

/technology

Java; what's next?

>>the wonderful world of programming languages

Programming languages are programmers' essential tools. These languages are undergoing constant evolution. They have to continually improve in order to allow programs to be written more quickly and efficiently and to allow mistakes to be detected beforehand. However, is every new language an improvement? And when should you switch to a new programming language?

Every year, processors get considerably faster. Faster computers enable software to do more for us, and programming languages provide the tools to create this software. These languages are evolving all the time, and the rapid flow of new programming languages allows programmers to work more efficiently. New computing languages are not created out of thin air. Although they often contain new elements, they always use tried and tested

elements of previous languages.

If the languages are somewhat similar, then the learning process is made simpler. However, if they are practically alike, no progress is made. This conundrum forces the users of these languages to make strategic choices, i.e. do we keep using the current language or do we switch to a new language? A new language may offer more efficient working methods, but it takes effort to learn and you have to hope that the rest of the world follows your lead.

Evolution

Programming languages have been changing since the very first language was developed. The first languages were linear. In the Basic dialects, you programmed on a line-by-line basis, with loops and jumps to other parts of the code. Subsequently, more user-friendly languages were created such as Pascal and C. Languages of this type are called imperative or procedural languages. Within the academic world, there is another trend: functional languages such as Lisp and Haskell. The key focus of these languages is functionality. However, these functional languages have never been very popular in the industrial sector.

One of the most major innovations was the step from imperative languages to object-oriented (OO) languages. Smalltalk is the

Functional Programming	Haskell / Clojure / F#	
	Scala	Ruby Python
Object Oriented Programming	C#	
	Java	Smalltalk
	Statically Typed	Dynamically Typed



```

switch(Label)
{
    case eNodeLabel.Amount:
        g.FillRectangle(BrushGray, new Rectangle(RectPercent.Width * Bau..
        string strperc = Bau.HybridAnzahl * 100.05;
        g.DrawString(strperc, MainFont, BrushBlack, RectPercent.Width * 0.5,
        break;
    case eNodeLabel.Amount:
        g.FillRectangle(BrushGray, RectPercent);
        string stram = Bau.Kombiniert.ToString();

        foreach(int hyb in Bau.HybridAnzahl)
            stram += (" "+hyb.ToString());

        g.DrawString(stram, MainFont, BrushBlack, RectPercent.Width * 0.5,
        break;
    case eNodeLabel.None:
    default:
        g.FillRectangle(BrushGray, RectPercent);
        break;
}

```

mother of the OO languages, although it was C++ that made this trend famous. Java is an language that is now extremely widely used, although the language alone is part of a greater whole: the Java platform. This is a trinity of the Java language, the Java Virtual Machine and the Java libraries.

Java language

The origins of Java date back to the 1990s, when hardware performance was a major issue. In order to get the very best performance whilst simultaneously making the step to the (then) new language as straightforward as possible, some concessions were made when developing the languages. Java is therefore not a pure, object-oriented language like Smalltalk: it is not all one entity. This results in ambiguity and exceptions in the language.

Java Virtual Machine (JVM)

The application of a virtual machine is one of the Java platform's most important innovations. During the creation of C, program code is generated in machine code for a specific processor. In order to be used by another processor, the C program has to be recompiled for this new processor. When compiling Java, the program is generated in bytecode. Bytecode is not run directly by a processor, but by a software layer: the JVM. The JVM is typically written in C and compiled for a specific processor. This means that as long as a JVM is available, a Java program can be run on any system (hence the slogan 'write once, run anywhere'). Furthermore, the JVM is an ideal processor that greatly simplifies programming. Important contributing factors to this include straightforward memory management and the ability to remove many types of errors by means of stringent checks during compilation and running.anywhere). Daarnaast doet de JVM zich voor als een ideale processor en vereenvoudigt het programmeerwerk.



Scala:

```
case class Clock (var hour:
Int, var min: Int);
```

Java:

```
public class Clock {
    private int hour;
    private int min;

    public Clock(int hour,
int min) {
        this.hour = hour;
        this.min = min;
    }

    public int getHour() {
        return hour;
    }

    public void setHour(int
hour) {
        this.hour = hour;
    }

    public int getMin() {
        return min;
    }

    public void setMin(int
min) {
        this.min = min;
    }

    public String toString() {
        ...
    }

    public boolean
equals(Object arg1) {
```

Java libraries

For the sake of productivity, it is also important that libraries are available. Libraries contain existing codes written by others. Popular languages may feature countless megabytes of existing code, to which you need add just a couple of lines to create, for example, a website. The library is therefore an important factor in deciding which programming language to use. Java has been around for over ten years, and in this time, a great number of libraries have been developed for Java. The Java code is mainly freely available from these libraries (open source).

>>every language has its own life cycle

Development environment

In addition to the Java platform, the development environment is also vital to productivity. They are known as 'Integrated Development Environments' (or IDEs). For Java, a number of these are available, with Eclipse being the most well-known. The IDE gives the programmer a great deal of support during programming. For example, it checks that the code is correct during input, and contains in-built debugging tools and links to libraries.

Law of the handicap of a head start

The degree of innovation in a programming language is inversely proportional to the age of the language and the number of users. Software products written in a language that is already in use require backwards compatibility. This is an aspect that former Java owner Sun has always paid a great deal of attention to. However, this slows the momentum of the innovation, eventually to the point that it stops altogether. When this occurs, it is time to look for a new language. This life cycle applies to all programming languages.

What's next?

The last few years have seen a constant flow of new programming languages. A number of these languages build on what was already available, and take advantage of the JVM and its associated benefits. Examples of this are Ruby, Python and Groovy, which in general create much more compact code than Java. Furthermore, innovations in these languages are implemented like closures, with functions being write out where necessary. For minor functions, this saves a great deal of typing. For years now, the Java community has been gathering information on how to incorporate this into the structure of Java.

An important disadvantage of the above-mentioned new languages is that they are all based on 'dynamic typing'. This means that errors are only discovered when the program is run. These languages are therefore not suitable for creating complex programs in large teams. It is therefore important that your compiler provides optimal assistance in detecting possible errors in the program before it is run.

Scala

One exception is the new language Scala. The person behind this language is Martin Odersky, professor at the University of Lausanne (EPFL) and an important figure in the development of Generics, one of the latest Java innovations. Scala programs are run on the JVM and can make use of the existing Java libraries. Scala is a programming language that is relatively high in innovation. It is a hybrid language that combines both OO and functional concepts. This method has now become a trend, with even Microsoft making use of it in the .NET platform by means of LINQ. Without going into too much detail, LINQ relies quite heavily on the style of the functional languages.

A further important factor is that Scala makes use of static typing. This means that the compiler is particularly strict, which helps to prevent a large number of errors. One noticeable innovation is 'type inference'. This means that programmers do not always have to explicitly repeat the types of variables and parameters, and as a result, the code is more compact. If possible, the compiler itself should specify the types. The scientific idea behind the determination of the types by the compiler is also based on the functional style. Traits are another innovation. These are Java interfaces in which implementation (fields and code) is permitted. Traits can help eradicate restrictions of single inheritance.

The Scala community is growing rapidly and has developed an IDE for Eclipse that is already working relatively well. Scala therefore has all of the ingredients to be a potential successor to Java...or does it?

Barrier

In general, Scala would seem to have all the right stuff. However, it is not quite ready to take over the world. For most users, the language barrier is too high as the many innovations that the Scala language incorporates means that it differs greatly from Java. This is often the case during the transfer to a new language. Investment is required before the benefits of improved productivity can be realised. The IDE also works adequately, but it is not yet as user-friendly as Java.

Furthermore, the Java community seems to be burying its heads in the sand somewhat, and would rather spend another year discussing new improvements to the language. However, it is likely that this has more to do with political matters rather than innovations in the language.

As long as major businesses do not adopt the new language, Scala will remain a niche product, although a business such as Google could give Scala the extra weight needed to tip the balance. The search giant has often shown that it has the courage to try something new. Scala has already been accepted in the Google Summer of Code programme, which sponsors university students to work on promising projects.

It is essential that Technolution continues to monitor the field of programming languages, as it allows us to separate the wheat from the chaff and to always know which horse to bet on and when. Only one thing is certain: one day, Java will once again be just the name of a coffee-producing island in the Indian Ocean...

