

ハンズオンガイド

Cursor基礎講座

架空のノースパイン商事の在庫管理システムに機能を足し、古い受注金額の計算をテストで守りながら書き直します。どの演習も、受講者が先に自分の考えを書き、そのあとでCursorに頼む順に進めます。

2026年10月23日（金）

演習4本と講師デモ

Cursorのエディタの画面

目次

演習の始め方

- 配布フォルダの開き方
- 4つの操作
- 演習のきまり
- STEP 1** 開始時の利用状況の記録

演習1 在庫管理コードへの機能追加

- STEP 1** 引当の関数
- STEP 2** 出荷の上限チェック
- STEP 3** 業務ルールの確認
- STEP 4** 保管棚の追加
- できあがりの目安
- 早く終わった方への追加と考察
- 発展課題

演習2 Cursor Rulesの作成

- STEP 1** 自分のチームの規約の書き出し
- STEP 2** Rulesなしでの依頼
- STEP 3** Rulesのファイルの作成
- STEP 4** 同じ依頼での見比べ
- できあがりの目安
- 早く終わった方への追加と考察
- 発展課題

演習3 モデルの使い分けと利用状況

- STEP 1** 2つのモデルでの同じ依頼
- STEP 2** 調べ物用のサブエージェント
- STEP 3** 調べ物と設計の分担
- STEP 4** 利用状況の見比べ
- できあがりの目安
- 早く終わった方への追加と考察
- 発展課題

演習4 レガシーコードの改修

- STEP 1** 自分で読む5分
- STEP 2** 仕様の読み取り
- STEP 3** 旧仕様メモとの突き合わせ
- STEP 4** 今の動きを固定するテスト
- STEP 5** 関数への分割
- STEP 6** テストを先に変えてからの修正
- できあがりの目安
- 早く終わった方への追加と考察
- 発展課題

講師デモ SQLとTerraform

振り返り

- STEP 1** 終了時の利用状況の記録
- 持ち帰るもの
- 職場で使う前に確かめること

よくある質問

00

演習の始め方

演習1から演習3は `inventory-app` のフォルダをCursorで開いて進めます。演習4では `legacy-order` のフォルダを開き直します。

配布フォルダの開き方

- Cursorのエディタの画面で `Open project` を押してください。
- デスクトップに展開した `1023_cursor` の中の `inventory-app` を選んでください。
- 左のファイル一覧に `index.html` と `js` のフォルダが表示されたら準備完了です。

4つの操作

操作	呼び出し方	向いている作業
Tab補完	書きかけのコードに出る薄い文字を <code>Tab</code> キーで確定	次の数行を書く作業
インライン編集	コードを選んで <code>Ctrl</code> + <code>K</code>	選んだ範囲だけを直す作業
チャット	<code>Ctrl</code> + <code>L</code> で開くサイドパネル	コードの意味を聞く作業
エージェント	サイドパネルで依頼を送る	複数のファイルにまたがる変更

モデルを切り替えるときは、サイドパネルの入力欄の近くにあるモデルの欄を押すか、`Ctrl` + `/` を押してください。

演習のきまり

演習で使うのは、配布フォルダの架空のコードとデータだけです。Cursorに入力したコードと依頼文は、回答を作るために外部のAIモデルへ送信されます。業務のコードや顧客の情報は入力しないでください。

エージェントがファイルを変えると、変更した箇所が差分で表示されます。内容を読んでから、採用するか取り消すかを選んでください。読まずに採用すると、どこが変わったのかを後から説明できなくなります。

STEP 1

開始時の利用状況の記録

AIを使わない

[2min]

Ctrl + Shift + J

で **Cursor Settings** を開き、 **Plan & Usage** を押してください。 **Included Usage** に出ている割合をメモします。研修の最後にもう一度見て、4時間で何%使ったかを比べます。

01

演習1 在庫管理コードへの機能追加

4つの操作を1回ずつ使い、在庫管理システムに3つの機能を足します。足すのは引当と出荷の上限チェックと保管棚です。

目安の時間は[35min]です。業務ルールは `docs/業務ルール.md` にあります。

STEP 1

引当の関数

Tab補完

[7min]

js/inventory.js

 を開き、 `ship` 関数のすぐ上の空行に、次のコメントと関数の1行目を自分で打ってください。打ち終わると、続きのコードが薄い文字で表示されます。

```
// 受注した数を引き当てる。利用可能数を超えては引き当てられない
function allocate(stocks, code, qty) {
```

薄い文字の中身を読み、業務ルールの「引当」と合っていれば `Tab` キーで確定してください。合っていなければ `Esc` キーで消し、自分で書き足してください。最後に、ファイルの末尾の `return` の一覧に `allocate` を足します。

Tips `Tab`補完は、カーソルの前後にあるコードとコメントから続きを推測します。コメントに「何を、どこまで」を書いておくと、提案が業務ルールに近づきます。

STEP 2 出荷の上限チェック

インライン編集 [7min]

今の `ship` 関数は、引当数より多い数を出荷しても止まりません。業務ルールの「出荷できるのは引当数まで」を、インライン編集で足します。

- 1 `ship` 関数の全体を選んでください。
- 2 `Ctrl` + `K` を押し、表示された入力欄に次の依頼文を貼り付けて送信してください。
- 3 差分を読んで、エラーの文言が日本語になっていることを確かめてから採用してください。

インライン編集の依頼文

出荷数が引当数を超える場合は、出荷せずに例外を投げてください。エラーの文言は「商品コード NP-1001 は引当数 3 を超えて出荷できません」の形にしてください。

STEP 3 業務ルールの確認

チャット [6min]

先に自分で答えを考えてから聞きます。 `docs/業務ルール.md` の発注点の節を読んでください。そのうえで `index.html` をブラウザで開き、発注が必要な商品が何件あるかを数えます。

次に `Ctrl` + `L` でサイドパネルを開き、次の質問を送ってください。自分で数えた件数と、Cursorが答えた件数と理由を比べます。

チャットの質問

@docs/業務ルール.md と @js/inventory.js を読んで教えてください。発注が必要かどうかは、どの数をどの値と比べて判定していますか。初期データで発注が必要な商品のコードを、理由と一緒に全部挙げてください。

答え合わせ

判定は利用可能数（在庫数から引当数を引いた数）と発注点を比べ、利用可能数が発注点以下なら発注が必要です。初期データではNP-1002、NP-2002、NP-3002、NP-4001、NP-5002の5件が該当します。NP-3002は在庫数35なので

見落としやすいのですが、引当数8を引くと利用可能数27になり、発注点30を下回ります。

STEP 4 保管棚の追加

エージェント

[15min]

倉庫の担当者から「商品がどの棚にあるかを一覧で見たい。棚の列（A、B、C）で絞り込みたい」と頼まれた想定です。データと計算と画面の3つのファイルにまたがるので、エージェントに頼みます。

依頼を送る前に、どのファイルのどこが変わるはずかを、自分で3行のメモに書いてください。エージェントの差分と見比べるために使います。

エージェントへの依頼文

商品に保管棚の項目 `location` を追加してください。値は「A-01」のように、棚の列を表す英字1文字とハイフンと2桁の番号にします。

1. `js/data.js` の10商品すべてに `location` を入れてください。用紙と梱包資材はA、筆記具とファイルはB、OA消耗品はCの列にしてください
2. `js/inventory.js` に、棚の列の英字で商品を絞り込む関数 `filterByZone` を足してください。空文字のときはすべて返します
3. 在庫の一覧に保管棚の列を足し、表の上に棚の列を選ぶ欄（すべて、A、B、C）を置いてください
4. `tests/inventory.test.js` に `filterByZone` のテストを1件足してください

差分を採用したら、`index.html` をブラウザで開き直して、棚の列で絞り込めることを確かめてください。続けて `tests/test.html` を開き、テストの結果を見ます。

できあがりの目安

- ☐ `tests/test.html` の結果が「13件中 13件 合格」になる。STEP 1とSTEP 2のテストは、Cursorに頼んで足してもよい
- ☐ 画面で引当数を超える数を出荷すると、日本語のエラーが表示されて在庫が変わらない
- ☐ 画面でCの列を選ぶと、トナーとインクの2行だけが表示される

早く終わった方への追加と考察

STEP 4の前に書いた3行のメモと、エージェントが実際に変えたファイルを比べてください。メモになかった変更があれば、それが必要だった理由をチャットで聞きます。自分の業務で、4つの操作のどれを一番使いそうかを1行で書いておくと、最後の振り返りで使えます。

発展課題

入庫の取り消し（間違えて入庫した数を戻す）を、Tab補完とエージェントの両方で作り、出てきたコードの違いを比べてください。演習2のRulesを作ったあとにもう一度試すと、違いがさらにはっきりします。

演習2 Cursor Rulesの作成

チームのコードの書き方をRulesのファイルに書き、Rulesを置く前と後で、同じ依頼から出てくるコードを比べます。目安の時間は[25min]です。Rulesは、チャットとエージェントへの依頼に使われます。Tab補完とインライン編集には使われません。

STEP 1 自分のチームの規約の書き出し

AIを使わない

[5min]

自分のチームで、コードレビューのたびに指摘している書き方を3つ、メモに書き出してください。関数名の付け方、エラーの扱い、テストの置き場所のように、レビューで直させた経験のあるものを選びます。

STEP 2 Rulesなしでの依頼

エージェント

[5min]

サイドパネルの右上から新しいチャットを開き、次の依頼を送ってください。出てきた関数名、引数の名前、エラーの文言、テストを足したかどうかをメモします。差分はまだ採用せずに取り消してください。

Rulesの前後で同じに使う依頼文

js/inventory.js に、間違えて在庫した数を取り消す関数を追加してください。

STEP 3 Rulesのファイルの作成

エージェント

[8min]

サイドパネルの入力欄に `/create-rule` と打ち、続けてSTEP 1で書いた規約と、次の規約を合わせて送ってください。エージェントが `.cursor/rules` のフォルダに `.mdc` のファイルを作ります。

`/create-rule` に続けて送る規約

- 在庫管理コードの書き方を、常に適用されるRulesにしてください。
- 関数名は「動詞＋名詞」のcamelCaseにする
 - 数量を表す引数と変数はqtyとする
 - 在庫の配列を直接書き換えず、mapやfilterで新しい配列を返す
 - エラーは日本語のメッセージで例外を投げ、先頭に「商品コード <コード> は」を付ける
 - 関数を足したり変えたりしたら、tests/inventory.test.js にテストを1件以上足す

できたファイルを開き、先頭の `alwaysApply: true` と、自分の規約が入っていることを確かめてください。

できあがりのファイルの例を見る

description: ノースパイン商事の在庫管理コードの書き方
alwaysApply: true

在庫管理コードの書き方

- 関数名は「動詞＋名詞」のcamelCaseにする（例: cancelReceipt、listReorderItems）
- 数量を表す引数と変数はqtyとし、quantityやnumとは書かない
- 在庫の配列を直接書き換ええない。mapやfilterで新しい配列を返す
- エラーは日本語のメッセージで例外を投げ、先頭に「商品コード <コード> は」を付ける
- コメントは日本語で、処理の理由だけを書く
- 関数を足したり変えたりしたら、tests/inventory.test.jsにテストを1件以上足す

適用のしかたは4種類です。今回の `alwaysApply: true` はAlways Applyで、すべてのチャットに読み込まれます。descriptionだけを書くとはApply Intelligentlyになります。その場合に読み込まれるのは、エージェントが関係すると判断したときだけです。

STEP 4 同じ依頼での見比べ

エージェント

[7min]

新しいチャットを開き、STEP 2と同じ依頼文を送ってください。STEP 2のメモと比べて、変わったところに印を付けます。見るのは関数名と引数の名前、エラーの文言、テストの有無の4点です。今度の差分は採用し、`tests/test.html` で全件合格することを確認してください。

できあがりの目安

- ☐ `.cursor/rules` のフォルダに `.mdc` のファイルが1つあり、先頭に `alwaysApply: true` がある
- ☐ STEP 2とSTEP 4の違いを、4点のうち2点以上メモに書けている
- ☐ `tests/test.html` が全件合格し、取り消しの関数のテストが1件以上増えている

早く終わった方への追加と考察

Rulesに書いたのに守られなかった規約があれば、その規約の書き方を具体的にしてもう一度試してください。「わかりやすい名前にする」のような書き方は守られにくく、「動詞＋名詞」のように判定できる書き方は守られやすくなります。

発展課題

テストのファイルにだけ使うRulesを、Apply to Specific Filesで作ってください。対象を `tests/**/*.js` に絞り、「テスト名は日本語で、何を確かめるかが読める文にする」と書きます。

演習3 モデルの使い分けと利用状況

同じ依頼を2つのモデルで比べ、調べ物だけを軽いモデルのサブエージェントに任せます。最後に、利用状況の画面で使った量を確認めます。

目安の時間は[30min]です。演習1と演習2で変えた `inventory-app` をそのまま使います。

STEP 1 2つのモデルでの同じ依頼

チャット

[10min]

モデルの欄でComposer 2.5を選び、新しいチャットで次の依頼を送ってください。続けて新しいチャットを開き、モデルの欄でComposerとGrok以外のモデルを1つ選んで、同じ依頼を送ります。

2つのモデルで比べる依頼文

発注が必要な商品について、仕入先に送る発注書の下書きを作る機能を設計してください。コードはまだ書かず、関数の名前と引数と戻り値、発注する数の決め方を、箇条書き10行以内で答えてください。

2つの答えを3点で比べ、メモに書いてください。3点は、返ってくるまでの時間と発注する数の決め方の具体さ、10行に収まったかどうかです。

Tips Composer 2.5とGrokは、Cursorのモデル向けの利用枠から使われます。ほかの会社のモデルが使うのは、別の利用枠です。どちらの枠から使ったかは、ブラウザで開く <https://cursor.com/dashboard/usage> の明細で確認られます。

STEP 2 調べ物用のサブエージェント

エージェント

[8min]

モデルの指定はRulesには書けません。作業ごとにモデルを変えたいときは、サブエージェントのファイルに `model` を書きます。次の内容を、`inventory-app` の中の `.cursor/agents/code-searcher.md` として保存してください。

`.cursor/agents/code-searcher.md`

```
---
name: code-searcher
description: コードベースの検索専用。関数やファイルの場所、変数の使われ方を調べるときに使う。コードは書き換えない。
model: composer-2.5
readonly: true
---
```

あなたは在庫管理システムのコードを調べる担当です。

依頼されたら、次の順で答えてください。

1. 関係するファイルと関数を、ファイルのパスと行番号つきで一覧にする
2. それぞれが何をしているかを1行で書く
3. 依頼の答えに必要な事実だけを書き、設計の提案はしない

`model: composer-2.5` は、このサブエージェントだけを軽いモデルで動かす指定です。`readonly: true` にすると、このサブエージェントはファイルを書き換えません。

STEP 3 調べ物と設計の分担

エージェント

[8min]

チャットのモデルを、STEP 1で選んだComposer以外のモデルにしたまま、次の依頼を送ってください。先頭の `/code-searcher` で、調べ物だけをサブエージェントに任せます。

サブエージェントを呼ぶ依頼文

`/code-searcher` 在庫数が引当数を変えているところをすべて洗い出してください。
そのあと、その結果をもとに、返品を受けたときに在庫を戻す関数の設計を5行以内で書いてください。コードはまだ書かないでください。

チャットに表示されるサブエージェントのカードに、どのモデルで動いたかが出ます。`composer-2.5` で動いたことを確かめてください。

STEP 4 利用状況の見比べ

AIを使わない

[4min]

`Plan & Usage` の `Included Usage` を開き、研修の最初にメモした割合と比べてください。依頼ごとのモデル名とトークン数は、ブラウザで開く明細の画面に並びます。明細の画面は <https://cursor.com/dashboard/usage> です。STEP 1の2つの依頼のトークン数をメモに書き写してください。

できあがりの目安

- ☐ STEP 1の2つの答えを、3点で比べたメモがある
- ☐ `.cursor/agents/code-searcher.md` があり、サブエージェントのカードに `composer-2.5` と表示された
- ☐ 利用状況の明細から、STEP 1の2つの依頼のトークン数を書き写せた

早く終わった方への追加と考察

自分の業務の作業を3つ挙げ、それぞれを軽いモデルに任せるか、上位のモデルに任せるかをメモに書いてください。判断の理由に「間違えたときに気づけるか」を入れると、使い分けの基準が説明しやすくなります。

発展課題

テストを流して結果を報告するだけのサブエージェント `test-runner` を作ってください。 `description` に「use proactively」と書くと、エージェントが自分から呼ぶようになるかを試します。

04

演習4 レガシーコードの改修

2012年から継ぎ足してきた受注金額の計算を、今の動きをテストで固定してから書き直し、旧仕様メモとの食い違いを直します。

目安の時間は[50min]です。Cursorで `legacy-order` のフォルダを開き直してから始めてください。

STEP 1 自分で読む5分

AIを使わない

[5min]

`js/order_legacy.js` の `calc` 関数を、上から自分で読んでください。値引、送料、消費税の3つについて、どう計算しているかを1行ずつメモに書きます。読み切れなかったところは「不明」と書いて構いません。

STEP 2 仕様の読み取り

エージェント

[8min]

仕様を読み取る依頼文

`js/order_legacy.js` の `calc` 関数が今どう動いているかを、`docs/仕様_読み取り.md` に書き出してください。

- 入力、エラー、値引、送料、消費税、合計の6つの見出しに分ける
- コードから読み取れる事実だけを書く。コメントとコードが食い違う箇所は、コードの動きを書いたうえで食い違いを注記する
- 全体でA4の1枚に収まる長さにする

できたファイルを、STEP 1の自分のメモと比べてください。自分が「不明」にしたところが、どう説明されたかを確認めます。

STEP 3 旧仕様メモとの突き合わせ

チャット

[7min]

`docs/旧仕様メモ.txt` を自分で読み、`docs/仕様_読み取り.md` と食い違っていそうな箇所に印を付けてください。そのあとでチャットに聞きます。

突き合わせの質問

@docs/旧仕様メモ.txt と @js/order_legacy.js を比べて、計算の結果が変わる食い違いを挙げてください。コメントだけの食

い違いは除いてください。それぞれ、食い違いが出る注文の例を1つずつ、会員ランクと地域と明細で示してください。

答え合わせ

計算の結果が変わる食い違いは2つあります。1つ目は送料無料の判定です。メモは値引後の税込金額で判定しますが、コードは値引前の小計で判定しています。GOLD会員が本州でNP-4001とNP-2001を1個ずつ買くと、メモでは送料800円、コードでは0円になります。

2つ目は消費税の端数です。メモは常に切り捨てですが、コードはGOLD会員だけ四捨五入しています。GOLD会員がNP-3002を1個買くと、メモでは37円、コードでは38円です。コメントの「消費税（8%）」は計算に影響しない古いコメントです。

STEP 4 今の動きを固定するテスト

エージェント

[12min]

書き直す前に、今の動きをテストで固定します。テストが全件合格のまま書き直せれば、外から見た動きを変えずに直せたことになります。

テストを書く依頼文

tests/order.test.js に、calc 関数の今の動きを固定するテストを10件足してください。

- 今のコードの結果を正解として書く。旧仕様メモと食い違う箇所も、今の結果のまま書く
- 一般、SILVER、GOLDの値引、4つの地域の送料、送料無料、重量の加算、在庫不足、登録のない商品コードを1件以上ずつ含める
- テスト名は、何を確かめるかが読める日本語にする

tests/test.html をブラウザで開き、「11件中 11件 合格」になることを確かめてください。合格しないテストがあれば、テストとコードのどちらが正しいかをチャットで聞いて直します。

STEP 5 関数への分割

エージェント

[10min]

書き直しの依頼文

js/order_legacy.js の calc 関数を、値引、送料、消費税をそれぞれ計算する関数に分けてください。

- calc の入力と戻り値の形は変えない (index.html から呼ばれているため)
- 計算の結果は1円も変えない。旧仕様メモとの食い違いも、この段階では直さない
- 割引率や送料のような数値は、名前の付いた定数にまとめる
- 終わったら tests/test.html の結果がどうなるかを教えてください

`tests/test.html` を開き直して、全件合格のままであることを確かめてください。不合格が出たら、書き直しで動きが変わった証拠なので、差分を取り消してやり直します。

STEP 6 テストを先に変えてからの修正

エージェント

[8min]

STEP 3 で見つけた送料無料の食い違いを直します。先に、旧仕様メモどおりの結果を正解にしたテストを1件足してください。そのテストだけが不合格になることを確かめてから、コードを直させます。

修正の依頼文

旧仕様メモでは、送料無料の判定は値引後の税込金額が11,000円以上かどうかで行います。

1. GOLD会員、本州、NP-4001とNP-2001を1個ずつの注文で、送料800円、合計11,690円を正解とするテストを `tests/order.test.js` に足してください
2. 今の時点で、そのテストだけが不合格になることを教えてください
3. そのあとで、送料無料の判定を旧仕様メモどおりに直してください

できあがりの目安

- ☐ `tests/test.html` が「12件中 12件 合格」になる
- ☐ `calc` 関数の中に、割引率や送料の数値が直接書かれていない
- ☐ 画面の `index.html` で、GOLD会員の11,000円の注文の送料が800円になる

早く終わった方への追加と考察

STEP 3 で見つけた2つ目の食い違い（消費税の端数）も、STEP 6 と同じ順で直してください。直す前に、旧仕様メモとコードのどちらを正とするかを誰が決めるのかを考え、メモに1行書きます。業務では、コードの動きが長年の運用で正になっていることもあるからです。

発展課題

`legacy-order` にも、書き直したコードを点検するRulesを作ってください。そのRulesを使って、エージェントにコードレビューをさせます。指摘のうち、直すものと直さないものを自分で分け、その理由を書きます。

05

講師デモ SQLとTerraform

講師が画面を共有し、SQLとTerraformのコードの生成からレビューまでを見せます。
目安の時間は[15min]です。使う見本と依頼文は、配布フォルダの `demo` にあります。

デモ	見せること	見るところ
SQL	<code>demo/sql/schema.sql</code> を読ませ、分類ごとの月次売上のSQLを書かせる	表定義を読ませると、列名を推測で書かなくなるか
SQLの手直し	会員ランクの内訳を足し、索引の候補を理由つきで3つ挙げさせる	理由が表定義の事実に基づいているか
Terraform	<code>demo/terraform/main.tf</code> に静的サイトの設定を足させる	既存のリソースグループの名前に合わせられるか
レビュー	足したコードを、公開範囲と暗号化と通信の制限の3点でレビューさせる	生成と点検を別の依頼に分けると、見落としが減るか

06

振り返り

終了時の利用状況を記録し、研修の成果物と、職場で使う前に確かめることを持ち帰ります。

STEP 1

終了時の利用状況の記録

AIを使わない

[3min]

Plan & Usage

 の

Included Usage

 を開き、研修の最初にメモした割合との差を計算してください。講師は、4時間の演習をAPIの料金に換算して5ドル程度と見込んでいます。

持ち帰るもの

成果物	場所
保管棚を足した在庫管理のコード	<code>inventory-app</code> のフォルダ
自分で書いたRules	<code>inventory-app/.cursor/rules</code>
調べ物用のサブエージェント	<code>inventory-app/.cursor/agents/code-searcher.md</code>
書き直した受注金額の計算とテスト	<code>legacy-order</code> のフォルダ
演習のメモと利用状況の記録	自分のメモ

職場で使う前に確かめること

研修で使ったアカウントとプランを、そのまま業務に使ってよいかは、社内の規程と研修事務局の案内に従ってください。業務のコードを入力する前に、プライバシーモードの設定と、社内で許可されたモデルを確かめます。どちらも、ここで決めずに社内の担当部署に確認するのが確実です。

07

よくある質問

Tab補完の薄い文字が表示されません

カーソルを関数の1行目の末尾に置いたまま、少し待ってください。それでも表示されない場合は、画面の下端の **Cursor Tab** の表示を押し、有効になっているかを確かめてください。

tests/test.htmlを開いても結果が表示されません

ファイルの一覧から開くとCursorの中で文字として表示されることがあります。エクスプローラーで **tests** のフォルダを開き、**test.html** をダブルクリックしてブラウザで開いてください。結果の表示が古いときは、ブラウザを再読み込みします。

エージェントが依頼と違うファイルまで変えました

差分を採用せずに取り消し、依頼文に「変えてよいファイルは〇〇と〇〇だけ」と書き足して送り直してください。変えてよい範囲を書くと、エージェントは範囲の外のファイルを変えにくくなります。

サブエージェントのカードに、指定したモデルと違う名前が表示されます

契約しているプランで使えないモデルを指定すると、Cursorは別のモデルに置き換えて動かします。**model** に書いたIDが正しいか、Cursor Settingsの **Models** に同じモデルがあるかを確かめてください。

演習の途中で利用上限に達しました

講師に知らせてください。残りの演習は講師の画面で見いただき、完成版のコードを配布します。業務用のアカウントに切り替えて続けることはしないでください。

