**Constructors:**

* A *constructor* initializes an object when it is created.
* It has the same name as its class and is syntactically similar to a method.
* However, constructors have no explicit return type.
* The general form of a constructor is shown here:

*access class*-*name*(*param-list*)

{

// constructor code

}

* Typically, you will use a constructor to give initial values to the instance variables defined by the class or to perform any other startup procedures required to create a fully formed object.
* Also, usually, *access* is **public** because constructors are normally called from outside their class.
* The *param-list* can be empty, or it can specify one or more parameters.
* All classes have constructors, whether you define one or not, because C# automatically provides a default constructor that causes all member variables to be initialized to their default values.
* For most value types, the default value is zero. For **bool**, the default is **false**. For reference types, the default is null.
* However, once you define your own constructor, the default constructor is no longer used.
* Here is a simple example that uses a constructor:

// A simple constructor.

using System;

class MyClass

{

public int x;

public MyClass()

{

x = 10;

}

}

class ConsDemo

{

public static void Main()

{

MyClass t1 = new MyClass();

MyClass t2 = new MyClass();

Console.WriteLine(t1.x + " " + t2.x);

}

}

In this example, the constructor for **MyClass** is

public MyClass()

{

x = 10;

}

* Notice that the constructor is specified as **public**.
* This is because the constructor will be called from code defined outside of its class.
* This constructor assigns the instance variable **x** of **MyClass** the value 10.
* This constructor is called by **new** when an object is created. For example, in the line

MyClass t1 = new MyClass();

* the constructor **MyClass( )** is called on the **t1** object, giving **t1.x** the value 10.
* The same is true for **t2**. After construction, **t2.x** has the value 10.
* **Thus, the output from the program is**

10 10

**Parameterized Constructors**:

* Will need a constructor that accepts one or more parameters.
* Parameters are added to a constructor in the same way they are added to a method: just declare them inside the parentheses after the constructor’s name.
* For example, here **MyClass** is given a parameterized constructor:

// A parameterized constructor.

using System;

class MyClass

{

public int x;

public MyClass(int i)

{

x = i;

}

}

class ParmConsDemo

{

public static void Main()

{

MyClass t1 = new MyClass(10);

MyClass t2 = new MyClass(88);

Console.WriteLine(t1.x + " " + t2.x);

}

}

**The output from this program is shown here:**

10 88

* In this version of the program, the **MyClass( )** constructor defines one parameter called **i**, which is used to initialize the instance variable, **x**.
* Thus, when the line MyClass t1 = new MyClass(10); executes, the value 10 is passed to **i**, which is then assigned to **x**.

**Add a Constructor to the Building Class:**

* We can improve the **Building** class by adding a constructor that automatically initializes the **Floors**, **Area**, and **Occupants** fields when an object is constructed.

// Add a constructor to Building.

using System;

class Building

{

public int Floors; // number of floors

public int Area; // total square footage of building

public int Occupants; // number of occupants

// A parameterized constructor for Building.

public Building(int f, int a, int o)

{

Floors = f;

Area = a;

Occupants = o;

}

// Display the area per person.

public int AreaPerPerson()

{

return Area / Occupants;

}

// Return the maximum number of occupants if each

// is to have at least the specified minimum area.

public int MaxOccupant(int minArea)

{

return Area / minArea;

}

}

// Use the parameterized Building constructor.

class BuildingDemo

{

public static void Main()

{

Building house = new Building(2, 2500, 4);

Building office = new Building(3, 4200, 25);

Console.WriteLine("Maximum occupants for house if each has " + 300 + " square feet: " +

house.MaxOccupant(300));

Console.WriteLine("Maximum occupants for office if each has " + 300 + " square feet: " +

office.MaxOccupant(300));

}

}

The output from this program is the same as for the previous version.

* Both **house** and **office** were initialized by the **Building( )** constructor when they were created.
* Each object is initialized as specified in the parameters to its constructor.
* For example, in the following line,

Building house = new Building(2, 2500, 4);

* the values 2, 2500, and 4 are passed to the **Building( )** constructor when **new** creates the object.
* Thus, **house**’s copy of **Floors**, **Area**, and **Occupants** will contain the values 2, 2500, and 4, respectively