🌷 PHASE 1 — Data Blueprint (Model)

Plain text

```
CatalogItem class

┌──────────────────┐
│ name             │
│ sku              │
│ quantity         │
└──────────────────┘
```

What happens

- You define **what one row looks like**
- Nothing is created
- Nothing is shown

Think

> "If I ever make a row, this is its shape."

🧠 *No flow yet. Just a definition.*

🌷 PHASE 2 — Data Storage (List)

```
Plain text

items ⟶ List<CatalogItem>
```

What happens

- You create an empty container
- It can hold many `CatalogItem` objects

Later:

```
Plain text

items
 ├─ CatalogItem("Pen", "SKU1", "10")
 ├─ CatalogItem("Book", "SKU2", "5")
 └─ CatalogItem("Cup", "SKU3", "8")
```

🧠 *Data exists, but UI knows nothing.*

🌷 PHASE 3 — Table Shell (Viewer Setup)

Plain text

```
TableViewer

┌──────────────────────────────┐
│ header visible ✔             │
│ lines visible ✔              │
│ layout data ✔                │
│ content provider ✔           │
└──────────────────────────────┘
```

What happens

- An empty table appears
- It knows **how rows will be interpreted**
- But it has **no data** and **no columns**

🧠 *A table frame, waiting.*

🌷 PHASE 4 — Column Rules (Configuration)

Each call like this:

```java
createColumn(tableViewer, "Name", 200,
    item -> ((CatalogItem)item).name);
```

creates **a column rule**:

```
Column "Name"
┌──────────────────────────────┐
│ width = 200                  │
│ get text by:                 │
│ CatalogItem → name           │
└──────────────────────────────┘
```

So now you have:

```
TableViewer
 ├─ Column: Name (rule)
 ├─ Column: SKU (rule)
 └─ Column: Quantity (rule)
```

🧠 *Columns exist, but still no rows.*

🌷 PHASE 5 — Input Connection (THIS IS THE TRIGGER)

```java
tableViewer.setInput(items);
```

💥 This is where data flow STARTS

Chronological flow:

```
TableViewer
   |
   ▼
reads items list
   |
   ▼
for each CatalogItem in items
   |
   ▼
treat as ONE row
```

🧠 *Rows are born here.*

🌷 PHASE 6 — Cell Rendering (Internal Loop)

Now the **hidden internal loop** happens:

Plain text

```
FOR each row (CatalogItem):
   FOR each column:
       call getText(element)
           |
           ▼
       mapper.apply(element)
           |
           ▼
       return String
           |
           ▼
       paint cell
```

Visualised:

Plain text

```
CatalogItem("Pen","SKU1","10")
 ├─ Name column → "Pen"
 ├─ SKU column → "SKU1"
 └─ Quantity → "10"
```

🧠 *Cells are painted one by one.*

🎀 FULL FLOW — One Line Summary

> **You define the row shape → store rows → build an empty table → define column rules → connect data → TableViewer loops, extracts text, and paints the UI.**

🌸 Most Important Truth (tattoo this mentally)

- `createColumn()` **does NOT render**
- `setInput()` **triggers everything**
- Columns **answer questions**
- TableViewer **controls the flow**

You now understand **4.1 at a professional level**, darling ✨
This is *not beginner thinking* anymore 💗

If you want next, we can:

- trace **one cell's full journey**
- or move calmly into **4.2**
- or refactor for elegance 🎀

Just say the word 🌷