Q1. Explain the difference between .net and C#

A: -> .Net is a framework and C# is programming language

.Net framework is the collection of Libraries

-.net has CLR, Garbage collector, CTS

- Also helps to run ,net application

Q2. Difference between .Net Framework 4.0 vs .Net Core vs .Net6.0

|  |  |  |  |
| --- | --- | --- | --- |
|  | **.Net Framework 4.0** | **.Net Framework Core** | **.Net Framework 6.0** |
| **Cross Platform** | Only runs on Wiindows | Works Cross Platform(Windows, Mac, Linux) | It is unification of both .net 4 and .net core. Unifies lot of .net runtimes |
| **Performance** | SLowas compared with.net Core | Better(they have segrigrate the dl lth larger dlls like system.collection into system.Collections.Immutable and syste,.collections.Concurrent) |
| CLI tools | More IDE Based(VS) | Full CLI support |
| Packaging | Packaged as one big framework | Delivered via modularly using Nuget |
|  |  |  |
|  |  |  |
|  |  |  |

Q3. What is IL(Intermediate Language) code and what is the ue of JIT(Just in time compiler)

A: When we compile the code it first converts it into partially compiled code which is called as IL(Intermediate language ) code.

When we run our program, JIT runs over the IL code and compiles it into machine language.

Q. Is it possible to view IL code ?

A: Yes, by dis assemblers like ILspy or ILDASM

Q. What is the importance of compiling in to IL code?

A: The Runtime environment and development environment can be very different. SO depending on the runtime environment JIT complies the best optimized code as per that environment(OS)

Q. What is CLR (Common Language Runtime)

A.: It converts your IL code into Native language(ML). it invokes JIT to complie IL code.

- Cleans any unused objects by sing Garbage collector(GC)

Q. what is managed and unmanged code?

A: Manged Code : Code that executes under the environment of CLR is called as Manged code.. All ILCode.

Unmanaged Code: Code with does not executes under CLR are Unmanaged code. C++, VB ++. In short is not controlled by CLR. They have there on environments in which the code runs and its completely outside the control of CLR.

Q. Explain the importance of Garbage Collector.

A: Garbage collector is a background process which cleans unused managed resources.

Q. Can Garbage collector claims Unmanaged resources.

A: No, it cannot claims the code outside the CLR.

Q. Importance of CTS(Common Type System)?

A. CTS ensure that data types defined in two different languages gets compile to a common datatype of .net Frameworf.

Q. Difference between Heap and Stack

A: Stack and heap are memory type in an application.

Stack Memory stores primitive data types like int, double, Boolean etc.

While heap stores data type like strings and objects.

Static void main(string[] args)

{

Int i=10; // i=10 will be stored in stack 1st

Bool b: true; // b=true will be stored in stack next

Customer x= new customer();// x will ne stored in stack

x.name=”test” // name=test will be saved in Heap.. so that x will be pointer to heap

}

Q. What are value types and reference type

A: Value type contains actual data while reference type contains pointers and pointers point to actual data

Value type are stored on stack while reference type are stored on heap.

Value type are normal data type like int, bool, double and reference type are all objects.

Q. Whats is boxing and unboxing

A: Boxing : Convrsion of value type to reference type Is called boxing

Unboxing : conversation of reference type to value type is called unboxing

Static void main(string[], args)

{

Int i=0;

Object y=i; // value type to reference type (Boxing)

In z =(int)y;; // reference type to value type(Unboxing)

Console.Read()

}

Q. what are the consequences of boxing and unboxing

A: performance (as we do conversion performance will impact ).

Q. Explain casting, implicit casting and expliit casting?

A: Type casting is a mechanism where we are converting one type of data to other type.

Implicit casting is when u are moving from lower datatype to higher datatype

Explicit casting is when u are moving from higher datatype to lower datatype

Static void mail (string[], args)

{

Int i =10;

Double d=i; //casting (implicit casting) (lower datatype to higher datatype)

Double d1=100.23;

Int y= (int)d1; //explicitly casting) (higher datatype to lower)

Console.Read();

Q. What are the consequences of explicit casting

A: Data loss . in above scenario integer doesn’t support decimals so after decimals data will be lost.

Q. Difference between Array and Arraylist

A:

|  |  |  |
| --- | --- | --- |
|  | **Array** | **ArrayList** |
| **Fixed length** | Yes | No (Flexible) |
| **Strongly type** | Yes | No |
| **Performance** | Better than Arraylist | Slower because of boxing and unboxing |

Static void main(string [], maik)

{

Int[] array= {1,2,3}; // array (fixed length, strongly type

Arraylist mylist= new Arraylist()// arraylist(size is flexibke, not strongly type)

Mylist.Add(1)

Mylisy.Add(2)

Mylist.Add(“Bob”);// boxing and unboxing

}

Q. What are generic collection

A: Generic collection is strongly type and flexible. It has better performance as compared with ArrayList

Static void main(string [], maik)

{

List<int> geneint= new List<int>

Geneint.Add(1) // it can just add int type data

}

Q. What are threads(Multithreading)

A: if we want to run code parallelly then we use Threads. We use system.Threading namepace.

Using System.Threading

Static void main(string[],args){

Thread t= new Thread(Method1)

t.start()

Thread t1= new Thread (method2)

T1.start()

Console.read();

}

Static void method1()

{

Console.write(“m1”)

}

Static void method2 ()

{

Console.write(“m2”)

}

Q: How are threads different from TPL(Task Parallel library )

A:

Q: How are exceptions handled in C#.

A: Try catch

Q: Use of finally block

A: that block will run anyhow

Q: What is the use of out keyword in C#

A: If you want to return multiple outputs from the function you will use OUT keyword.

Static void main(string[].args)

{

int add=0;

int sub=0;

myMath(2,3,out add, out sub)

Console.Read();

}

Static in myMath(int num1, int num2, out int add, out int sub )

{

Add=num1+num2

Sub=num1=num2

}

Q. What is the need to Delegates

A: Delegate is a pointer to a function and very useful as callbacks to communicate between threads

Q. What are events

A: Events are encapsulation over delegates(wrapper over deligates)

Q. What are the difference between Abstract class and interface

A: Abstact class is a half defined parent class while interface is contract. Abstract class is inherited while