DAY 1　2026年9月3日（木）

題材を掴む ＋ Claude Code の3操作

受注管理システムが何をするものかを掴んだうえで、Claude Code の3操作（ターミナル対話・Planモード・スラッシュコマンド）を自分の手で一通り動かします。読みにくい JSON を自分の画面に変えるところまでを、この日のうちに通します。

この日の演習

演習は上から順に進みます。全員が終わるのを待ってから次へ進むので、詰まったら手を挙げてください。早く終わった方には各演習に発展課題があります。

D1-1 起動確認とコードベース質問 [15min]

ねらい

claude が起動して、CLAUDE.md の中身まで踏まえた答えを返す状態を自分の手で確かめます。

さわるファイル

読む `CLAUDE.md`
何が書いてあるか目を通す

読む `src/main/java/com/example/order/service/OrderServiceImpl.java`
説明と実物を突き合わせる

手順

1. **考える** controller から先はどの順に呼ばれるか、自分の答えを一言メモします。あとで返ってきた説明と突き合わせます。
2. **実行** VSCode で handson フォルダを開き、Ctrl+@ でターミナルを出して .\mvnw.cmd spring-boot:run を打ちます。Started OrderApplication が出たらそのまま放置し、ターミナル右上の＋で2枚目を開いて claude と打ちます。
3. **書く** 2枚目のターミナルで「変更はせずに、このプロジェクトの層構造と OrderServiceImpl の役割を説明して」と送ります。続けて「CLAUDE.md に書かれている受注ステータスの遷移ルールを教えて」と聞きます。
4. **答え合わせ** 返ってきた遷移を、ページ上部の「システムの歩き方」を別タブで開いて図と見比べます。最後に / を打ち、一覧から /help を選んで Enter を押します。

できたら

- [ ] 1枚目のターミナルに Started OrderApplication、2枚目に Claude Code の入力欄が出ている
- [ ] 遷移ルールを聞くと PENDING → CONFIRMED → SHIPPED → DELIVERED が返る
- [ ] / を打つとスラッシュコマンドの一覧が出て、/help の結果が画面に出る

考えること

> 同じ質問を CLAUDE.md が置かれていないプロジェクトで投げたら、答えのどこが変わるか。

AI の出方

説明の粒度は毎回変わります。controller / service / repository / model の4層がそろわないときは「controller から repository までの呼び出し順を、ファイル名つきで説明して」と聞き直してください。調べ始めて長くなったら Esc で止めて構いません。

D1-1+ src/main/java/com/example/order/model/Order.java の STATUS_PENDING に、日本語の Javadoc を付けさせます。出てきた説明が「受注直後の、まだ確定前の状態」という実際の意味とずれていないかを自分で判定してください。STATUS_CONFIRMED から STATUS_CANCELLED までの残り4定数をまとめて生成させると、途中から意味のずれた説明が混じりやすくなります。1定数ずつ頼んだ場合と見比べると、依頼の粒度で提案の正確さが変わるのが分かります。

くわしく（背景・詰まったときの対処）

この演習の前に、講師と一緒に「システムの歩き方」で受注ステータスの遷移図と層構造を見て、ブラウザに出る生の JSON を一度眺めます。Eclipse からターミナルに移った初日にいちばん引っかかるのは、指示が届いているのか分からない状態です。ここで起動と応答を先に確かめておくと、あとの演習では提案の中身を読むことに時間を使えます。

ターミナルは2枚使います。1枚目はアプリを動かしっぱなしにする窓、2枚目は claude と話す窓です。1枚で兼ねると、アプリを止めるつもりで押した Ctrl+C で Claude Code のほうが終わります。2枚目はターミナル右上の＋（新しいターミナル）で開きます。Ctrl+@ が効かないときは、表示メニューのターミナルからでも開けます。アプリの初回起動は依存関係の取得に時間がかかるので、D1-1 の頭で先に走らせておくと後が楽です。

claude を初めて起動すると、配色の選択とフォルダを信頼するかの確認が出ます。どちらも Enter で進めて構いません。入力欄が出れば起動できています。

handson フォルダの直下には CLAUDE.md が置いてあり、技術スタック、コーディング規約、受注の業務ルールが書かれています。claude を起動するとこの内容が読み込まれるので、前提を毎回打ち直す必要はありません。遷移ルールを聞いて正しく返ってくれば、読み込まれている証拠になります。

層の説明は毎回同じ形では返りません。4つの層の名前が一度でそろわなくても、聞き直せれば十分です。応答が返らない、そもそも claude というコマンドが見つからないというときは、ターミナルの現在地が handson になっているかを最初に確かめてください。

この演習ではファイルを作りません。コードにも手を入れないので、終わったらそのまま D1-2 に進めます。入力欄で @ を打つとファイル参照の候補が出ます。D1-2 以降で使うので、時間が余ったら一度眺めておいてください。/ の一覧にはこのプロジェクト用の /review も並んでいます。コードを直したあとに打つと、規約と業務ルールに照らして見てくれます。

D1-2 日付範囲検索の実装 [20min]

ねらい

受注日の範囲で受注を絞り込む検索を、意図を言葉で伝えて実装します。

さわるファイル

書く `src/main/java/com/example/order/service/OrderServiceImpl.java`
findByDateRange の本体を埋める

書く `src/main/java/com/example/order/controller/OrderController.java`
/date-range の窓口を足す

手順

1. **考える** どのリポジトリメソッドを呼び、何の型に詰め替え、何を返すか。この3つを先に決めてから頼みます。
2. **書く** 2枚目のターミナルの Claude Code に、@ で OrderServiceImpl.java を渡して findByDateRange の実装を依頼します。例外を投げている行を消すことも伝えます。
3. **書く** 続けて OrderController.java に、クエリパラメータ from と to を受け取る GET /date-range を追加させます。既存の findByStatus と書き方をそろえてと添えます。
4. **実行** Java を直したので、1枚目のターミナルで Ctrl+C を押し、.\mvnw.cmd spring-boot:run で起動し直します。ブラウザで http://localhost:8080/api/orders/date-range?from=2026-04-01&to=2026-04-03 を開きます。

できたら

- [ ] 開いた JSON を Ctrl+F で ORD- と検索するとヒットが5件で、ORD-20260401-001 と ORD-20260403-005 がどちらも含まれる（並び順は問わない）
- [ ] from と to をどちらも 2026-04-01 にすると、その日の2件が返る（両端を含む）
- [ ] from=2026-05-01&to=2026-05-31 は空配列 [] が返る（null ではない）

考えること

> from に新しい日付、to に古い日付を渡したとき、空で返すのと入力ミスとして弾くのと、どちらが仕様として正しいか。

AI の出方

stream で詰め替える案と for ループの案のどちらも出ます。リポジトリを呼んで OrderDTO に変換していれば、書き方の違いは正解の範囲です。変更内容では @Transactional(readOnly = true) が付いているかだけ、すぐ上の findByStatus と見比べてください。

D1-2+ OrderRepository の findByCustomerAndStatus は、顧客名とステータスの両方が一致する受注を返す検索です。これを呼ぶ薄いラッパーを OrderService と OrderServiceImpl の両方に足し、findByStatus と同じ形にそろえてください。似ているぶん引数の順番を取り違えた案が出やすいので、呼び先と引数だけは承認前に目で追ってください。findByDateRange には触らないので、後続の演習の前提を壊しません。

くわしく（背景・詰まったときの対処）

findByDateRange は、受注日が from から to の間に入る受注だけを取り出す検索です。配布時点では本体が例外を投げる一行だけになっていて、呼ぶと必ず止まります。

手がかりは2つあります。取り出す側はリポジトリにもう用意されていること、詰め替えのお手本が同じファイルの中にあることです。手順1で自分の答えを決めてから、OrderRepository と、すぐ上の findByStatus を開いて確かめてください。リポジトリのメソッドはメソッド名から SQL が組み立てられる仕組みなので、こちらで SQL を書く必要はありません。範囲の指定は両端を含むため、from と to に同じ日を入れてもその日の受注が取れます。

実装前に /api/orders/date-range を叩くと HTTP 400 が返ります。エンドポイントを足すと解消するので、400 が出ても手順どおり進めてください。

触るのは OrderServiceImpl の findByDateRange の本体と、OrderController に足す /date-range の2か所だけで、新しいファイルは作りません。Java の変更は動いているアプリには届かないので、確認の前に起動し直します。実装後に OrderController.java へ赤い波線が残ったら、java.time.LocalDate と org.springframework.format.annotation.DateTimeFormat の import を補ってください。OrderServiceImpl 側は LocalDate を最初から import しています。Claude Code に「足りない import を補って」と続けて頼んでも構いません。

動くところまで来たら、入力欄で /review と打ってみてください。このプロジェクト用のレビュー観点で、直したところを見てくれます。

D1-3 受注ダッシュボードの生成 [25min]

ねらい

受注一覧を表で見られる画面を1枚つくらせて、見ながら自分で育てます。

さわるファイル

新規 `src/main/resources/static/dashboard.html`
自分の画面を1枚つくる

手順

1 **考える** 画面に何が見えたら嬉しいかを2つ決めます。並べる列と、金額の見せ方あたりから選びます。

2 **書く** Shift+Tab を、入力欄の下に plan mode と出るまで押します（1回目は auto-accept edits なので、もう1回押します）。「この API を叩いて受注一覧を表で見られる画面を、src/main/resources/static/dashboard.html に単一の HTML で作ってください」とだけ頼みます。

3 **実行** 出てきた計画を読み、矢印キーで Yes を選んで Enter を押します。承認すると dashboard.html ができるので、アプリは1枚目のターミナルで動かしたまま、ブラウザで http://localhost:8080/dashboard.html を開きます。

4 **答え合わせ** 決めた2つと違うところを、1回に1つずつ短い日本語で直します。直すたびにブラウザを再読み込みして、変わった箇所を見ます。

できたら

- [ ] http://localhost:8080/dashboard.html に受注10件が表で並ぶ
- [ ] 自分で決めた2つのうち、1つ以上が画面に出ている
- [ ] 「金額を3桁区切りにしてください」のような1行の依頼を出し、再読み込みして見た目が変わるところまで確認した

考えること

> 一発で全部指示した場合と、一言から始めて1つずつ足した場合で、どちらが速く欲しい形に届いたか。

AI の出方

同じ依頼でも配色や列構成は毎回変わります。派手すぎたら「装飾を減らして、白背景に細い罫線だけの落ち着いた見た目にしてください」と言い直してください。一度に直しすぎて壊れたときは「さきほどの変更を取り消して」で戻せます。

D1-3+ ステータスごとの件数と、受注金額の合計を一覧の上に出してください。「ステータスごとの件数と、受注金額の合計を一覧の上にまとめて出してください」の一言で足せます。集計を JavaScript 側でやるか、API を足してサーバ側でやるかは自分で決めてください。どちらを選んだかと理由を一言で言えるようにしておくと、Day3 のレビュー演習につながります。

くわしく（背景・詰まったときの対処）

src/main/resources/static は空です。フォルダごと見当たらないときは、Claude Code が作ってくれます。Spring Boot はこのフォルダに置いたファイルを http://localhost:8080/ファイル名 で配るので、dashboard.html を1枚置くだけで画面ができます。Java には触らない演習です。Planモードは計画を出すところで一度止まるので、Yes を選んで承認するまで dashboard.html は作られません。

アプリは起動したままで構いません。この配布物は動いているアプリが src/main/resources を直接読むので、dashboard.html を新しく置いてもブラウザを再読み込みするだけで出ます。中身を直したときも同じで、再読み込みだけで変わります。起動し直しが要るのは Java を直したときだけです。頼む、見る、直すを止めずに回してください。

この演習の軸になる API は、GET /api/orders と GET /api/orders/{id}、D1-2 で足した GET /api/orders/date-range の3本です。ほかに顧客名検索の /api/orders/search?customerName= とステータス検索の /api/orders/status/{status} も動いているので、余力があれば使ってください。D1-5 でステータス変更ボタンを足すときは、PATCH /api/orders/{id}/status を使います。

何が見たいかを決めずに「いい感じの画面を作って」と頼むと、返ってきたものを評価できません。評価できないものは直せないので、先に見たい形を決めておきます。この画面は D1-4 以降でも手を入れ続ける前提です。明細を返せるようにしたら明細欄、ステータス遷移を実装したら変更ボタンを、自分で足していってください。

完成度は問いません。表が出れば達成です。手が止まる人はプロンプトが長すぎることが多いので、短く言い直してください。ファイル名を変えると URL も変わるため、dashboard.html の名前のまま以降の演習でも使います。

休憩

[10min]　ここで一度手を止めます。詰まっている方はこの間に声をかけてください。

D1-4 受注明細のレスポンス追加 [25min]

ねらい

受注のレスポンスに明細を載せ、金額の内訳を追えるようにします。

さわるファイル

新規 `src/main/java/com/example/order/dto/OrderItemDTO.java`
明細を返すための入れ物

書く `src/main/java/com/example/order/dto/OrderDTO.java`
items を足して変換する

手順

1 **考える** OrderItem が持つ7つのフィールドのうち、API に返すものと返さないものを決めます。親への参照 order を含めるとどうなるかも考えます。

2 **書く** Shift+Tab を plan mode と出るまで押し（1回目は auto-accept edits です）、@OrderItem.java と @OrderDTO.java を渡します。OrderItemDTO の新規作成と、OrderDTO への items 追加を依頼します。

3 **書く** 出てきた計画に親への参照 order が入っていたら、矢印キーで No, keep planning を選び「order は含めないでください」と伝えます。直った計画を Yes で承認します。

4 **答え合わせ** Java を直したので、1枚目のターミナルで Ctrl+C を押し、.\mvnw.cmd spring-boot:run で起動し直します。ブラウザで http://localhost:8080/api/orders/1 と /api/orders/2 を開き、items 配列と小計を読みます。

できたら

- [ ] /api/orders/1 のレスポンスに items が現れ、RACK-42U-BK の明細1件が入る
- [ ] 明細の小計 540,000 に 1.10 を掛けた 594,000 が、ヘッダーの totalAmount と一致する
- [ ] /api/orders/2 では L3スイッチと L2スイッチの明細が2件並ぶ

考えること

> 一覧の /api/orders も同じ変換を通る。10件返すとき、明細を取りにいく SELECT は何回走るか。

AI の出方

生成された OrderItemDTO に親への参照 order が紛れることがあります。明細が無い受注で items が null になる案も出るので、空リストになっているかを変更内容で確かめてください。赤い波線が残ったら、そのまま「足りない import を補って」と続けて頼めます。

D1-4+ src/main/java/com/example/order/model/Customer.java を手本に、配送先を表す ShippingInfo エンティティを同じフォルダに新規作成してください。@Table のテーブル名、shippingCode の unique = true、@Column のカラム名がスネークケースになっているかを、承認する前に点検します。余力があれば schema.sql に CREATE TABLE shipping_info も追記してください。Java を足したので、確かめる前に1枚目のターミナルで Ctrl+C を押し、.\mvnw.cmd spring-boot:run で起動し直します。受注 API の本筋には触らないので、後続の演習に影響しません。

くわしく（背景・詰まったときの対処）

GET /api/orders/1 を叩くと、受注番号も顧客名も合計金額も返るのに、明細の配列だけが返りません。データベースの order_items には13件の明細が入っていて、エンティティの Order も items を持っています。足りないのは外に出る側です。OrderDTO に明細のフィールドが無く、変換メソッドの fromEntity が明細に触れていません。

DTO は、返したい項目だけを詰め替えて外に出すための入れ物です。エンティティの OrderItem をそのまま返すと、データベースの持ち方がそのまま API の形になります。いまは親参照の order に @JsonIgnore が付いているので JSON の循環こそ止まりますが、LAZY の親参照や内部項目に外向きの形が引きずられます。だから明細専用の OrderItemDTO を新しく起こします。

D1-2 は既存メソッドの穴を埋める作業でした。ここはクラスを一枚まるごと新設して、関連ファイルも一緒に直す場面です。Planモードで計画を先に見てから承認してください。承認せずに直したい点を伝える操作も、ここで一度やっておきます。

作るのは src/main/java/com/example/order/dto/OrderItemDTO.java の1本、直すのは同じフォルダの OrderDTO.java です。どちらも Java なので、確認の前にアプリを起動し直します。返すフィールドは productCode、productName、quantity、unitPrice、subtotal の5つが目安になります。id=2 は L3スイッチと L2スイッチの明細を2件持つので、複数明細も合わせて確かめておくと安心です。

時間が余ったら、@src/main/resources/static/dashboard.html を対象に「行を開いたら明細を表示してください」と1行で頼み、自分の画面にも明細を出してください。こちらは HTML だけの変更なので、ブラウザの再読み込みだけで反映されます。

D1-5 ステータス遷移ルールの実装 [25min]

ねらい

CONFIRMED からキャンセルできるよう遷移ルールを直し、自分の画面から実際に通します。

さわるファイル

書く	`src/main/java/com/example/order/service/OrderServiceImpl.java`	CONFIRMED の遷移を直す
書く	`src/main/resources/static/dashboard.html`	ステータス変更ボタンを足す

手順

1	**考える**	case Order.STATUS_CONFIRMED に CANCELLED を足すとき、いまある SHIPPED への遷移をどう残すかを先に決めます。
2	**書く**	claude で validateStatusTransition の case Order.STATUS_CONFIRMED を対象に指定し、CANCELLED への遷移も許可するよう依頼します。SHIPPED は今までどおり残すと添えます。
3	**書く**	@src/main/resources/static/dashboard.html を対象に「行から PATCH /api/orders/{id}/status を叩いてステータスを変更できるボタンを足してください」と頼みます。叩き先が /cancel ではなく /status であることを、必ず言葉で指定します。
4	**答え合わせ**	Java を直したので、1枚目のターミナルで Ctrl+C を押し、.\mvnw.cmd spring-boot:run で起動し直します。画面のボタンから id=2 を CANCELLED にし、続けて id=3 と id=4 でも試します。

できたら

- [ ] 画面のボタンから id=2 を CANCELLED にでき、再読み込みすると一覧の id=2 が CANCELLED で表示される
- [ ] CONFIRMED の id=3 を SHIPPED に変える操作は、今までどおり通る
- [ ] SHIPPED の id=4 を CANCELLED にしようとすると通らない（共通エラーハンドラが無いので 500 で返る）

考えること

> InvalidOrderStateException は業務エラーなのに 500 で返る。本来はどの HTTP ステータスが正しいか。

AI の出方

文字列の "CANCELLED" を直に埋め込む案が出ます。Order.STATUS_CANCELLED の定数に直っているかを変更内容で確かめてください。case ブロックごと書き換えて SHIPPED への遷移を消してしまう案も出るので、そこも見ます。画面側では PATCH /api/orders/{id}/cancel を叩くボタンが出ることがあります。cancelOrder は遷移ルールを通らず、直す前でも 200 を返してしまうので、fetch のパスが /status になっているかを確かめてください。

D1-5+ @RestControllerAdvice の共通エラーハンドラを足し、OrderNotFoundException を 404、InvalidOrderStateException を 400 に対応づけてください。実装後に1枚目のターミナルで Ctrl+C を押し、.\mvnw.cmd spring-boot:run で起動し直してから SHIPPED の受注を叩くと、500 だったものが 400 に変わります。余力があれば updateOrder と deleteOrder に、PENDING 以外を弾くステータスチェックも足してみてください。

くわしく（背景・詰まったときの対処）

OrderServiceImpl の validateStatusTransition は、受注ステータスをある値から別の値へ変えてよいかを判定するメソッドです。配布時点では case Order.STATUS_CONFIRMED が SHIPPED への遷移しか許していないため、確定済みの受注をキャンセルできません。業務ルールでは PENDING と CONFIRMED からのキャンセルを認めます。出荷済み以降は認めません。モノが動いたのに帳簿だけ消える状態になるためです。

確認は PATCH /api/orders/{id}/status で行います。内部で validateStatusTransition が呼ばれるからです。PATCH /api/orders/{id}/cancel は cancelOrder という別のメソッドを呼ぶので、遷移ルールの確認には使いません。手順3で作るボタンも、押したときに叩くのは /status のほうです。

ステータス値は文字列の直書きではなく Order の定数を使います。綴り間違いをコンパイルの時点で見つけられるからです。

画面のボタンがうまく動かないときは、ターミナルから直接叩いても確かめられます。PowerShell なら Invoke-RestMethod -Method Patch -Uri http://localhost:8080/api/orders/2/status -ContentType "application/json" -Body (@{status="CANCELLED"} | ConvertTo-Json) の1行です。返ってきた status が CANCELLED になっていれば通っています。

配布データでは id=2 と id=3 が CONFIRMED、id=4 が SHIPPED で入っています。id=2 を一度 CANCELLED にすると遷移元が変わるので、やり直すときはアプリを起動し直してください。H2 はメモリ上で動くので、起動のたびに配布時の10件へ戻ります。

新しいファイルは作りません。編集は OrderServiceImpl.java の中と、自分のダッシュボードだけです。手順4で起動し直すのは OrderServiceImpl を直したためで、ボタンを足した HTML のほうは再読み込みだけで反映されます。ボタンの見た目や文言をあとから直すときは、起動したまま再読み込みで確かめてください。

到達チェック

この日の終わりに、自分で確かめてください。全部埋まっていなくても構いません。

- [ ] ターミナルを2枚使い分け、1枚目でアプリを動かしたまま2枚目で claude を動かした
- [ ] / からスラッシュコマンドを1つ実行し、@ でファイル参照の候補も確かめた
- [ ] /api/orders/date-range が両端を含む5件を返し、範囲外は空配列で返ることを確かめた
- [ ] http://localhost:8080/dashboard.html に自分で作った画面を出し、受注が表で並ぶところまで確認した
- [ ] ダッシュボードを直してブラウザを再読み込みし、起動し直さずに変わることを確かめた
- [ ] /api/orders/1 の items に明細が並び、小計の1.10倍が totalAmount と一致することを確かめた
- [ ] 画面のボタンから id=2 を CANCELLED にでき、SHIPPED の id=4 では通らないことを確かめた
- [ ] Planモードで計画を出させ、No, keep planning での差し戻しと承認を1回ずつ行った

つまずいたとき

よく出る詰まりどころです。当てはまるものがなければ、その場で声をかけてください。

生成されたコードが正しいか判断できません。

判断の材料を先に持ってから頼むと変わります。D1-2 なら、呼ぶリポジトリメソッド・詰め替える型・返すものの3つを自分で決めてから依頼し、変更内容がその3つと合っているかだけを見てください。全部を読もうとすると手が止まります。この日の合否は、叩いた結果で決めるものだと思ってください。コードを読み切れなくても、ブラウザに出た結果が合っていれば正解の範囲です。

起動し直したらエラーが出て立ち上がりません。

ターミナルに出た [ERROR] で始まる行を、上から3行か4行そのままコピーしてください。2枚目のターミナルの Claude Code に貼り、「これが出ています。直してください」と伝えます。多くは import の不足です。起動に失敗した時点でプロセスは終わり、プロンプトが戻っています。直ったら1枚目のターミナルでそのまま .\mvnw.cmd spring-boot:run を打ち直してください。

どこを直したらアプリを起動し直すのですか。

Java のファイルを直したときだけです。動いているアプリは起動時のクラスファイルを掴んだままなので、1枚目のターミナルで Ctrl+C を押し、.\mvnw.cmd spring-boot:run で入れ直します。HTML と CSS と JavaScript は src/main/resources を直接読んでいるので、ブラウザの再読み込みだけで反映されます。コンパイルだけ先に通したいときは、ターミナル右上の＋でもう1枚開き、そこで .\mvnw.cmd compile を打ってください。2枚目は claude と話す窓なので、そこに打つとプロンプトとして送られます。

dashboard.html を開くと 404 か真っ白になります。

404 のときは保存先とファイル名を確かめてください。src/main/resources/static/dashboard.html に保存されていれば、アプリを動かしたままブラウザを再読み込みするだけで出ます。真っ白のときは F12 を押して Console タブを開き、赤い文字が出ていたらそのまま Claude Code に貼って「これが出ています」と伝えるのが最短です。fetch のパス違いか、レスポンスの構造の取り違えがほとんどです。

ダッシュボードを直したのに画面が変わりません。

まずブラウザを再読み込みしてください。それでも変わらないときは Ctrl+Shift+R で読み直します。保存できていない、あるいは別のファイルを直しているのが次に多い原因なので、パスが src/main/resources/static/dashboard.html になっているかを確かめてください。

/api/orders/date-range を叩いたら HTTP 400 が返ります。

D1-2 でエンドポイントを足す前は 400 になります。date-range という文字列が /api/orders/{id} の窓口に吸い込まれ、数値に変換できないためです。OrderController に /date-range を足して起動し直すと解消します。

id=2 のステータスを変えたあと、やり直したいです。

1枚目のターミナルで Ctrl+C を押し、.\mvnw.cmd spring-boot:run で起動し直してください。データベースはメモリ上にあるので、起動のたびに配布時の10件へ戻ります。壊しても戻るので、気にせず試してください。