# Java 10 Local Variable Type Inference Advantage? [closed]

Asked 5 years, 11 months ago  Modified 5 years, 11 months ago  Viewed 3k times

5

**Closed**. This question needs to be more focused. It is not currently accepting answers.

**Want to improve this question?** Update the question so it focuses on one problem only by editing this post.

Closed 5 years ago.

Improve this question

I am trying to understand Java 10 Local Variable Type Inference. I seem to understand what it is, but I do not see any advantage from this. So I was wondering what is the idea behind introducing this feature. These are few of my observations. Please correct me if I am wrong here.

1. For example, (unlike other languages) I can not just declare a variable like this

   ```
   var abc;
   ```

   I need to initialize it(But cannot initialize to null). So I don't really see any advantage whatsoever.

2. One of the other arguments I saw is that previously we had to declare a variable like this with its explicit types.

   ```
   Map<User, String> userChannels = new HashMap<>();
   ```

   Now I can do it like this

   ```
   var userChannels = new HashMap<User, String>();
   ```

With modern day IDE's(like IntelliJ IDEA) and their support for code completion. I cannot think of any added advantage that this brings to the table(in the above context).

Some of the other points I read were that

Polymorphic code doesn’t play nice with var.
And also I cannot use var for non denotable types like Anonymous class.

Given all these, why was it necessary to introduce this feature? Can someone please clarify If I am missing something here.

java java-10

Share Improve this question

Follow

edited Jan 3, 2019 at 11:46

asked Jan 3, 2019 at 11:41


pvpkiran

27k ● 12 ● 100 ● 145

6 1. IDEs are not browsers. 2. Listing advantages are off-topic here I believe. 3. Here is the JEP openjdk.java.net/jeps/286 for the motivation and goal behind it. – Naman Jan 3, 2019 at 11:46

2 It was not necessary. But a variablr declaration like `Properties properties = new Properties() ;` can now be declared with `var properties = new Properties();` which some people prefer. – NielsNet Jan 3, 2019 at 11:46

3 @pvpkiran The question is still off-topic. Please read the JEP for advantage, motivation, and goals and be specific about any issues/doubts that you may face while either using or comparing the feature with specific implementation. – Naman Jan 3, 2019 at 11:51

1 I blame Oracle.They are only introducing syntactical sugars. They are ruining the essence of versions. Java was in safe hands of SunMicrosystems. Once java was a pure object oriented programming langauage. Alas! – Jabongg Jan 3, 2019 at 11:55

3 For me, the advantage is less redundant text which makes the code less cluttered and thus easier to read. The fact it reduces the amount of typing, IDE or no, is a small bonus. In cases where multiple variables are initialized "together" it can also better align the variable names. – Slaw Jan 3, 2019 at 12:32

## 2 Answers

Sorted by: Highest score (default)

10

One of the reasons is that it makes your code shorter and easier to read. Consider the following example, where you will use the variable only once or twice after the declaration.

```
ReallyLongClassNameBecauseBigEnterpriseProject reallyLongClassAbv = new
ReallyLongClassNameBecauseBigEnterpriseProject(foo);
OtherAnnoyingLongClassName otherAnnoyingLongClassName =
reallyLongClassAbv.getOtherAnnoyingLongClassName();
```

If the variable name has the same name than the class (or a shortened name because you can deduce the class by the context or by the constructor), then the class name doesn't add much info. But if you write

```
var reallyLongClassAbv = new
ReallyLongClassNameBecauseBigEnterpriseProject(foo);
var otherAnnoyingLongClassName =
reallyLongClassAbv.getOtherAnnoyingLongClassName();
```

It's already nicer and faster to read, and since you already have the class name, you don't lose any info. On a small bonus, your variable names are even aligned!

You might think that it doesn't make a big difference, but in my experience, I have worked in projects full of those statements and I really wished I didn't have to read two or three times the class name at each declaration. The `var` keyword could increase the information/text ratio and make your code less verbose.

**NOTE** : to make it clear, even with `var`, you should still avoid to give uselessly long names to your class, but sometimes you don't have the choice or it comes from another library.

---

As stated in the comments, you could have a look at the JEP to have a more complete answer and the Style Guidelines for Local Variable Type Inference by Stuart Marks.

---

On a humorous note : you can check here a satire of how to Enterprisify your Java Class Names, or real examples like InstantiationAwareBeanPostProcessorAdapter.

Share Improve this answer

Follow

edited Jan 3, 2019 at 13:06

answered Jan 3, 2019 at 11:46

Ricola

2,882 ● 13 ● 27

I didn't edited to make it fit into the screen, it's just that I remembered than in my project that sometimes the variable name didn't have the full class name. Even if the variable name has the same name, it already removes one useless apparition of the long class name. You could indeed say that people have to come with shorter class names but sometimes in projects you don't always have the choice, especially if they come from an external library. – Ricola Jan 3, 2019 at 11:58

2 One example amongst others : AbstractSingletonProxyFactoryBean, what if I'd like to call my variable `bean` because I only use it once or twice and in with the context you don't need the full name? – Ricola Jan 3, 2019 at 12:02

1 Well, reverted the vote for the sake of experience sharing. Though that's primarily an opinion which is supposedly off-topic on SO. Anyway, better way for people focussing on using LVTI, the style guide by Stuart Marks - openjdk.java.net/projects/amber/LVTIstyle.html – Naman Jan 3, 2019 at 12:10

Only one person has downvoted your answer! It wasn't me, but one possible reason that you got a downvote is that you are answering a question that (in my opinion) is off topic for Stack Overflow. – skomisa Jan 4, 2019 at 0:05

8

## Advantage: shorter code

```
StringBuilder buffer = new StringBuilder();  // traditional
var buffer = new StringBuilder();            // new
```

## Disadvantage: less clarity in some cases

```
Person theFirst = phoneRegister.values().iterator.next();   // At least you
know it's a Person
var theFirst = phoneRegister.values().iterator.next();      // Quite a task to
find out the type.
```

Hopefully, you won't find such a line in any production code, even the first one.

## Disadvantage: implementation type instead of interface

```
Map<User, String> userChannels = new HashMap<>();   // interface type
var userChannels = new HashMap<User, String>();     // this gives a HashMap
variable
```

With the "var" version, you get a variable declared as `HashMap`, not `Map`, silently allowing you to use all additional methods implemented in the `HashMap` class, not only the ones from the `Map` interface. So you face a harder job if you later want to use a different implementation class.

Share Improve this answer Follow

answered Jan 3, 2019 at 13:10

Ralf Kleberhoff
7,290 ● 1 ● 14 ● 10