1. Console.WriteLine()

Console.WriteLine is a useful Console method that renders a line of text on the console. In its simplest form it outputs a string argument with a trailing newline. With no arguments it outputs a blank line.

10. A short list with information about the most popular programming languages. How do they differ from C#?

10.1. Java.

Features of C# absent in Java

- C# includes more primitive types and the functionality to catch arithmetic exceptions.
- Includes a large number of notational conveniences over Java, many of which, such as operator overloading and user-defined casts, are already familiar to the large community of C++ programmers.
- Event handling is a "first class citizen"—it is part of the language itself.
- Allows the definition of "structs", which are similar to classes but may be allocated on the stack (unlike instances of classes in C# and Java).
- C# implements properties as part of the language syntax.
- C# allows switch statements to operate on strings.
- C# allows anonymous methods providing closure functionality.
- C# allows iterator that employs co-routines via a functional-style yield keyword.
- C# has support for output parameters, aiding in the return of multiple values, a feature shared by C++ and SQL.
- C# has the ability to alias namespaces.
- C# has "Explicit Member Implementation" which allows a class to specifically implement methods of an interface, separate from its own class methods. This allows it also to implement two different interfaces which happen to have a method of the same name. The methods of an interface do not need to be public; they can be made to be accessible only via that interface.
- C# provides integration with COM.
- Following the example of C and C++, C# allows call by reference for primitive and reference types.

Features of Java absent in C#

- Java's strictfp keyword guarantees that the result of floating point operations remain the same across platforms.
- Java supports checked exceptions for better enforcement of error trapping and handling.

Philosophical differences between the languages

- There are no unsigned primitive numeric types in Java. While it is universally agreed that mixing signed and unsigned variables in code is bad, Java's lack of support for unsigned numeric types makes it somewhat unsuited for low-level programming.
- C# does not include checked exceptions. Some would argue that checked exceptions are very helpful for good programming practice. Others, including Anders Hejlsberg, chief C# language architect,

argue that they were to some extent an experiment in Java and that they haven't been shown to be worthwhile [1] [2].

- C#'s namespaces are more similar to those in C++. Unlike Java, the namespace does not specify the location of the source file. (Actually, it's not strictly necessary for a Java source file location to mirror its package directory structure.)
- C# includes delegates, whereas Java does not. Some argue that delegates complicate the method invocation model, because they are handled through reflection, which is generally slow. On the other hand, they can simplify the code by removing the need to declare new (possibly anonymous) classes to hook to events.
- Java requires that a source file name must match the only public class inside it, while C# allows multiple public classes in the same file.
- C# allows the use of pointers, which some language designers consider to be unsafe, but certain language features try to ensure this functionality is not misused accidentally. Pointers also greatly complicate technologies such as Java's RMI (Remote Method Invocation), where program objects resident on one computer can be referenced within a program running on an entirely separate computer. Some have speculated that the lack of memory pointers in Java (substituted by the more abstract notion of object references) was a nod towards the coming of grid computing, where a single application may be distributed across many physical pieces of hardware.
- C# supports the goto keyword. This can occasionally be useful, but the use of a more structured method of control flow is usually recommended.
- C# has true multi-dimensional arrays, as well as the array-of-arrays that is available to Java (which C# calls jagged arrays). Multi-dimensional arrays are always rectangular (in the 2D case, or analogous for more dimensions), whereas an array-of-arrays may store rows (again in the 2D case) of various lengths. Rectangular arrays may speed access if memory is a bottleneck (there is only one memory reference instead of two; this benefit is very dependent on cache behavior) while jagged arrays save memory if it's not full but cost (at the penalty of one pointer per row) if it is. Rectangular arrays also obviate the need to allocate memory for each row explicitly.
- Java does not include operator overloading, because abuse of operator overloading can lead to code that is harder to understand and debug. C# allows operator overloading, which, when used carefully, can make code terser and more readable. Java's lack of overloading makes it somewhat unsuited for certain mathematical programs. Conversely, .NET's numeric types do not share a common interface or superclass with add/subtract/etc. methods, restricting the flexibility of numerical libraries.
- Methods in C# are non-virtual by default. In Java however, methods are virtual by default. Virtual methods guarantee that the most overridden method of an object will be called which is determined by

the runtime. You always have to keep that in mind when calling or writing any virtual method! If the method is declared as non-virtual, the method to invoke will be determined by the compiler. This is a major difference of philosophy between the designers of the Java and .NET platforms.
- Java 1.5's generics use type-erasure. Information about the generic types is lost when Java source is compiled to bytecode. .NET 2.0's generics are preserved after compilation due to generics support starting in version 2.0 of the .NET Common Language Runtime, or CLR for short. Java's approach allows Java 1.5 binaries to be run in the 1.4 JRE, at the cost of additional runtime typechecks.
- C# is defined by ECMA and ISO standards, whereas Java is proprietary, though largely controlled through an open community process.
- The C# API is completely controlled by Microsoft, whereas the Java API is managed through an open community process.
- The .NET run-time allows both managed and unmanaged code, enabling certain classes of bugs that do not exist in Java's pure managed code environment but also allows interfacing with existing code.

10.2. C/C++.

- Both C and C++ give a lower level of abstraction that, with increased complexity, provides a breadth of access to underlying machine functionality that are not necessarily exposed with other languages.
- C++ adds the convenience (reduced development time) of a fully object oriented language which can, potentially, add an additional performance cost.
- C# provides a managed memory model that adds a higher level of abstraction. This level of abstraction adds convenience and improves development times, but complicates access to lower level APIs and makes specialized performance requirements problematic.
- It is possible to implement high performance software in a managed memory environment, but awareness of the implications is essential.
- The syntax of C# is less demanding and error prone than that of C/C++ and has, for the initiated programmer, a shallower learning curve.
- In terms of real world applications, these languages can be applied best in the following domains:
  C
  - Kernel level software.
  - Hardware device drivers.
  - Applications where access to old, stable code is required.

  C, C++
  - Application or Server development where memory management needs to be fine-tuned and can't be left to generic garbage collection solutions;

- Development environments that require access to libraries that do not interface well with more modern managed languages.
- Although managed C++ can be used to access the .NET framework, it is not a seamless transition.

C#

- Rapid client application development.
- High performance Server development that benefits from the .NET framework.
- Applications that require the benefits of the .NET framework in the language it was designed for.

10.3. PHP.

PHP

- PHP generates directly the HTML/JS /CSS used to compose the page and produce the end HTML page directly from the code or templates engines. To be able to develop PHP web sites you must know at least HTML.
- By default PHP does not make any separation between looks and logic. The programmer has to either use additional packages or write a template engine. Otherwise it would be hard changing the look (besides changing CSS files).
- As far as software distribution – when giving something written in PHP normally you give out your sources as nothing is compiled. To avoid that special software must be used to encrypt the code.
- PHP is an interpreted language. For most web pages it is fine but it uses a lot of CPU and time if more complex operations are attempted.
- Background tasks on PHP are basically impossible to handle.
- PHP doesn't have any native support for AJAX.
- PHP needs the following minimal tools – a web server with PHP support and a text editor.
- Debugging a PHP application is very hard.

C#

- A page is composed out of "web components" which are aware of their state and can fire events like "button click". Once an event is fired, the C# logic changes the state of the page components or adds new ones on the fly. Components are then invoked by the framework to render themselves. No knowledge of HTML is required.
- The logic code is separated from the presentation code. The presentation code is nearly a pure HTML code with some additional tags for the controls. The logic on the other side is a pure C# file and is compiled to produce a .dll file.
- As the logic is normally compiled, the software can be distributed safely.
- Performances varies between Linux and Windows machines. On Linux the performances are still 10x the one of a PHP equivalent. On Windows C# is nearly as fast as pure C++ code.
- There can be many background threads. They will continue to run in the background even after a page is loaded. You can cache things in

the memory (for example from a database) or you can even communicate between different web sessions (for example to create a chat).

- C# has full AJAX support. That means there can be events which call back the server without reloading the page, timers could pool data at regular intervals and much more.
- C# needs the following minimal tools – a web server with ASPx support, some text editor and a C# compiler. You can use MonoDevelop witch is free and open source or Visual Studio witch is much better but is not free other than the Express Edition.
- Debugging a C# application is very easy. Inside Visual Studio (as well as MonoDevelop) there is a full debugger.

11. The difference between C# and .NET Framework

C# is a programming language, whereas .NET covers both the .NET Framework (an application framework library) and the Common Language Runtime which is the runtime in which .NET assemblies are run.

Microsoft's implementation of C# is heavily integrated with the .NET Framework so it is very difficult to use it without using the .NET Framework. However it is important to understand that they are two very different things. There are other implementations other than Microsoft’s of C# available.

The knowledge of C# implies some knowledge of .NET because the C# object model corresponds to the .NET object model and you can do anything interesting in C# just by using .NET libraries. The opposite isn't necessarily true as you can use other languages to write .NET applications.

The distinction between a language, runtime and a library is stricter in .NET/C# than for example in C++, where the language specification also includes some basic library functions. The C# specification says very little about the environment (basically, that it should contain some types such as *int*, but that's more or less all).