# 1. What is CLR in .NET?

Options:

A) Common Language Runtime

B) Compiled Language Runtime

C) Common Language Routine

D) Compiled Language Routine

Correct Answer: A) Common Language Runtime

Explanation: The Common Language Runtime (CLR) is a crucial part of the .NET framework, acting as the execution engine that runs .NET applications. It provides key services such as memory management, type safety, exception handling, garbage collection, and thread management. The CLR allows different .NET languages to be used together and ensures that applications are executed in a managed environment, improving security and robustness.

# 2. Which of the following is true about the .NET Framework?

Options:

A) It supports only C#.

B) It is a platform-independent framework.

C) It includes a large library of coded solutions to prevent common programming problems.

D) It is a programming language.

Correct Answer: C) It includes a large library of coded solutions to prevent common programming problems.

Explanation: The .NET Framework is not just for C#; it supports multiple programming languages. It is primarily a Windows-based framework, making it platform-dependent. However, with the introduction of .NET Core (now part of .NET 5 and onwards), some aspects of .NET have become cross-platform. The framework is not a programming language but includes a vast collection of pre-built classes and libraries known as the Framework Class Library (FCL), designed to handle common programming tasks, thereby speeding up the development process.

# 3. What does the 'var' keyword in C# signify?

Options:

A) The variable must be initialized.

B) The variable type is mutable.

C) The variable type is determined at compile time.

D) The variable can only store variant types.

Correct Answer: C) The variable type is determined at compile time.

Explanation: The 'var' keyword in C# is used for implicitly typed local variable declarations. The type of the variable is determined by the compiler at compile time based on the type of the initializer expression. This means that once initialized, the type of the variable cannot change, making it strongly typed, not mutable in terms of its type, and not specifically for storing variant types. It requires initialization at the point of declaration.

# 4. Which of the following is a value type in C#?

Options:

A) String

B) Class

C) Struct

D) Interface

Correct Answer: C) Struct

Explanation: In C#, value types include simple types (e.g., integer, float), enum types, struct types, and nullable value types. Structs are value types and are stored on the stack, which means they hold their data directly, in contrast to reference types (like classes and strings) that are stored on the heap and hold a reference to their data. Interfaces are not types per se but rather contracts that classes and structs can implement.

# 5. What is MVC in ASP.NET?

Options:

A) Model-View-Controller

B) Model-View-Connector

C) Module-View-Controller

D) Model-Vector-Controller

Correct Answer: A) Model-View-Controller

Explanation: MVC stands for Model-View-Controller. It's a design pattern widely used in web development, including ASP.NET, to separate the application into three main components. The Model represents the application's data structure, the View is responsible for displaying the data (the UI component), and the Controller handles the input, processes requests, and returns the appropriate views or data. This separation allows for modular development, making it easier to manage and scale web applications.

# 6. Which method in ASP.NET MVC is used to return an HTTP 404 error?

Options:

A) HttpNotFound()

B) NotFound()

C) Error404()

D) HTTPNotFound()

Correct Answer: A) HttpNotFound()

Explanation: In ASP.NET MVC, the HttpNotFound() method is used within a controller to return an HTTP 404 status code, indicating that the requested resource could not be found on the server. This method is part of the controller base class and helps in handling requests for non-existent resources gracefully, by returning a standard HTTP 404 response.

# 7. What is Code First in Entity Framework?

Options:

A) A design where the database is manually created before writing the code

B) A design that allows developers to work directly with SQL

C) A design that enables developers to define database models using code

D) A design that requires a pre-existing database

Correct Answer: C) A design that enables developers to define database models using code

Explanation: Code First is an approach within Entity Framework that allows developers to create database models using C# code. Instead of designing a database schema first and then generating the corresponding code, the Code First approach lets developers define their data models in code, and Entity Framework generates the database schema based on these models. This approach is particularly useful for a code-centric development workflow, where the emphasis is on the code rather than the database design.

# 8. Which of these is an advantage of using Entity Framework?

Options:

A) Increased performance compared to raw SQL

B) Direct control over the SQL code generated

C) Reduction in the amount of code for data access

D) Complete elimination of the need for data validation

Correct Answer: C) Reduction in the amount of code for data access

Explanation: One of the primary advantages of using Entity Framework (EF) is the significant reduction in the amount of code required for data access in applications. EF is an Object-Relational Mapping (ORM) framework that abstracts the database layer, allowing developers to interact with the database using high-level entities and LINQ queries instead of writing verbose SQL queries. This not only speeds up the development process but also makes the code more maintainable. While EF might not always outperform optimized raw SQL in terms of performance, and it doesn't eliminate the need for data validation, its benefits in reducing boilerplate code and simplifying data operations are considerable.

# 9. What is Dependency Injection in .NET?

Options:

A) A design pattern used for creating loosely coupled code

B) Injecting code at runtime to change the application behavior

C) A method to increase the coupling between different modules of the application

D) A technique used to enhance the performance of applications

Correct Answer: A) A design pattern used for creating loosely coupled code

Explanation: Dependency Injection (DI) is a design pattern that allows for the creation of loosely coupled code, making systems more modular, easier to test, and maintain. In .NET, DI involves automatically providing a class with the instances of the objects (dependencies) it needs to perform its function instead of the class creating these objects directly. This is usually achieved through the use of a DI container or framework, which manages the instantiation and life cycle of objects and their dependencies. DI helps in reducing class dependencies, making code more flexible and easier to manage.

# 10. What feature does the async/await keyword pair in C# implement?

Options:

A) Parallel programming

B) Synchronous programming

C) Asynchronous programming

D) Sequential programming

Correct Answer: C) Asynchronous programming

Explanation: The async and await keywords in C# are used to implement asynchronous programming, allowing for non-blocking operations. When a method is marked with the async

# Value And references Type

* **Primitive Types**: These include:
  + **int**
  + **double**
  + **float**
  + **bool**
  + **char**
  + **byte**
  + **sbyte**
  + **short**
  + **ushort**
  + **uint**
  + **ulong**
  + **long**
* **Structs**: Custom structures and many built-in types are also value types, including:
  + **DateTime**
  + **TimeSpan**
  + **Guid**
  + All custom structs defined using the **struct** keyword
* **Enumerations (Enums)**: Defined using the **enum** keyword, e.g., **enum Color { Red, Green, Blue }**

**Reference Types**

Reference types store a reference to their data, which is stored in the heap. This allows them to be shared and modified by different parts of a program. Common reference types include:

* **Classes**: Any type declared with the **class** keyword, such as:
  + **string**
  + **object** (the base type from which all other types derive)
  + All custom classes, such as **FileStream**, **StringBuilder**, etc.
* **Arrays**: Any type of array (single-dimensional, multi-dimensional, or jagged), e.g., **int[]**, **double[,]**, **string[][]**
* **Delegates**: Types that represent references to methods with a specific parameter list and return type, such as:
  + **Func<T>**
  + **Action**
* **Interfaces**: Such as **IEnumerable**, **IDisposable**, etc.
* **Dynamic Types**: Declared using the **dynamic** keyword in C#.