**Reflection**

Reflection provides objects (of type [Type](https://docs.microsoft.com/en-us/dotnet/api/system.type)) that describe assemblies, modules and types. You can use reflection to dynamically create an instance of a type, bind the type to an existing object, or get the type from an existing object and invoke its methods or access its fields and properties. If you are using attributes in your code, reflection enables you to access them.

Reflection is the ability of a managed code to read its own metadata for the purpose of finding assemblies, modules and type information at runtime. In other words, reflection provides objects that encapsulate assemblies, modules and types. A program reflects on itself by extracting metadata from its assembly and using that metadata either to inform the user or to modify its own behavior. Reflection is similar to C++ RTTI (Runtime Type Information), but much broader in scope and capability. By using Reflection in C#, one is able to find out details of an object, method, and create objects and invoke methods at runtime. The System.Reflectionnamespace contains classes and interfaces that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types. When writing a C# code that uses reflection, the coder can use the typeof operator to get the object's type or use the GetType() method to get the type of the current instance.

