**Object Oriented Programming - Classes & Objects In C#**

**Object-Oriented Programming**

* Programming languages are based on two fundamental concepts, data and ways to manipulate data.
* Traditional languages such as Pascal and C used the procedural approach which focused more on ways to manipulate data rather than on the data itself.
* This approach had several drawbacks such as lack of re-use and lack of maintainability.
* To overcome these difficulties, OOP was introduced, which focused on data rather than the ways to manipulate data.
* The object-oriented approach defines objects as entities having a defined set of values and a defined set of operations that can be performed on these values.

**Object-oriented programming provides the following features:**

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

**Abstraction**

Abstraction is the feature of extracting only the required information from objects. For example, consider a television as an object. It has a manual stating how to use the television. However, this manual does not show all the technical details of the television, thus, giving only an abstraction to the user.

**Encapsulation**

Details of what a class contains need not be visible to other classes and objects that use it. Instead, only specific information can be made visible and the others can be hidden. This is achieved through encapsulation, also called data hiding. Both abstraction and encapsulation are complementary to each other.

**Inheritance**

Inheritance is the process of creating a new class based on the attributes and methods of an existing class. The existing class is called the base class whereas the new class created is called the derived class. This is a very important concept of object-oriented programming as it helps to reuse the inherited attributes and methods.

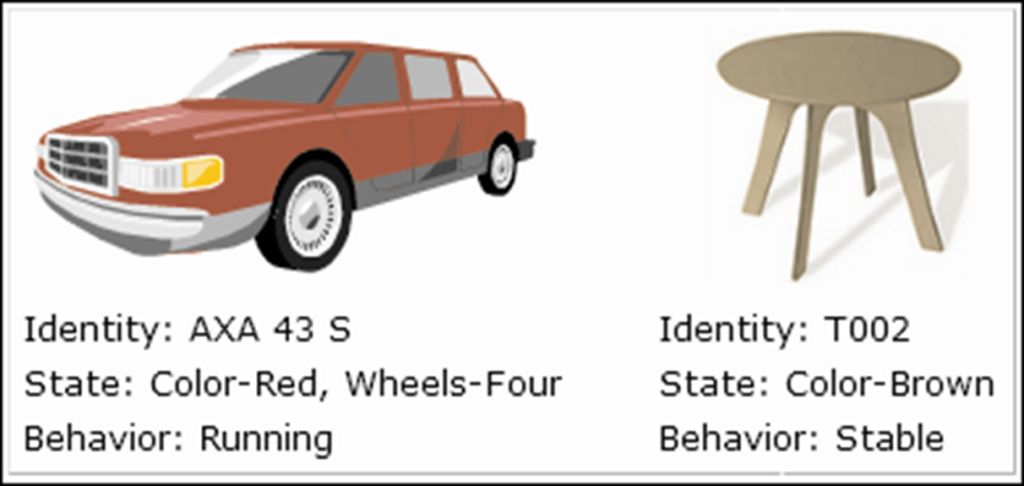
**Polymorphism**

Polymorphism is the ability to behave differently in different situations. It is basically seen in programs where you have multiple methods declared with the same name but with different parameters and different behavior.

**Classes and Objects**

* C# programs are composed of classes that represent the entities of the program which also include code to instantiate the classes as objects.
* When the program runs, objects are created for the classes and they may interact with each other to provide the functionalities of the program.
* An object is a tangible entity such as a car, a table, or a briefcase. Every object has some characteristics and is capable of performing certain actions.
* The concept of objects in the real world can also be extended to the programming world. An object in a programming language has a unique identity, state, and behavior.
* The state of the object refers to its characteristics or attributes whereas the behavior of the object comprises its actions.
* An object has various features that can describe it which could be the company name, model, price, mileage, and so on.

**The following figure shows an example of objects:**



* An object stores its identity and state in fields (also called variables) and exposes its behavior through methods.
* Several objects have a common state and behavior and thus, can be grouped under a single class.

**Example**

A Ford Mustang, a Volkswagen Beetle, and a Toyota Camry can be grouped together under the class Car. Here, Car is the class whereas Ford Mustang, Volkswagen Beetle, and Toyota Camry are objects of the class Car.

**The following figure displays the class Car:**



**Creating Classes**

* The concept of classes in the real world can be extended to the programming world, similar to the concept of objects.
* In object-oriented programming languages like C#, a class is a template or blueprint which defines the state and behavior of all objects belonging to that class.
* A class comprises fields, properties, methods, and so on, collectively called data members of the class. In C#, the class declaration starts with the class keyword followed by the name of the class.

**The following syntax is used to declare a class:**

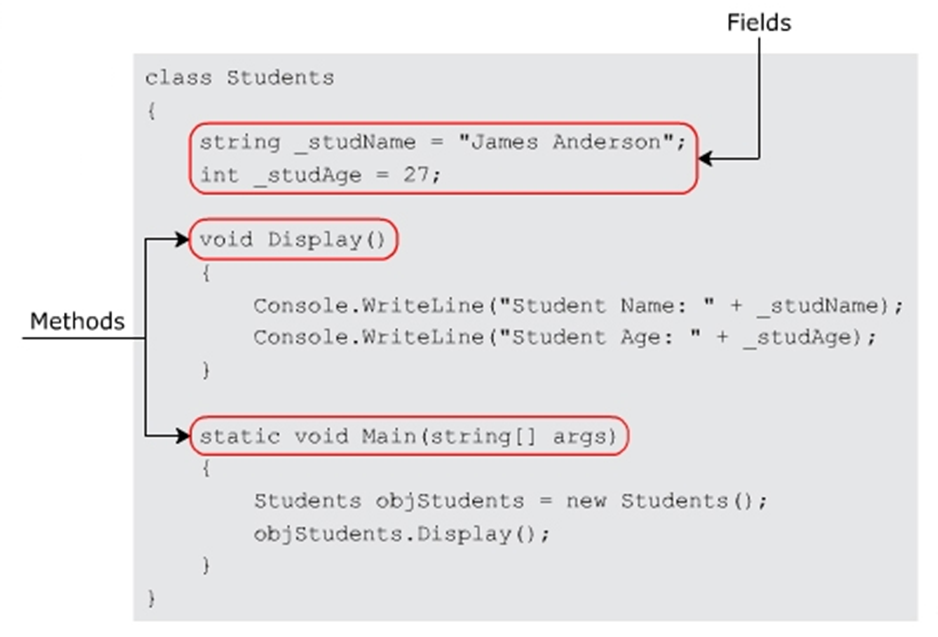
class className

{

// class members

}

**The following figure displays a sample class:**



**Guidelines for Naming Classes**

**There are certain conventions to be followed for class names while creating a class that help you to follow a standard for naming classes.**

**These conventions state that a class name:**

* Should be a noun and written in initial caps, and not in mixed case.
* Should be simple, descriptive, and meaningful.
* Cannot be a C# keyword.
* Cannot begin with a digit but can begin with the ‘@’ character or an underscore (\_).

**Example**

* Valid class names are: Account, @Account, and \_Account.
* Invalid class names are: 2account, class, Acccount, and Account123.

**Creating & Instantiating Objects**

* It is necessary to create an object of the class to access the variables and methods defined within it.
* In C#, an object is instantiated using the new keyword. On encountering the new keyword, the Just-in-Time (JIT) compiler allocates memory for the object and returns a reference of that allocated memory.

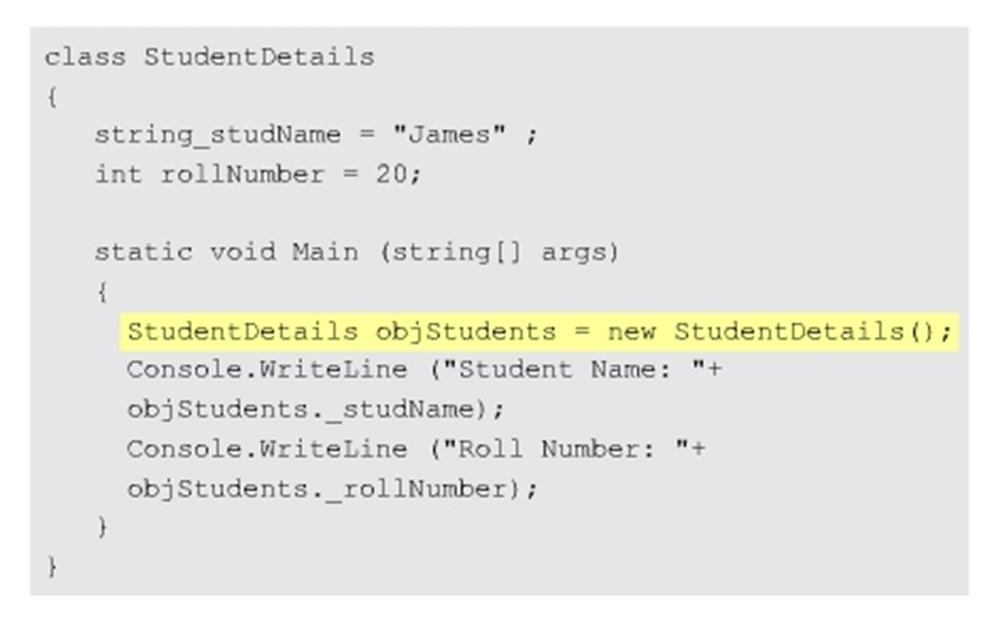
**The following syntax is used to instantiate an object.**

<ClassName><objectName> = new <ClassName>();

**In Above Code,**

* **ClassName**: Specifies the name of the class.
* **objectName**: Specifies the name of the object.

**The following figure displays an example of object instantiation:**



**The this keyword**

* The this keyword is used to refer to the current object of the class to resolve conflicts between variables having same names and to pass the current object as a parameter.
* You cannot use the this keyword with static variables and methods.

**In the following code, the this keyword refers to the length and \_breadth fields of the current instance of the class Dimension:**

using System;

class Dimension

{

double \_length;

double \_breadth;

public double Area(double \_length, double \_breadth)

{

this.\_length = \_length;

this.\_breadth = \_breadth;

return \_length \* \_breadth;

}

static void Main(string[] args)

{

Dimension objDimension = new Dimension();

Console.WriteLine(“Area of rectangle = “ +

objDimension.Area(10.5, 12.5));

}

}

**In Above Code,**

* The Area() method has two parameters \_length and \_breadth as instance variables. The values of these variables are assigned to the class variables using the this keyword.
* The method is invoked in the Main()method. Finally, the area is calculated and is displayed as output in the console window.

**The following figure displays the use of the this keyword:**

