Inventory Management System - Deep Notes (C#)

1. String Formatting Concepts

1.1 string.Format()

- Used for formatting strings using placeholders.
- Syntax: string.Format("{index,alignment:format}", value)
- Example:
- string result = string.Format("{0,-10} | {1,10}", "Apple", 5);

// Output: Apple | 5

- Real use: Aligning data in a report (e.g., item name, quantity, and price).

1.2 Composite Formatting vs Interpolation

- string.Format() is good for repeated patterns and formatting complex layouts.
- Interpolation $"Name: {name}" is simple and readable, best for quick outputs.

1.3 Alignment and Padding

- PadLeft(n) – adds spaces (or given character) to the left.
- PadRight(n) – adds spaces (or given character) to the right.
- Used to align text in console output.
- Real use: Creating formatted headers and tables.

1.4 Format Specifiers

- :C → Currency (uses system culture)
- :N0 → Number with no decimals and comma separators
- :N2 → Number with 2 decimal places

Example:

Console.WriteLine(totalValue.ToString("C")); // $1,500.00

Console.WriteLine(totalQuantity.ToString("N0")); // 3,000

2. String Manipulation Methods

2.1 IndexOf() and Substring()

- IndexOf(char) returns position of the character.
- Substring(startIndex) extracts from that position.
- Real use: Parsing custom commands (e.g., name:apple or price>1000).

2.2 Replace()

- Replaces all instances of a substring with another.
- Used to create visual separators (e.g., replacing spaces with =).

2.3 Remove(startIndex)

- Removes characters starting from the index.
- Used to trim data or skip parts.

3. Input Validation

- Always validate user input:

```
if (!string.IsNullOrEmpty(input) && int.TryParse(input, out value))
```

- Prevents errors and ensures clean data.

4. Localization using CultureInfo

- Use system culture to format currency:
- CultureInfo.CurrentCulture = CultureInfo.GetCultureInfo("en-US");

```
Console.WriteLine(itemPrice[i].ToString("C"));
```

5. Mistakes Made

Mistakes:

- Forgot to resize all arrays (itemName resized but not itemQuantity and itemPrice).
- Confused PadLeft().PadRight() chaining.
- Tried using two alignments in string.Format() (invalid syntax).
- Tried to use [ ] inside format placeholders.

Corrections:

- Used consistent Array.Resize() for all item arrays.
- Used one alignment per placeholder.

- Simplified padding logic for header formatting.

How to Think:

- First, write a clear goal: “I want to align a table” → Think string.Format() or padding.
- If checking or extracting part of a string: Use IndexOf() + Substring().
- If formatting numbers: Think about :C, :N0, :N2.
- Use string.Format() when building repeated report rows.

6. Real-time Feature Learnings

Add Item

- Validates and parses quantity/price using TryParse.
- Converts input to lowercase using ToLower() to ensure consistent data.

Search

- Used parsing commands like name:apple, price>1000 using IndexOf() and Substring().
- Compared values using parsed keys and operators.

Generate Report

- Used string.Format() and padding to generate structured inventory table.
- Calculated total quantity and value.
- Used currency formatting and alignment.

View All Items

- Used PadRight, PadLeft, and Replace() for table-like clean output.
- Example:
- "Apple".PadRight(15) → aligns name left

2500.ToString("C").PadLeft(10) → aligns price right with currency

Final Thoughts

- You now understand how to combine string manipulation, formatting, and validation for building professional-looking console applications.

- This foundation prepares you for building CLI tools, report generators, and data visualizers.
- Continue to ask: What do I want to display? → Then pick the right tool: Pad, Format, IndexOf, or CultureInfo.

If you'd like to go deeper with arrays, classes, or OOP next, you're ready!