** What is Microsoft Dotnet?

->It is frame work / software platform and developed by Microsoft

-> It contains language and technologies

** What is Datatype?

-> It is a container place for storing data

** Types of Datatypes?

-> Fixed size Datatype and Varying size Datatype

-- Fixed Size DT --

long - 8 bytes (signed)

ulong - 8 bytes (unsigned)

int - 4 bytes (signed)

uint - 4 bytes (unsigned)

short - 2 bytes (signed)

ushort - 2 bytes (unsigned)

byte - 1 byte (unsigned)

sbyte - 1 byte (signed)

-- Varying Size DT

string, class, array, enum, interface, object, structure

** What is signed and unsigned data type?

-> in signed we can store both positive and negative values

-> in unsigned we can store only positive values

** What is Datatype conversion?

-> copying data from one data type to another data type

-> Implicit Copying, Explicit Copying, Copying Data using convert class methods

1. Implicit Copying

-> Directly we can copy the data

-> Syntax: DDT DV = SV;

2. Explicit Copying

-> Forcefully we can copying the data

-> Syntax: DDT DV = (DDT) SV;

3. Copying Data using Convert Class Methods

-> we can copy the data using convert class methods

-> Syntax: Convert.ToInt32();Convert.ToInt64();Convert.ToString();Convert.ToSingle();

** Why it is called a virtual entity?

-> No memory allocation in RAM for storing class data

** What is the output for printing the object variable?

-> namespace_name.class_name

** What is simple datatype?

-> data type which can store only one value

-> we can access variable by it's name

-> ex: fixed size data type and strings also treated like simple data type

** What is complex datatype?

-> data type can store more than one value

-> we can access variable through the object variable

-> ex: all varying size data type

** What is method?

-> method is used for performing some task / to do some action

-> a method can return only one value

** What is naming conventions and why we use?

-> it is used by the industries

-> it improves code readability and code maintainability

-> two types:

1. pascal casing -> first letter of each word should be uppercase (all)

2. camel casing -> first letter of each word should be uppercase except first word should be first character lowercase (local variables)

** Types of variables?

-> global variable - inside the class we declared the variables is called global variables

-> local variable - inside the method we declared the variables is called local variables

** What is naming standard?

-> class = complex nouns

-> variables = simple nouns

-> methods = verbs

** What is debugging?

-> it is execution time process

-> using debugging, to understand the logic written by programmers

** RAM ARCHITECTURE?

1. Stack - Local variables stored

2. Heap - objects & arrays created

3. Code Segment - methods will be loaded -- once method will be completely executed in code segment method will be deleted

4. Data Segment - static members / class members will be created

5. String Buffer / String Pool - strings will be stored

** What is array?

-> array is a collection which is used to store group of related data's under by a single variable name

-> Syntax: DT[] VN/AN = new DT[3]{"value1","value2"};

-> Why we used array: to avoid memory wastage caused by variable names

-> array index will start from zero

** What is loops?

-> using loops, to reducing code repetition/duplicate codes

-> types of loops in c#

1. For loop
2. For each loop
3. while loop - > it is a entry control loop
4. do while loop -> it is a exit control loop

-> what is iteration: one complete loop execution is called iteration

** What is recursion?

-> a method calling itself is known as recursion

** Difference between strings and character array?

-- strings --

-> strings are stored into string buffer/string pool

-> strings are immutable can't change the value

-> string is treated like a simple data type we can accessing by it's name

-- character array --

-> character array is created in heap memory

-> char array is mutable we can change the value

-> char array is complex data type we can accessing char array name with index number

** C# Escape characters?

1. \a - to produce some sounds
2. \t - to produce tab space / 5 white space
3. \n - new line
4. \' - print single quote
5. \"- print double quote
6. \\ - print backslash
7. \b - backspace

** What is verbatim strings?

-> string starting with @ is known as verbatim strings

-> all character present in double quotes are treated as a normal character in verbatim strings

_____END OF C# BASIC_____

OOPS CONCEPT

** What is inheritance?

-> the process of accessing one class members in another class

-> using inheritance, we can reduce code duplication

-> it will be improving code maintainability

** Types of inheritance?

1. single level inheritance -> one parent, one child

2. multi-level inheritance -> one parent, one child, one grand child

3. hierarchical inheritance -> 1 parent, multiple child’s

4. multiple inheritance -> one child, multiple parents (it's not support directly but we can achieve using interface)

** What is object class?

-> it is a inbuilt class

-> the class name is object

-> also called universal data type

-> if class is not containing any other user defined parent class it as automatically compiler adding the object class as parent class

** What is static?

-> static is a keyword which is used to create static members

-> always static variables executed first

-> accessing is faster than instance members

-> static members and variables only one executed in program life time

** What is assembly?

-> it is a file

-> which is created by c# compiler

** What is method hiding?

-> child class method will be hiding parent class method

ADVANCED EXTRA QUESTIONS

** What is contains Framework?

-> it contains tools & resources

** What is tools?

-> C# compiler

-> debugging tool

-> CLR (JIT COMPILER, GARBAGE COLLECTOR)

** What is resources?

-> base class libraries

-> resources contain rules

1. CTS - Common Type System ===> Common rules for all language present in CTS

2. CLS - Common Language Specification

=> specific rules for each language present in CLS

** What is Manifest?

-> Information about assembly

** What is Metadata?

-> Information about code/program

** What .dll file?

-> the application is not executable directly is called .dll file

** What is .exe file?

-> the application is executable directly is called .exe file

**What is garbage collector?

-> garbage collector is responsible for deleting/removing dead object/unused objects

-> to avoid memory wastage

** What is dead object?

-> object is not having any reference then that object is called dead objects

** What is IDispossable?

-> inbuilt interface

-> whenever the class is implementing IDispossable interface compulsory create object of the class within using block

** What is Collections?

-> collections are similar like inbuilt classes

-> it is used for storing collection of related data

-> two types

1. Non Generic Collections :

No type safety = casting conversion will be required

* ArrayList
* ListDictionary
* Hashtable
* SortedList
* Stack
* Queue
* StringDictionary
* HybridDictionary
* Array

2. Generic Collections :

Type safety = casting conversion will not be required

* List<>
* Dictionary<>
* SortedList<>
* Stack<>
* Queue<>

OOPS CONCEPT

1. What is class?

-> class is a virtual entity or a model or a blue print or template

-> class is used for storing related variables and methods.

2. What is object?

-> object is a physical entity or a real-world entity

-> when object is created for a class there will be memory allocation in RAM for storing class data.

3. Difference Between String and StringBuilder?

-- string --

-> strings are immutable

-> in strings if we try to modify strings new strings will be create hence memory wastage will be more

-- string builder --

-> string builder is mutable

-> in string builder if we try to modify strings new strings will not be created hence no memory wastage

4. What is the purpose of constructor?

-> using constructor, we can initialize object or assign data into object

-> constructor name must be same as class name

-> constructor is a special type of method

-> in c# always parent class constructor executed first then only child class constructor executed

5. What is default constructor?

-> default constructor is used for initializing object with default values

-> when we don't create any constructor default constructor will be created by compiler

6. What is an object-oriented programming language?

-> object-oriented programming language is language

-> which supports encapsulation, abstraction, inheritance, polymorphism

7. What is encapsulation?

-> Grouping or binding related variables and methods in a common related place is called encapsulation

8. What is abstraction?

-> Hiding implementation details exposing the required details to the users.

9. What is upcasting?

-> identifying parent class object reference by using child class object reference.

-> Syntax: PC PCV= CV;

10. What is down casting?

-> identifying child class object reference by using parent class object reference.

-> syntax:CC CCV= (CC)PCV;

11. What is the purpose of new keyword?

-> it is used for creating of a class.

-> new keyword is also used for hiding base class members in derived class

12. What is the use of base keyword?

-> using base keyword, we can access base class members in derived class

-> it is always referring immediate parent class

13. What is the purpose of this keyword?

-> this keyword refers to the current object or current instance

-> in the time execution this keyword will be converted current object address

14. What is static constructor?

-> static constructor is used for initializing static variables.

-> static constructor can't have access modifier

-> static constructor can't have parameters

-> we can't call explicitly

-> always child class static constructor is executed first then only parent class static constructor executed

15. What is the purpose of instance constructor?

-> instance constructor is used for initializing instance variables

16. What is namespace?

-> namespace is used for grouping logically related classes, structs, enums, interfaces, etc..

-> after compilation all the namespace members converted into namespace_name.members_name

17. What is namespace aliasing?

-> assigning a shortcut name for bigger namespace name is called as name space aliasing.

-> namespace aliasing will improve code readability

18. What is the purpose of access modifiers and explain?

-> access modifiers are used for providing security to the program members.

-> we have 5 types of access modifiers public, private, protected, internal, protected internal

1. public -- members will be accessible from anywhere

2. private -- members will be accessible only within the declared class

3. protected -- members are accessible only within declared class and derived class

4. internal -- members are accessible only within the declared assembly

5. protected internal -- members are accessible within declared assembly and also in the declared class and derived class

19. What is the default access modifier for namespace level members?

-> internal

20. What is the default access modifier for class level members?

-> private

21. What is the default access modifier for interface level members?

-> public

22. What is polymorphism?

-> when a method is present with the same name in different forms then that method is said to be exhibiting polymorphism

23. What are the different types of polymorphism?

-> compile time polymorphism and runtime polymorphism

1. compile time polymorphism (static/early binding)-> overloading=method, constructor, operator

2. runtime polymorphism (dynamic/late binding)-> overriding=method overriding

24. What is overloading?

-> when 2 or more methods having same name with different signature in the given class then we can say the method is overloaded

25. What is virtual method?

-> virtual method is a low priority method. we can change the behaviour of virtual method in the derived class

26. What is overriding?

-> overriding is used for changing the behaviour of parent class method in the child class

-> for overriding a method its name in child & parent class must be same and also the method signature must be same

27. Is it possible to override a virtual method?

-> yes

28. Is it possible to override a override method?

-> yes

29. Is it possible to override a static method?

-> no

30. Is it possible to overload a static method?

-> yes

31. What is sealed method?

-> sealed method is a method whose behaviour we can't change in derived class.

-> we can declare only an override method as sealed method

32. Difference between instance variables and static variables?

-- instance variables --

-> instance variables are object specific

-> we can access instance variables using object name

-- static variables --

-> static variables are class specific

-> we can access static variables using class name

33. Difference Between instance method and static method?

-- instance method --

-> instance method must be called using object name

-> execution of instance method is slower than static method

-> inside instance method we can use this keyword

-- static method --

-> static method must be called using class name

-> execution of static method is faster than instance method

-> inside static method we can't use this keyword

34. Is it possible to use this keyword for accessing instance variables?

-> yes

35. Is it possible to access instance variables from static methods?

-> no

36. What are the different types of classes?

1. Instance class
2. Static class
3. Sealed class

4. Nested class

5. Abstract class

6. Partial class

37. What is static class?

-> static class is a class which can contain only static members

-> static class can't participate in inheritance

-> we can't create object of static class

-> when a class is having only static members it is a better to declare class as static class

38. What is sealed class?

-> sealed class is a class which will not contain child

-> we can create one or more objects for a sealed class

-> virtual method is not allowed in sealed class

-> abstract methods are not allowed in sealed class

39. What is nested class?

-> nested class is a class. class which is declared inside another class

-> we can create object for inner class using outer class name dot inner class name

40. What is partial class?

-> Using partial class, we can split single class definition into multiple files.

-> during compilation time all partial classes definition which are present inside same name space will merge into single class

41. What is an abstract class?

-> abstract class is a class which contains zero or more abstract methods.

-> we can't create object for an abstract class

42.What is abstract method?

-> abstract method is a method without body (or) unimplemented methods

-> it contains only header

-> after compilation it converted to pure virtual methods

-> pure virtual method : vm without body

43. When to create abstract class?

-> it is recommended to create abstract class when we know the implementation of some method and when we don't know the implementation for some methods.

44. What is interface?

-> interface is like class it can contain only unimplemented methods

-> we can't directly create an object for interface

45. When to create interface?

-> when two developers having code dependency, we can eliminate the dependency using interfaces. interface will act as a contract between developers

46. Is it possible to implement multiple interfaces in a single class?

-> yes

47. Difference Between Abstract Class and Interface?

-- Abstract Class --

-> It supports implemented and unimplemented methods

-> It's support versioning

-> A class can inherit from only one abstract class

-- Interface --

-> It supports only unimplemented methods

-> interface will not support versioning

-> A class can implement any number of interfaces

48. What is versioning?

-> when you add an implemented method to the abstract class it doesn't affect the child class (versioning)

49. what is property?

-> property is a similar like a variable

-> property is used for reading & modifying data present in private variables

50. What is constant variable?

-> class specific constants

-> it's a global variable

-> internally it will be static

-> always constant variable has to be an initialized variable

51. What is read-only variable?

-> object specific constants

-> read-only variables data can be modified in the same class constructor

52. What is var keyword?

-> declaring implicitly typed variables

-> implicitly types means c# compiler specify the datatype

-> after compilation c# compiler is going to specify the required data type

50. What is Difference Between value type and reference type?

-- value type --

-> value type will directly store data in the variable

-> reading data from value type is faster

-- reference type --

-> reference type will store address of the actual data in the variable

-> reading data from reference type is slow

51. What is structure?

-> structure is similar to a class

-> structure is a value type

-> structure can't contain initialized instance variables

-> structure can't have parameter less constructor

-> structure can't have destructor

-> structure can't participate in inheritance

-> passing structure var to some other variable then the entire structure will be copied

52. What is the difference between class and structure?

-- class --

-> class is a reference type

-> class can contain initialized instance variables

-> class can contain parameter less constructor

-> class can participate in inheritance

-- structure --

-> structure is a value type

-> structure can't contain initialized instance variables

-> structure can't contain parameter less constructor

-> structure can't contain a destructor

-> structure can't participate in inheritance

53. what is the use of out keyword?

-> using out keyword we can pass address of variable

54. what is the use of ref keyword?

-> using ref keyword we can pass address of variable

55. what is the difference between ref and out keyword?

-- ref --

-> we must initialize ref variables before passing to a function

-> using ref we can't return multiple values

-- out --

-> we must initialize or re-initialize within the called function

-> using out we can return multiple values from a function

56. what is the difference between for and foreach loop?

-> foreach loop is faster than for loop.

-> we can't modify data present in a for each loop iteration variable

-> Syntax : foreach(individual_element DT iteration_variable in collection_name)

57. what is boxing?

-> converting a value type to object type is called as boxing

58. what is unboxing?

-> converting object type to value type is called unboxing

59. what is the use of is keyword?

-> is keyword is used for checking the data type of variable

60. what is the use of as keyword?

-> as keyword is used for data type conversion

61. What is the difference between cosnt and read-only variable?

-- constant --

-> we must initialize a const variable while declaring

-> constant variables must be accessed using class name

-- read only --

-> read only variables can be initialized during declaration or inside a constructor

-> read only variables must be accessed using object name

62. what is an enum?

-> enum is used for grouping related constants

-> enum is a value type

63. what is an exception?

-> exception is a run time error

-> exception is an abnormal event which occurs at the time of execution & it disturbs the normal program execution

64. what is exception handling?

-> it is used for avoiding the abrupt termination of program. exception handling is done using try catch blocks

65. what is the purpose of a finally block?

-> finally block will be executed in all situations

-> inside the finally block we can't write return statement

-> important code like database connection closing will be usually written in finally block. clr guarantees the execution of finally block code.

66. what is the use of throw keyword?

-> using throw keyword we can re throw the exceptions back to the callers

67. MSIL is platform dependent or independent?

-> MSIL is platform independent

68. which class is the base class for all exception classes in c#?

-> exception

69. what is manifest in an assembly?

-> manifest is used for storing the current assembly name, version number, public & private key also the culture details

70. what is metadata in an assembly?

-> metadata will store all class names method name property names etc...

-> present in the assembly. it will also contain the info about which method belongs to which class

71. Who will call GC?

-> CLR will call GC when there is a scarcity of the memory

72. GC can delete which type of objects?

-> only managed objects GC can delete

73. what is destructor?

-> destructor is used for destroying un managed resources.

-> destructor can't have parameter

-> in destructor we can't specify the access modifier

-> execution of destructor is child to parent

74. JIT compiler and CLR is platform dependent or independent?

-> JIT & CLR are platform dependent

75. What are the difference between destructor and IDispossable interface Dispose method?

-> destructor will be called by the GC while deleting the object. Dispose method must be called by programmer after completing usage of the object

76. What is the problem with non generic collections?

-> type conversions will happen while inserting or reading data from non generic collections

77. what is the difference between stack and queue?

-> stack is used for retrieving data in LIFO order & queue used for reading data in FIFO order

78. what is a nullable type?

-> nullable type is a value type which is useful for storing null values along with respective value type values

79. what is an attribute?

-> attributes are useful for adding custom meta data into the meta data section of the assembly

-> within the square braces we can specify the custom metadata's or attribute

80. what is the use of reflection?

-> using reflections, we can read metadata present in the assembly.

81. what is delegate?

-> delegate is a type safe function pointer

-> delegate is used for implementing callback function

-> delegates are internally considered as classes

-> Access_Modifier delegate ReturnType delegate_name (parameter);

82. what is the purpose of multicast delegate?

-> a delegate that points to multiple method is called a multicast delegate

-> the "+" operator adds a function to the delegate object and the "-" operator removes an existing function from a delegate object

83. What is an event?

-> in c# events are used for implementing notifications. event will store one or more delegate objects. events are usually used in GUI programming

84. What is Extension Methods?

-> extension method is a static method. using extension methods we can add new methods to an existing class without modifying source code

85. What is the C# using block and why should i use it?

-> using block is used for creating objects

-> which implements Idispossable interface

-> using block will be automatically converted to try finally pattern and dispose method will be called in the finally block