Generic instances in Java Byte Code

Martin Plümicke

Baden-Wuerttemberg Cooperative State University
Stuttgart/Horb

28. April 2014

Overview

Brian Goetz: Why Java 8 has no function types?

- ...
- ...
- ...
- It is unlikely that there would be a runtime representation for each distinct function type, meaning developers would be further exposed to and limited by erasure. For example, it would not be possible (perhaps surprisingly) to overload methods `m(T->U)` and `m(X->Y)`.

Example class

```
package Testfiles;

class AA<T> {
    T attr;

    void set(T attr) { this.attr = attr; }

    T get () { return attr; }

}
```

instanceof

```
class Main {
    void m () {
      AA<?> ac;
      Boolean b1 = ac instanceof AA<String>;
      Boolean b2 = ac instanceof AA<Character>;
    }
}
```

instanceof

```
class Main {
    void m () {
      AA<?> ac;
      Boolean b1 = ac instanceof AA<String>;
      Boolean b2 = ac instanceof AA<Character>;
    }
}
```

```
> javac Main.java
```

```
Main.java:5: error: illegal generic type for instanceof
        Boolean b1 = ac instanceof AA<String>;
                                   ^
Main.java:6: error: illegal generic type for instanceof
        Boolean b2 = ac instanceof AA<Character>;
                                   ^
```

Generic Exceptions

```
class MyException<T> extends Exception { }
```

Generic Exceptions

```
class MyException<T> extends Exception { }
```

```
> javac MyException.java
```

```
error: a generic class may not extend java.lang.Throwable
class MyException<T> extends Exception
                             ^
```

Generic Exceptions

```
class MyException<T> extends Exception { }
```

```
> javac MyException.java

error: a generic class may not extend java.lang.Throwable
class MyException<T> extends Exception
                             ^
```

Background:

```
try { }
catch ( MyException<Typ1> e ) { }
catch ( MyException<Typ2> e ) { }
```

is impossible.

Generic overloading

```
package Testfiles;

class GenericsTestReduced {
  void meth1(AA<BB> x) { System.out.println("BB"); }

  void meth1(AA<Character> x) { System.out.println("Character"); }
}
```

Generic overloading

```
package Testfiles;

class GenericsTestReduced {
  void meth1(AA<BB> x) { System.out.println("BB"); }

  void meth1(AA<Character> x) { System.out.println("Character"); }
}


> javac GenericsTestReduced.java

GenericsTestReduced.java:9: error: name clash:
meth1(AA<Character>) and meth1(AA<BB>) have the same erasure
    void meth1(AA<Character> x)  System.out.println("Character");
         ^
1 error
```

Byte-code

```
//Constant-pool
...
16| tag = CONSTANT_Utf8, length = 5, meth1
17| tag = CONSTANT_Utf8, length = 17, (LTestfiles/AA;)V
18| tag = CONSTANT_Utf8, length = 9, Signature
19| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA<LTestfiles/BB;>;)V
20| tag = CONSTANT_Utf8, length = 5, meth1
21| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA<Ljava/lang/Character;>;)V
//Methods
...
method[2] = {
  access_flags = 0
  name_index = 16  //meth1
  descriptor_index = 17  //(LTestfiles/AA;)V
  ...
  attribut[1] = {
    attribut_name_index = 18  //Signature
    attribut_length = 2
    descriptor_index = 19 //(LTestfiles/AA<LTestfiles/BB;>;)V }}
```

Byte-code

```
//Constant-pool
...
16| tag = CONSTANT_Utf8, length = 5, meth1
17| tag = CONSTANT_Utf8, length = 17, (LTestfiles/AA;)V
18| tag = CONSTANT_Utf8, length = 9, Signature
19| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA<LTestfiles/BB;>;)V
20| tag = CONSTANT_Utf8, length = 5, meth1
21| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA<Ljava/lang/Character;>;)V
//Methods
...
method[3] = {
  access_flags = 0
  name_index = 20  //meth1 (meth2)
  descriptor_index = 17  //(LTestfiles/AA;)V

  attribut[1] = {
    attribut_name_index = 18  //Signature
    attribut_length = 2
    descriptor_index = 21  //(LTestfiles/AA<Ljava/lang/Character;>;)V }}
```

First approach: Change the links (original)

```
//Constant-pool
...
16| tag = CONSTANT_Utf8, length = 5, meth1
17| tag = CONSTANT_Utf8, length = 17, (LTestfiles/AA;)V
18| tag = CONSTANT_Utf8, length = 9, Signature
19| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA<LTestfiles/BB;>;)V
20| tag = CONSTANT_Utf8, length = 5, meth1
21| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA<Ljava/lang/Character;>;)V

//Methods
...
method[2] = {
  access_flags = 0
  name_index = 16  //meth1
  descriptor_index = 17  //(LTestfiles/AA;)V
  ...
  attribut[1] = {
    attribut_name_index = 18  //Signature
    attribut_length = 2
    descriptor_index = 19  //(LTestfiles/AA<LTestfiles/BB;>;)V }}
```

First approach: Change the links (changed)

```
//Constant-pool
...
16| tag = CONSTANT_Utf8, length = 5, meth1
17| tag = CONSTANT_Utf8, length = 17, (LTestfiles/AA;)V
18| tag = CONSTANT_Utf8, length = 9, Signature
19| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA<LTestfiles/BB;>;)V
20| tag = CONSTANT_Utf8, length = 5, meth1
21| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA<Ljava/lang/Character;>;)V

//Methods
...
method[2] = {
  access_flags = 0
  name_index = 16  //meth1
  descriptor_index = 19  //(LTestfiles/AA<LTestfiles/BB;>;)V
  ...
  attribut[1] = {
    attribut_name_index = 18  //Signature
    attribut_length = 2
    descriptor_index = 19  //(LTestfiles/AA<LTestfiles/BB;>;)V }}
```

JVM–output

```
> java Testfiles.GenericsTestReduced
Exception in thread "main" java.lang.ClassFormatError:
  Method "meth1" in class Testfiles/GenericsTestReduced
  has illegal signature "(LTestfiles/AA<LTestfiles/BB;>;)V"
  at java.lang.ClassLoader.defineClass1(Native Method)
  at java.lang.ClassLoader.defineClass(ClassLoader.java:760)
  at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
  at java.net.URLClassLoader.defineClass(URLClassLoader.java:455)
  at java.net.URLClassLoader.access$100(URLClassLoader.java:73)
  at java.net.URLClassLoader$1.run(URLClassLoader.java:367)
  at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
  at java.security.AccessController.doPrivileged(Native Method)
  at java.net.URLClassLoader.findClass(URLClassLoader.java:360)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
  at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
  at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:495)
```

Second approach: New JVM Type syntax

`LTy<Lp11/.../p1m1/T1;Lp21/.../p2m2/T2;...;Lp1n/.../pnmn/Tn;>`

becomes

`LTy%Lp11#...#p1m1#T1%Lp21#...#p2m2#T2%...%Lp1n#...#pnmn#Tn%%`

New translation

```
class GenericsTestReduced {
  void meth1(AA<BB> x) { System.out.println("BB"); }

  void meth1(AA<Character> x) { System.out.println("Character"); }
}
```

void meth1(AA<BB> x)

```
//Constant-pool
17| tag = CONSTANT_Utf8, length = 17, (LTestfiles/AA;)V
18| tag = CONSTANT_Utf8, length = 9, Signature
19| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA<LTestfiles/BB;>;)V
20| tag = CONSTANT_Utf8, length = 5, meth1
21| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA<Ljava/lang/Character;>;)V
22| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA%LTestfiles#BB%%;)V
23| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA%Ljava_lang_Character%%;)V

//Methods
method[2] = {
access_flags = 0
name_index = 20  //meth1
descriptor_index = 22  //(LTestfiles/AA%LTestfiles#BB%%;)V
...
  attribut[1] = {
    attribut_name_index = 18  //Signature
    attribut_length = 2
    descriptor_index = 19  //(LTestfiles/AA<LTestfiles/BB;>;)V }}
```

void meth1(AA<Character> x)

```
//Constant-pool
17| tag = CONSTANT_Utf8, length = 17, (LTestfiles/AA;)V
18| tag = CONSTANT_Utf8, length = 9, Signature
19| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA<LTestfiles/BB;>;)V
20| tag = CONSTANT_Utf8, length = 5, meth1
21| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA<Ljava/lang/Character;>;)V
22| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA%LTestfiles_BB%%;)V
23| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA%Ljava#lang#Character%%;)V

//Methods
method[3] = {
access_flags = 0
name_index = 20  //meth1
descriptor_index = 23  //(LTestfiles/AA%Ljava#lang#Character%%;)V
...
  attribut[1] = {
    attribut_name_index = 26  //Signature
    attribut_length = 2
    descriptor_index = 21  //(LTestfiles/AA<Ljava/lang/Character;>;)V }}
```

Method call

```
package Testfiles;

public class Main {

    public static void main (String[] args) {
     GenericsTestReduced gt = new GenericsTestReduced ();
     gt.meth1(new AA<BB>());
     gt.meth1(new AA<Character>());
    }
}
```

Byte-code (original)

```
8| tag = CONSTANT_Class, name_index = 32
9| tag = CONSTANT_Methodref, class_index = 8, name_and_type_index = 28
10| tag = CONSTANT_Methodref, class_index = 2, name_and_type_index = 33
14| tag = CONSTANT_Utf8, length = 6, <init>
15| tag = CONSTANT_Utf8, length = 3, ()V
19| tag = CONSTANT_Utf8, length = 17, (LTestfiles/AA;)V
28| tag = CONSTANT_NameAndType, name_index = 14, descriptor_index = 15
32| tag = CONSTANT_Utf8, length = 12, Testfiles/AA
33| tag = CONSTANT_NameAndType, name_index = 37, descriptor_index = 19
37| tag = CONSTANT_Utf8, length = 5, meth1
```

```
26 new 8 //new AA<BB>()
29 dup
30 invokespecial 9 //<init> of AA<BB>()
33 invokevirtual 10 //meth1(AA<BB>)
...
37 new 8 //new AA<Character>()
40 dup
41 invokespecial 9 //<init> of AA<Character>()
44 invokevirtual 10 //meth1(AA<Character>)
```

New Byte-code-File in Java syntax

```
package Testfiles;

public class Main {

    public static void main (String[] args) {
     GenericsTestReduced gt = new GenericsTestReduced ();
     gt.meth1(new AA%LTestfiles#BB%%());
     gt.meth1(new AA%Ljava#lang#Character%%());
    }
}
```

```
6| tag = CONSTANT_Methodref, class_index = 2, name_and_type_index = 21
7| tag = CONSTANT_Methodref, class_index = 2, name_and_type_index = 22
10| tag = CONSTANT_Utf8, length = 6, <init>
11| tag = CONSTANT_Utf8, length = 3, ()V
18| tag = CONSTANT_NameAndType, name_index = 10 descriptor_index = 11
21| tag = CONSTANT_NameAndType, name_index = 25 descriptor_index = 29
22| tag = CONSTANT_NameAndType, name_index = 27 descriptor_index = 28
25| tag = CONSTANT_Utf8, length = 5, meth1
27| tag = CONSTANT_Utf8, length = 5, meth1
28| tag = CONSTANT_Utf8, length = 40, (LTestfiles/AA%Ljava#lang#Character%%;)V
29| tag = CONSTANT_Utf8, length = 33, (LTestfiles/AA%LTestfiles#BB%%;)V
30| tag = CONSTANT_Utf8, length = 28, Testfiles/AA%LTestfiles#BB%%
31| tag = CONSTANT_Class, name_index = 30
32| tag = CONSTANT_Methodref, class_index = 31, name_and_type_index = 18
33| tag = CONSTANT_Utf8, length = 35, Testfiles/AA%Ljava#lang#Character%%
34| tag = CONSTANT_Class, name_index = 33
35| tag = CONSTANT_Methodref, class_index = 34, name_and_type_index = 18

  10 new 31 //new AA%Ljava#lang#BB%%()
  13 dup
  14 invokespecial 32 //<init> auf AA%Ljava#lang#BB%%()
  17 invokevirtual 6 //meth1(AA%Ljava#lang#BB%%())
  ...
  21 new 34 //new AA%Ljava#lang#Character%%()
  24 dup
  25 invokespecial 35 //<init> auf AA%Ljava#lang#Character%%()
  28 invokevirtual 7 //meth1(AA%Ljava#lang#Character%%())
```

.class–Files

For each instance an own .class–Files has to be built:

- AA%LTestfiles#BB%%.class
- AA%Ljava#lang#Character%%.class

AA.class

```
1| tag = CONSTANT_Class, name_index = 2
2| tag = CONSTANT_Utf8, length = 12, Testfiles/AA
3| tag = CONSTANT_Class, name_index = 4
4| tag = CONSTANT_Utf8, length = 16, java/lang/Object
...
31| tag = CONSTANT_Utf8, length = 28, Testfiles/AA%LTestfiles#BB%%

this_class = 1
super_class = 3
```

AA%LTestfiles#BB%%.class

```
1| tag = CONSTANT_Class, name_index = 31
2| tag = CONSTANT_Utf8, length = 12, Testfiles/AA
3| tag = CONSTANT_Class, name_index = 4
4| tag = CONSTANT_Utf8, length = 16, java/lang/Object
...
31| tag = CONSTANT_Utf8, length = 28, Testfiles/AA%LTestfiles#BB%%

this_class = 1
super_class = 3
```

Subtyping

```
package Testfiles;

public class CC<T1, T2> extends AA<T1> { }

public class Main_Subtyping {

    public static void main (String[] args) {
     GenericsTestReduced gt = new GenericsTestReduced ();
     gt.meth1(new CC<BB,Integer>());
     gt.meth1(new CC<Character,BB>());
    }
}
```

CC%LTestfiles#BB%Ljava#lang#Integer%%.class

```
...
2| tag = CONSTANT_Class, name_index = 15
3| tag = CONSTANT_Class, name_index = 16
...
15| tag = CONSTANT_Utf8, length = 47,
          Testfiles/CC%LTestfiles#BB%Ljava#lang#Integer%%
16| tag = CONSTANT_Utf8, length = 28, Testfiles/AA%LTestfiles#BB%%

this_class = 2
super_class = 3
```

CC%Ljava#lang#Character%LTestfiles#BB%%.class

```
...
2| tag = CONSTANT_Class, name_index = 15
3| tag = CONSTANT_Class, name_index = 16
...
15| tag = CONSTANT_Utf8, length = 49,
          Testfiles/CC%Ljava#lang#Character%LTestfiles#BB%%
16| tag = CONSTANT_Utf8, length = 35, Testfiles/AA%Ljava#lang#Character%%

this_class = 2
super_class = 3
```

Translation Algorithm

Translated syntax of instatiated types:

`LTy<Lp11/.../p1m1/T1;Lp21/.../p2m2/T2;...;Lp1n/.../pnmn/Tn;>`

⇓

`LTy%Lp11#...#p1m1#T1%Lp21#...#p2m2#T2%...%Lp1n#...#pnmn#Tn%%`

Translation Algorithm

Translated syntax of instatiated types:

```
LTy<Lp11/.../p1m1/T1;Lp21/.../p2m2/T2;...;Lp1n/.../pnmn/Tn;>
                                ⇓
LTy%Lp11#...#p1m1#T1%Lp21#...#p2m2#T2%...%Lp1n#...#pnmn#Tn%%
```

The algorithm:

1. Translation of all instatiated types (method declarations, expressions, `instanceof`, ...)
2. Signature attributes of method declarations are unchanged
3. For each used instatiated class generate a new `.class`–clase file, where the generic variable is instatiated in all types.
4. For each superclass generate also a new `.class`–clase file, analogously.

Consideration of raw types

- Until now raw types can be used for each type instance.
- In the new design raw types can only be used for the uninstantiated case.

Class-Loader

- Problem: If there are many different type instances of one class each new generated class is loaded during runtime.

Class-Loader

- Problem: If there are many different type instances of one class each new generated class is loaded during runtime.
- Idea to solve:
  - The different type instances differ only in type annotations
    $\Longrightarrow$ for code-running only one instance is needed.

Class-Loader

- Problem: If there are many different type instances of one class each new generated class is loaded during runtime.
- Idea to solve:
  - The different type instances differ only in type annotations
    $\Longrightarrow$ for code-running only one instance is needed.
  - Either: Implementing a new class-loader (without changing the JVM)
  - Or: Adapting the JVM.

Summary and Outlook

Summary:

- The JVM uses no type parameteres.
  This leads to some unbeautiful properties.
- Solution-approach: Generation for each type-instance an own .class-file.
- Problem: Multiple class-loading during runtime.

Summary and Outlook

Summary:

- The JVM uses no type parameteres.
  This leads to some unbeautiful properties.
- Solution-approach: Generation for each type-instance an own .class-file.
- Problem: Multiple class-loading during runtime.

Outlook

- Solving class-loading problem.
- Implementation.