



2026・生成AIプログラミング入門

# 生成AIプログラミング入門編

## ハンズオンガイド

### 受講者向け手元資料

VSCode 上の Claude Code で、自然言語の指示だけから業務アプリを立ち上げる体験と、既存の Java プロジェクトを Issue 駆動で修正する体験の2本を、この1ページに沿って進めます。前半の課題Aでは一覧・検索・詳細ドリルダウンのアプリをテンプレートなしで作成、完成後に危険な点を AI 自身に指摘させます。後半の課題Bではバグ入りの Spring Boot プロジェクトを AI で読み解き、Issue を起票してから直します。各ステップは「自分で考える → 指示を書く → 実行する → 答え合わせ」の4段で進みます。まず自分の言葉で指示を組み立ててから、hints の参考プロンプトと突き合わせてください。

**形式** 会場 / 4時間 (240分) **対象** 生成AIプログラミング未経験～初心者 **環境** VSCode × Claude Code (Amazon Bedrock 経由 / Sonnet 4.6)

**配布素材** handson フォルダ (この中の kadaia / kadaib を VSCode で開いて使います)

Claude Code

VSCode

Spring Boot

### SECTION 01・OVERVIEW

## この研修で身につくこと

AI の基礎理論には踏み込みません。VSCode のターミナルで Claude Code に自然言語で指示し、動くものを作る・既存コードを理解する・Issue 単位で直す、という手元の動きに範囲を限定します。作って終わりではなく、生成物の危険な点を確認する目線までを持ち帰ります。

#### SKILL 01

### Claude Code の基本操作

ターミナルでの `claude` 起動、自然言語での指示、`@` でのファイル参照、Plan モード (Shift+Tab 2回) での計画確認と Agent モードでの実行を、場面で使い分けます。

#### SKILL 02

### テンプレートなしのアプリ構築

要件 (お題シート) と元データ (CSV) だけを渡し、一覧→検索→詳細の業務アプリをゼロから立ち上げます。一気に頼まず、動く最小形から積み上げる指示の出し方を体験します。

### SKILL 03

#### AIによるコード理解

初見の Spring Boot プロジェクトに日本語コメントを挿入させ、画面と処理の流れを短時間で掴みます。読めないコードを前に止まらない読み方を覚えます。

### SKILL 04

#### Issue駆動のバグ修正

見つけた不具合を Issue（ローカルの Markdown ファイル）として起票し、Issue 1件ずつを AI に渡して修正します。勢いだけのバيبコーディングとの違いを自分の手で確かめます。

#### 本研修のスタンス

前半は勢いよく作る楽しさに振り、後半でその限界に自分でぶつかってもらう構成です。AI の出力は同じ指示でも毎回ばらつきます。隣の人と画面が違ってても正常です。各ステップの OK 基準を満たしていれば正解と考えてください。

## SECTION 02 ・ TIMETABLE

### タイムテーブル（240分）

前半で座学とデモ、中盤からハンズオン2本、最後にまとめのデモと振り返りです。各セッションは[Nmin]で配分しています。

Session	内容	時間
01	オリエンと支援の段階遷移 ― ゴール共有、VSCode と Claude Code の起動確認、補完から対話委任までの位置づけ	[20min]
02	座学とデモ ― AIコーディングの現在地、VSCode での基本操作、Plan モードと Agent モードの使い分け	[40min]
03	ハンズオン 課題A ― CRUD業務アプリをテンプレートなしで構築、完成後に危険性を AI に指摘させる	[70min]
04	ハンズオン 課題B ― Java / Spring Boot のコメント挿入とソース理解、Issue 起票、Issue 駆動のバグ修正	[70min]
-	まとめとAI設計デモ ― バイブコーディングの強み弱み、コンテキストエンジニアリング、ハーネス動作のデモ	[20min]
05	振り返りと質疑応答	[20min]

#### この資料の使い方

Session 03 と 04 のハンズオン中は、このページを手元に置いて上から順に進めてください。各ステップの参考プロンプトと詰まり対処は配布素材の [hints/](#) フォルダにあります。まず自分で指示を考え、行き詰まったら hints を開く、の順で使うと身につきます。

## SECTION 03 ・ MATERIALS

# 配布素材の構成

配布した **handson** フォルダに、課題A・課題Bの素材と参考プロンプト一式が入っています。課題ごとに開くフォルダが変わる点に注意してください。課題Aは **kadaiA**、課題Bは **kadaiB** を VSCode で開きます。

```
handson/
├── README.md          この案内
├── CLAUDE.md         Claude Code が起動時に自動で読み込む共有文脈
├── AGENTS.md          複数AIツール共通の指示
├── kadaiA/           課題A: CRUD業務アプリをゼロから作る
│   ├── README.md     進め方・OK基準・発展課題
│   ├── dummy_data.csv  備品データ50件（書き換えない）
│   └── docs/お題シート.md  作るアプリの要件
├── kadaiB/           課題B: 備品管理システム（Spring Boot・不具合入り）
│   ├── src/          Java ソース一式
│   ├── issues/       Issue テンプレートと記入例
│   └── mvnw / mvnw.cmd  Maven Wrapper（Maven のインストール不要）
├── docs/             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
├── hints/           各ステップの参考プロンプトと詰まり対処（handson 直下）
│   ├── step01_first_prompt.md  課題A: 最初の1指示
│   ├── step02_crud_build.md    課題A: CRUDアプリ構築
│   ├── step03_warning_check.md  課題A: 危険性の指摘
│   ├── step04_comment_insert.md  課題B: コメント挿入とソース理解
│   └── step05_issue_driven.md  課題B: Issue駆動のバグ修正
```

## 研修素材と外部入力的前提

配布素材に含まれるソースコード・備品データ・不具合はすべて研修用に用意した架空のものです。研修中は Claude Code への入力を自由に行えます。実業務でのお客様コードや本番データの入力可否は、所属組織のセキュリティポリシーと AI 利用ガイドラインに従って判断してください。

## SESSION 03・課題A・[70MIN]

### 課題A：CRUD業務アプリをテンプレートなしで作る

Claude Code への指示だけで、備品データ50件を扱う一覧・検索・詳細ドリルダウンの業務アプリをゼロから作ります。雛形もサンプルコードもありません。要件は **kadaiA/docs/お題シート.md**、データは **kadaiA/dummy\_data.csv** です。全員が動くものを完成させることを最優先にし、仕上げに危険な点を AI 自身に指摘させます。

1. 自分で考える

→

2. 指示を書く

→

3. 実行する

→

4. 答え合わせ

各ステップ共通の進め方です。指示文を先に自分で組み立て、実行して結果を見てから、hints の参考プロンプトと見比べます。参考プロンプトを最初に写すと、指示を設計する力がつきません。

#### A-1 環境確認と最初の1指示

開くフォルダ handson/kadaiA（VSCode でこのフォルダを開く）

全員で同じ1指示を実行し、Claude Code が動くことと操作のリズムを確認します。ここでのつまずきは遠慮なく挙手して解消してください。以降の70分がスムーズになります。

## 1

## CSV を読ませて説明させる

## 目的

Claude Code がフォルダ内のファイルを読み、日本語で応答するまでの一連の動きを体験します。

## 操作

1. VSCode で `kadaiA` フォルダを開きます
2. ターミナル（表示 → ターミナル）で `claude` と入力して起動します
3. `dummy_data.csv` の内容を説明させる指示を、自分の言葉で入力します

## 自分で考える

「何が」「何件」「どんな項目で」入っているかを AI に答えさせるには、どんな日本語で頼めばよいでしょうか。続けて、カテゴリごとの件数を表で出させる指示も考えてみてください。

## 生成物の名前と場所

この段階ではファイルは作りません。ターミナル上の応答（データの説明と集計表）が成果です。

## OK 基準

- `dummy_data.csv` が備品データ50件であることを AI が説明した
- カテゴリごとの件数が表形式で返ってきた

## 追加と考察

ファイルを読むだけの依頼と、集計する依頼で AI の動き方がどう違ったかを見比べてください。隣の席と出力の言い回しが違うのも正常です。生成は毎回ばらつきます。

## 答え合わせ

参考プロンプトと詰まり対処は `hints/step01_first_prompt.md`（handson 直下の hints フォルダ）を参照してください。

## A-2

## 一覧・検索・詳細のアプリを構築する

要件 `kadaiA/docs/お題シート.md` / 元データ `kadaiA/dummy_data.csv`

課題Aの本編です。お題シートの要件（一覧・備品名の部分一致検索・詳細ドリルダウン・一覧へ戻る）を満たすアプリを、Claude Code への指示だけで作ります。技術スタックは自由です。迷ったら「ブラウザで開くだけで動く形にしてください」と添えると、環境構築なしで確認できます。

## 2

## CRUD業務アプリの構築

## 目的

自然言語の指示だけで動く業務アプリが立ち上がる体験を通じて、AIコーディングの威力と、指示の分割の仕方を掴みます。

## 操作

1. `docs/お題シート.md` を開き、3つの画面要件と「変えないこと」を確認します
2. 一覧表示だけの最小形をまず作らせ、ブラウザで動作を確認します
3. 動いたら検索、次に詳細ドリルダウンと、1機能ずつ追加させます
4. 思った画面と違うときは、作り直しではなく「ここをこう変えてください」と差分で伝えます

## 自分で考える

お題シートの要件を、どの順番で・どこまで一度に頼むかを自分で設計してください。全部を1指示に詰め込む場合と、3回に分ける場合で結果がどう変わるかも試す価値があります。

## 生成物の名前と場所

`kadaiA` フォルダ直下にアプリ一式を生成させます。メインの画面ファイルは `index.html`（または起動手順が README に書かれた形）とします。`dummy_data.csv` は移動・削除・書き換えをしないでください。

## OK 基準

- 備品50件の一覧が画面に表示される（備品コード / 備品名 / カテゴリ / 在庫数 / 保管場所）
- 備品名の部分一致で一覧を絞り込める
- 一覧の1件をクリックすると全項目（購入日・備考を含む）の詳細が表示される
- 詳細から一覧に戻る

## 追加と考察

生成中の画面を眺め、AI がどんなファイルをどんな順で作るかを観察してください。承認を求められたら、差分を読んでから承認する癖をここでつけます。

## 発展課題（時間が余ったら）

- カテゴリでの絞り込みを追加する
- 在庫数が少ない備品を目立たせる表示を追加する
- 新規登録・編集・削除（CRUDのC/U/D）を追加する

## 答え合わせ

参考プロンプトと詰まり対処は `hints/step02_crud_build.md` を参照してください。

## Tips 生成が止まった・違うものができたとき

生成が途中で止まったら「続けてください」で再開できます。見当違いのものができたら、直してほしい箇所だけを具体的に伝えます。Plan モード（Shift+Tab を2回）に切り替えると、実装前に計画を確認してから進められます。

## A-3 危険性・Warning を AI に指摘させる

**物差し** docs/セキュリティチェックリスト.md（handson 直下の docs フォルダ）

完成したアプリの危険な点・曖昧な点・Warning になりそうな点を、作った本人である AI に指摘させます。作っている最中は何も警告しなかった AI が、聞かれれば山ほど挙げる。ここに勢いだけで作るパイプコーディングの限界が表れます。

### 3 セルフレビューの依頼と突き合わせ

#### 目的

「動くこと」と「安心して使えること」の距離を自分の目で確認し、チーム開発でルール（ハーネス）が要る理由を体感します。

#### 操作

1. いま作ったアプリの危険な点・曖昧な点・Warning をすべて挙げさせる指示を入力します
2. 観点（セキュリティ / 入力チェック / エラー処理 / 読みやすさ）を分けて指摘させると整理しやすくなります
3. 指摘が出たら、チーム開発で一番問題になるものはどれかを続けて聞きます

## 自分で考える

AI の指摘を `docs/セキュリティチェックリスト.md` と突き合わせ、AI が挙げなかった項目がないかを探してください。AI のセルフレビューにも抜けがあります。

## 生成物の名前と場所

ターミナル上の指摘リストが成果です。余裕があれば、指摘を `kadaiA/review_memo.md` として保存させると振り返りに使えます。

## OK 基準

- 自作アプリへの指摘が観点別に3件以上挙がった
- チェックリストと突き合わせ、AI の指摘の過不足を1つ以上自分で見つけた

## 追加と考察

なぜ AI は作っている最中に警告しなかったのでしょうか。指示に品質の観点が含まれていなければ、AI は「動くもの」を最短で作ります。この観点をあらかじめ渡しておく仕組みが、後半とまとめて扱うハーネスです。

## 発展課題（時間が余ったら）

- 指摘された点を1つ選び、修正まで依頼して動作を確認する

## 答え合わせ

参考プロンプトと詰まり対処は `hints/step03_warning_check.md` を参照してください。

## 課題A 到達チェック

- ☐ 一覧・検索・詳細ドリルダウンが動くアプリを、テンプレートなしで完成させた
- ☐ 1機能ずつ積み上げる指示の出し方を体験した
- ☐ AI のセルフレビューで、自作アプリの危険な点を確認した
- ☐ 「動く」と「安心して使える」の間に距離があることを自分の言葉で言える

## SESSION 04 ・ 課題B ・ [70MIN]

### 課題B：Spring Boot のバグを Issue 駆動で直す

題材は備品管理システム（Java 17 / Spring Boot 3.4 / H2 インメモリDB / Thymeleaf）。意図的な不具合が仕込まれています。初見のコードを AI で読み解き、見つけた不具合を Issue として書き、Issue 1件ずつを AI に渡して修正します。Git や GitHub は使いません。Issue は `kadaiB/issues/` フォルダの Markdown ファイルです。

ステップ	内容	時間
B-1	コメント挿入とソース理解	[25min]
B-2	Issue 駆動開発の説明と起票練習	[20min]
B-3	Issue 駆動でバグを修正する	[25min]

## 起動のしかた

VSCode で `kadaib` フォルダを開き、ターミナルで Mac は `./mvnw spring-boot:run`、Windows PowerShell は `.\mvnw.cmd spring-boot:run` を実行します。初回はライブラリのダウンロードで数分かかります。起動後にブラウザで `http://localhost:8080/equipments` を開きます。停止は Ctrl+C です。

## B-1 コメント挿入とソース理解

開くフォルダ `handson/kadaib` / 中心ファイル `src/main/java/com/example/equipment/service/EquipmentServiceImpl.java`

初見のプロジェクトを、AI に日本語コメントを入れさせながら読み解きます。構成は controller → service（インターフェース）→ serviceImpl → repository → model の流れです。自分で全部読まなくても、AI に説明させれば処理の流れは短時間で掴めます。

### 4 プロジェクト把握とコメント挿入

#### 目的

既存コードの理解に AI を使う型（全体像 → 中心ファイル → 動作確認）を身につけます。

#### 操作

- プロジェクトの構成と処理の流れを日本語で説明させます
- `EquipmentServiceImpl.java` の各メソッドに、意図がわかる日本語コメントを追加させます（ロジックは変更させない）
- アプリを起動し、一覧・検索・詳細・貸出の画面を一巡します

#### 自分で考える

「ロジックは変えずにコメントだけ足す」を守らせるには、指示にどんな一文が要るでしょうか。制約を明示しない指示と比べてみてください。

#### 生成物の名前と場所

日本語コメントが入った `EquipmentServiceImpl.java`。ファイルの場所は変わりません。

#### OK 基準

- 画面が2つ（一覧・詳細）あり、処理がどのファイルをどう流れるかを自分の言葉で説明できる
- コメント追加の差分にロジック変更が混ざっていないことを確認した
- アプリが起動し、一覧画面が表示された

#### 追加と考察

README の「業務ルール」5項目を読み、画面の実際の挙動と食い違う箇所の当たりをつけてください。この食い違いが次のステップで起票する不具合です。

#### 答え合わせ

参考プロンプトと詰まり対処は `hints/step04_comment_insert.md` を参照してください。

## B-2 Issue 駆動開発の説明と起票練習

テンプレート `kadaib/issues/ISSUE_TEMPLATE.md` / 記入例 `kadaib/issues/ISSUE_001_詳細画面がエラーになる備品がある.md`

Issue とは、不具合や修正したいことを1件ずつ独立した文書にしたものです。チーム開発では「誰が見ても再現でき、直ったかどうか判定できる」粒度で書かれた Issue が作業の単位になります。本研修では `issues/` フォルダの Markdown ファイルとして起票します。

### 目的

「なんか変」を、再現手順・期待する動き・実際の動きに分解して書けるようになります。

### 操作

1. アプリを触って不具合を探します。一覧・検索・詳細・貸出を一巡し、存在しない ID の URL や、在庫より大きい貸出数など、意地悪な操作も試します
2. 記入例 ISSUE\_001 を読み、書き方の粒度を確認します
3. 見つけた不具合を `ISSUE_TEMPLATE.md` の形式で `issues/` に起票します（例: `ISSUE_002_内容がわかる名前.md`）

### 自分で考える

README の業務ルール5項目が探索の地図です。「在庫5個以下は要補充表示」「在庫を0未満にしない」「日付は yyyy/MM/dd」「存在しないIDはエラー画面にしない」「備考未設定でも詳細が表示される」。仕様と挙動の食い違いを自分の操作で再現してから書きます。

### 生成物の名前と場所

`kadaiB/issues/ISSUE_00N_内容がわかる名前.md` を2件以上。

### OK 基準

- Issue を2件以上起票した
- 各 Issue に再現手順・期待する動き・実際の動きが書かれている

### 追加と考察

不具合探しを AI に手伝わせることもできます。ただし AI の挙げた候補は、必ず自分の操作で画面上に再現してから起票してください。再現できない指摘は Issue になりません。

### 答え合わせ

起票のヒントは `hints/step05_issue_driven.md` の前半を参照してください。

**対象** 自分が起票した `kadaiB/issues/` の Issue

起票した Issue を1件選び、その Issue だけを AI に渡して修正させます。課題Aの「思いつきで指示する」進め方との違いは、修正の根拠と完了条件が文書に固定されていることです。直ったかどうかは Issue の再現手順で自分が判定します。CI/CD は扱いません。

### 目的

Issue を1件ずつ渡し、修正 → 再現手順で確認 → 次へ、というチーム開発の基本サイクルを回します。

### 操作

1. Issue を1つ選び、`@` でその Issue ファイルを参照させて修正を依頼します
2. 提示された差分を読み、Issue の範囲を超えた変更が混ざっていないか確認してから承認します
3. アプリを再起動し、Issue の再現手順どおりに操作して直ったことを確認します
4. 時間の許す範囲で2〜3件目の Issue に進みます



## 自分で考える

複数の不具合を一度に全部直させる指示と、Issue 1件ずつの指示で、差分の読みやすさと確認のしやすさがどう変わるかを意識してください。

## 生成物の名前と場所

修正済みのソースコード（`kadaiB/src/` 配下）。修正した Issue ファイルの末尾に「修正済み・確認日」を追記しておく与管理の練習になります。

## OK 基準

- Issue を1件以上修正し、再現手順ごとの操作で直ったことを確認した
- 差分を読んでから承認する手順を守れた

## 追加と考察

修正の途中で別の不具合に気づいたら、その場で直さず新しい Issue として起票します。1つの修正に1つの目的。これが崩れると、何を直したのか誰にもわからなくなります。

## 発展課題（時間が余ったら）

- 残りの Issue を修正する
- 修正後のコードを `docs/コーディング規約.md` に照らしてレビューさせる

## 答え合わせ

参考プロンプトと詰まり対処は `hints/step05_issue_driven.md` を参照してください。

## 課題B 到達チェック

- ☐ AI のコメント挿入で初見コードの流れを掴めた
- ☐ Issue を2件以上、再現手順つきで起票した
- ☐ Issue 1件以上を修正し、再現手順で直ったことを確認した
- ☐ 課題Aの進め方との違い（根拠と完了条件が文書にある）を説明できる

## SESSION 04 後半～まとめ・発展

# 発展トピックとまとめの観察ポイント

ハンズオン後の座学とデモで扱う内容です。手を動かす場面はありませんが、今日の2つの体験がどうつながるかを掴む時間です。デモを見るときの観察ポイントを挙げておきます。

## TOPIC 01

### ハルシネーションと注意点

AI は存在しない API や設定を、正しそうな顔で出力することがあります。課題Bで「AI の指摘は自分の操作で再現してから起票する」と決めていたのは、この対策そのものです。

## TOPIC 02

### コンテキストエンジニアリング

会社のナレッジや個人の暗黙知を、AI が読める形（規約・チェックリスト・CLAUDE.md）に整えておく考え方です。配布素材の `docs/` と `CLAUDE.md` がその実例です。

## TOPIC 03

### コンテキスト → ハーネス → ループ

整えた文脈（コンテキスト）を、毎回自動で効かせる仕組み（ハーネス）に載せ、生成 → 検証 → 修正を回し続ける（ループ）。チーム開発で AI を安全に速く使う設計思想です。

## TOPIC 04

### AIエージェントの発展

今日使った対話型の先には、複数の専用エージェントが分担して動く形があります。まとめのデモで実物を見ます。

#### まとめのAI設計デモ 観察ポイント

デモでは、1つのコマンドから必要な Skills や Subagents を AI が自分で選択して呼び出し、質の高い出力に至るまでを見せます。次の3点を意識して見てください。

- 人間が打った指示は1つだけなのに、裏で何段階の処理が動いたか
- 課題Aで自分が手で打った「観点を分けて指摘して」のような指示が、どこに仕組みとして埋め込まれているか
- 出力の検証を誰が（人間か、仕組みか）やっているか

#### 今日の2つの体験のつながり

課題Aで体験した「速いが危うい」と、課題Bで体験した「文書を挟むと確実になる」。この2つの間を埋めるのがハーネスです。次回以降の研修では、この仕組みを自分たちで作る側に回ります。

## FAQ & TROUBLESHOOTING

### よくあるトラブルと対処

VSCode と Claude Code、課題Bの起動まわりでつまずきやすい点をまとめます。解消しないときは挙手してください。

#### CLAUDE CODE

##### claude が起動できない

VSCode のターミナルで `claude --version` を実行し、バージョンが表示されるか確認します。表示されない場合は事前セットアップガイドの手順を見直してください。本研修は Amazon Bedrock 経由の接続設定を使うため、設定ファイルの場所もガイドに記載があります。

#### CLAUDE CODE

##### 応答が返らない・止まる

生成が途中で止まったら「続けてください」で再開できます。文脈が膨らんで重いときは `/compact` で要約圧縮、話題を切り替えるときは `/clear` でリセットします。

## CLAUDE CODE

### Plan モードに切り替わらない

Plan モードへの切り替えは Shift+Tab を2回です。IME が日本語入力の状態だと効かないことがあるので、半角英数に切り替えてから試します。指示が思ったファイルに届かないときは `@` で対象を明示します。

## 生成物

### 思ったものと違うものができた

作り直しを頼むより、「ここをこう変えてください」と差分で伝えるほうが速く確実に直ります。同じ指示でも出力は毎回ばらつくため、隣の席と画面が違うのは正常です。OK 基準を満たしていれば正解です。

## 課題B / 起動

### mvnw が動かない・起動に失敗する

`java -version` で 17 と表示されるか確認します。Windows PowerShell では `.\mvnw.cmd spring-boot:run` と先頭の `.\` を忘れずに入力します。初回はライブラリのダウンロードで数分かかるため、エラーでなければそのまま待ちます。

## 課題B / 起動

### 8080 番ポートが使えない

前回起動したアプリが残っている可能性があります。起動したターミナルで Ctrl+C を押して停止するか、ターミナルごと閉じてから起動し直します。

## 課題B / DB

### H2 コンソールに繋がらない

`http://localhost:8080/h2-console` を開き、JDBC URL に `jdbc:h2:mem:equipmentdb`、ユーザー名 `sa`、パスワード空欄で接続します。既定値の `jdbc:h2:~/test` のままだと繋がりません。

## REFERENCES

## 参考リンク

研修後の自習に使える公式情報です。

[Claude Code ドキュメント](#)[CLAUDE.md \(メモリ\)](#)[Claude Code よくある使い方](#)[Claude Code × Amazon Bedrock](#)[VSCode ドキュメント](#)[Spring Boot](#)[H2 Database](#)[Maven](#)

生成AIプログラミング入門編 ハンズオンガイド

Givery, Inc.