Table of Contents

---

1. Comments

Types of Comments

```
// Single-line comment

/*
 * Multi-line comment
 * Can span multiple lines
 */

#region RegionName
// Code that can be collapsed
#endregion
```

Advantages

- Improve code readability
- Document complex logic
- Regions help organize large code sections

Disadvantages

- Over-commenting makes code cluttered
- Outdated comments mislead developers

When to Use

- Explain WHY, not WHAT

- Document complex algorithms
- Use regions for grouping related code in large files

---

2. Variables & Data Types

Integer Types

```
int x = 123;           // 4 bytes (-2.1B to 2.1B)
short y = 56;          // 2 bytes (-32K to 32K)
long l = 123456789;    // 8 bytes (very large range)
```

Memory Diagram:

```
Stack Memory:
```

Variable	Value
x (int)	123
y (short)	56
l (long)	123...

Floating-Point Types

```
float x = 1.2F;        // 4 bytes, 7 digits precision
double y = 1.234;      // 8 bytes, 15-16 digits precision
decimal z = 1.2345M;   // 16 bytes, 28-29 digits precision
```

When to Use Each

- **int**: Most common, general counting
- **long**: Large numbers (file sizes, timestamps)
- **float**: Graphics, game development (less precision needed)
- **double**: Scientific calculations
- **decimal**: Financial calculations (no rounding errors)

Disadvantages

- **float/double**: Precision errors in financial calculations

- **decimal**: Slower performance, more memory

Character & Boolean

```
char c = 'a';          // Single character (2 bytes, Unicode)
int x = c;             // Implicit conversion: x = 97 (ASCII)
bool b = true;         // true or false (1 byte)
```

Nullable Types

```
// Regular type - cannot be null
int x = 123;
// x = null;   ❌ Compile error

// Nullable types
Nullable<int> y = 123;
y = null;   ✅

// Shorthand syntax
int? z = 123;
z = null;   ✅
```

Memory Diagram:

Variable	HasValue	Value
z (int?) after null	true false	123 0

Advantages of Nullable

- Represent "no value" in databases
- Optional parameters
- Distinguish between 0 and "unknown"

Disadvantages

- Extra memory overhead
- Must check for null before use

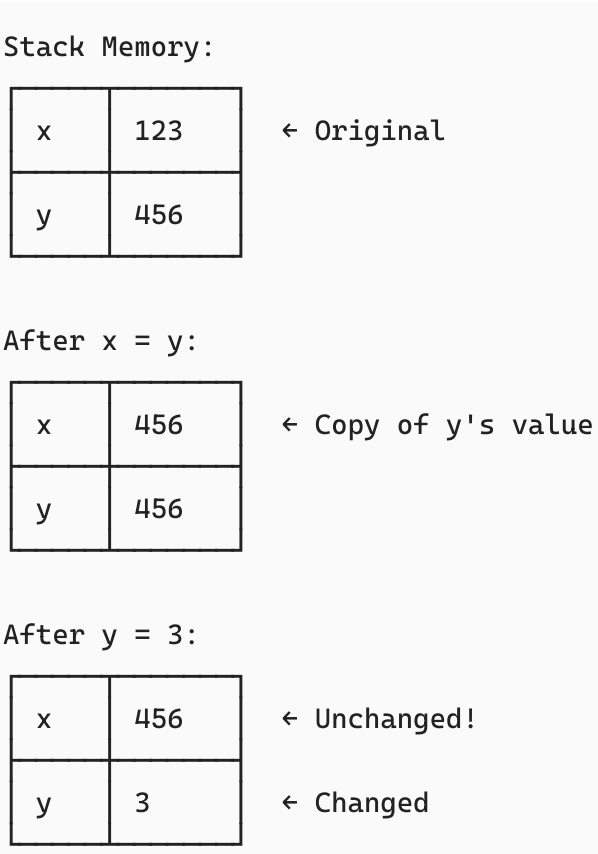
---

3. Value Types vs Reference Types

Value Types (Stack)

```
int x = 123;
int y = 456;
x = y;      // Copies the VALUE
y = 3;
// x = 456, y = 3 (independent)
```

Memory Diagram:

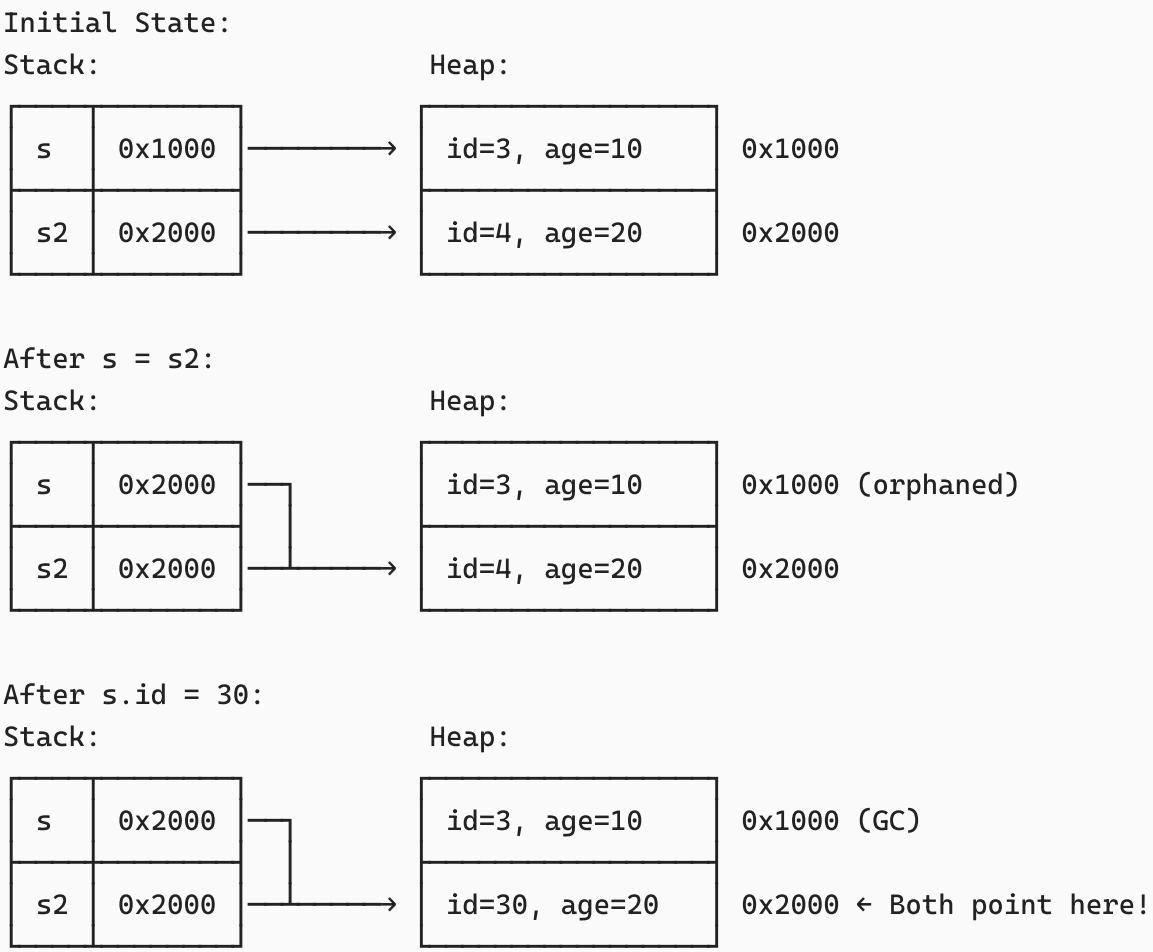


Reference Types (Heap)

```
student s = new student();
s.id = 3;
s.age = 10;

student s2 = new student();
s2.id = 4;
s2.age = 20;

s = s2;         // Copies the REFERENCE
```

```
s.id = 30;
// Both s and s2 point to same object!
// s.id = 30, s2.id = 30
```

Memory Diagram:



Value Types Advantages

- Fast allocation (stack)
- Automatic cleanup
- Thread-safe by default

Value Types Disadvantages

- Copying large structs is expensive
- Limited stack space

Reference Types Advantages

- Efficient for large objects

- Can share data between methods
- Polymorphism support

Reference Types Disadvantages

- Garbage collection overhead
- Risk of null references
- Memory leaks if not managed

When to Use

- **Value Types**: Small data (<16 bytes), immutable data, high-performance scenarios
- **Reference Types**: Large objects, need sharing/modification, object-oriented design

---

4. String Formatting

String Concatenation

```
int id = 1;
string name = "ahmed";
int age = 14;

// Method 1: Concatenation (❌ Slow)
Console.WriteLine("id=" + id + ",name=" + name + ",age=" + age);
```

String Placeholders

```
// Method 2: Composite Formatting
Console.WriteLine("id={0}, name={1}, age={2}", id, name, age);
```

String Interpolation (Recommended)

```
// Method 3: String Interpolation (✅ Best)
Console.WriteLine($"id={id}, name={name}, age={age}");
```

Escape Sequences

```
// \t = tab, \n = newline
Console.WriteLine($"id={id}\t name={name}\t age={age}");
```

```
// Verbatim strings (@) - ignore escapes
Console.WriteLine(@"C:\Users\ITI\Desktop\C#46\Day2");
```

Console Input/Output

```
// Output methods
Console.Write("text");      // No newline
Console.WriteLine("text");  // With newline

// Input methods
string txt = Console.ReadLine();           // Read entire line
int x = Console.Read();                    // Read single char as ASCII
ConsoleKeyInfo k = Console.ReadKey();      // Read key press
```

Advantages

Method	Pros	Cons
Concatenation	Simple	Slow, creates many objects
Composite	Positional reuse	Less readable
Interpolation	Readable, fast	Requires C# 6+

When to Use

- **String Interpolation**: Default choice (readable + fast)
- **Composite Formatting**: When reusing same string with different values
- **Concatenation**: Avoid for multiple strings

---

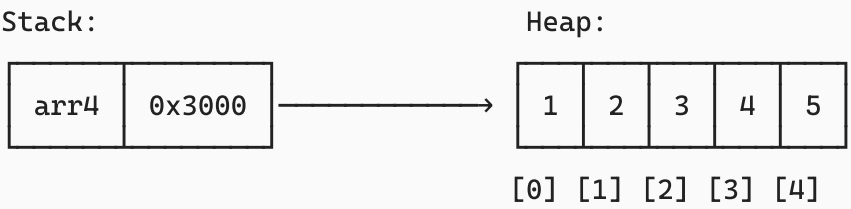
5. Arrays

Array Declaration

```
// Method 1: Fixed size
int[] arr = new int[5];

// Method 2: With initialization
int[] arr1 = new int[] { 1, 2, 3, 4 };

// Method 3: Implicit type
```

```
int[] arr3 = { 1, 2, 3, 4, 5 };

// Method 4: Collection expression (C# 12+)
int[] arr4 = [1, 2, 3, 4, 5];
```

Memory Diagram:



Array Operations

```
// Access/Modify
arr4[3] = 45;
Console.WriteLine(arr4[3]);  // Output: 45

// Common methods
Array.Sort(arr);       // Sort ascending
Array.Reverse(arr);    // Reverse order
int len = arr.Length; // Get length
```

Practical Example: Student Ages

```
Console.WriteLine("Enter student number:");
int studentNumber = int.Parse(Console.ReadLine());

int[] studentAges = new int[studentNumber];

// Input
for(int i = 0; i < studentNumber; i++)
{
    Console.WriteLine($"Enter age of student {i+1}:");
    studentAges[i] = int.Parse(Console.ReadLine());
}

// Process
int sum = 0;
for(int i = 0; i < studentAges.Length; i++)
{
    sum += studentAges[i];
    Console.WriteLine($"Age of student {i+1} = {studentAges[i]}");
```

```
}

Console.WriteLine($"Average = {sum/studentNumber}");
```

Advantages

- Fast random access O(1)
- Memory efficient
- Type-safe

Disadvantages

- Fixed size (cannot grow)
- Insertion/deletion is expensive
- Must know size at creation

When to Use

- Known size at compile time
- Need fast index access
- Storing homogeneous data
- **Avoid**: When size changes frequently (use List instead)

By Abdullah Ali

Contact : +201012613453