**C# | Abstraction**

Data Abstraction is the property by virtue of which only the essential details are exhibited to the user. The trivial or the non-essentials units aren’t exhibited to the user.   
Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

**Example:** Consider a real-life scenario of withdrawing money from ATM. The user only knows that in ATM machine first enter ATM card, then enter the pin code of ATM card, and then enter the amount which he/she wants to withdraw and at last, he/she gets their money. The user does not know about the inner mechanism of the ATM or the implementation of withdrawing money etc. The user just simply knows how to operate the ATM machine, this is called abstraction.

In C# abstraction is achieved with the help of **Abstract classes**.

**Abstract Classes**

* An abstract class is declared with the help of **abstract**keyword.
* In C#, you are **not** allowed to create objects of the abstract class. Or in other words, you cannot use the abstract class directly with the new operator.
* Class that contains the abstract keyword with some of its methods (not all abstract method) is known as an **Abstract Base Class**.
* Class that contains the abstract keyword with all of its methods is known as pure Abstract Base Class.
* You are not allowed to declare the abstract methods outside the abstract class.
* You are not allowed to declare an abstract class as [**Sealed Class**](https://www.geeksforgeeks.org/c-sealed-class/).

There are situations in which we want to define a superclass that declares the structure of a given abstraction without providing a complete implementation of every method. That is, sometimes we want to create a superclass that only defines a generalized form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details.

Consider a classic “shape” example, perhaps used in a computer-aided design system or game simulation. The base type is “shape” and each shape has a color, size and so on. From this, specific types of shapes are derived (inherited)-circle, square, triangle, and so on – each of which may have additional characteristics and behaviors. For example, certain shapes can be flipped. Some behaviors may be different, such as when you want to calculate the area of a square.

**Example:**

|  |
| --- |
| *// C# program to calculate the area*  *// of a square using the concept of*  *// data abstraction*  using System;  namespace Demoabstraction {  *// abstract class*  abstract class Shape {  *// abstract method*      public abstract int area();  }  *// square class inheriting*  *// the Shape class*  class Square : Shape {    *// private data member*      private int side;    *// method of  square class*      public Square(int x = 0)      {          side = x;      }  *// overriding of the abstract method of Shape*  *// class using the override keyword*      public override int area()      {          Console.Write("Area of Square: ");          return (side \* side);      }  }    *// Driver Class*  class GFG {    *// Main Method*      static void Main(string[] args)      {  *// creating reference of Shape class*  *// which refer to Square class instance*          Shape sh = new Square(4);  *// calling the method*          double result = sh.area();          Console.Write("{0}", result);      }  }  } |

**Output:**

Area of Square: 16

### **Encapsulation vs Data Abstraction**

* [**Encapsulation**](https://www.geeksforgeeks.org/c-encapsulation/)is data hiding(information hiding) while Abstraction is detail hiding(implementation hiding).
* While encapsulation groups together data and methods that act upon the data, data abstraction deal with exposing to the user and hiding the details of implementation.

### **Advantages of Abstraction**

* It reduces the complexity of viewing things.
* Avoids code duplication and increases reusability.
* Helps to increase the security of an application or program as only important details are provided to the user.