What Happened To Java (Specifically The Language)? [closed]

Asked 16 years, 1 month ago Modified 11 years, 1 month ago

Viewed 2k times

13

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 11 years ago.

Back in 2000 (when .NET was unleashed upon us IIRC) it was an innovative cutting edge language (last time I used it was 2003).

From what I read though, Sun has only evolved the language exceedingly slowly. I've even read that where the language has evolved, such as the addition of Generics, developers have complained about poor implementation.

Are these accurate perceptions, and if so, any thoughts on why, particularly with the seemingly obvious

competition from C#?

c# java sun

Share

Improve this question

Follow

edited Nov 2, 2013 at 18:05

Gabriele Petronella

108k ● 21 ● 221 ● 236

asked Nov 20, 2008 at 17:50

Ben Aston

55.6k ● 69 ● 219 ● 344

I will answer with another question. Why does Microsoft, putting so much effort into "fancy language" features like linq, expression trees and all that stuff, why are they just ignoring the experience from Java world? 5 years ago people in Java did Hibernate... – Paul Kapustin Nov 20, 2008 at 17:57

now they came out with Linq to entities which is something not too very exciting to be honest.... They are so way behind in some other things... – Paul Kapustin Nov 20, 2008 at 17:57

4 The community can do libraries - eg NHibernate. It makes sense for MS to do 'core' work such as language and runtime improvements. C# 3 is *much* nicer to use than Java 6 - and much nicer than C# 1. – Jon Skeet Nov 20, 2008 at 18:11

1 badbadboy: You do know that hibernate has been ported to .NET, right? And LINQ, lambda expressions, proper generics and other .NET features are pretty cool, whether you'll admit it or not. I'd like you to point me to Java equivalents. – Stack Overflow is garbage Nov 21, 2008 at 8:55

7 Answers

Sorted by: Highest score (default)

21

From an enterprise point of view, evolving a language is not a good thing, it's actually pretty bad.

That's why you hear older languages like cobol, fortran and even C written with year numbers after them--many enterprises stick with that year.

On top of that, larger teams means more of a chance that someone in your team will do stuff others don't understand, so there is an important but under-rated value in keeping a language simple and clean. This means not adding too many alternative ways to do things.

I worked with Ruby and had a ball with the language, but looking at it from an enterprise point of view, it was an absolutely terrifying language. I couldn't count the ways a bad programmer could mess up a large team, forcing them to spend days untangling a mess created in minutes.

There are companies who refuse to go to java 5.0 because of the complexity of generics. (We're still working on 1.3x but that's for another reason).

And honestly, most "Improvements" buy you very little. Some syntax change, the ability to remove a few levels of braces.

I can't think of a single case where Java has forced me to repeat business logic (Which is what I worry about when

I'm trying to make my code "DRY")--it's a clean enough language to be completely DRY if you're a good programmer.

For instance, anything you can do with a closure you can do with a subclass without repeating business logic--what you end up with looks worse because of layers of braces/extra class definitions, but is often more reusable (You can extend the class you use to implement your callback, but you can't extend a closure method, you have to rewrite it.)

I didn't feel this way about code for the first few decades of my career (I LOVE language tricks, the funkier the better), but now I've been at this a long time--It could be my old age sneaking in, or it could be experience, but now I see huge benefits to simple, explicit, stable code (offered by a language that won't let you play tricks) and can't really find a single advantage to many alternative methods, even if they save a line or two of typing.

If you are looking for a java upgrade, however, look at Scala. It's pretty freaking amazing, still runs on the JVM, interacts with Java, etc.

edited Nov 20, 2008 at 18:27

answered Nov 20, 2008 at 18:10

Bill K
62.8k • 18 • 112 • 158

5 The benefits of most language improvements can be summed up in a single word: readability. Compare the readability of filtering, sorting and projecting a collection using LINQ to Objects with the equivalent code in Java...
– Jon Skeet Nov 20, 2008 at 19:48

1 Possibly, but looking at LINQ it seems to be way behind Active Record--MUCH less readable/usable. Does that make it wrong? I'm not sure how it compares to Java+annotations+Hibernate. – Bill K Nov 20, 2008 at 20:01

Also, a good programmer should be able to express himself readably in any decent language. In Java I notice it takes a bit of programming talent to do so, in Ruby much less.
– Bill K Nov 20, 2008 at 20:03

3 @Bill K - Linq has nothing do do with persistence. It can be used for that, but it can also be used on plain old collections of normal objects without going near a database.
– Greg Beech Nov 20, 2008 at 21:54

3 @Bill: I think it would be worth waiting until you *have* used/understood LINQ before saying what's possible in other languages. In particular, you're still talking about databases, when LINQ gives much more than that. There are things I've done easily in LINQ which would just be painful in Java.
– Jon Skeet Nov 21, 2008 at 7:32

20

Most languages have one strong hand involved in their origin and evolution. Think: Larry Wall/Perl, Guido/Python, Matz/Ruby, Odersky/Scala, Hickey/Clojure, etc. These guys are all brilliant language dudes. I'd give my left arm to be half as smart as any of them.

Java has actually had the distinction of having not just one but a series of amazing language guys at the helm - starting with Gosling, but I also think of Guy Steele, Bill Joy, Gilad Bracha, Neal Gafter, etc - all amazing guys. That actually has been a good thing (I think). It's made the language better but prevented stagnation.

But for the last couple years, there's been a real vacuum of language leadership. At the moment, no one's minding the store. No one's making the hard decisions about what fits with the Java mold and makes sense to add (or more importantly to not add). I don't know what that means. I'm hopeful that the enormous popularity and reach of Java and the strong base of the JVM mean that this vacuum is too attractive not to be filled and given direction at some point. But I'm only cautiously hopeful because I don't know who that will be.

John Rose is that dude on the JVM side. Though if I can only get innovation in one or the other, I'd take JVM right now anyways. :)

edited Nov 20, 2008 at 21:46

answered Nov 20, 2008 at 18:14

Alex Miller
70.1k • 25 • 124 • 168

Alex, Alex, Alex Terrocotta is perfect, unique, state of art technology. I wish we can hava .net version, too. Sometimes, I'm thinking of the possibility to beat the performance of

terracotta integrated with hibernate combo in .net, but it's not possible. – sirmak Nov 20, 2009 at 22:10

I'd like to note (3 years later) that Brian Goetz has really emerged as the new guiding hand in how Java the language will evolve and imho that's the best possible thing that could have happened. – Alex Miller Nov 21, 2011 at 15:51

11

Java has certainly been evolving very slowly - especially if you compare it with C# and VB. I personally feel that they made the wrong decision with generics in terms of keeping backward compatibility at the cost of execution-time safety and efficiency. The .NET approach works a lot better in *almost* every way, IMO.

Java 7 has a long list of *potential* features - both language and platform - but it's been an awfully long time in the making, and there's still significant question marks over many of the features.

I wouldn't like to place any "blame" on why this has happened, however.

Share Improve this answer

Follow

answered Nov 20, 2008 at 17:58

Jon Skeet

1.5m ● 889 ● 9.3k ● 9.3k

Thanks for the plug on the feature list. I also maintain a Java 7 link blog at java7.tumblr.com if you're interested in tracking things that way. Everything on the link blog ends up on that feature page (eventually). – Alex Miller Nov 20, 2008 at 18:15

More than two years later and the current version of Java still is "Version 6 Update 24". – tiago2014 Mar 9, 2011 at 0:34

6

The evolution of the Java language has been slow, but deliberately so.

Generics provides a good example. Compatibility with prior versions of Java was a requirement. *Given the goals of the project*, the generics implementation performs a very useful function exactly as designed. However, it doesn't meet the expectations of many developers that expected the behavior of reified generics.

Innovation in the JVM, on the other hand, has been extremely rapid, often leading the way for other VMs, and promoting competition in the performance arena.

In my opinion, the Java language should be as stable as possible. I like the idea of closures, but I don't believe Java is the language for them. (If something must go in, I'd prefer the conservative FCM.) I work with a team of developers that need training, to build and maintain a complex production application. The Java language gives us a nice blend of power and structure as it is.

Other languages, like Scala and Groovy, and ports to the JVM like Ruby and Python, will continue to give life to the Java platform, even after the Java language itself has gone the way of COBOL.

Share Improve this answer

Follow

answered Nov 20, 2008 at 18:05

erickson

269k ● 59 ● 401 ● 497

3

Java has different problems then .Net at the moment, resulting in different choices.

.Net, being relatively new and with a chance to avoid some of Java's mistakes has the chance to do thing differently. This gives it two main advantages:

1. MS was able to more clearly distinguish between run time platforms. C# 3.0 will clearly not run on the 1.1 framework, and C# 1.1 won't run on the 3.0 framework. Of course, there is some fuzziness there, but in general you have a better idea where you stand. On the other hand, the client JVM auto-updates by default on windows. So it can be much trickier keeping an old system working.
2. The .Net framework, being younger, has accumulated much less cruft and internal complexity. Those things will kill the speed with which you can add new features. We're now starting to see some of this in .Net also. For example: there are a number of BCL functions that require you to pass in or return an array that should instead use an IEnumerable. The history of those functions make it nearly impossible to ever change them.

Those two things conspire together to make it possible (at the moment) for the the .Net languages to advance more

quickly. However, like I've already said we're starting to see these effects catch up with .Net as well.

Share Improve this answer Follow

edited Nov 21, 2008 at 21:53

answered Nov 20, 2008 at 18:48

Joel Coehoorn

415k ● 114 ● 577 ● 813

C# 1.1 code *will* run on the 3.0 framework. While the compatibility isn't perfect, it's pretty good. If you've got a program built with .NET 1.1, chances are pretty good that it'll run if you've got .NET 3.0 installed. – Jon Skeet Nov 20, 2008 at 19:49

hence the 'fuzziness' remark :) – Joel Coehoorn Nov 21, 2008 at 21:54

It is also worth noting you can have multiple versions of .NET installed and you can bind to specific versions. Practically old systems shouldnt stop working when updates occur. – Andrew T Finnell Mar 27, 2011 at 18:50

1

I think that Sun are now being conservative and ensuring they make the right decisions after making a few poor ones.

In addition there was a lot of politicking and messing around before Java 1.5 came out. Many companies were using third party hacks like Generic Java, because these

core language features weren't available but strongly desired.

I think they have sped up again since 1.5, with 1.7 on the horizon, and it looks like each of these has delivered very useful new features. The open sourcing of the language is good as well.

And whatever some people say, retaining backwards compatibility has been a very important feature in the Java language.

I would say that .NET has kicked Sun's arses into gear, and that it is good that they both exist.

Share Improve this answer

Follow

answered Nov 20, 2008 at 18:07

JeeBee

17.5k ● 5 ● 52 ● 60

3 Sun has sped up since 1.5? That was released over 4 years ago. Since then we've had C# 2, C# 3, and I wouldn't be *entirely* surprised to see C# 4 ship before Java 7. At least I've got a pretty firm idea what's going to be in C# 4 - the feature list for Java 7 still seems very up in the air. – Jon Skeet Nov 21, 2008 at 7:36

Having said that, I do entirely agree that it's a good thing for both Java and .NET to exist :) – Jon Skeet Nov 21, 2008 at 7:37

2 .NET 4.0 has now shipped before Java 7. :)
– Andrew T Finnell Mar 27, 2011 at 18:51

1

For a long time, Java has (for better or for worse) preferred to introduce new functionality through frameworks, rather than in the language itself.

You could argue that, for example, someone writing a Spring/Hibernate/Struts application is actually writing in a specialized dialect of Java, since their code will be largely inoperable without all the reflection/injection/instrumentation magic performed by the frameworks.

Personally, I'd prefer for the language to evolve a little bit more, and for the framework authors to quit monkeying with the language semantics via bytecode manipulation.

As a side note, I also strongly disagree with Sun's decision to use type erasure in their implementation of Generics. Supposedly, they wanted to ensure backward compatibility, but to me that seems totally spurious, since Annotations (added at exactly the same time, in Java 5) created their own set of incompatibilities. At any rate, code compiled for Java 5 can't ever be executed on an earlier JVM, so the backward-compatibility claim seems very dubious to me.

Share Improve this answer

Follow

answered Nov 20, 2008 at 21:58

benjismith

16.7k ● 9 ● 61 ● 83

Type erasure is about forward compatibility. Other solutions were considered for the problem, but erasure was selected

as a clean way to allow code compiled for Java 1.4 to keep working without modification in a 1.5 runtime. – erickson Nov 23, 2008 at 6:13

Ah, yes. I hadn't thought of that. When .NET introduced generic types, they did so by adding a whole new namespace and differentiating the parametric collection classes from the old vanilla collection classes. Both solutions are pretty ugly. – benjismith Nov 26, 2008 at 16:24