1. **What is Object-Oriented Programming (OOP)?**
   OOP is a programming paradigm based on the concept of objects that contain data and methods.
2. **What are the four main principles of OOP?**
   The four main principles are encapsulation, inheritance, polymorphism, and abstraction.
3. **What is encapsulation?**
   Encapsulation is the bundling of data and methods that operate on that data within a single unit, restricting direct access to some components.
4. **What is inheritance?**
   Inheritance is a mechanism that allows one class to inherit the properties and methods of another class.
5. **What is polymorphism?**
   Polymorphism allows objects to be treated as instances of their parent class, enabling method overriding and overloading.
6. **What is abstraction in OOP?**
   Abstraction is the concept of hiding complex implementation details and showing only the essential features of an object.
7. **What is a class in C#?**
   A class is a blueprint for creating objects that defines properties and methods.
8. **What is an object in C#?**
   An object is an instance of a class that contains data and can perform actions defined by its class.
9. **What is a constructor in C#?**
   A constructor is a special method called when an object is instantiated, used to initialize its fields.
10. **What is a destructor in C#?**
    A destructor is a method called when an object is garbage collected, used for cleanup purposes.

11. **What is method overloading?**
    Method overloading allows multiple methods in the same class to have the same name with different parameters.
12. **What is method overriding?**
    Method overriding allows a derived class to provide a specific implementation of a method already defined in its base class.
13. **What is an interface in C#?**
    An interface is a contract that defines a set of methods and properties that implementing classes must provide.
14. **What is an abstract class?**
    An abstract class is a class that cannot be instantiated and may contain abstract methods that must be implemented in derived classes.
15. **What is a property in C#?**
    A property is a member of a class that provides a flexible mechanism to read, write, or compute the value of a private field.
16. **What is encapsulation achieved through in C#?**
    Encapsulation is achieved through access modifiers (public, private, protected, internal) that restrict access to class members.
17. **What is a sealed class in C#?**
    A sealed class cannot be inherited, preventing other classes from deriving from it.
18. **What is a static class in C#?**
    A static class cannot be instantiated and can contain only static members.
19. **What is a delegate in C#?**
    A delegate is a type that references methods with a specific parameter list and return type, allowing method references to be passed around.
20. **What is an event in C#?**
    An event is a special kind of delegate that provides a notification mechanism for other classes when something of interest occurs.

21. **What is the difference between a class and a struct in C#?**
    A class is a reference type, while a struct is a value type, with different memory allocation and behavior characteristics.
22. **What is a base class in C#?**
    A base class is a class from which other classes derive, providing common functionality.
23. **What is a derived class in C#?**
    A derived class inherits from a base class and can extend or override its functionality.
24. **What is polymorphism achieved through in C#?**
    Polymorphism is achieved through method overriding and interface implementation.
25. **What is an object initializer in C#?**
    An object initializer allows you to create an object and set its properties in a single statement.
26. **What is the purpose of the `virtual` keyword in C#?**
    The `virtual` keyword allows a method to be overridden in derived classes.
27. **What is the purpose of the `override` keyword in C#?**
    The `override` keyword is used to provide a new implementation for a method inherited from a base class.
28. **What is the difference between `==` and `Equals()` in C#?**
    `==` checks for reference equality, while `Equals()` checks for value equality based on the implementation.
29. **What is a constructor chaining in C#?**
    Constructor chaining occurs when one constructor calls another constructor in the same class or base class.
30. **What is the purpose of the `new` keyword in C#?**
    The `new` keyword is used to create instances of objects or hide a member inherited from a base class.

31. **What is a namespace in C#?**
    A namespace is a container that holds a set of related classes, structs, enums, and interfaces for organizing code.
32. **What is a tuple in C#?**
    A tuple is a data structure that holds a fixed number of elements of potentially different types.
33. **What is an abstract method in C#?**
    An abstract method is a method declared without an implementation in an abstract class, requiring derived classes to implement it.
34. **What is a concrete class?**
    A concrete class is a class that can be instantiated and provides implementations for all its members.
35. **What is an interface versus an abstract class?**
    An interface cannot provide any implementation, while an abstract class can provide both complete and incomplete methods.
36. **What is the significance of the protected access modifier?**
    The protected access modifier allows members to be accessible only within their class and by derived class instances.
37. **What is a constructor with parameters?**
    A constructor with parameters allows passing arguments to initialize an object' s properties upon instantiation.
38. **What is a factory method in OOP?**
    A factory method is a design pattern that uses a method to create objects, allowing for the instantiation of different classes based on conditions.
39. **What is the Singleton pattern?**
    The Singleton pattern restricts a class to a single instance and provides a global access point to that instance.
40. **What is a static member in C#?**
    A static member belongs to the class itself rather than to any specific instance, shared across all instances.

41. **What is the purpose of the const keyword in C#?**
    The const keyword defines a constant field or local variable whose value cannot be modified.
42. **What is the readonly keyword in C#?**
    The readonly keyword defines a field that can only be assigned at its declaration or in a constructor.
43. **What is an iterator in C#?**
    An iterator is a method that uses the yield keyword to return elements one at a time, allowing for easier iteration over collections.
44. **What is a lambda expression in C#?**
    A lambda expression is a concise way to represent an anonymous function that can be used to create delegates or expression tree types.
45. **What is a dynamic type in C#?**
    A dynamic type can hold any data type and is resolved at runtime, allowing for more flexibility.
46. **What is a generic type in C#?**
    A generic type allows classes, interfaces, and methods to operate on a specified type without specifying that type until runtime.
47. **What is a sealed method in C#?**
    A sealed method cannot be overridden in any derived class.
48. **What is the purpose of the is keyword in C#?**
    The is keyword is used to check if an object is compatible with a given type.
49. **What is the purpose of the as keyword in C#?**
    The as keyword is used to perform a safe type conversion, returning null if the conversion fails.
50. **What is a ValueTuple in C#?**
    A ValueTuple is a lightweight structure that can hold multiple values and supports named fields.
51. **What is the difference between an interface and a delegate?**
    An interface defines a contract for classes, while a delegate is a type-safe function pointer that can refer to methods.

52. **What is the `yield return` statement in C#?**
    The `yield return` statement returns each element one at a time from an iterator method.
53. **What is a default constructor?**
    A default constructor is a constructor that does not take parameters and initializes an object with default values.
54. **What is a copy constructor?**
    A copy constructor creates a new object as a copy of an existing object of the same class.
55. **What is a parameterized constructor?**
    A parameterized constructor is a constructor that accepts parameters to initialize object properties.
56. **What is a base constructor?**
    A base constructor is a constructor of the base class that can be invoked from a derived class constructor using `base`.
57. **What is a `sealed` class?**
    A `sealed` class cannot be inherited, preventing other classes from deriving from it.
58. **What is an object-oriented design pattern?**
    An object-oriented design pattern is a reusable solution to a commonly occurring problem in software design.
59. **What is a composition in OOP?**
    Composition is a design principle where a class is composed of one or more objects from other classes to achieve functionality.
60. **What is aggregation in OOP?**
    Aggregation is a relationship between classes where one class contains a reference to another class, representing a "has-a" relationship.
61. **What is an interface with default implementation?**
    An interface with default implementation allows methods to have a default behavior that can be overridden by implementing classes.

62. **What is method hiding in C#?**
    Method hiding occurs when a derived class defines a method with the same name as a method in its base class, using the new keyword.
63. **What is a polymorphic variable?**
    A polymorphic variable is a variable that can hold references to objects of different types but shares a common base type.
64. **What is a class diagram in OOP?**
    A class diagram visually represents the classes, their attributes, methods, and relationships in a system.
65. **What is a design pattern?**
    A design pattern is a general reusable solution to a commonly occurring problem in software design.
66. **What is an object lifecycle?**
    An object lifecycle refers to the various stages an object goes through from creation to destruction.
67. **What is the difference between compile-time and run-time polymorphism?**
    Compile-time polymorphism (method overloading) occurs during compilation, while run-time polymorphism (method overriding) occurs at runtime.
68. **What is dependency injection in OOP?**
    Dependency injection is a design pattern that allows for the removal of hard-coded dependencies, making classes more testable and maintainable.
69. **What is a software interface?**
    A software interface defines the methods and properties that a class must implement without providing the implementation details.
70. **What is a module in OOP?**
    A module is a separate unit of functionality within a system, encapsulating a specific aspect of the system' s functionality.

71. **What is a method in C#?**
    A method is a block of code that performs a specific task and can be called to execute that task.
72. **What is a class member?**
    A class member is a property, method, event, or field that is defined within a class.
73. **What is a static method in C#?**
    A static method belongs to the class itself and can be called without creating an instance of the class.
74. **What is a non-static method?**
    A non-static method requires an instance of the class to be called and operates on instance data.
75. **What is a multi-tier architecture?**
    A multi-tier architecture is a client-server architecture where the application is divided into multiple layers or tiers.
76. **What is a wrapper class in C#?**
    A wrapper class is a class that provides a way to use primitive data types as objects.
77. **What is an interface segregation principle?**
    The interface segregation principle states that no client should be forced to depend on methods it does not use, promoting smaller and specific interfaces.
78. **What is a single responsibility principle?**
    The single responsibility principle states that a class should have only one reason to change, focusing on a single responsibility.
79. **What is the open-closed principle?**
    The open-closed principle states that software entities should be open for extension but closed for modification.
80. **What is the Liskov substitution principle?**
    The Liskov substitution principle states that objects of a superclass should be

replaceable with objects of a subclass without affecting the correctness of the program.

81. **What is a method signature?**
    A method signature consists of the method' s name and its parameter types, uniquely identifying the method.
82. **What is a class hierarchy?**
    A class hierarchy is the arrangement of classes in a parent-child relationship, showing inheritance relationships.
83. **What is an object state?**
    An object state refers to the current values of its properties at a given time.
84. **What is the purpose of using interfaces?**
    Interfaces provide a way to achieve polymorphism and decouple code, allowing for flexible design.
85. **What is type safety in C#?**
    Type safety ensures that variables are used only in ways that are compatible with their data types, preventing type errors.
86. **What is a reference type?**
    A reference type holds a reference to the actual data, meaning multiple references can point to the same object.
87. **What is a value type?**
    A value type directly contains its data, and each variable holds a separate copy of the data.
88. **What is the difference between reference equality and value equality?**
    Reference equality checks if two references point to the same object, while value equality checks if two objects contain the same data.
89. **What is the purpose of the override modifier?**
    The override modifier allows a derived class to provide a new implementation of a method inherited from a base class.
90. **What is a concrete implementation?**
    A concrete implementation is a complete and functional definition of a method or property.

91. **What is a virtual method?**

    A virtual method is a method defined in a base class that can be overridden in a derived class.

92. **What is the difference between abstract and virtual methods?**

    An abstract method has no implementation in the base class, while a virtual method has a default implementation that can be overridden.

93. **What is a method call?**

    A method call is an instruction to execute a method, passing control to that method and optionally passing parameters.

94. **What is a composite pattern?**

    A composite pattern is a structural design pattern that allows you to compose objects into tree structures for representing part-whole hierarchies.

95. **What is a state pattern?**

    A state pattern is a behavioral design pattern that allows an object to alter its behavior when its internal state changes.

96. **What is a strategy pattern?**

    A strategy pattern is a behavioral design pattern that defines a family of algorithms, encapsulating each one and making them interchangeable.

97. **What is an adapter pattern?**

    An adapter pattern is a structural design pattern that allows incompatible interfaces to work together by converting one interface into another.

98. **What is a proxy pattern?**

    A proxy pattern is a structural design pattern that provides a surrogate or placeholder for another object to control access to it.

99. **What is the purpose of the using statement?**

    The using statement ensures that disposable objects are properly disposed of when no longer needed, managing resource cleanup.

100. **What is a class responsibility collaboration (CRC) model?**

    A CRC model is a modeling technique that helps identify the responsibilities of classes and their interactions with other classes.