C# Data Types, Value vs Reference Types, and Conversion Deep Dive

Deep Dive into C# Data Types and Conversions

==========
Value Types vs Reference Types
==========
- Value types store actual data in stack (fast, short-lived)
  Examples: int, float, double, bool, char, struct

- Reference types store reference (memory address) in stack and actual object in heap
  Examples: string, arrays, class, object

Example:
```
int a = 5;
int b = a;
b = 10;
// a remains 5

string[] names1 = new string[] {"Abhi"};
string[] names2 = names1;
names2[0] = "GPT";
// names1[0] is also "GPT" because both point to same memory
```

==========
Choosing the Right Data Type
==========
- Use int for whole numbers, decimal for money, bool for true/false, string for text.
- Avoid premature optimization, choose data type based on value range and usage.
- Consider library and DB compatibility (e.g., DateTime for dates).

Examples:
- Use decimal for currency like price = 12.99m;
- Use float/double for geometry or scientific calculations
- Prefer string for names, address, etc.

==========
Data Type Conversions
==========
Implicit Conversion (Safe)
```
int a = 10;
decimal b = a; // widening
```

Explicit Conversion (May lose data)
```
decimal a = 5.9m;
int b = (int)a; // b is 5
```

Convert class (Rounds)
```
decimal a = 5.9m;
int b = Convert.ToInt32(a); // b is 6
```

TryParse (Safe for user input)

C# Data Types, Value vs Reference Types, and Conversion Deep Dive

```
string input = "123";
if(int.TryParse(input, out int result)) {
   Console.WriteLine(result);
}

Parse (Throws exception if invalid)
int val = int.Parse("123");

======================
Real-World Example: Task Cost Estimation
======================

Useful when calculating project budgets or time-based costs.
Here's a practical feature implemented in the Task Manager:

void estimateTaskCost() {
    Console.WriteLine("Enter hours:");
    string hoursInput = Console.ReadLine().Trim();

    Console.WriteLine("Enter hourly cost:");
    string costInput = Console.ReadLine().Trim();

    decimal hours, cost;
    bool isHourValid = decimal.TryParse(hoursInput, out hours);
    bool isCostValid = decimal.TryParse(costInput, out cost);

    if (!isHourValid || !isCostValid) {
        Console.WriteLine("Invalid input.");
        return;
    }

    decimal total = hours * cost;
    int truncated = (int)total;
    int rounded = Convert.ToInt32(total);

    Console.WriteLine($"Exact: {total}");
    Console.WriteLine($"Truncated: {truncated}");
    Console.WriteLine($"Rounded: {rounded}");
}

Concepts Learned:
- TryParse for safe input handling
- decimal for financial calculations
- (int) cast truncates, Convert rounds
```