C# and .NET Comprehensive Question Bank

# Question 1

What is the primary purpose of the .NET Framework?

* Web development
* Mobile app development
* Desktop application development
* All of the above

✅ Answer: All of the above

💡 Explanation: The .NET Framework supports building web, desktop, and mobile applications using a unified platform.

# Question 2

Which of the following is NOT a feature of .NET Core?

* Cross-platform
* Open-source
* Windows-only
* High performance

✅ Answer: Windows-only

💡 Explanation: .NET Core is designed to be cross-platform, running on Windows, Linux, and macOS.

# Question 3

What is the file extension for a C# source file?

* .csharp
* .cs
* .c#
* .cpp

✅ Answer: .cs

💡 Explanation: C# source files use the `.cs` extension.

# Question 4

Which component of .NET is responsible for executing code?

* CLR
* FCL
* MSIL
* JIT

✅ Answer: CLR

💡 Explanation: The Common Language Runtime (CLR) is the execution engine for .NET applications.

# Question 5

What is the role of the Common Type System (CTS)?

* To compile code
* To manage memory
* To define how types are declared and used
* To handle exceptions

✅ Answer: To define how types are declared and used

💡 Explanation: CTS ensures that types are used consistently across different .NET languages.

# Question 6

Which keyword is used to define a class in C#?

* struct
* class
* interface
* object

✅ Answer: class

💡 Explanation: The `class` keyword is used to define a class in C#.

# Question 7

What is the output of the following code?  
int x = 5;  
int y = x++;  
Console.WriteLine(y);

* 4
* 5
* 6
* 7

✅ Answer: 5

💡 Explanation: `x++` is post-increment, so `y` gets the value before incrementing.

# Question 8

What is the default access modifier for class members in C#?

* public
* private
* protected
* internal

✅ Answer: private

💡 Explanation: Class members are private by default unless specified otherwise.

# Question 9

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

* string
* object
* int
* array

✅ Answer: int

💡 Explanation: `int` is a value type, while others are reference types.

# Question 10

What does the `using` directive do in C#?

* Declares a namespace
* Imports a namespace
* Defines a class
* Initializes a variable

✅ Answer: Imports a namespace

💡 Explanation: The `using` directive allows access to classes in a namespace.

# Question 11

Which of the following is NOT a principle of Object-Oriented Programming?

* Encapsulation
* Inheritance
* Compilation
* Polymorphism

✅ Answer: Compilation

💡 Explanation: Compilation is a process, not a principle of OOP. The main principles are Encapsulation, Inheritance, and Polymorphism.

# Question 12

What is the purpose of a constructor in a class?

* To define methods
* To initialize objects
* To inherit from other classes
* To implement interfaces

✅ Answer: To initialize objects

💡 Explanation: Constructors are special methods used to initialize new objects of a class.

# Question 13

Which keyword is used to inherit a class in C#?

* inherits
* base
* extends
* : (colon)

✅ Answer: : (colon)

💡 Explanation: In C#, inheritance is denoted using the colon `:` symbol.

# Question 14

What is method overloading?

* Defining multiple methods with the same name but different parameters
* Replacing a method in a derived class
* Calling a method from a base class
* Using a method from another namespace

✅ Answer: Defining multiple methods with the same name but different parameters

💡 Explanation: Method overloading allows multiple methods with the same name but different signatures.

# Question 15

What is the purpose of the `virtual` keyword in C#?

* To define a static method
* To define a method that can be overridden
* To define a method that cannot be changed
* To define a constructor

✅ Answer: To define a method that can be overridden

💡 Explanation: The `virtual` keyword allows a method to be overridden in a derived class.

# Question 16

What is the output of the following code?  
class A {  
 public virtual void Show() { Console.WriteLine("A"); }  
}  
class B : A {  
 public override void Show() { Console.WriteLine("B"); }  
}  
A obj = new B();  
obj.Show();

* A
* B
* AB
* Compile-time error

✅ Answer: B

💡 Explanation: Since `Show()` is overridden in class B and the object is of type B, it prints "B".

# Question 17

What is encapsulation in OOP?

* Hiding implementation details
* Inheriting from a base class
* Overloading methods
* Using multiple classes

✅ Answer: Hiding implementation details

💡 Explanation: Encapsulation hides internal state and requires all interaction to be performed through an object’s methods.

# Question 18

Which access modifier allows access only within the same class?

* public
* protected
* private
* internal

✅ Answer: private

💡 Explanation: The `private` modifier restricts access to the containing class only.

# Question 19

What is the base class for all classes in C#?

* Object
* Base
* Root
* Class

✅ Answer: Object

💡 Explanation: All classes in C# implicitly inherit from the `System.Object` class.

# Question 20

What is the difference between `abstract` and `interface` in C#?

* Abstract classes can have implementations; interfaces cannot
* Interfaces can have constructors
* Abstract classes cannot be inherited
* Interfaces can contain fields

✅ Answer: Abstract classes can have implementations; interfaces cannot

💡 Explanation: Abstract classes can include method implementations, while interfaces cannot (prior to C# 8.0).

# Question 21

Which design pattern provides a way to create objects without specifying the exact class to instantiate?

* Singleton
* Factory Method
* Observer
* Decorator

✅ Answer: Factory Method

💡 Explanation: The Factory Method pattern defines an interface for creating objects, but allows subclasses to alter the type of objects that will be created.

# Question 22

Which design pattern ensures a class has only one instance and provides a global point of access to it?

* Singleton
* Prototype
* Adapter
* Facade

✅ Answer: Singleton

💡 Explanation: The Singleton pattern ensures a class has only one instance and provides a global point of access to it.

# Question 23

Which design pattern allows an object to alter its behavior when its internal state changes?

* State
* Strategy
* Command
* Visitor

✅ Answer: State

💡 Explanation: The State pattern allows an object to alter its behavior when its internal state changes.

# Question 24

Which design pattern is used to encapsulate a request as an object, thereby allowing parameterization of clients with queues, requests, and operations?

* Command
* Chain of Responsibility
* Mediator
* Observer

✅ Answer: Command

💡 Explanation: The Command pattern encapsulates a request as an object, thereby allowing parameterization of clients with queues, requests, and operations.

# Question 25

Which design pattern provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation?

* Iterator
* Composite
* Flyweight
* Proxy

✅ Answer: Iterator

💡 Explanation: The Iterator pattern provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

# Question 26

Which design pattern allows you to define a family of algorithms, encapsulate each one, and make them interchangeable?

* Strategy
* Template Method
* Visitor
* Interpreter

✅ Answer: Strategy

💡 Explanation: The Strategy pattern allows you to define a family of algorithms, encapsulate each one, and make them interchangeable.

# Question 27

Which design pattern provides a simplified interface to a complex subsystem?

* Facade
* Adapter
* Bridge
* Composite

✅ Answer: Facade

💡 Explanation: The Facade pattern provides a simplified interface to a complex subsystem.

# Question 28

Which design pattern allows an object to be notified when another object changes state?

* Observer
* Decorator
* Chain of Responsibility
* Command

✅ Answer: Observer

💡 Explanation: The Observer pattern allows an object to be notified when another object changes state.

# Question 29

Which design pattern allows you to add new functionality to an object without altering its structure?

* Decorator
* Adapter
* Composite
* Flyweight

✅ Answer: Decorator

💡 Explanation: The Decorator pattern allows you to add new functionality to an object without altering its structure.

# Question 30

Which design pattern allows you to compose objects into tree structures to represent part-whole hierarchies?

* Composite
* Flyweight
* Proxy
* Chain of Responsibility

✅ Answer: Composite

💡 Explanation: The Composite pattern allows you to compose objects into tree structures to represent part-whole hierarchies.

# Question 31

Which collection class is used to store key-value pairs?

* ArrayList
* HashTable
* Stack
* Queue

✅ Answer: HashTable

💡 Explanation: The HashTable class is used to store key-value pairs.

# Question 32

Which collection class represents a first-in, first-out (FIFO) collection of objects?

* ArrayList
* HashTable
* Stack
* Queue

✅ Answer: Queue

💡 Explanation: The Queue class represents a first-in, first-out (FIFO) collection of objects.

# Question 33

Which collection class represents a last-in, first-out (LIFO) collection of objects?

* ArrayList
* HashTable
* Stack
* Queue

✅ Answer: Stack

💡 Explanation: The Stack class represents a last-in, first-out (LIFO) collection of objects.

# Question 34

Which collection class is used to store elements in a non-sequential manner?

* ArrayList
* HashTable
* LinkedList
* Queue

✅ Answer: LinkedList

💡 Explanation: The LinkedList class is used to store elements in a non-sequential manner.

# Question 35

Which collection class is used to store elements in a sequential manner?

* ArrayList
* HashTable
* LinkedList
* Queue

✅ Answer: ArrayList

💡 Explanation: The ArrayList class is used to store elements in a sequential manner.

# Question 36

Which interface is used to compare objects for sorting?

* IEnumerable
* IComparable
* IComparer
* ICloneable

✅ Answer: IComparable

💡 Explanation: The IComparable interface is used to compare objects for sorting.

# Question 37

Which interface is used to compare objects for equality?

* IEnumerable
* IComparable
* IComparer
* ICloneable

✅ Answer: IComparer

💡 Explanation: The IComparer interface is used to compare objects for equality.

# Question 38

Which interface is used to iterate over a collection?

* IEnumerable
* IComparable
* IComparer
* ICloneable

✅ Answer: IEnumerable

💡 Explanation: The IEnumerable interface is used to iterate over a collection.

# Question 39

Which interface is used to clone objects?

* IEnumerable
* IComparable
* IComparer
* ICloneable

✅ Answer: ICloneable

💡 Explanation: The ICloneable interface is used to clone objects.

# Question 40

Which collection class is used to store elements in a binary tree structure?

* ArrayList
* HashTable
* LinkedList
* BinaryTree

✅ Answer: BinaryTree

💡 Explanation: The BinaryTree class is used to store elements in a binary tree structure.

# Question 41

Which keyword is used to define a generic class in C#?

* generic
* template
* class
* T

✅ Answer: class

💡 Explanation: The `class` keyword is used to define a generic class in C#.

# Question 42

Which keyword is used to define a generic method in C#?

* generic
* template
* method
* T

✅ Answer: method

💡 Explanation: The `method` keyword is used to define a generic method in C#.

# Question 43

Which keyword is used to define a generic delegate in C#?

* generic
* template
* delegate
* T

✅ Answer: delegate

💡 Explanation: The `delegate` keyword is used to define a generic delegate in C#.

# Question 44

Which keyword is used to define a generic interface in C#?

* generic
* template
* interface
* T

✅ Answer: interface

💡 Explanation: The `interface` keyword is used to define a generic interface in C#.

# Question 45

Which keyword is used to define a generic collection class in C#?

* generic
* template
* collection
* T

✅ Answer: collection

💡 Explanation: The `collection` keyword is used to define a generic collection class in C#.

# Question 46

Which keyword is used to define a generic constraint in C#?

* generic
* template
* constraint
* T

✅ Answer: constraint

💡 Explanation: The `constraint` keyword is used to define a generic constraint in C#.

# Question 47

Which keyword is used to define a generic parameter in C#?

* generic
* template
* parameter
* T

✅ Answer: parameter

💡 Explanation: The `parameter` keyword is used to define a generic parameter in C#.

# Question 48

Which keyword is used to define a generic type in C#?

* generic
* template
* type
* T

✅ Answer: type

💡 Explanation: The `type` keyword is used to define a generic type in C#.

# Question 49

Which keyword is used to define a generic argument in C#?

* generic
* template
* argument
* T

✅ Answer: argument

💡 Explanation: The `argument` keyword is used to define a generic argument in C#.

# Question 50

Which keyword is used to define a generic return type in C#?

* generic
* template
* return
* T

✅ Answer: return

💡 Explanation: The `return` keyword is used to define a generic return type in C#.