1.

Managed Code

“The code, which is developed in .NET framework is known as managed code. This code is directly executed by CLR with the help of managed code execution. Any language that is written in .NET Framework is managed code”.

Unmanaged Code

The code, which is developed outside .NET framework is known as unmanaged code.

“Applications that do not run under the control of the CLR are said to be unmanaged, and certain languages such as C++ can be used to write such applications, which, for example, access low - level functions of the operating system. Background compatibility with the code of VB, ASP and COM are examples of unmanaged code”.

Unmanaged code can be unmanaged source code and unmanaged compile code. Unmanaged code is executed with the help of wrapper classes.

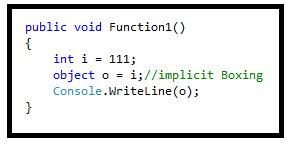
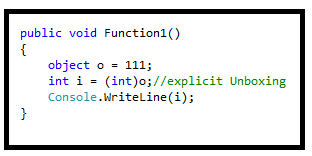
Wrapper classes are of two types:

CCW (COM Callable Wrapper).

RCW (Runtime Callable Wrapper).

Managed code and unmanaged code in .NET

2.

**Answer:** Boxing and Unboxing both are used for type conversion but have some difference:  
  
**Boxing:**  
  
Boxing is the process of converting a value type data type to the object or to any interface data type which is implemented by this value type. When the CLR boxes a value means when CLR is converting a value type to Object Type, it wraps the value inside a System.Object and stores it on the heap area in application domain.   
  
**Example:**  
  
**Unboxing:**  
  
Unboxing is also a process which is used to extract the value type from the object or any implemented interface type. Boxing may be done implicitly, but unboxing have to be explicit by code.   
  
**Example:**  
  
The concept of boxing and unboxing underlines the C# unified view of the type system in which a value of any type can be treated as an object.

3.

**Answer:**

Class and struct both are the user defined data type but have some major difference:  
 **Struct**

* The struct is value type in C# and it inherits from System.Value Type.
* Struct is usually used for smaller amounts of data.
* Struct can’t be inherited to other type.
* A structure can't be abstract.
* No need to create object by new keyword.
* Do not have permission to create any default constructor.

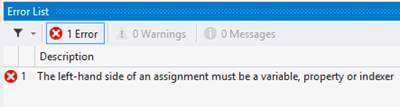
**Class**

* The class is reference type in C# and it inherits from the System.Object Type.
* Classes are usually used for large amounts of data.
* Classes can be inherited to other class.
* A class can be abstract type.
* We can’t use an object of a class with using new keyword.
* We can create a default constructor.

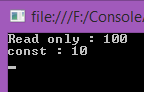
4.

**Answer:**  
  
**Constant** (const) and **Readonly**(readonly) both looks like same as per the uses but they have some differences:   
  
**Constant**is known as “const” keyword in C# which is also known immutable values which are known at compile time and do not change their values at run time like in any function or constructor for the life of application till the application is running.  
  
**Readonly** is known as “readonly” keyword in C# which is also known immutable values and are known at compile and run time and do not change their values at run time like in any function for the life of application till the application is running. You can assay their value by constructor when we call constructor with “new” keyword.  
  
**See the example**  
  
We have a Test Class in which we have two variables one is readonly and another is constant.

1. **class** Test {
2. **readonly** **int** read = 10;
3. **const** **int** cons = 10;
4. **public** Test() {
5. read = 100;
6. cons = 100;
7. }
8. **public** **void** Check() {
9. Console.WriteLine("Read only : {0}", read);
10. Console.WriteLine("const : {0}", cons);
11. }
12. }

Here I was trying to change the value of both the variables in constructor but when I am trying to change the constant it gives an error to change their value in that block which have to call at run time.  
  
  
  
So finally remove that line of code from class and call this Check() function like the following code snippet:

1. **class** Program {
2. **static** **void** Main(**string**[] args) {
3. Test obj = **new** Test();
4. obj.Check();
5. Console.ReadLine();
6. }
7. }
8. **class** Test {
9. **readonly** **int** read = 10;
10. **const** **int** cons = 10;
11. **public** Test() {
12. read = 100;
13. }
14. **public** **void** Check() {
15. Console.WriteLine("Read only : {0}", read);
16. Console.WriteLine("const : {0}", cons);
17. }
18. }

**Output:**

5.

Value types are stored in the Stack whereas reference types stored on heap.

Value types:

int, enum , byte, decimal, double, float, long

Reference Types:

string , class, interface, object

6.

No, because they are not accessible outside the class.

7.

Философский вопрос

8.

In method overriding, we change the method definition in the derived class that changes the method behavior. Method overloading is creating a method with the same name within the same class having different signatures.

9.

Encapsulation is defined 'as the process of enclosing one or more items within a physical or logical package'. Encapsulation, in object oriented programming methodology, prevents access to implementation details.

10.

The three ways to pass a parameter to a method are detailed below.

* **Value Parameters**: Passing a parameter to a method by value creates a new storage location for the value parameter. Any changes to the value parameter by the method have no effect on the argument.
* **Reference Parameters**: Passing a parameter to a method by reference can be achieved by using the ref keyword. Instead of creating a new storage location for the parameter, the method accesses the memory location of the argument and passes it as a parameter. Changes made to the parameter will also affect the argument.
* **Output Parameters**: The out keyword allows a method to return two values from a function. It’s similar to passing a reference parameter, except in this case data is being transferred out of the method.

11.

Странный вопрос)

– In c# a static class cannot implement an interface. When a single instance class needs to implement an interface for some business reason or IoC purposes, you can use the Singleton pattern without a static class.  
– You can clone the object of Singleton but, you can not clone the static class object  
– Singleton object stores in Heap but, static object stores in stack  
– A singleton can be initialized lazily or asynchronously while a static class is generally initialized when it is first loaded

12.

Yes, We can overload main() method. A C# class can have any number of main() methods.  
But to run the C# class, class should have main() method with signature as “public static void main(String[] args)”. If you do any modification to this signature, compilation will be successful. But, You will get run time error as main method not found.

13.

int x = 20;

int y = 20;

Console.WriteLine( x == y);

Console.WriteLine(x.Equals(y));

Output:

**True**

**True**

**For Reference Type**:

**==** performs an identity comparison, i.e. it will only return true if both references point to the same object. While **Equals**() method is expected to perform a value comparison, i.e. it will return true if the references point to objects that are equivalent.

For Example:  
StringBuilder s1 = new StringBuilder(“Yes”);

StringBuilder s2 = new StringBuilder(“Yes”);

Console.WriteLine(s1 == s2);

Console.WriteLine(s1.Equals(s2));

Output:

**False**

**True**

In above example, s1 and s2 are different objects hence “==” returns false, but they are equivalent hence “Equals()” method returns true. Remember there is an exception of this rule, i.e. when you use “==” operator with string class it compares value rather than identity.

**When to use “==” operator and when to use “.Equals()” method?**

For value comparison, with Value Type use “==” operator and use “Equals()” method while performing value comparison with Reference Type.

14.

It’s a way to allow declaring reserved keywords as vars.

Example

void test(int @string)

15.

In C#5 and earlier, you can do it using constructor.

class Employee

{

public Employee()

{

Name = "Default Name";

}

public string Name { get; set; }

}

In C#6.0, You can do:

public string Name { get; set; } = "Default Name"

16.

//Example:

Random r = new Random();

int n = r.Next();

17.

You can catch multiple exceptions using condition statements in C#.

Example:

catch (Exception ex)

{

if (ex is FormatException || ex is OverflowException)

{

testid = "";

return;

}

throw;

}

In C#6.0,

catch (Exception ex) when (ex is FormatException || ex is OverflowException)

{

testid = "";

return;

}

18.

To avoid creating a clone of our singleton class, we can do the following :  
– Implement MethodwiseClone()  
– Override the clone() method and throw CloneNotSupportedException from it.