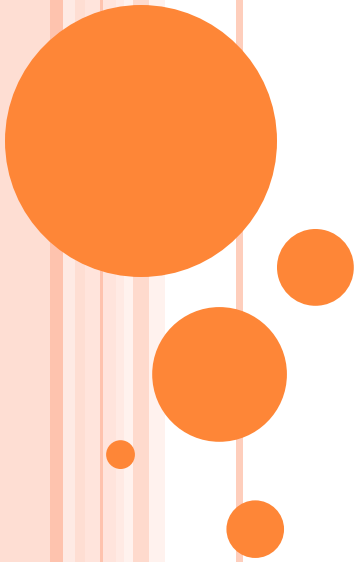


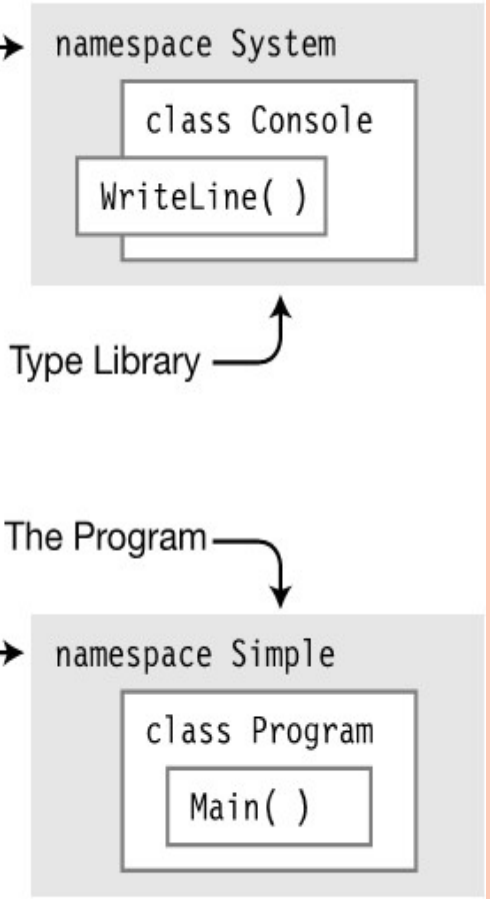
INTRODUCTION

Asma Kanwal
Lecturer
Department of Computer Science
GC University, Lahore

C# Program Structure

SimpleProgram.cs

```
1   using System;
2
3   namespace Simple
4   {
5      class Program      // declare a class
6      {
7         static void Main()
8         {
9            Console.WriteLine("Hi there!");
10        }
11     }
12  }
```



Produces the following output:

```
Hi there!
```

Starting Point of Program

- The starting point of execution of every C# program is at the first instruction in Main.
- The name Main must be capitalized.

IDENTIFIERS

- The alphabetic and underscore characters (a through z, A through Z, and _) are allowed at any position.
- Digits are not allowed in the first position but are allowed everywhere else.
- The @ character is allowed in the first position of an identifier but not at any other position. Although allowed, its use is generally discouraged.

C# KEYWORDS

abstract	const	extern	int	out	short	typeof
as	continue	false	interface	override	sizeof	uint
base	decimal	finally	internal	params	stackalloc	ulong
bool	default	fixed	is	private	static	unchecked
break	delegate	float	lock	protected	string	unsafe
byte	do	for	long	public	struct	ushort
case	double	foreach	namespace	readonly	switch	using
catch	else	goto	new	ref	this	virtual
char	enum	if	null	return	throw	void
checked	event	implicit	object	sbyte	true	volatile
class	explicit	in	operator	sealed	try	while

TEXT OUTPUT

Write()

```
Console.Write("This is trivial text.");
```

Output string

Example:

```
System.Console.Write ("This is text1. ");
System.Console.Write ("This is text2. ");
System.Console.Write ("This is text3. ");
```

This is text1.	This is text2.	This is text3.
First statement	Second statement	Third statement

TEXT OUTPUT

WriteLine()

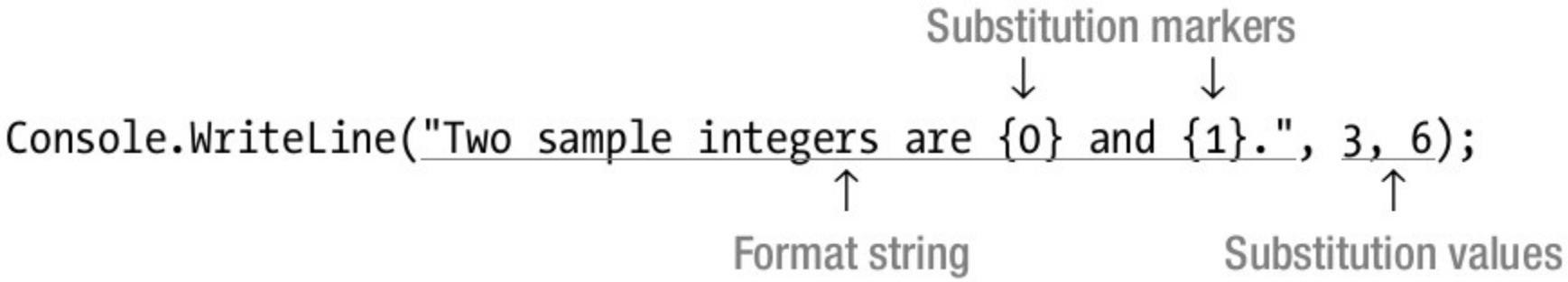
```
System.Console.WriteLine("This is text1.");
System.Console.WriteLine("This is text2.");
System.Console.WriteLine("This is text3.");
```

```
This is text1.
This is text2.
This is text3.
```

FORMAT STRING

- If there is more than a single parameter, the parameters are separated by commas.
- The first parameter must always be a string and is called the format string . The format string can contain substitution markers.
  1. A substitution marker marks the position in the format string where a value should be substituted in the output string.
  2. A substitution marker consists of an integer enclosed in a set of matching curly braces. The integer is the numeric position of the substitution value to be used. The parameters following the format string are called substitution values . These substitution values are numbered, starting at 0.

Format String



COMMENTS

- Delimited comments have a two-character start marker (/*) and a two-character end marker (*/).
- Text between the matching markers is ignored by the compiler.
- Delimited comments can span any number of lines.

Documentation Comments

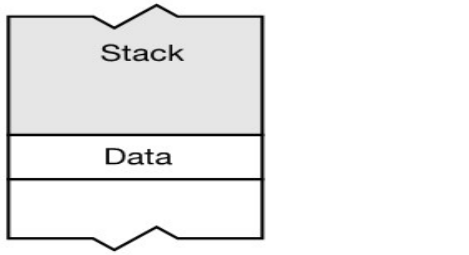
```
/// <summary>
/// This class does...
/// </summary>
class Program
{
    ...
```

COMMENTS

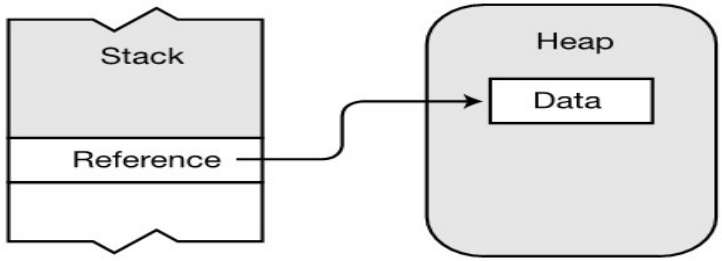
Type	Start	End	Description
Single-line	//		The text from the beginning marker to the end of the current line is ignored by the compiler.
Delimited	/* */		The text between the start and end markers is ignored by the compiler.
Documentation	///		Comments of this type contain XML text that is meant to be used by a tool to produce program documentation. This is described in more detail in Chapter 25.

VALUE TYPES AND REFERENCE TYPES

- Value types require only a single segment of memory, which stores the actual data.
- Reference types require two segments of memory:
  - The first contains the actual data and is always located in the heap.
  - The second is a reference that points to where the data is stored in the heap.



Value Type Data
– The data is stored on the stack.



Reference Type Data
– The data is stored in the heap.
– The reference is stored on the stack.

Value Types and Reference Types

- A value type stores its contents in memory allocated on the stack.

```
int x = 42;
```

- In contrast, a reference type, such as an instance of a class or an array, is allocated in a different area of memory called the heap.

```
int[] numbers = new int[10];
```