1. What is Object Oriented Programming?

OOP is a technique to develop logical modules, such as classes that contains properties, fields and events. OOP provides many concepts such as inheritance, data binding, polymorphism etc.

Object means a real word entity such as pen,paper, chair, table etc.

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects.

It simplifies the software development and maintenance by providing some concepts:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

Simula is considered as the first object-oriented programming language.

2. What is a class?

A class can be defined as the primary building block of OOP. A class contains data and behavior of an entity.

A class in C# can contain:

1. data member  
2. properties  
3. constructor  
4. methods

Notes:

1. Class name should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.  
2. The name of the constructor is always same as the class name  
3. A class can have any number of data members, properties, constructors and methods  
4. Data member defined using a class is called as object reference.  
5. A class can have a data member which is an object reference of the same class Like the manager of the employee is also a employee.

3. What is an Object?

An entity that has state and behavior is known as an object e.g. pen, table, car etc. It can be physical or logical.

An object has three characteristics:

* state: represents data (value) of an object.
* behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.
* identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But,it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Parker, color is black etc. known as its state. It is used to write, so writing is its behavior.  
Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance(result) of a class. For example, you have a class called Vehicle and car is the object of that class.

4. Explain the basic features of OOPs ?

OOPs has four basic features.

* Abstraction : Abstraction is the process of exposing only the relevant data to the users without showing unnecessary information
* Polymorphism : I allows you to use an entity in multiple forms
* Encapsulation : Prevents the data from unwanted access by binding of code and data in a single unit called object
* Inheritance : Promotes the reusability of code and eliminates the use of redundant code.

5. What are Abstract classes ?

An abstract class is a class that can not be instantiated and is always used as a base class.

Characteristics of an abstract class:

* You can not instantiate an abstract class directly
* You can have abstract as well as non abstract members in an abstract class
* You must declare one abstract method in the abstract class
* An abstract class is always public
* An abstract class is declared using abstract keyword

6. Explain the features of an Interface

* An Interface contains only the signature of methods
* An Interface has no Implementation on its own
* An Interface is used to implement multiple inheritance in code.
* It defines a static set of methods and their arguments
* Variables in Interface must be declared as public,static and final
* Methods in an Interface must be declared as public and abstract
* A class implementing an Interface must implement all of its method
* An Interface can derive from more than one Interface

7. Difference between an abstract class and an Interface ?

|  |  |
| --- | --- |
| Abstract Class | Interface |
| A class can extend only one Abstract class | A class can implement several interfaces |
| The member of an abstract class can be private and protected | An Interface can only have public members |
| Abstract classes should have subclasses | Interfaces must have Implementations by classes |
| Any class can extend an abstract class | Only an Interface can extend another Interface |
| Methods in an abstract class can be abstract as well as concrete | All methods in an Interface should be abstract |
| There can be a constructor for Abstract class | Interface does not have constructor |
| An abstract class can Implement methods | Interfaces can not contain body of any of its method |

8. Explain different types of Inheritance

There are four types of Inheritance in Object oriented programming.

* Single Inheritance : contains one base class and one derived class
* Hierarchical Inheritance : Contains one base class and multiple derived classes of same base class
* Multilevel Inheritance : Contains a class derived from a derived class
* Multiple Inheritance : Contains several base class and a derived class

9. What is Constructor?

C# constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor.  
A constructor is a special method that is used to initialize an object. Every class has a constructor,if we don’t explicitly declare a constructor for any C# class the compiler builds a default constructor for that class. A constructor does not have any return type. Constructors are responsible for object initialization and memory allocation of its class.

There are basically two rules defined for the constructor.

1. Constructor name must be same as its class name  
2. Constructor must have no explicit return type

You can read more on constructor [here](http://www.csharpstar.com/csharp-constructor/).

10. What is Destructor?

A Destructor is automatically invoked when an object is finally destroyed. The name of the Destructor is same as class and prefixed with a tilde (~)

A Destructor is used to free the dynamic allocated memory and release the resources. You can Implement a custom method that allows to control object destruction by calling the destructor.

* Destructors don not have any return type
* Destructors are always public
* Destructors can not be overloaded

11. What is a Static Constructor?

Static constructor is used to initialize static data of a class. Compiler calls the static constructor before the first instance is created.

* No access specifier is required to define it
* You can not pass parameters to static constructor
* A class can have only one static constructor
* It can access only static members of class
* It is invoked only once, when the program execution begins

12. What is Method Overloading?

Method Overloading lets you have 2 methods with same name and different signature.  
Overloading can be achieved:  
-By changing the number of parameters used.  
-By changing the order of parameters.  
-By using different data types for the parameters.

13. What is Access Specifier?

An access specifier defines the scope and visibility of a class member. C# supports the following access specifiers:

* Public
* Private
* Protected
* Internal
* Protected internal

14. What is Encapsulation?

* Encapsulation is a process of hiding the members from outside of class and implemented using access specifiers
* Encapsulation is also called as information hiding.
* Encapsulation provides a way to preserve the integrity of state data. Rather than defining public fields, private data fields should be defined.
* Well-encapsulated class should hide its data and the details of how it operates on data from the outside world. This is termed black box programming.
* Using this,implementation of the method can be changed by the class author without breaking any existing code making use of it.

15. What is the difference between Method Overloading and Method Overriding?

|  |  |
| --- | --- |
| **Method Overloading** | **Method Overriding** |
| Method Overloading lets you have 2 methods with same name and different signature | Method Overloading lets you have 2 methods with same name and same signature |
| Overloading is called as compile time polymorphism or early binding | Overriding is called as run time polymorphism or late binding or dynamic polymorphism |
| Overloading can be achieved: -By changing the number of parameters used. -By changing the order of parameters. -By using different data types for the parameters. | Overriding can be achieved: -Creating the method in a derived class with same name, same parameters and same return type as in base class is called as method overriding |
| Method overloading can be overloaded in same class or in the child class. | Method overriding is only possible in derived class not within the same class where the method is declared |
| Example:  public class Methodoverloading  {  public int addition(int a, int b) //Two int parameters with same method  {  return a + b;  }  public int addition(int a, int b,int c) //Three int Parameters with same method  {  return a + b + c;  }  public float addition(float a, float b,float c,float d) //four float type Parameters with same method same as above two method  {  return a + b + c + d;  }  } | Example:  Public class test  {  public virtual int addition()  {  return 10;  }  }  public class Amount:test  {  public override int balance()  {  return 500;  }  } |

16. What is the difference between Abstraction and Encapsulation ?

Encapsulation is wrapping, just hiding properties and methods. Encapsulation is used for hide the code and data in a single unit to protect the data from the outside the world. Class is the best example of encapsulation. Abstraction refers to showing only the necessary details to the intended user.

17. What is Operator Overloading ?

Operator overloading is a technique by which operators used in a programming language are implemented in user-defined types with customized logic that is based on the types of arguments passed.

Operator overloading facilitates the specification of user-defined implementation for operations wherein one or both operands are of user-defined class or structure type. This helps user-defined types to behave much like the fundamental primitive data types. Operator overloading is helpful in cases where the operators used for certain types provide semantics related to the domain context and syntactic support as found in the programming language. It is used for syntactical convenience, readability and maintainability

18. What is a Delegate ?

Delegates are type safe function pointer. It holds reference to a function.

The signature of delegates matches the signature of the function that it points to else you will get compilation error.  
Delegates are typesafe pointers because it points to a function and holds the signature of the function.

19. What is multicast Delegate ?

A multicast delegate is a delegate that has references to more than one function.When you inoke a multicase delegate,all the functions the delegate is pointing to are also invoked.

There are 2 ways to create multicast delegate.

+ or += to register a method with the delegate  
– or -= to unregister a method with the delegate  
A multicast delegate invokes the methods in invocation list in same order in which they are added.

20. What are Events ?

Events are user actions such as key press, clicks, mouse moves, etc., or some occurrence such as system generated notifications. Applications need to respond to events when they occur.

21. What is the difference between Array and Collection?

|  |  |
| --- | --- |
| Array | Collection |
| You need to specify the size of an array at the time of its declaration.It can not be resized. | The size of a collection can be adjusted dynamically as per users requirement.It does not have fixed size |
| The member of an array should be of the same data type | Collection can have elements of different types |

22. What is the difference between Shadowing and Overriding?

Shadowing:

1.You can shadow a base class member in the derived class by using the keyword Shadows.  
2.The method signature, access level and return type of the shadowed member can be completely different than the base class member.  
3.A method or function of the base class is available to the child (derived) class without the use of the “overriding” keyword.  
4.The compiler hides the function or method of the base class. This concept is known as shadowing or method hiding.  
5.In the shadowing or method hiding, the child (derived) class has its own function, the same function is also available in the base class.

Overriding:

1.Method overriding is an important feature of OOPS that allows us to re-write a base class function or method with a different definition.  
2.Overriding is also known as “Dynamic Polymorphism” because overriding is resolved at runtime.  
3.The method signature, access level and return type of the hidden member has to be same as the base class member  
4.In other words both methods (base class method and derived class method) have the same name, same number and same type of parameter in the same order with the same return type.  
5.The overridden base method must be virtual, abstract or override.

You can look at C# implementation on Shadowing and Overriding [here](http://www.csharpstar.com/overriding-vs-shadowing-in-csharp/).

23. What is the difference between Class and Structure?

|  |  |
| --- | --- |
| Class | Structure |
| A Class is a reference type | A Structure is a value type |
| By default, the members of a Class are private | By default, the members of a Structure are public |
| Class supports Inheritance | Structure does not support Inheritance |
| Class can contain constructor/destructor | Structure does not require Constructor/Destructor |
| Variables of a Class can be assigned as null | Structure members can not have null values |

24. What is the similarities between Class and Structure?

* Access specifiers are identically used in structure and classes to restrict the access of their data and methods outside their body
* Both can have constructors,methods, properties,fields, constants etc..
* Both can implement Interfaces to use multiple-inheritance in code
* Both can have Delegates and Events

25. What is Enum?

* An enum is a distinct value typewith a set of named constants
* It is declared by using keyword Enum
* Every Enum type has an underlying type which can be any integral type except char. The default underlying type is int.
* The first enumerator has the value 0 by default and the value of each successive enumerator is increased by 1
* Enumeration bridge the gap between numbers and objects

26. What is a Nested Class?

Nested class is nothing but defining a class within another class.

Nested classes has the ability to specify private as an access modifier for the class itself.The use of the private access modifier defines the intended accessibility of the class and prevents access from outside the class.

27. What is an Indexer?

An Indexer is a member that enables an object to be indexed like Arrays.

28. What is Sealed keyword in C#?

You can disable inheritance by using the sealed keyword on a class or a method. When used on a class, you can’t derive other classes from it. When used on a method, derived classes can’t override the method.

29. What is a hashtable?

Hashtable is store multiple items and each of these items is associated with unique string key. Each item can be accessed using the key associated with it.

30. What is Polymorphism and explain different types of Polymrphism?

The word polymorphism means having many forms.  
polymorphism is ‘one interface, multiple functions’. Polymorphism can be static or dynamic.

Static Polymrphism:

The mechanism of linking a function with an object during compile time is called early binding. It is also called static binding. C# provides two techniques to implement static polymorphism. They are:

1.Function overloading  
2.Operator overloading

Dynamic Polymorphism:

The mechanism of linking method to an object during run time is called late binding.It is also known as run-time polymorphism. Method overriding helps to implement Dynamic Polymorphism.

Thanks for visiting !!