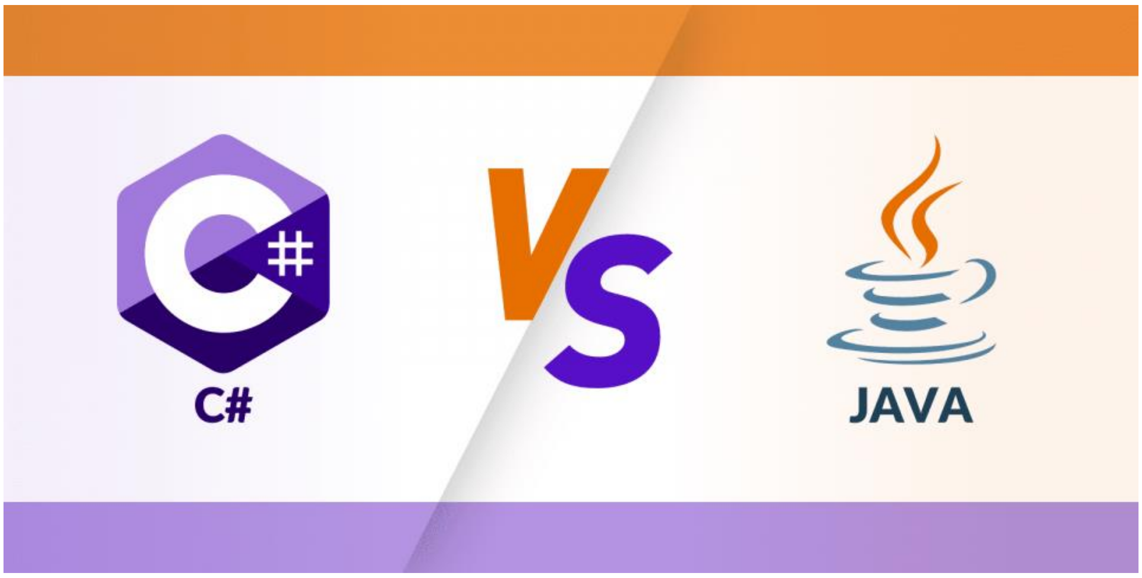

# MIGRATING FROM C# TO JAVA

## Content

# 1. What is C#?

Developed by Microsoft around 2000, as its .NET initiative, C# is a multi-paradigm programming language. It offers programming disciplines like strongly typed, lexically, scoped, generic, object-oriented, and component-oriented. C# is one of the programming languages that is designed for the CLI (Common Language Infrastructure).

The language was soon declared as an international languages standard by ECMA and ISO with the following design goals:

- The language is intended to be a modern, simple, object-oriented programming language.
- C# implementations provide support for software engineering principles. Principles such as array bounds checking, garbage collection, and more.
- The language offers portability and thus can be executed on different platforms.
- C# provides support for internationalization.
- Applications for both embedded systems and OS are supported by C#

## Features of C#

- It was derived from C++ and Java.
- It is part of Microsoft Visual Studio
- **Simple**
    - No pointers
    - No operators like "::" or "->"
    - Varying ranges of primitive types.
    - Memory management and garbage collection is automatic and does not need explicit code.
- **Modern**
    - Built according to the current trend.
    - Powerful, scalable, robust.
    - Any component can be turned into a web service with built-in support.
- **Object-Oriented**
    - Supports OOP concepts such as inheritance, encapsulation, polymorphism, interfaces, operator overloading, etc.
    - C# introduces structures enabling primitive types to become objects.
- **Type-Safe**
    - It does not allow us to perform unsafe casts.

    - Reference types are initialized to null and value types are initialized to zero by the compiler automatically.
    - Arrays are bound checked and are zero-based indexed.
- **Interoperability**
    - Includes native support for the COM and windows based applications.
    - Components from VB NET can directly be used in C#.

## 2. What is JAVA?

Developed by Sun Microsystems and designed by James Gosling in 1995, Java is a class-based, object-oriented multi-paradigm language. It is designed with the WORA concept i.e. Write Once Run Anywhere and thus is called platform-independent language. Applications in Java are compiled to bytecode that can run execute on any JVM regardless of its underlying OS and architecture. It offers its applications in web development and other Android-based software for several devices. High-level applications of Java include embedded systems, desktop, and applications. Devices such as smartphones, ATMs, home security systems and more are all supported by Java.

Primary goals of Java:

- It must be simple and object-oriented.
- It must be robust and secure.
- It must execute the high performance.
- It must be interpreted, threaded, and dynamic.

### Features of Java

- **Simple:** It has a simple syntax and is easy to learn.
- **Object-Oriented:** It follows OOPs concepts such as inheritance, abstraction, polymorphism, encapsulation.
- **Robust:** Java offers Memory Management and mishandled Exceptions by automatic Garbage Collector and Exception Handling.
- **Platform Independent**: It follows its WORA functionality by running its applications on any platform with JRE's help (Java Runtime Environment)
- **Secure**, Java's stable features enable us to develop a virus-free, temper free system.
- **Multithreading**: Java multithreading designs a program to do several tasks simultaneously. Thus, it utilizes the same memory and other resources to execute multiple threads at the same time.

- **Architectural Neutral**: The compiler generates bytecodes, which is independent of computer architecture.
- **High Performance**: Java enables high performance with the use of a just-in-time compiler.
- **Distributed:** Programs can be designed to run on computer networks.

## 3. Advantages and disadvantages

### Advantages of C#

- C# provides lambda and generics support.
- Language-Integrated Query (LINQ).
- The language has secure extension methods.
- Properties with getting/set methods.
- Memory management.
- Best in class cross-platform support.
- Backward compatibility.
- Operator overloading support

### Disadvantages of C#

- Poor GUI.
- The application must be windows based as C# is an internal part of the .NET framework.
- Software is proprietary, so it requires an upfront purchase.
- C# mostly depends on the .Net framework, and so is less flexible.
- C# executes slowly, and the program needs to be compiled each time when any changes are made.

### Advantages of Java

- It provides detailed documentation.
- The large pool of skilled developers available.
- Java allows developing standard programs and reusable code.
- It offers a multi-threaded environment.
- Excellent and high performance.
- A huge array of 3rd party libraries.
- Easy to navigate libraries.

### Disadvantages of Java

- JIT compiler makes the program low.
- The hardware of maintaining Java programs is high, as Java demands high memory and processing requirements.
- The language does not provide any support for low-level programming constructs like pointers or operator overloading.
- The user has no control over garbage collection as Java does not provide functions like delete(), free().

## 4. Key differences between C# and JAVA

### 4.1. Main Function

**C#:**

```
static void Main(string[] args)
```

`string` is shorthand for the `System.String` class in C#. Another interesting point is that in C#, your Main method can actually be declared to be parameter-less

```
static void Main()
```

**Java:**

```
public static void main(String[] args)
```

### 4.2. Print statements

**C#:**

`System.Console.WriteLine("Hello world!");` or

```
Console.WriteLine("Hello again!");
```

**Java:**

```
System.out.println("Hello world!");
```

## 4.3. Constants

### C#:

To declare constants in C# the `const` keyword is used for compile time constants while the `readonly` keyword is used for runtime constants. The semantics of constant primitives and object references in C# is the same as in Java.

```
const int K = 100;
```

### Java:

In Java, compile-time constant values are declared inside a class using the keyword `final`:

```
static final int K = 100;
```

## 4.4. Inheritance

C# uses C++ syntax for inheritance, both for class inheritance and interface implementation as opposed to the `extends` and `implements` keywords.

### C#:

```
class B:A, IComparable
{
        ............
}
```

### Java:

```
class B extends A implements IComparable
{
        ...............
}
```

## 4.5. Polymorphism & Overriding / Overloading

The means of implementing polymorphism typically involves having methods in a base class that may be overridden by derived classes. These methods can be invoked even though the client has a reference to a base class type which points to an object of the derived class. Such methods are bound at runtime instead of being bound during compilation and are typically called virtual methods.

In Java all methods are virtual methods while in C#, as in C++, one must explicitly state which methods one wants to be virtual since by default they are not.

To mark a method as virtual in C#, one uses the `virtual` keyword. Also, implementers of a child class can decide to either explicitly override the virtual method by using the `override` keyword or explicitly choose not to by using the `new` keyword instead

Example:

```
using System;

public class Parent {
      public virtual void DoStuff(string str) {
          Console.WriteLine("In Parent.DoStuff: " + str);
      }
}
public class Child: Parent {
      public void DoStuff(int n) {
            Console.WriteLine("In Child.DoStuff: " + n);
      }
      public override void DoStuff(string str) {
             Console.WriteLine("In Child.DoStuff: " + str);
      }
 }
```

```
public new void DoStuff(string str)
```

To call a base class method, you use the keyword `base` in C# and `super` in Java.

## 4.6. Multiple Classes in a Single File

Multiple classes can be defined in a single file in both languages with some significant differences.

In Java, there can only be one class per source file that has public access and it must have the same name as the source file.

C# does not have a restriction on the number of public classes that can exist in a source file and neither is there a requirement for the name of any of the classes in the file to match that of the source file.

## 4.7. Primitive Types

In Java, all integer primitive types (byte, short, int, long) are signed by default. In C# there are both signed and unsigned varieties of these types:

| **Unsigned** | **Signed** | **Size** |
|---|---|---|
| `byte` | `sbyte` | 8 bits |
| `ushort` | `short` | 16 bits |
| `uint` | `int` | 32 bits |
| `ulong` | `long` | 64 bits |

The only significantly different primitive in C# is the `decimal` type, a type which stores decimal numbers without rounding errors. Eg:

```
decimal dec = 100.44m;
```

## 4.8. Strings

Java does not allow operator overloading. Thus, the following valid sentence in C#:

```
String myString;

if ( myString == “value” )
```

It’s not valid in Java. Instead, we have to use the equals method:

```
if ( myString.equals( “value” ) )
```

## 4.9. Array Declaration

Java has two ways in which one can declare an array:

```
int[] iArray = new int[100]; //valid

float fArray[] = new float[100]; //valid
```

C# uses only the latter array declaration syntax:

```
int[] iArray = new int[100]; //valid

float fArray[] = new float[100]; //ERROR: Won't compile
```

## 4.10. Switch Statements

There are two major differences between the switch statement in C# versus that in Java. In C#, switch statements support the use of string literals and do not allow fall-through unless the label contains no statements.

```
 switch(foo) {

   case "A": Console.WriteLine("A seen");
             break;
   case "B": case "C":
             Console.WriteLine("B or C seen");
             break;
 /* ERROR: Won't compile due to fall-through at case "D" */
   case "D": Console.WriteLine("D seen");
   case "E": Console.WriteLine("E seen");
             break;

}
```

## 4.11. Importing Libraries

Both languages support this functionality and C# follows Java's technique for importing libraries:

### C#: `using` keyword

```
using System;

using System.IO;

using System.Reflection;
```

### Java: `import` keyword

```
import java.util.*;

import java.io.*;
```

## 4.12. Properties

Properties are a way to abstract away from directly accessing the members of a class, similar to how accessors (getters) and modifiers (setters) are used in Java.

Particularly for read/write properties, C# provides a cleaner way of handling this concept. The relationship between a get and set method is inherent in C#, while has to be maintained in Java.

It is possible to create, read-only, write-only or read-write properties depending on if the getter and setter are implemented or not.

**C#:**

```
public int Size {
   get { return size; }
   set { size = value; }
}
```

OR

```
public int Size {
   get -> size;
   set -> size = value;
}
```

**Java:**

```
public int getSize() {
   return size;
}
public void setSize ( int value ) {
   size = value;
}
```

## 4.13. Pass by Reference

In Java the arguments to a method are passed by value meaning that a method operates on copies of the items passed to it instead of on the actual items.

In C#, it is possible to specify that the arguments to a method actually be references.

In Java trying to return multiple values from a method is not supported.

## 4.14. Exception handling

Both C# and Java include an exception handling mechanism, but while C# only handles unchecked exceptions, JAVA supports both checked and unchecked. You can learn more about the JAVA handling mechanism here:

https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/

### 4.15. Collections

While both C# and Java offer a wide range of collections classes, C# collections (like `Sets` or `Lists`) can be directly instantiated while Java are abstract interfaces and should be used through any of the underlying implementations. For example, to use a `Set` in Java, we can use a `HashSet` as underlying implementation:

```
Set<String> mySet = new HashSet();
```

## 5. Activities

1. Download the .cs files (C#) in the platform . You should translate them to English (that's to say, name of methods/properties/members/variables, and then convert them into JAVA format (the easiest way to do that is to paste them into a new project/file in IntelliJ IDEA and compile to detect the differences between C# and JAVA.
2. In exercise 1, add two more classes: AlumnoFCT and AlumnoErasmus. Both of them must inherit from Alumno. AlumnoFCT will have three specific attributes: empresa, tutor e instructor (the three of them are strings). AlumnoErasmus will have the starting and ending date of the internship, and the origin country.

## 6. Bibliography

https://en.wikipedia.org/wiki/Comparison_of_C_Sharp_and_Java

https://www.janbasktraining.com/blog/java-vs-c-sharp/

https://hackr.io/blog/c-sharp-vs-java

https://www.educba.com/java-vs-c-sharp/

https://www.geeksforgeeks.org/java-vs-c-sharp/