Using the OpenJDK to investigate covariance in Java

Raoul-Gabriel Urma & Janina Voigt

UNIVERSITY OF CAMBRIDGE

You know about…

```
Fruit f = new Apple();
```

What about…

```
Fruit[] f = new Apple[5];
```

```
Apple[] apples = new Apple[5];
Fruit[] fruits = apples;
```

```
Apple[] apples = new Apple[5];
Fruit[] fruits = apples;
fruits[0] = new Banana();
```

```
Apple[] apples = new Apple[5];
Fruit[] fruits = apples;
fruits[0] = new Banana();
```

Not Type Safe!

Covariance, Contravariance, Invariance

- Given **A** is a subtype of **F**

- *Covariance*: **A** is used where **F** is required
- *Contravariance*: **F** is used where **A** is required
- *Invariance*: neither covariant nor contravariant

- In our work, we focus on *array covariance* in Java: **A[]** is a subtype of **F[]**

Array Covariance in Java

- Array covariance accepted by Java compiler
- At runtime, type checks are required for *every* array operation
- This causes performance overhead

Why does Java allow this?

Without generics, how do you write a sort() for arrays?

```
public static void sort(Object[] a)
{
   …
}
```

(from java.util.Arrays)

UNIVERSITY OF CAMBRIDGE

Generics – a better alternative

```
public static <T> void sort(T[] a)
{
   …
}
```

Type safe! ☺

Java

Generics are invariant:

```
ArrayList<Fruit> f
        = new ArrayList<Banana>();
```

Java

Generics are invariant:

```
ArrayList<Fruit> f
        = new ArrayList<Banana>();
```

Java has *use-site variance,* but this leads to restrictions on how the collection can be used:

```
ArrayList<? extends Fruit> f
        = new ArrayList<Banana>();
f.add(new Banana());
```

UNIVERSITY OF CAMBRIDGE

Scala

Supports *definition-site variance:*

```
Bowl[T]{}          // invariant
CoBowl[+T]{ }      // covariant

val bowl : Bowl[Fruit]
         = new Bowl[Banana];
val coBowl : CoBowl[Fruit]
         = new CoBowl[Banana];
```

UNIVERSITY OF CAMBRIDGE

Google Dart

Covariance is allowed but this may cause errors at runtime:

```
List<Fruit> fruits
      = new List<Apple>(5);
fruits[0] = new Banana();
fruits[0].peelApple();
```

Do we care about this?

- Empirical study of Java arrays: how often are covariant arrays used?
- **64** open source programs
  - Ant, Lucene, Hibernate, Tomcat, JUnit…
- **5.000.000**+ lines of code

UNIVERSITY OF CAMBRIDGE

Modifying the OpenJDK

- We made arrays invariant
- We get compile errors if there's any covariant array use

```
public boolean isSubtypeUnchecked
   (Type t, Type s, Warner warn)
```

In com.sun.tools.javac.code.Types.java

UNIVERSITY OF CAMBRIDGE

$ javac Main.java

```
public class Main
{
   public static void main(String[] args)
   {
      Object[] str = new String[10];
   }

   public static Object[] test2()
   {
      return new String[10];
   }
}
```

$ javac Main.java

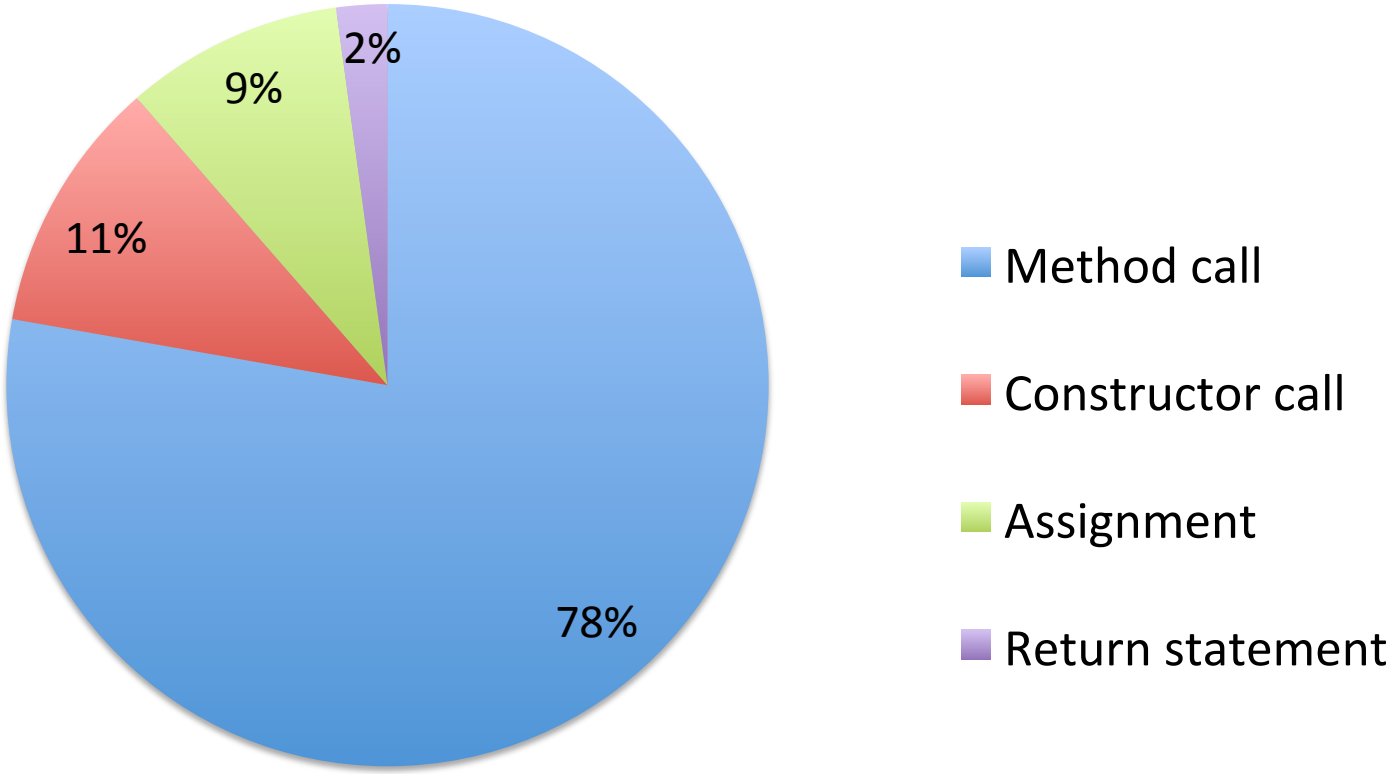
```
Main.java:5: incompatible types
found   : java.lang.String[]
required: java.lang.Object[]
            Object[] str = new String[10];

Main.java:10: incompatible types
found   : java.lang.String[]
required: java.lang.Object[]
            return new String[10];
```

Results

- Number of programs: **64**
- LOC: **5.000.000**+
- Number of covariance uses: **324**
- **9**/64 programs use covariance

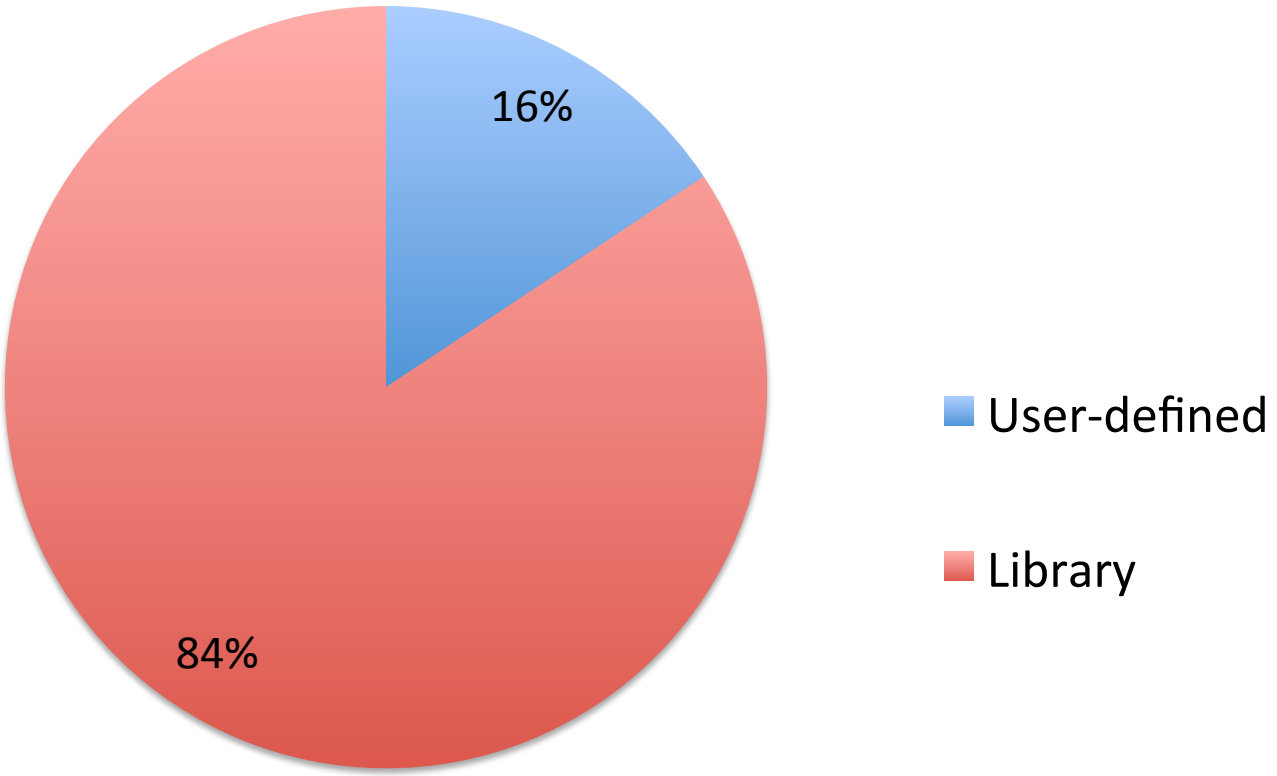
UNIVERSITY OF CAMBRIDGE

Breakdown of Covariant Uses by Type



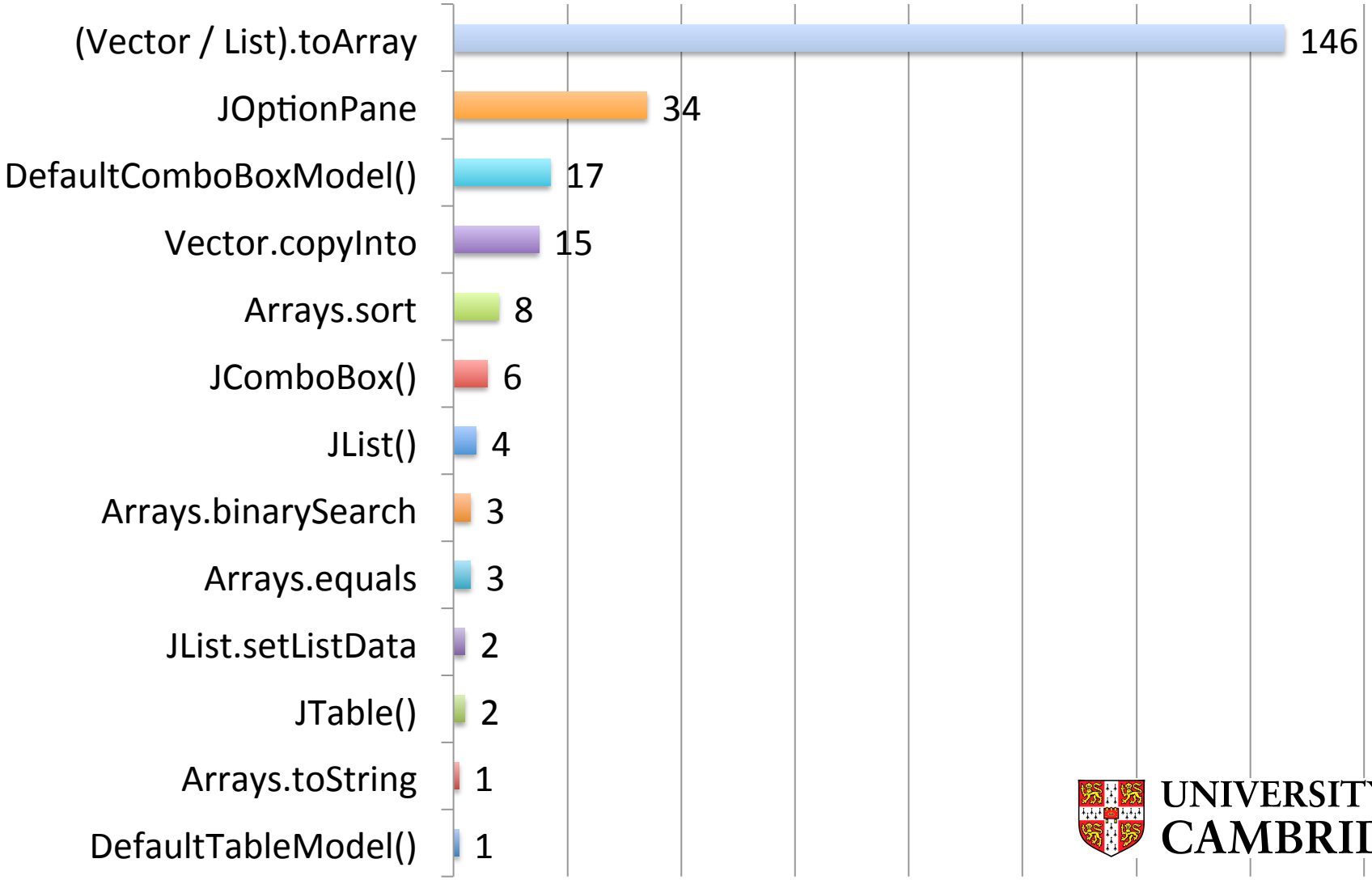
UNIVERSITY OF CAMBRIDGE

User-defined vs Library Calls



UNIVERSITY OF CAMBRIDGE

Breakdown of Covariant Library Calls



UNIVERSITY OF CAMBRIDGE

Some Examples

```
public static boolean
   arrayContains(Object[] array, Object key)
{
   for (int i = 0; i < array.length; i++)
   {
      if (array[i].equals(key))
         return true;
   }
   return false;
}
```

Found in Cobertura

UNIVERSITY OF CAMBRIDGE

With Generics

```
public static <T> boolean
   arrayContains(T[] array, T key)
{
   for(T k : array)
   {
      if(key.equals(k))
         return true;
   }
   return false;
}
```

Some Examples

```
public Object[] toArray(Object[] a)
```

(in java.util.List)

```
String[] args = (String[]) arglist.toArray(
                    new String[arglist.size()]
                );
```

“Returns an array containing all of the elements in this list in proper sequence; the runtime type of the returned array is that of the specified array.”

Found in Cobertura

With Generics

```
public <T> T[] toArray(T[] a)
```

(in java.util.List)

```
String[] strA = list.toArray(
                    new String[0]
                );
```

Conclusion

- Very few user-defined uses of covariance
- Can be converted to use generics, which is type safe
- Since array covariance is not used often, perhaps it should not be included in future programing languages (now that we have generics)?
- Modifying the OpenJDK is an effective way to conduct studies of Java

UNIVERSITY OF CAMBRIDGE

Further Work

- Looking at the use of wildcards in Java
- Extending our corpus to include a larger variety of programs
- Making our corpus publicly available for other studies
- Using the OpenJDK as a platform for empirical studies

UNIVERSITY OF CAMBRIDGE

Thanks!

raoul.urma@cl.cam.ac.uk

janina.voigt@cl.cam.ac.uk

Twitter: @raoulUK

UNIVERSITY OF CAMBRIDGE