Java 8 - 类型推断优化

理解Java 8 类型推断需理解几个问题:

- 什么是泛型?
- Java7对泛型推断做了哪些优化?
- Java8对此有做了哪些优化?

简单理解泛型

泛型是Java SE 1.5的新特性，泛型的本质是参数化类型，也就是说所操作的数据类型被指定为一个参数。通俗点将就是“类型的变量”。这种类型变量可以用在类、接口和方法的创建中。

理解Java泛型最简单的方法是把它看成一种便捷语法，能节省你某些Java类型转换(casting)上的操作:

List box = new ArrayList();box.add(new Apple());Apple apple =box.get(0);

上面的代码自身已表达的很清楚: box是一个装有Apple对象的List。get方法返回一个Apple对象实例，这个过程不需要进行类型转换。没有泛型，上面的代码需要写成这样:

Apple apple = (Apple)box.get(0);

泛型的尴尬

泛型的最大优点是提供了程序的类型安全同时可以向后兼容，但也有尴尬的地方，就是每次定义时都要写明泛型的类型，这样显示指定不仅感觉有些冗长，最主要是很多程序员不熟悉泛型，因此很多时候不能够给出正确的类型参数，现在通过编译器自动推断泛型的参数类型，能够减少这样的情况，并提高代码可读性。

java7的泛型类型推断改进

在以前的版本中使用泛型类型，需要在声明并赋值的时候，两侧都加上泛型类型。例如:

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

你可能觉得:老子在声明变量的的时候已经指明了参数类型，为毛还要在初始化对象时再指定? 幸好，在Java SE 7中，这种方式得以改进，现在你可以使用如下语句进行声明并赋值:

```
Map<String, String> myMap = new HashMap<>(); //注意后面的"<>"
```

在这条语句中，编译器会根据变量声明时的泛型类型自动推断出实例化HashMap时的泛型类型。再次提醒一定要注意new HashMap后面的“<>”，只有加上这个“<>”才表示是自动类型推断，否则就是非泛型类型的HashMap，并且在使用编译器编译源代码时会给出一个警告提示。

但是: Java SE 7在创建泛型实例时的类型推断是有限制的: 只有构造器的参数化类型在上下文中被显著的声明了，才可以使用类型推断，否则不行。例如: 下面的例子在java 7无法正确编译(但现在在java8里面可以编译，因为根据方法参数来自动推断泛型的类型):

```
List<String> list = new ArrayList<>();
list.add("A");// 由于addAll期望获得Collection<? extends String>类型的参数，因此下面的语句无法通过
list.addAll(new ArrayList<>());
```

Java8的泛型类型推断改进

java8里面泛型的目标类型推断主要2个:

1.支持通过方法上下文推断泛型目标类型

2.支持在方法调用链路当中，泛型类型推断传递到最后一个方法

让我们看看官网的例子

```
class List<E> {
   static <Z> List<Z> nil() { ... };
   static <Z> List<Z> cons(Z head, List<Z> tail) { ... };
   E head() { ... }
}
```

根据JEP101的特性，我们在调用上面方法的时候可以这样写

```
//通过方法赋值的目标参数来自动推断泛型的类型
List<String> l = List.nil();
//而不是显示的指定类型
//List<String> l = List.<String>nil();
//通过前面方法参数类型推断泛型的类型
List.cons(42, List.nil());
//而不是显示的指定类型
//List.cons(42, List.<Integer>nil());
```

总结

以上是JEP101的特性内容了，Java作为静态语言的代表者，可以说类型系统相当丰富。导致类型间互相转换的问题困扰着每个java程序员，通过编译器自动推断类型的东西可以稍微缓解一下类型转换太复杂的问题。 虽然说是小进步，但对于我们天天写代码的程序员，肯定能带来巨大的作用，至少心情更愉悦了