|  |  |  |
| --- | --- | --- |
| Access Modifiers and Keywords | | |
| Sr No. | Names | IMP Points |
| 1 | Public | 1.Allows access to the class, method, or variable from any other class or project. |
| 2 | Private | 1.Can only be accessed within the same class.  2. restrict access to other classes can’t access outside not even by derived classes. |
| 3 | Protected | 1.Can be accessed within the same class and its derived (child) classes.  2.Used in inheritance to allow child classes to reuse parent class functionality. |
| 4 | Internal | 1. Allows access to the member within the same assembly (project) but not from other assemblies.  2. Cannot be accessed from outside the assembly, even if the class is public. |
| 5 | Static | 1. Used to define members (methods, fields, properties) that belong to the class itself rather than an instance of the class.  2. Static cannot access non-static members directly.  3. Static fields and methods require a class reference to access. |
| 6 | Non-Static | 1. Requires an instance of the class to access.  2. Used when different objects need to maintain their own separate values. |
| 7 | Const | 1. Used to declare a constant value that cannot be changed.  2. Must be initialized at declaration. |
| 8 | Readonly | 1.Used to define variables that can only be assigned during declaration or inside a constructor.  2.Cannot be modified outside the constructor or declaration.  3.Can be assigned runtime values, unlike const.  4.Useful when values need initialization but should not change after object creation. |
| 9 | Var | 1. Automatically determines the data type at compile-time.  2. Must be initialized at declaration, and the type cannot be changed later. |
| 10 | Dynamic | 1. Allows variables to store any type and determines their type at runtime. |
| 11 | Params | 1. Allows a method to accept a variable number of arguments as an array. |
| 12 | Ref | 1. Passes arguments by reference instead of by value.  2. ref requires the variable to be initialized before passing  3. ref can use for modifying existing values. |
| 13 | Out | 1. Passes arguments by reference  2. out must be assigned a value within the method.  3. Used when a method needs to return multiple values. |
| 14 | Get and Set | 1. Used to encapsulate fields by providing controlled access.  2. get retrieves the value, while set assigns a new value.  3. Must be inside a property declaration. |
| 15 | Sealed | 1. Prevents a class from being inherited or a method from being overridden.  2. Used to restrict further modifications and ensure security of class.  3. Cannot be used with abstract classes or methods. |
| 16 | Singleton | 1. Ensures a class has only one instance and provides a global point of access to it.  2. Requires a private constructor and a static instance variable.  3. Ensure thread safety when implementing the Singleton pattern.  4. Used for managing shared resources. |
| 17 | Lock | 1. Ensures thread safety by allowing only one thread to access a block of code at a time.  2. Requires an object (reference type) as a lock token. |
| 18 | Interface | 1. Defines a contract that implementing classes must follow.  2. Cannot contain implementation details (only method).  3. A class can implement multiple interfaces, but only inherit from one class. |
| 19 | Virtual | 1. Virtual allows method overriding, Allows a method in a base class to be overridden in a derived class.  2.Virtual methods must have a base implementation.  3.Virtual is optional to override. |
| 20 | Abstract | 1. Defines a method or class that must be implemented in a derived class.  2. abstract methods can only exist in abstract classes.  3. Abstract classes cannot be instantiated. |
| 21 | Method Overloading | 1.Overloading allows multiple methods with the same name but different parameters in the same class.  2. Overloading needs a different parameter signature. (name and parameters)  3. Overloading is determined at compile time.  4.Also called asStatic Binding. |
| 22 | Method Overriding | 1. Allows a derived class to provide a new implementation for a method defined in the base class.  2. Overriding requires the method to be virtual or abstract in the base class.  3. Overriding is determined at run time.  4. Also called asDynamic Binding. |
| 23 | Async | 1. Async is a modifier that indicates a method is asynchronous.  2. Must be used with a method (async method).  3. Async alone does not make a method asynchronous; it requires await inside. |
| 24 | Await | 1. Await is an operator used inside an async method to pause execution until the awaited task completes.  2. It ensures that the method resumes only after the awaited operation finishes.  3. Can only be used inside an async method.  4. If await is not used inside an async method, it will run synchronously. |