C# - Indexers
C# - Properties

C# - Indexers

- An **indexer** allows an object to be indexed like an array.
- When you define an indexer for a class, this class behaves like a **virtual array**.
- You can then access the instance of this class using the array access operator ([ ]).

Use of Indexers

- Declaration of behavior of an indexer is to some extent similar to a property.
- Like properties, you use **get** and **set** accessors for defining an indexer.

C# - Indexers

- However, properties return or set a specific data member, whereas indexers returns or sets a particular value from the object instance.
- In other words, it breaks the instance data into smaller parts and indexes each part, gets or sets each part.
- Defining a property involves providing a property name.
- Indexers are not defined with names, but with the **this** keyword, which refers to the object instance.

C# - Indexers

- **Syntax of a one dimensional indexer**

```
element-type this[int index]
{
  // The get accessor.
  get   {
    // return the value specified by index
  }

  // The set accessor.
  set   {
    // set the value specified by index
  }
}
```

C# - Indexers

Overloaded Indexers

- Indexers can be overloaded.
- Indexers can also be declared with multiple parameters and each parameter may be a different type.
- It is not necessary that the indexes have to be integers. C# allows indexes to be of other types, for example, a string.

C# - Properties

- If a class contains any values in it and if we want to access those values outside of that class, then we can provide access to those values in 2 different ways. They are as follows:

1. By storing the value under a public variable, we can give direct access to the value outside of the class.
2. By storing that value in a private variable, we can also give access to that value outside of the class by defining a property for that variable.

C# - Properties

- A Property in C# is a member of a class that is used to set and get the data from a data field (i.e. variable) of a class.
- The most important point is that a property in C# is never used to store any data, it just acts as an interface or medium to transfer the data.
- We use the Properties as they are public data members of a class, but they are actually special methods called accessors.

C# - Properties

- **What are Accessors in C#?**
- The Assessors are nothing but special methods which are used to set and get the values from the underlying data member (i.e. variable) of a class. Assessors are of two types. They are as follows:
- **Set Accessor**
- **Get Accessor**

C# - Properties

- **What is a Set Accessor?**
- The **set** accessor is used to set the data (i.e. value) into a data field i.e. a variable of a class.
- This set accessor contains a fixed variable named **value**.
- Whenever we call the property to set the data, whatever data (value) we are supplying will come and store inside the variable called **value** by default. Using a set accessor, we cannot get the data.
- **Syntax: set { Data_Field_Name = value; }**

C# - Properties

- **What is Get Accessor?**
- The get accessor is used to get the data from the data field i.e. variable of a class. Using the get accessor, we can only get the data, we cannot set the data.
- **Syntax: get {return Data_Field_Name;}**

C# - Properties

- Example of Accessor

```
// Declare a Code property of type string:
   public string Code
   {
      get   {
         return code;
      }
      set   {
         code = value;
      }
   }
```

C# - Properties

- **Properties** are named members of classes, structures, and interfaces.
- Member variables or methods in a class or structures are called **Fields**.
- Properties are an extension of fields and are accessed using the same syntax.
- They use **accessors** through which the values of the private fields can be read, written or manipulated.
- Properties do not name the storage locations. Instead, they have **accessors** that read, write, or compute their values.

C# - Properties

Accessors

- The **accessor** of a property contains the executable statements that helps in getting (reading or computing) or setting (writing) the property.
- The accessor declarations can contain a get accessor, a set accessor, or both.

C# - Properties

- **Abstract Properties**
- An abstract class may have an abstract property, which should be implemented in the derived class.
- Same way interface may have an abstract property, which should be implemented in the derived class.