


Members (C# Programming Guide)

07/20/2015 • 2 minutes to read •  +3

In this article

[See also](#)

Classes and structs have members that represent their data and behavior. A class's members include all the members declared in the class, along with all members (except constructors and finalizers) declared in all classes in its inheritance hierarchy. Private members in base classes are inherited but are not accessible from derived classes.

The following table lists the kinds of members a class or struct may contain:

Member	Description
Fields	Fields are variables declared at class scope. A field may be a built-in numeric type or an instance of another class. For example, a calendar class may have a field that contains the current date.
Constants	Constants are fields whose value is set at compile time and cannot be changed.
Properties	Properties are methods on a class that are accessed as if they were fields on that class. A property can provide protection for a class field to keep it from being changed without the knowledge of the object.
Methods	Methods define the actions that a class can perform. Methods can take parameters that provide input data, and can return output data through parameters. Methods can also return a value directly, without using a parameter.
Events	Events provide notifications about occurrences, such as button clicks or the successful completion of a method, to other objects. Events are defined and triggered by using delegates.

Member	Description
Operators	Overloaded operators are considered type members. When you overload an operator, you define it as a public static method in a type. For more information, see Operator overloading .
Indexers	Indexers enable an object to be indexed in a manner similar to arrays.
Constructors	Constructors are methods that are called when the object is first created. They are often used to initialize the data of an object.
Finalizers	Finalizers are used very rarely in C#. They are methods that are called by the runtime execution engine when the object is about to be removed from memory. They are generally used to make sure that any resources which must be released are handled appropriately.
Nested Types	Nested types are types declared within another type. Nested types are often used to describe objects that are used only by the types that contain them.

See also

- [C# Programming Guide](#)
- [Classes](#)

Is this page helpful?

 Yes  No