C# VS JAVA

Assignment # 2

ABSTRACT

C# is used primarily on the .NET Framework, Mono, and Portable.NET implementations of the CLI. If your software or web application is being built for Windows, C# will work best with the .NET suite of technologies. If are to develop for Unix, Linux, or other platforms outside of the Microsoft platform, Java's large open-source ecosystem is the better choice.

COURSE

Java | Object Oriented programming

SUBMITTED BY

Usama Sarwar | CIIT/FA17-BS(CS)-090/SWL

SUBMITTED TO

Mr. Shehzad

DATED

April 9, 2018

COMSATS Institute of Information Technology

Sahiwal Campus

C#

C# is a general-purpose programming language that first appeared in 2000, as part of Microsoft's .NET initiative. It was designed for the Common Language Infrastructure (CLI)—an open specification developed by Microsoft and standardized by ISO and ECMA. C# applications are compiled into bytecode that can run on implementations of the CLI.

JAVA

Initially released by Sun Microsystems in 1995, Java is a general-purpose programming language that was designed with the specific goal of allowing developers to "write once, run anywhere." Java applications are compiled into bytecode that can run on implementations of the Java Virtual Machine (JVM). Like CLI, JVM helps bridge the gap between source code and the 1s and 0s that the computer understands.

C# VS. JAVA: MAJOR SIMILARITIES

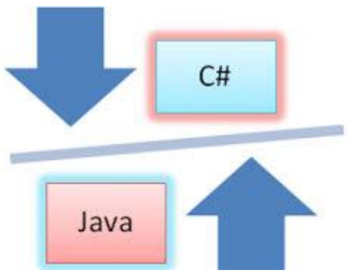
The origins of both Java and C# are closely tied to the transition from lower-level programming languages like C++ to higher-level programming languages that compile into bytecode that can be run on a virtual machine. This comes with a number of benefits, most notably the ability to write human readable code once that can run on any hardware architecture that has the virtual machine installed on it. Syntactic quirks aside, it's not surprising how similar these two languages are from the top-level perspective of an application developer. Here are a few of the main similarities between C# and Java:

- **Type-Safe.** A type error occurs when the data type of one object is mistakenly assigned to another object, creating unintended side effects. Both C# and Java make a good effort to ensure that illegal casts will be caught at compile time and exceptions will be thrown out at runtime if a cast cannot be cast to a new type.

- **Garbage Collection.** In lower-level languages, memory management can be tedious because you have to remember to properly delete new objects to free up resources. That's not the case in C# and Java, where built-in garbage collection helps prevent memory leaks by removing objects that are no longer being used by the application. While memory leaks can still occur, the basics of memory management have already been taken care of for you.

- **Single Inheritance.** Both C# and Java support single inheritance—meaning only one path exists from any base class to any of its derived classes. This limits unintended side effects that can occur when multiple paths exist between multiple base classes and derived classes. The diamond pattern is a textbook example of this problem.

- **Interfaces.** An interface is an abstract class where all methods are abstract. An abstract method is one that is declared but does not contain the details of its implementation. The code governing any methods or properties defined by the interface must be supplied by the class that implements it. This helps avoid the ambiguity of the diamond pattern, because it's always clear which base class is implementing a given derived class during runtime. The result is the clean, linear class hierarchy of single inheritance combined with some of the versatility of multiple inheritance. In fact, using abstract classes is one-way multiple inheritance languages can overcome the diamond pattern.



C# VS. JAVA: MAJOR DIFFERENCES

As similar as the two languages are in terms of purpose, it's important to remember that C# holds its origins in Microsoft's desire to have a proprietary "Java-like" language of their own for the .NET framework. Since C# wasn't created in a vacuum, new features were added and tweaked to solve issues Microsoft developers ran into when they initially tried to base their platform on Visual J++. At the same time, Java's open-source community continued to grow, and a technical arms race developed between the two languages. These are some of the major differences between C# and Java.

- **Windows vs. Open-Source.** While open-source implementations exist, C# is mostly used to develop for Microsoft platforms—the .NET Framework's CLR being the most widely used implementation of the CLI. On the other end of the spectrum, Java has a huge open-source ecosystem and gained a second wind in spite of its age, thanks in part to Google adopting the JVM for Android.

- **Support for Generics.** Generics improve compiler-assisted checking of types largely by removing casts from source code. In Java, generics are implemented using erasures. Generic type parameters are "erased" and casts are added upon compilation into bytecode. C# takes generics even further by integrating it into the CLI and allowing type information to be available at runtime, yielding a slight performance gain.

- **Support for Delegates (Pointers).** C# has delegates which essentially serve as methods that can be called without knowledge of the target object. To achieve the same functionality in Java, you need to use an interface with a single method or some other workaround that may require a nontrivial amount of additional code, depending on the application.

- **Checked Exceptions.** Java distinguishes between two types of exceptions—checked and unchecked. C# chose a more minimalist approach by only having one type of exception. While the ability to catch exceptions can be useful, it can also have an adverse effect on scalability and version control.

- **Polymorphism:** C# and Java take very different approaches to polymorphism. While Java enables polymorphism by default, C# must invoke the "virtual" keyword in a base class and the "override" keyword in a derived class.

- **Enumerations (Enums):** In C#, enums are simple lists of named constants where the underlying type must be integral. Java takes the enum further by treating it as a named instance of a type, making it easier to add custom behavior to individual enums.

C# OR JAVA | SUMMARY

With so much in common, the language you ultimately choose to use will depend largely upon the platform you have chosen for your project. Today, C# is used primarily on the .NET Framework, Mono, and Portable.NET implementations of the CLI. If your software or web application is being built for Windows, C# will work best with the .NET suite of technologies.

That said, if you wish to develop for Unix, Linux, or other platforms outside of the Microsoft platform, Java's large open-source ecosystem is the better choice. The community is constantly creating new libraries and tools. New powerful languages like Scala, Clojure, and Groovy have also appeared, all based off of the JVM. It also doesn't hurt that most JVM implementations are open-source and free. Java is also the main language used by Google to develop for Android, which is currently the largest mobile operating system in the world.

Keep in mind that the advantages listed above are slight, and neither language is likely to disappear anytime soon. Both languages have been around long enough that there's not really anything you can't build in one that you couldn't build in the other. Bottom line: Choose the language that works best for your project's platform of choice.