどの画面を作ったかが後から追えます。

自分で考える 「ブラウザで開くだけで動く形」の一文が無いとどうなるか

ブラウザには、ローカルのファイルから別のローカルファイルを読み込むことへの制限があります。`index.html` の中から `dummy_data.csv` を通常の方法で読もうとすると、この制限に当たって読めません。AI はそれを避けるため、簡易サーバーを起動する前提の作りを選ぶことがあります。そうなると、アプリを見るたびにコマンドを実行することになり、確認の手間が増えます。

指示に添えた一文が、環境構築の有無を決めています。指示の中でどこを譲れないと伝えるか、という判断がそのまま実装の形に出た例です。自分の指示のどこがこの結果を決めたのか、`memo.md` の `## A-1` に1行書いておいてください。

生成物 handson/kadaiA/index.html

単一ファイルが基本です。AI が CSS や JavaScript を別ファイルに分けた場合も、`kadaiA/` の直下にまとめて置かせてください。別の場所に置かれると、A-5 の `/selfcheck` が点検対象として拾えないことがあります。

> **dummy_data.csv は動かさない、書き換えない**
>
> 元データが変わると、50件という判定基準が使えなくなります。A-3 で配置する `.claude/settings.json` は、このファイルの編集を禁止する設定を持っています。それまでは自分で気をつけてください。

OK基準 ブラウザで index.html を開いた状態で確認する4つ

1. 一覧に備品が **50件** 表示される
2. 検索欄に「ノート」と入力すると **3件** に絞り込まれる
3. 一覧の1件をクリックすると詳細が表示され、一覧に戻れる
4. `index.html` をダブルクリックするだけで動く（コマンドの実行が不要）

3番の詳細は、7項目すべてが出ていることを確認してください。備考が空欄の備品も50件中22件あります。そのうち1件を開いて、表示が崩れないかも見ておきます。ここで崩れていても、いまは直さなくて構いません。A-2 で書き出す材料になります。

追加と考察 分けるか、まとめるか

指示を3回に分けたことで、1回ごとに画面を見て方向を足せました。1回目で全部を頼んだ場合と比べて、どちらが自分の意図に近いものになったと思いますか。

分けることには手間もあります。3回聞けば3回待ちます。それでも分けたほうがよい場面と、まとめて頼んだほうが速い場面があります。境目はどこにありそうか、`memo.md` に1行だけ書いておいてください。A-5 で `/selfcheck` を回すと、この「まとめて頼む」の極端な形を体験することになります。

もう1つ気づいておきたいことがあります。ここまでの応答には、AI が何をしたかの一覧が付いていません。どのファイルを読んで、どこを書き換えたのかは、応答の本文を読んで自分で拾う必要があります。A-4 でこの状態が変わります。

発展課題 理由を引き出す問いを作る

カテゴリでの絞り込みを足したいとき、AI にどう問いかければ「なぜその実装にしたのか」まで返ってくるでしょうか。指示の文面を自分で考えて試してください。

作業指示ではなく、理由を引き出す問いを作るのがこの課題です。「カテゴリで絞り込めるようにしてください」だけでは、動くコードは返ってきても判断の理由は返ってきません。何を足せば理由が付いてくるかを、実際に2通り試して比べてみてください。

手順まで含めた発展版は `exercises/challenges/ch01_category_filter.md` にあります。時間が余った人向けです。A-2 の開始時刻になったら、途中でも切り上げてください。

うまくいかないとき（A-1）

一覧に何も表示されない。画面が真っ白

多くはデータの読み込みで止まっています。ブラウザで `F12`（Mac は `Cmd` + `Option` + `I`）を押し、Console タブに出ている赤いメッセージをそのまま AI に貼り付けて「このエラーが出ています。原因と直し方を教えてください」と送ってください。エラーメッセージは、そのまま渡すのが一番速い伝え方です。

件数が50件ではなく、数件しか出ない

サンプルとして数件だけを埋め込んだコードが生成されている可能性があります。`index.html` の中に備品名が直接書かれていないかを確認してください。書かれていた場合は「CSV の全件を読み込む形にしてください。データを HTML に直接書かないでください」と伝えます。

検索しても件数が変わらない

入力してからボタンを押す作りになっているかもしれません。まず画面上のボタンを探してください。ボタンが無く、入力しても反応しない場合は「入力するたびに絞り込まれるようにしてください」と伝えます。

詳細を開くと画面が壊れる

備考が空欄の備品で起きやすい症状です。50件のうち22件が空欄です。壊れる備品と壊れない備品を1件ずつ見つけて、違いを `memo.md` に書いておいてください。A-2 の懸念点としてそのまま使えます。

生成が途中で止まった

「続けてください」と送ると再開します。何度か繰り返しても同じ位置で止まる場合は、依頼の範囲が広すぎることがあります。「一覧だけ先に作ってください」のように、範囲を狭めて頼み直してください。

思っていた画面と違うものができた

作り直しは最後の手段です。まず「ここをこう変えてください」と、変えたい箇所だけを伝えてください。作り直すと、いま動いている部分まで作り替わります。

時間が足りない

一覧50件が出ていれば、検索と詳細が未完成でも次に進んで構いません。A-2 は「できていないこと」も書き出す対象です。未完成であること自体が材料になります。

SECTION 04

A-2 バイブコーディングの懸念点を自分で洗い出す

勢いで作ったものに何が足りないかを、AI に聞く**前に**自分の言葉で書き出します。ここで書いたものは、A-5 のパイプラインが読み込む入力になります。このステップから講師が巡回します。

A-2 バイブコーディングの懸念点を自分で洗い出す

触るファイル	`handson/memo.md`
生成物	`## A-2` の下に、自分の懸念5個以上と AI との差分
終わったと言える状態	自分の5個、AI の指摘3件以上、差分1行が揃っている
難易度	2 / 3（順番を守れるかが勝負です）

目的 AI に聞く前に自分の視点を確保する

自分が何を不安に思っているかを、先に言語化します。順番には理由があります。先に AI へ聞くと、返ってきた指摘が「そういうものか」と頭に入り、自分の視点が上書きされます。あとから自分の懸念を思い出そうとしても、AI の指摘をなぞったものしか出てきません。

先に5分だけ自分で考えると、AI が挙げなかった項目が手元に残ります。この残りが、あなたが持っていて AI が持っていない情報の正体です。

実務上の理由もあります。ここで書いた懸念は A-5 で `/selfcheck` の入力になります。AI に点検させるとき、汎用の観点だけでなく「自分が気にしていること」を混ぜられると、点検の当たりが変わります。その効果を A-6 で比較します。

操作 自分で5分、AI に2分、突き合わせ1分

1. 5分間、自分だけで書き出す（AI に聞かない）

`memo.md` の `## A-2` 見出しの下に、いま作ったアプリについて不安な点を **5個以上** 書きます。箇条書きで構いません。文の途中で切れていても構いません。手が止まったら、次の5つの観点を順に眺めてください。答えを与える観点ではなく、視点を移すための手がかりです。

観点	自分に問いかける形
データの扱い	元の CSV を壊す動きは入っていないか。データが増えたときはどうなるか
入力値	検索欄に思いがけない文字を入れたら何が起きるか
エラー時の挙動	ファイルが読めなかったとき、画面には何が出るか
他人が読めるか	明日、隣の席の人がこのファイルを開いて意味がわかるか
明日の自分が直せるか	1か月後の自分が、どこを直せばよいか見当を付けられるか

> **具体的に書いてください**
> 「セキュリティが不安」ではなく「検索欄にタグらしき文字を入れたら、そのまま画面に反映されそう」と書くほうが、あとで照合できます。抽象的な1行は、A-5 で点検観点に混ぜても拾われません。書き方の粒度が、そのまま点検の当たりに出ます。

2. AI に指摘させる

5分経ったら、Claude Code に `@kadaiA/index.html` を対象として、危険な点・曖昧な点・実行時に警告が出そうな箇所を挙げてもらいます。深刻な順に並べるよう添えると読みやすくなります。

HINTS 参考プロンプトは `hints/step03_warning_check.md` にあります。

3. 突き合わせる

返ってきた指摘と、自分が書いた5個を並べて見ます。次の3つに仕分けてください。

- 自分も AI も挙げた（重なった）
- 自分は挙げられず、AI が挙げた
- 自分は挙げたが、AI は触れなかった

3つ目があれば、その1件を `memo.md` に目立つ形で残しておいてください。太字でも記号付きでも構いません。A-6 の比較で使います。

自分で考える AI が触れなかったものは、なぜ見えなかったのか

自分が挙げられなかった項目のうち、AI が挙げたものはどれですか。それは知識が
足りなかったからでしょうか、それとも見る時間が足りなかったからでしょうか。

逆に、自分は気づいたのに AI が触れなかったものはありますか。あるとしたら、それはなぜ AI に見えなかったのでしょう。考えられるのは次のような理由です。

- そのファイルの中に書かれていない情報だった（運用の都合、社内の慣習、過去の失敗）
- 画面を実際に操作した人にしかわからない挙動だった
- 「このアプリを誰が使うか」を AI が知らなかった

3つ目に当たるものが出てきたら、それは A-3 で配置する `CLAUDE.md` が扱う領域です。
設定ファイルで渡せる情報と、渡しても伝わらない情報の境目が、ここで一度見えます。

生成物 memo.md の ## A-2 に3種類

1. 自分が書いた懸念（5個以上）
2. AI が挙げた指摘（要点だけで構いません）
3. 両者の差分（1行以上）

> **見出しの文字列 ## A-2 は変えないでください**
>
> A-5 で `/selfcheck` がこの見出しを目印に中身を読み込みます。読み込まれた懸念は、点検の観点 A-13「受講者の懸念」として扱われます。空のままだと、点検が汎用の観点だけで行われ、A-6 の比較で差が出なくなります。

OK基準 3つ揃えば次へ進めます

- ① 自分の言葉で **5個以上** 書けた（AI に聞く前に書いたもの）
- ② AI の指摘が **3件以上** 返ってきた
- ③ 自分と AI の差分を **1行以上** 書けた

3番は、重なった件数を数えるだけでも構いません。「5個中2個が重なった。AI は入力値の扱いを3件挙げたが、自分は1件も挙げられなかった」のように書けていれば十分です。

追加と考察 指摘がばらつくこと自体が状態を表している

AI の指摘は毎回同じにはなりません。隣の受講者と内容が違っていても、どちらかが失敗しているわけではありません。

指摘の粒度がばらつくこと自体が、文脈を渡していない状態の特徴です。何を重く見るかの基準を渡していないので、AI はそのつど自分で基準を決めます。決め方が毎回変われば、出てくる指摘も変わります。

A-5 では、観点リストを持った状態で同じ点検をします。そのときに何がどう変わるかを見るために、いまの指摘の件数だけでもメモしておいてください。

発展課題 /clear してから同じ質問を送る

同じ質問を、`/clear` で会話の文脈をリセットしてからもう一度送ってみてください。`/clear` は、それまでのやり取りを AI の記憶から外すコマンドです。

指摘の内容はどれくらい変わりましたか。1回目は A-1 の作業の流れが会話に残っていました。2回目はそれがありません。同じファイルを見ているのに結果が変わるなら、AI が見ているのはファイルだけではないということです。

この観察は、`CLAUDE.md` が「毎回同じ前提から始められる仕組み」であることの裏返しです。A-3 で置いたあとにもう一度この発展課題を試すと、変わり方の幅が縮んでいることを確認できます。

うまくいかないとき（A-2）

5個も思いつかない

いま画面でできないことを書いても構いません。「削除ができない」「並べ替えができない」も立派な懸念です。作ったものへの不満と、危なそうなところを分けずに、まず数を出してください。

AI の指摘が抽象的で、何を言われているかわからない

「どのファイルの何行目のことですか」と聞き返してください。場所を答えられない指摘は、そのアプリを見て言っているものではない可能性があります。指摘に場所と根拠を付けさせることは、A-5 の点検でも同じ考え方で扱います。

AI が指摘ではなく、勝手に修正を始めた

止めて構いません。「修正はしないでください。指摘だけを挙げてください」と伝えます。いま直してしまうと、A-5 のパイプラインで拾える指摘が減り、比較の材料が薄くなります。直すのはあとです。

指摘が20件以上出てきて読み切れない

「深刻な順に上位5件だけ、理由を1行ずつ付けて挙げてください」と絞ってください。全部読む必要はありません。件数だけメモしておけば、A-5 との比較には足ります。

A-1 が未完成で、指摘するものが少ない

未完成な部分をそのまま懸念として書いてください。「詳細画面が未実装。実装したときに備考の空欄で崩れそう」のような書き方で構いません。予想を書いておくと、A-5 の結果と突き合わせられます。

SECTION 05

A-3 ハーネスを配置する

配布フォルダの中で待機していた設定ファイル一式を、初めてプロジェクトに置きます。演習Aの折り返し地点です。**このステップは全員同時に行います。**講師の合図を待ってから始めてください。

> **ハーネスとは**
>
> AI を働かせるための足場のことです。本研修では、`CLAUDE.md`（共有の前提）と `.claude/` の中の設定・コマンド・観点・サブエージェント、この一式をまとめてそう呼びます。馬具のハーネスと同じで、力を出させるためではなく、力を意図した方向に向けるための道具です。

A-3 ハーネスを配置する

触るフォルダ	`handson` 直下
生成物	`handson/CLAUDE.md` と `handson/.claude/`（コピーによる配置）
終わったと言える状態	AI が `CLAUDE.md` の中身に触れて答え、応答の末尾にサマリーが付く
難易度	1 / 3（コピーと開き直しだけです）

目的 置くだけで振る舞いが変わることを確かめる

「AI に指示を出す」以外に、「AI が読む場所にあらかじめ置いておく」という渡し方があります。この2つは実務での効き方が違います。指示は毎回書き直しになりますが、置いたものは全員に同じように効きます。その差を体験するのがこのステップです。

操作 コピーして、パネルを開き直して、同じ質問をもう一度

1. コピー元を開く

VSCode 左のエクスプローラで `handson` > `_harness_kit` > `step1_kadaiA` の順にフォルダを開きます。中には7つ入っていますが、**handson 直下へコピーするのは上の3つだけ**です。

```
handson/
├── _harness_kit/
│   └── step1_kadaiA/
│       ├── CLAUDE.md              ← これを handson 直下へ
│       ├── .claude/               ← これも handson 直下へ
│       │   ├── settings.json          禁止する操作の一覧
│       │   ├── commands/selfcheck.md  A-5 で叩くコマンドの手順書
│       │   ├── skills/kadaia-review/SKILL.md   点検の観点13項目
│       │   └── agents/reviewer.md     点検専任のサブエージェント
│       ├── docs/                  ← これも handson 直下へ（/selfcheck が読む観点4本）
│       │   ├── コーディング規約.md
│       │   ├── 生成物チェックリスト.md
│       │   ├── セキュリティチェックリスト.md
│       │   └── セルフチェックの観点.md
│       ├── AGENTS.md                  コピーしません（他のAIツール向けの見本）
│       └── .github/ .cursor/ .codex/  コピーしません（同上）
└── ...
```

> **docs を忘れないでください**
>
> A-5 で回す `/selfcheck` は、`docs/` の4本を点検の観点として読みます。ここが無いと、観点が足りないまま点検が進み、指摘が数件しか出ません。結果として A-6 の比較で差が出なくなります。**コピーするのは `CLAUDE.md` ・ `.claude` ・ `docs` の3つ**です。

2. handson 直下にコピーする

- Windows：エクスプローラで `step1_kadaiA` を開き、`CLAUDE.md` をクリック、`Ctrl` を押しながら `.claude` と `docs` をクリックして3つ選びます。`Ctrl`+`C` でコピーし、`handson` フォルダを開いて `Ctrl`+`V` で貼り付け
- Mac：Finder で `step1_kadaiA` を開き、`CLAUDE.md` をクリック、`Cmd` を押しながら `.claude` と `docs` をクリックして3つ選びます。`Cmd`+`C` でコピーし、`handson` フォルダを開いて `Cmd`+`V` で貼り付け
- VSCode のエクスプローラ内でドラッグして移動させても構いませんが、**移動ではなくコピー**にしてください。`_harness_kit` の中身は残しておきます

`.claude` のように先頭にドットが付くフォルダは、Windows のエクスプローラや Mac の Finder では既定で見えません。Windows は「表示」タブの「隠しファイル」にチェック、Mac の Finder は `Cmd` + `Shift` + `.` で表示が切り替わります。VSCode のエクスプローラからは最初から見えています。

移動してしまった場合も演習は続けられます。元に戻したいときだけ講師に声をかけてください。

3. 配置されたことを確認する

VSCode のエクスプローラで `handson` の直下を見て、`CLAUDE.md` ・ `.claude` ・ `docs` の3つが並んでいることを確認します。`docs` を開くとファイルが4本あります。A-0 で「無い」ことを確認した場所に、いま「ある」状態です。

4. Claude Code パネルを閉じて開き直す

> **この手順を飛ばすと、以降がすべて動きません**
> 設定ファイルはパネルの起動時に読み込まれるため、開いたままでは反映されません。アクティビティバーの Claude Code アイコンをクリックしてパネルを閉じ、もう一度クリックして開き直します。会話の履歴が残っている場合は `/clear` を送って区切ってから進めると、前の文脈が混ざりません。

5. A-0 と同じ質問をもう一度送る

A-0 で送ったのと**同じ文**を、そのまま送ります。文面を控えていない場合は、いま参照している設定ファイルと、そこに書かれているルールの要点を尋ねる形にしてください。

同じ問いを2回送ることに意味があります。問いが同じで答えが違えば、変わったのは問いの側ではなく環境の側です。

自分で考える AI が新しく知ったのは、具体的にどの情報か

`CLAUDE.md` を開いて、見出しを上から順に見てください。次の9つが、いま AI が毎回読む前提です。

#	CLAUDE.md の見出し	そこに書かれていること
1	**フォルダ構成**	どこに何があるか。`handson` 直下を起点に見ること
2	**作業対象の指定**	`@kadaiA/` のようにパスで指定する決まり
3	**課題A の要件**	一覧・検索・詳細と、ブラウザで開くだけで動く条件
4	**課題B の技術スタック**	Spring Boot・H2・Thymeleaf。演習Bで効きます
5	**業務ルール（課題B）**	仕様の正本となる5項目
6	**コーディング規約**	命名・コメント・関数の長さの決まり
7	**出力の作法**	日本語で書く、対象ファイルのパスを先に示す、など
8	**してはいけないこと**	元データの書き換え、依頼していない機能追加、`git` の実行など
9	**応答の最後に必ず出すもの**	実行サマリーの書式。A-4 の主題です

このうち、A-2 で自分が書いた懸念に関係する見出しはどれですか。1つ選んで `memo.md` の `## A-3` に書いてください。関係するものが見つからない場合は、「自分の懸念は `CLAUDE.md` では手当てされていない」と書いてください。それも正しい観察です。

生成物 配置した2つ（中身は編集しません）

- `handson/CLAUDE.md`
- `handson/.claude/`（`settings.json` / `commands/selfcheck.md` / `skills/kadaia-review/SKILL.md` / `agents/reviewer.md`）

A-3 は置くだけのステップです。書き換えたくなる箇所を見つけたら、`memo.md` に控えておいて、A-4 の発展課題で試してください。

OK基準 3つ確認したら A-4 へ

1. `handson` 直下に `CLAUDE.md` ・ `.claude` ・ `docs` の3つがある
2. `handson/docs/` の中にファイルが **4本** ある
3. AI の回答が、`CLAUDE.md` の内容（技術スタック・業務ルール・してはいけないこと のいずれか）に触れている
4. 回答の末尾に「実行サマリー」のブロックが付いている

3番は A-4 の主題です。ここでは「何か出るようになった」ことに気づければ十分です。
もし出ていなければ、パネルの開き直しができていないか、コピー先が `handson` 直下ではない可能性があります。

追加と考察 「これを読んで」とは一度も言っていない

コピーしただけで、AI に「これを読んでください」とは一度も言っていません。`CLAUDE.md` は Claude Code が起動時に自動で読み込む決まりになっています。

この自動読込があるから、チーム全員が同じ前提で AI を使えます。1人が毎回うまい指示を書いて成果を出す状態と、全員が同じ前提から始められる状態は、再現性の面で別物です。前者は書いた人が休むと止まります。

`.claude/settings.json` も同時に効き始めています。中を見ると `deny` という一覧があり、そこに書かれた操作は AI が実行できません。`dummy_data.csv` の書き換えもここで止めています。A-5 で点検と修正を任せても元データが壊れないのは、この1ファイルが理由です。

```
"deny": [
  "Read(./**/.env*)",
  "Read(./**/*credentials*)",
  "Bash(rm -rf:*)",
  "Bash(git push:*)",
  "Edit(./kadaiA/dummy_data.csv)",
  "Edit(./kadaiB/src/main/resources/data.sql)"
]
```

発展課題 deny の1行を選んで、無かった場合を考える

`.claude/settings.json` を開いて中身を読んでください。`deny` の一覧に並んでいる行のうち、1行を選びます。その1行が無かったら何が起きるでしょうか。`memo.md` に書いてください。

たとえば `Edit(./kadaiA/dummy_data.csv)` を外したとすると、AI が「データ形式を整えます」と判断したときに元データを書き換えられる状態になります。書き換わったことに気づくのは、たいてい件数が合わなくなったあとです。

もう1歩進めるなら、自分の職場で「AI にやらせたくない操作」を1つ挙げ、それを `deny` に足す書き方を AI に聞いてみてください。書き方の正解より、何を止めたいかを言語化するほうが実務では重要です。

うまくいかないとき（A-3）

回答の末尾にサマリーが出ない
パネルを開き直したかを確認してください。閉じずにコピーだけした場合、設定は反映されません。開き直しても出ない場合は、handson 直下に CLAUDE.md があるかを見ます。handson/kadaiA/CLAUDE.md のように1階層深い場所に入っていることがあります。

AI が「設定ファイルはありません」と答えたまま
VSCode で開いているフォルダが handson ではない可能性が高いです。エクスプローラの一番上に表示されているフォルダ名を確認してください。kadaiA になっていたら、handson を開き直します。

.claude フォルダが貼り付けられない、見えない
先頭にドットが付くフォルダは、環境によってはエクスプローラで非表示になります。Windows はエクスプローラの「表示」タブで「隠しファイル」にチェックを入れます。Mac の Finder は `Cmd` + `Shift` + `.` で表示が切り替わります。

_harness_kit から中身が消えた
コピーではなく移動になっています。演習は続けられます。handson 直下に CLAUDE.md・.claude・docs があれば問題ありません。B-0 で step2_kadaiB を使うので、そちらのフォルダが残っていることだけ確認してください。

中身を書き換えてしまった
書き換えた箇所を覚えていれば戻してください。覚えていない場合は、_harness_kit/step1_kadaiA/ から CLAUDE.md をコピーし直して上書きします。元ファイルは残っています。

SECTION 06

A-4 実行サマリーの出方を確認する

AI が「何をしたか」を毎回自己申告する状態を体験します。作業のログが応答に残ると、確認の手間がどう変わるかを見ます。あわせて、書かれていることと実際が一致しているかを自分で1回照合します。

A-4 実行サマリーの出方を確認する

触るフォルダ	`kadaiA/`
生成物	更新された `kadaiA/index.html` と、`memo.md` の `## A-4`
終わったと言える状態	サマリーの記載と実ファイルの変更が一致していることを確認できた
難易度	2 / 3（読み方と照合が主題です）

目的 AI の作業を人間が追える形で受け取る

コードレビューで一番手間がかかるのは、変更内容を読むことではなく「どこが変わったかを探すこと」です。AI に作業させると、この探す作業が毎回発生します。応答の末尾に「どのファイルに何をしたか」が出ていれば、探す時間がなくなります。

もう1つ、書かれていることと実際が一致しているかを自分で確かめます。自己申告は便利ですが、鵜呑みにする前に1回は照合してください。1回照合すると、どの欄が信用できて、どの欄を毎回見るべきかの感覚がつかめます。

操作 機能を1つ足して、末尾を読んで、照合する

1. 機能を1つ追加させる

`@kadaiA/index.html` に対して、在庫が少ない備品の行を目立たせる表示を足してもらいます。伝える条件は次の2つです。

- 対象は在庫が **5個以下** の備品（5個ちょうども含みます）
- 一覧を見たときに、その行が他と違って見えること

見せ方は指定しません。背景色でも、文字色でも、印を付けるのでも構いません。AI が選んだ見せ方を見てから、気に入らなければ次の指示で変えられます。

> **「5個以下」に5を含めるかを明示している理由**
>
> ここが取り違えやすい箇所だからです。「以下」と「未満」の取り違えは、点検の観点 A-08「数値と境界」でも見られます。課題Bの業務ルールでも同じ論点が出てきます。条件を書くときに境目を明示する癖は、AI に頼むときも人に頼むときも効きます。

2. 応答のいちばん下を読む

応答の末尾に「実行サマリー」というブロックが出ています。次の4項目を上から順に読んでください。

欄	何が書かれているか	どう使うか
動作	読み取り・編集・新規作成・コマンド実行を何回ずつ行ったか	想定より数が多ければ、頼んでいない作業が混ざっている合図
従った設定	`CLAUDE.md` のどの見出し、どの Skill、どのコマンドに従ったか	見出し名が書いてあるので、その箇所を開いて照合できる
対象ファイル	どのパスに何をしたか	変更箇所を探さずに開ける
未実施・保留	頼まれたが今回やらなかったこと	人間が判断すべき残件がここに出る

「従った設定」の欄には、`## 出力の作法` のように `CLAUDE.md` の見出し名がそのまま書かれているはずです。書かれていたら、実際に `CLAUDE.md` を開いてその見出しを探してください。設定と応答が1対1でつながっていることが目で見えます。

図11 実行サマリーの読み方。4つの欄それぞれに役割があります。上から順に読むより、まず「未実施・保留」を見て、そのあと「対象ファイル」で照合するほうが確認は速く終わります。

操作（続き） 照合とブラウザ確認

3. 書かれているとおりか照合する

サマリーの「対象ファイル」に挙がっているファイルを開き、書かれているとおりの変更が入っているかを目で確かめます。見るのは2点です。挙がっているファイルが実際に変わっているか。そして、挙がっていないファイルが変わっていないか。VSCode のエクスプローラでは変更のあったファイルに印が付くわけではないため、気になる場合はファイルの更新日時を見てください。

4. ブラウザで確認する

ブラウザを再読み込みして、在庫の少ない備品の行が目立っているかを確認します。`dummy_data.csv` の中で在庫5個以下の備品は **4件** です。目立っている行が4件あれば、条件の解釈も表示も合っています。

3件しかない場合は、5個ちょうどの備品が対象から外れています。境目の取り違えです。「5個ちょうども含めてください」と伝えて直してもらってください。

5. サマリーを memo.md に貼る

出てきた実行サマリーの中身を、`memo.md` の `## A-4` にそのまま貼り付けます。A-6 の比較で、ハーネスを入れる前後の応答の違いを説明する材料になります。

自分で考える 「未実施・保留」に何が書かれていたか

書かれていた場合、それは AI の判断で今回やらなかったことです。省いてよかったものでしたか。それとも、頼んだつもりだったのに落ちていたものでしたか。

「なし」と書かれていた場合も1つ考えてみてください。本当に残件はゼロでしょうか。たとえば、在庫のしきい値である5という数字は、いまコードのどこに書かれているでしょうか。直書きされているとしたら、しきい値を変えたい人はコードを探すことになります。これは残件と呼べる状態かもしれません。判断したことを1行、`memo.md` に書いてください。

生成物 更新した index.html と memo.md の ## A-4

- `handson/kadaiA/index.html`（更新）
- `handson/memo.md` の `## A-4`（実行サマリーの貼り付けと、気づいたこと1行）

OK基準 4つ揃えば A-5 へ

1. 応答の末尾に実行サマリーの4項目（動作／従った設定／対象ファイル／未実施・保留）がすべて出ている
2. サマリーの「対象ファイル」に書かれたファイルと、実際に変更されたファイルが一致している
3. ブラウザで、在庫5個以下の **4件** が他の行と区別できる表示になっている
4. `memo.md` の `## A-4` にサマリーを貼り付けた

追加と考察 設定ファイル1節で振る舞いが変わる

A-1 のときの応答には、このサマリーは出ていませんでした。出るようになったのは、`CLAUDE.md` の末尾に `## 応答の最後に必ず出すもの` という節が1つあるからです。開いて読んでみてください。書かれているのは日本語の指示文です。プログラムではありません。

読み取りだけでも出させているのには理由があります。出たり出なかったりすると、出なかったときに「壊れたのか、そういう仕様なのか」がわからなくなります。毎回出ると決めておけば、出ないこと自体が異常の合図になります。

設定ファイルの1節で、AI の振る舞いをここまで変えられます。逆に言えば、書いていないことは変わりません。

発展課題 サマリーに自分の欲しい項目を足す

実行サマリーに、あなたの仕事で欲しい項目を1つ足すとしたら何ですか。例を挙げます。「参照した外部情報の一覧」「変更前の状態に戻す方法」「この変更で影響を受けそうな他のファイル」。自分の職場のレビューで毎回聞かれることを思い出すと出てきます。

決まったら `CLAUDE.md` の `## 応答の最後に必ず出すもの` を書き換えて、実際にその項目が出るか試してください。書き換えたあとはパネルを開き直します。出なかった場合、書き方に問題があります。何を書けば出るようになるかを2通り試して、違いを `memo.md` に書いてください。

この演習で書き換えた `CLAUDE.md` は元に戻さなくて構いません。A-5 の点検には影響しません。在庫の表示そのものを深めたい場合は `exercises/challenges/ch02_low_stock_highlight.md` に進んでください。しきい値を画面から変えられるようにする課題です。

うまくいかないとき（A-4）

サマリーが出ない、または途中で切れる

まずパネルを開き直してください。それでも出ない場合は CLAUDE.md の末尾を確認します。## 応答の最後に必ず出すもの の節がそのまま残っているかを見てください。A-3 の発展課題で書き換えた場合、書式が崩れていることがあります。_harness_kit/step1_kadaiA/CLAUDE.md からコピーし直せば元に戻ります。

サマリーは出るが「従った設定」がいつも同じ

依頼の内容が変わっても同じ見出しが並ぶことはあります。読み取りだけの依頼なら ## フォルダ構成 だけ、コードを書き換える依頼なら ## コーディング規約 が加わる、といった違いが出ているかを見てください。まったく変わらない場合は、「今回の依頼で実際に参照した見出しだけを書いてください」と一言添えます。

目立つ行が4件より多い、または少ない

多い場合は条件が「10個以下」などになっていないかを確認します。少ない場合は境目の取り違えです。どちらも、いま画面に出ている件数を伝えて「在庫5個以下の4件が対象です」と直してもらいます。件数で伝えるのが一番速い指摘の仕方です。

サマリーに書かれていないファイルが変更されていた

その旨をそのまま AI に伝えてください。「サマリーに kadaiA/style.css が載っていませんが、更新されています」と伝えると、書き漏らしか、意図的に省いたかが返ってきます。自己申告を1回検算する経験として、むしろ良い状態です。memo.md に記録しておいてください。

表示は変わったが、元の一覧が壊れた

件数を数えてください。50件が出ていれば表示だけの問題です。50件でなければ、データの読み込みに手が入っています。「一覧は50件のまま、表示だけを変えてください」と伝えて直します。

SECTION 07

A-5 コマンド一つでパイプラインを回す

コマンド1本で、AI が観点に沿って成果物を点検し、中間データを残し、直し、直しきれたかを再判定して、人間向けの報告書を出すところまでを体験します。演習Aの山場で、70分のうち20分をここに使います。

A-5 コマンド一つでパイプラインを回す

触るフォルダ	`handson` 全体（`.work/` が新しくできます）
生成物	`kadaiA/REPORT.md` と `.work/` の中間データ4本
終わったと言える状態	報告書に指摘・修正・残件の数字が入り、中間データが4本揃っている
難易度	3 / 3（実行は簡単、読み解きに時間を使います）

目的 依頼を「会話」ではなく「決まった手順の実行」として扱う

A-1 から A-4 まで、依頼はすべて会話でした。会話は柔軟ですが、毎回書き方が違います。同じ点検を明日もう一度やろうとすると、同じようには書けません。手順を1本のファイルに書いておけば、コマンド名を打つだけで同じ手順が動きます。

このステップでは、実行中に何が起きているかを見ることに時間を使います。
結果だけ受け取ると、中で何をしたのかがわからないまま報告書を信じることになります。`.work/` に残る途中経過を開けば、どのファイルを見て、何を根拠に、何を直すと決めたのかが全部追えます。

図12　/selfcheck のパイプライン全体図。入力に memo.md の ## A-2 を入れているのが設計上の要点です。受講者が自分で書いた不安を点検の観点に混ぜることで、A-6 の比較が成立します。

操作 実行して、眺めて、番号順に開く

1. コマンドを実行する

Claude Code パネルで次を送ります。

```
/selfcheck kadaiA
```

`/` を打つと候補が出ます。`selfcheck` を選び、引数に `kadaiA` を付けて送信してください。引数を省略しても `kadaiA` が対象になりますが、明示するほうが何を点検したかが記録に残ります。
3分から5分かかります。途中で止めないでください。

2. 実行中に `.work/` を眺める

実行が始まったら、VSCode のエクスプローラで `handson/.work/` フォルダを開いたままにします。
フォルダは実行開始時に作られます。ファイルが順に増えていく様子が見えます。

```
.work/
├── 01_inventory.md     点検対象として拾ったファイルの一覧
├── 02_findings.json    観点ごとの指摘（深刻度つき）
├── 03_fix_plan.md      どれを直すと決めたか
└── 04_loop_log.md      何周回って、各周で何件残ったか
```

番号順に増えます。増えていく間、AI は点検専任のサブエージェント
（`.claude/agents/reviewer.md`）を呼び出しています。このサブエージェントは
読み取りしかできない設定になっていて、コードを書き換えません。点検する役と、
直す役を分けてあります。

3. 報告書を読む

完了したら `handson/kadaiA/REPORT.md` を開きます。6つの節が並んでいます。

節	何が書かれているか
1. 結論	3行以内。何件指摘し、何件直し、何件残したか
2. 数字のまとめ	指摘の内訳（高／中／低）、修正件数、残件数、ループ何周
3. 直したもの	観点・ファイル・変更内容・確認方法
4. 直さなかったもの	なぜ直さないと判断したか
5. 人間に判断してほしいこと	AI が決められなかったもの
6. 使った観点と出どころ	どのファイルのどの見出しに基づくか

まず1節と2節を読み、数字を頭に入れてください。そのあと5節に飛びます。**5節がこの報告書で一番重要です。**AI が「自分では決められない」と判断した箇所が並んでいます。全部を AI が決めて終わる報告書にしていないのは、そこが実務での線引きだからです。

4. 中間データを番号順に開く

報告書を読んだあと、`.work/` の4本を順に開きます。読む観点を挙げておきます。

ファイル	ここを見てください
01_inventory.md	点検対象に何が挙がっているか。`dummy_data.csv` は「参照のみ」になっているはずです。データを点検の材料にはするが、書き換え対象からは外す、という区別が最初に付けられています
02_findings.json	指摘1件を開いて、`ファイル` `行` `内容` `根拠` の4つが埋まっているか。`根拠` には、どのファイルのどの観点に基づく指摘かが書かれています。根拠のない指摘は出さない決まりです
03_fix_plan.md	全部は直していないはずです。判断の基準は、深刻度「高」は必ず直す、「中」は依頼外の機能追加にならず既存の動作を壊さないものだけ、「低」は直さない、というものです
04_loop_log.md	開始時に高が何件あって、何件直して、終了時に何件残ったか。そして次の周に進んだのか、止まったのか

5. ループが何周で止まったかを確認する

`04_loop_log.md` の一番下を見ます。止まった理由が書かれています。次の3通りのどれかです。

- 深刻度「高」が0件になったので停止
- 上限3周に達したので打ち切り（残件は報告書の未対応欄へ）
- 同じ指摘の修正が2周続けて失敗したので打ち切り

自分がどれだったかを `memo.md` の `## A-5` に書いてください。周回数と止まった理由の2行で構いません。

6. アプリがまだ動くことを確かめる

ブラウザを再読み込みして、一覧50件・検索・詳細が動くことを確認します。点検の過程で修正が入っているため、直した結果として壊れていないかを人間の目で見ます。

> **HINTS** 実行が止まったときの対処は `hints/step04_pipeline.md` にあります。

図14 ループの回り方。1周は「修正 → 修正箇所の再点検 → 全体が壊れていないかの確認 → 残件を数える」の4つです。止まる条件が2つ（高が0件／3周到達）あり、どちらで止まったかは 04_loop_log.md の最終行に書かれます。

自分で考える A-2 で書いた5個と、02_findings.json を比べる

数は増えましたか。増えたとしたら、増えた分は何を見ているから出てきたのでしょうか。観点の一覧は `.claude/skills/kadaia-review/SKILL.md` に書かれています。開くと、A-01 から A-13 までの観点が表になっています。

観点ID	見るもの	深刻度
A-01	ブラウザで開くだけで動くか	**高**
A-02	データが全件（50件）出るか	**高**
A-03	入力値の扱い（検索キーワードをそのまま画面に流していないか）	**高**
A-04	元データの保全（`dummy_data.csv` を書き換えていないか）	**高**
A-05〜A-11	検索の挙動・詳細表示・未設定の値・数値と境界・日付の表記・エラー時の挙動・定数と重複	中
A-12	読みやすさ（コメント・関数の長さ・未使用の変数）	低
A-13	**受講者の懸念**（`memo.md` の `## A-2` のうち他で拾えなかったもの）	内容による

この表を見てから、もう一度自分の5個を読み返してください。同じことを別の言葉で書いていた項目はありませんか。逆に、この表に無い視点を自分が書いていませんでしたか。

最後の観点 A-13 に注目してください。自分の書いた懸念が指摘に入っていたら、`内容` の冒頭に「受講者の懸念より:」と付いています。探してみてください。

生成物 報告書1本と中間データ4本

- `handson/kadaiA/REPORT.md`（最終報告・人間向け）
- `handson/.work/01_inventory.md`
- `handson/.work/02_findings.json`
- `handson/.work/03_fix_plan.md`
- `handson/.work/04_loop_log.md`
- `handson/kadaiA/index.html`（修正が入っている場合）

> **.work/ は消さないでください**
>
> 実行が終わったあとに開けることが、この演習の中身です。研修後に社内で説明するときも、途中経過が残っているほうが伝わります。`CLAUDE.md` の「してはいけないこと」にも `.work/` の削除禁止が書かれています。

OK基準 5つ確認したら A-6 へ

1. `kadaiA/REPORT.md` が生成され、指摘件数・修正件数・残件数が **数字で** 書かれている
2. `.work/` に中間ファイルが **4本** 揃っている
3. `04_loop_log.md` に **2周以上** の記録がある（1周で深刻度「高」が0件になった場合は1周でも可。その旨がログに書かれていること）
4. `REPORT.md` の第5節「人間に判断してほしいこと」に **1件以上** 書かれている
5. 修正後もブラウザでアプリが動く（一覧50件・検索・詳細）

4番が空の報告書は、この演習の趣旨から外れています。空だった場合は「判断が分かれた箇所か、置いた前提を1件挙げてください」と追加で依頼してください。

追加と考察 コマンドを作るとは、手順を文章で1本書くこと

`/selfcheck` は `.claude/commands/selfcheck.md` に書かれた手順書です。**中身は日本語の文章で、プログラムではありません。**

開いて読んでみてください。「0. 前提と引数」から始まり、入力を読む、棚卸しを書く、点検する、修正する、ログを書く、報告書を書く、という順に手順が並んでいます。表の書式まで見本付きで指定されています。書いてあるとおりに AI が動いた結果が、いま手元にある `.work/` と `REPORT.md` です。

「コマンドを作る」とは、この文章を1本書くことです。プログラミングの知識は要りません。必要なのは、自分がいつもやっている手順を、抜けなく順番どおりに書き出せることです。手順が言語化できていない作業は、コマンドにもできません。

もう1点、実行中にサブエージェントを1件ずつ順番に呼んでいたことにも触れておきます。20名が同時に実行する研修環境では、並列で走らせると接続先の上限に当たります。手順書の中で「並列で起動しないでください」と1行指定してあります。動く仕組みを書くだけでなく、動かす環境の都合も手順書に書ける、という例です。

発展課題 観点を1つ足して、もう一度回す

`.claude/skills/kadaia-review/SKILL.md` の観点リストに、自分の職場でレビューされる観点を1つ足してください。観点IDは `A-14` から続けます。

足すときに難しいのは「具体的に見るもの」の欄です。ここが抽象的だと、点検時に拾われません。「可読性を確認する」では拾えません。「関数が50行を超えていないか」なら拾えます。コードのどの記述を見れば判定できるかまで書くのが条件です。

足したら `/selfcheck kadaiA` をもう一度回します。指摘は増えましたか。増えなかった場合、原因は2つ考えられます。そもそも該当する箇所が無かったか、観点の書き方が拾える形になっていなかったかです。どちらかを切り分けるには、`index.html` を自分の目で見て、その観点に該当しそうな箇所があるかを探します。あるのに拾われなかったなら、書き方の問題です。

2回目以降の `.work/` は上書きされます。1回目の結果を残したい場合は、`REPORT.md` を `REPORT_1.md` に名前を変えてから回してください。

うまくいかないとき（A-5）

指摘が数件しか出ず、報告書が薄い

A-3 で docs のコピーを飛ばしている可能性があります。handson/docs/ にファイルが4本あるかを確認してください。無い場合、点検の観点が読めないまま進みます。.work/01_inventory.md の「読めなかったもの」の欄に、読めなかったファイル名が出ています。_harness_kit/step1_kadaiA/docs をコピーし、パネルを開き直してから /selfcheck kadaiA を回し直してください。

/selfcheck が候補に出てこない

.claude/commands/selfcheck.md が handson 直下に配置されているかを確認してください。A-3 でコピーしたあと、パネルを開き直していない場合も出ません。開き直してから、もう一度 / を打ってみてください。

実行が5分以上返ってこない

20名が同時に実行しているため、混み合うと時間がかかります。10分を超えても終わらない場合は、いったん停止して /selfcheck kadaiA をもう一度送ってください。.work/ にファイルが1本でもできていれば、途中まで進んでいた証拠です。

.work/ にファイルが2本しかできていない

点検の途中で止まっています。01_inventory.md を開いて、「読めなかったもの」の欄を見てください。入力のどれかが見つからずに止まっている場合があります。多いのは memo.md の ## A-2 が空のケースです。空でも進む作りにしてありますが、書いてあるほうが結果は濃くなります。

指摘が0件だった

そのまま報告書に「指摘0件」と書かれていれば、それも結果です。ただし A-2 で自分が5個書けたのに AI が0件なら、点検対象が正しく拾えていない可能性があります。01_inventory.md の点検対象一覧に kadaiA/index.html が入っているかを確認してください。

修正が入ったらアプリが動かなくなった

ブラウザの `F12`（Mac は `Cmd` + `Option` + `I`）で Console のエラーを確認し、そのまま AI に貼り付けて直してもらいます。03_fix_plan.md を見れば、どの指摘に対して何を変えたかが書かれているので、変更箇所の見当が付きます。この状況自体が「AI の修正も検算が要る」という体験です。memo.md に1行残しておいてください。

04_loop_log.md が1周で終わっている

深刻度「高」が最初から0件だった場合は1周で止まります。ログにその旨が書かれていれば正常です。書かれていない場合は、「ログに停止理由を追記してください」と依頼してください。

元データを書き換えられていないか不安

kadaiA/dummy_data.csv を開いて、1行目が見出し行、データが50行であることを
確認してください。.claude/settings.json の deny でこのファイルの編集を止めているため、
書き換えは起きない設定になっています。

SECTION 08

A-6 ハーネスの有無を比較する

設定なしで作ったときと、設定を置いてコマンド1本を回したあとで、何がどう変わったかを比較ファイルとして手元に残します。研修の価値は当日できたことではなく、月曜日に職場で再現できることで決まります。この10分はその差を埋めるために使います。

A-6 ハーネスの有無を比較する

触るフォルダ	`kadaiA/`
生成物	`handson/kadaiA/COMPARE.md`
終わったと言える状態	3列の表があり、3列目に自分の手で1行足してあり、持ち帰りの節が書けている
難易度	2 / 3（AI に任せない部分があります）

目的 70分を社内で説明できる1ファイルに落とす

「Claude Code が便利だった」では、聞いた人は何をすればよいかわかりません。
「設定ファイルを1本置いたら、レビューの観点が毎回同じになった」なら、置く場所と中身を聞けば真似できます。この10分で作るのは表1つと数行の文章です。

操作 作らせて、読んで、自分で足す

1. 比較表を作らせる

Claude Code に、次の2つのファイルを突き合わせて `kadaiA/COMPARE.md` を作らせます。

- `@memo.md` の `## A-2`：素の状態で自分が書いた懸念
- `@kadaiA/REPORT.md`：ハーネスを入れて回した点検の結果

表の列は次の3つにします。

列	中身	どこから来るか
素の状態で気づけたこと	A-2 で自分が書き出せた項目	`memo.md` の `## A-2`
ハーネスを入れてから拾えたこと	`/selfcheck` の指摘のうち、A-2 に無かったもの	`REPORT.md` と `.work/02_findings.json`
まだ誰も拾えていないこと	どちらにも出ていないが、実務では問題になりそうなこと	あなたの頭の中だけ

HINTS 参考プロンプトは `hints/step04_pipeline.md` にあります。3列の意味だけ伝えれば、文面は自由です。

2. できた表を読む

`COMPARE.md` を開いて、1列目と2列目を突き合わせます。自分が書いた懸念のうち、`/selfcheck` でも拾われたものはどれですか。自分だけが挙げていた項目は、2列目に対応するものがないはずです。その行が、あなたが持っていて仕組みが持っていない視点です。

3. 3列目に自分で1行足す

> **この操作は AI に頼まないでください**
>
> 3列目「まだ誰も拾えていないこと」には、自分の手で1行以上書き足します。AI が拾えなかったのは、能力の問題ではなく、渡していない情報があるからです。何を渡していなかったのかを1行で書けると、`CLAUDE.md` に何を書き足すべきかもわかります。

書きにくければ、次の問いから考えてください。

- このアプリを実際に社内で使うとしたら、最初に困るのは誰か
- データが50件ではなく5000件になったら何が起きるか

- 備品を借りた人・返した人の記録は、どこにも残っていない
- このファイルを引き継いだ人が、最初に開くべきファイルはどれか

4. 持ち帰りの節を書く

`COMPARE.md` の末尾に `## 自分の職場に持ち帰るなら` という見出しを作り、1行から3行書きます。自分の仕事のどの作業で、今日の何を使えそうか。使うとしたら、最初に用意するファイルは何か。ここも AI に書かせず、自分の言葉で書いてください。他人が読んで意味が通るかは気にしなくて構いません。あとで読み返すのは自分です。

自分で考える 増えたのは指摘の数だけか

もう1つ、数に表れない変化があります。**毎回同じ観点で見てくれる**という再現性です。A-2 で AI に聞いたときは、聞くたびに指摘の内容が変わる状態でした。`/selfcheck` は観点リストを持っているので、明日回しても同じ観点で見ます。

1人で使うときと、5人のチームで使うときとで、どちらの価値が大きくなると思いますか。1人なら、頭の中に観点があれば足ります。5人だと、頭の中は共有できません。誰が回しても同じ観点で点検されることが、そのままレビュー品質の下限になります。逆に言えば、観点リストに書き忘れたものは、チームの誰も見なくなります。ファイルに書くとは、そういう責任を持つことでもあります。

この考えを1行、`COMPARE.md` か `memo.md` に足しておいてください。

生成物 handson/kadaiA/COMPARE.md

1. 3列の比較表
2. 3列目に自分の手で足した1行以上
3. `## 自分の職場に持ち帰るなら` の節（1行以上）

OK基準 3つ揃えば演習Aは終わりです

① `COMPARE.md` に **3列** の表がある

② 3列目に **自分の手で1行以上** 足してある

③ `## 自分の職場に持ち帰るなら` に **1行以上** 書いてある

表の行数は問いません。1列目が3行しかなくても、それが A-2 で書けた分なら正しい記録です。実際より多く見せる必要はありません。

追加と考察 一緒に持ち帰るとよいファイル

このステップの成果物は、研修後にそのまま社内共有できる形にしてあります。ファイル名と場所を控えておいてください。`handson/kadaiA/COMPARE.md` です。

ファイル	何が書いてあるか	誰に見せると効くか
kadaiA/COMPARE.md	導入前後の差	上司、チームの意思決定者
kadaiA/REPORT.md	点検の結果と、人間に残した判断	同僚、レビュー担当
.work/02_findings.json	指摘の根拠付き一覧	仕組みを作りたい人
CLAUDE.md	何を前提として渡したか	同じ設定を自部署で作りたい人
.claude/commands/selfcheck.md	手順を文章で書くとどうなるか	同上

社外に持ち出す前に、業務データや社名が混ざっていないかは必ず確認してください。本研修のデータは架空の備品データです。

発展課題 読み手をどこまで具体的に伝えると精度が上がるか

`COMPARE.md` を、上司に見せる1枚の報告として整えるとしたら、どの情報を足し、どの情報を削りますか。足す候補は、たとえば所要時間、何人で何分かかったか、この仕組みを自部署に置くのに必要な作業。削る候補は、演習の手順そのものや、技術的な用語です。

AI に「この読み手向けに書き直してください」と頼むとき、次の3通りを実際に試して、出力の違いを見比べてください。

1. 「上司向けに書き直してください」
2. 「開発経験のない部門長向けに、A4半ページで書き直してください」
3. 「開発経験のない部門長向けに、導入判断に必要な情報だけを A4半ページで書き直してください。技術用語には初出で1行の補足を付けてください」

3番まで書くと出力がどう変わるかを見ておくと、明日から書く指示の粒度が決まります。余力があれば `exercises/challenges/ch04_csv_export.md` にも進めます。作った一覧を CSV で書き出す課題で、`dummy_data.csv` を書き換えない制約の中でどう実現するかを考えます。

うまくいかないとき（A-6）

表が3列にならない、2列で出てくる

「3列目は空欄で構わないので、列だけ作ってください」と伝えてください。3列目は自分で埋める欄です。空欄の列があるほうが、埋める場所がはっきりします。

REPORT.md が無いと言われる

A-5 が完了していない可能性があります。kadaiA/ の中を確認してください。REPORT.md が無い場合は、.work/02_findings.json を代わりに使って比較表を作ることもできます。指摘の一覧はそちらにも入っています。

memo.md の ## A-2 が空だった

いまから3行だけでも書いてください。作ったアプリを思い出して、不安なところを書けば十分です。空のまま比較しても、1列目が空の表しかできません。

AI が3列目まで勝手に埋めてしまった

埋まっている行はそのまま残して構いません。その下に、自分で1行足してください。AI が書いた行と自分が書いた行が並ぶと、視点の違いが見えます。どちらが自分の行かがわかるよう、行頭に「(自分)」と付けておくと後で読みやすくなります。

時間が足りない

3列の表と、3列目の1行だけを優先してください。## 自分の職場に持ち帰るなら は、休憩時間や研修後に書き足せます。書く場所を作っておくだけでも、あとから戻ってこられます。

SECTION 09

演習Aで手元に残るもの

70分が終わった時点で、`handson` フォルダには次のものが残っています。どれも研修後に開き直せます。会場では時間の都合で読み飛ばした箇所も、ファイルは残っているので後から追えます。

ファイル	いつできたか	中身
kadaiA/index.html	A-1・A-4・A-5	設定なしで作り、機能を1つ足し、設定ありで点検・修正したアプリ本体
memo.md	A-0 〜 A-5	自分の懸念と、各ステップの気づき。A-2 の内容は点検の入力にもなりました
CLAUDE.md	A-3	AI が起動時に自動で読む共有の前提。9つの見出しで構成されています
.claude/	A-3	設定・コマンド・観点・サブエージェント。`settings.json` の `deny` が元データを守りました
.work/ の4本	A-5	点検の途中経過。棚卸し・指摘・修正計画・ループログ
kadaiA/REPORT.md	A-5	人間向けの報告書。第5節に「人間に判断してほしいこと」が残っています
kadaiA/COMPARE.md	A-6	導入前後の比較と、自分の職場への持ち帰り

この70分で確かめたこと

同じ AI が、設定ファイルの有無で振る舞いを変えました。変わったのは3つです。応答の形（実行サマリーが毎回付く）、点検の観点（毎回同じ13項目で見る）、そして守られる範囲（`deny` に書いた操作は実行されない）。どれもファイルを置いただけで、AI に「こうしてください」と毎回書いてはいません。

それでも、A-6 の3列目は空きませんでした。設定ファイルに書いていないことは、AI には見えないままです。何を書けば見えるようになるか、何は書いても伝わらないかの線引きは、自分の職場のファイルを書きながら見つけることになります。演習Bでは、この設定ファイル一式を別の題材に持ち込んで、初見の Java プロジェクトを相手にします。

関連する資料

手順の本体は配布フォルダの `exercises/` にあります。詰まったときの参考プロンプトは `hints/` にまとまっています。基本操作・用語集・トラブル対処は手元ガイドの共通編を、演習Bは課題B編を参照してください。

Claude Code 公式ドキュメント
CLAUDE.md（メモリ）の仕様
スラッシュコマンドの作り方
サブエージェント
settings.json と権限設定
Claude Code × Amazon Bedrock
VSCode 画面の各部名称
ローカルファイルの読み込み制限について

SOURCES

Claude Code 公式ドキュメント（Anthropic）
CLAUDE.md（メモリ）の仕様（Anthropic）
スラッシュコマンドの作り方（Anthropic）
サブエージェント（Anthropic）
settings.json と権限設定（Anthropic）
Claude Code を Amazon Bedrock 経由で使う（Anthropic）
VSCode 画面の各部名称（Microsoft）
ローカルファイルの読み込み制限（MDN）

生成AIプログラミング入門編 手元ガイド 課題A編

Givery, Inc.