Analysis of Polymorphism detection

Bachelor's thesis

at the
Software Composition Group, University of Bern, Switzerland
http://scg.unibe.ch/

by
Michael Morelli
July 2013

led by
Prof. Dr. Oscar Nierstrasz

Abstract

This document analyses the possibility of guessing polymorphic fields at compile time with the bytecode manipulation library Javassist.

Acknowledgments

I would like to thank Professor Oscar Nierstrasz for providing me with guidance and helpful support.
Also thanks to Dr. Mircea F. Lungu for his support.

Contents

1 Introduction

In object oriented programming polymorphism via inheritance and interface can only be detected at runtime. Of course we can have a look at our code and assume some polymorphism. But we have not a 100% guarantee that our assumption is correct. Only at runtime polymorphism can be detected, due to late-binding.
But is not there a way to guess polymorphism at development time? How good and precise can we assume polymorphism with some heuristics?
In this project we intend to see how well polymorphic fields are predictable at compile time. To gain this knowledge we implement an application which explores the polymorphic fields at development time and runtime and compare these results.

2 Polymorphism in a Nutshell

By definition polymorphism in object oriented programming (OOP) is the possibility of a variable or function argument to take on values of many types. In other words, a variable can have multiple types during runtime. This happens via inheritance or interfaces.

Via inheritance:
A declared variable can refer to any object of any class that is derived from its declared type by inheritance.
For example, if the class A is the parent of class B, then an A reference can be used to refer to any object of class B. So in this case class A can be instantiated with the type A or B. The reference variable can have two values with different types, so it is polymorphic.

Via interface:
Similarly to polymorphism via inheritance an interface name can be used as the type of a reference variable. So an interface reference variable can be used to refer to any object of any class that implements that interface.

3 Analysis

In this document we want to see how good the results of our guessing are. Do the guessed polymorphic fields match with the actual result at runtime?

3.1 Procedure

For our analysis we use Apache projects which are open source. We download the source code and import it to the Eclipse IDE. Through the import Eclipse generates the bytecode of the project we need for our detection on-the-fly. All the needed jars are imported manually.
The Apache Commons libraries contain a lot of unit tests. But to run our Dynamic Detection we have to declare the main class we want to use for the dynamic case.
To gain such a main class we have to write one by ourself. This class is responsible to run all unit tests of an external project one after the other. So for every project we want to analyse at runtime, we have to write our own main class to run the simulative run. To have a consistent nomination over all external projects we call our test runner main class `MainClass`. This way we can only change the path to the external project in the `Controller` class and we do not have to change the main class name for every Apache project.

For the Static Detection we do not need a main class. We can only set the path to the external project and “parse” the whole external Apache project.

3.2 Heuristics

To get a good guessing of polymorphic fields at compile time we have to implement heuristics. The main heuristics are:

- Ignore primitive field data types since they cannot be polymorphic at runtime. (The primitive data types are: `Boolean`, `char`, `byte`, `short`, `int`, `long`, `float` and `double`)
- (Fields which have more than one type at compile time and have an Interface type can be assumed to be polymorphic at runtime.)

The heuristic "fields with type Interface are assumed to be polymorphic" has been cancelled. Because this heuristic gives only a good match for the snakes and ladders project as we can see in the table below. The guessed fields would look as follows if we would put this heuristic.

Project	**# polymorphic fields at compile time (static detection)**	**# polymorphic fields at runtime (dynamic detection)**
Apache Commons JXPath 1.3	54	0
Apache Commons Collections 3.2.1	239	2
Apache Commons Codec 1.8	15	0
Apache Commons Pool 1.6	39	0
Apache Commons Daemon 1.0.15	16	0
Apache Commons CLI 1.2	5	0
P2 Snakes and Ladders	4	1

Due to the mismatch of the guessed fields at development time and polymorphic fields at runtime we cancelled this heuristic.

3.3 Projects

We analyse the following apache projects in detail:

Project name	#Unittests	#Errors	#Failures	Runner
JXPath 1.3	365	53	0	Unittests only
Collections 3.2.1	39143	0	66	Unittests only
Snakes and Ladders	15	0		Unittests run the whole application

Note: The number of unit tests, errors and failures are the ones on my machine running all unit tests with JUnit 4.0.

You find the whole Static and Dynamic output in the Appendix. The following references of the analysis refer to the Appendix output tables of the corresponding project.

3.3.1 Apache Commons JXPath

The external project Apache Commons JXPath is a Java-based implementation of XPath 1.0 that, in addition to XML processing, can inspect/modify Java object graphs and even mixed Java/XML structures. The open source includes 365 unit tests which we want to run for our dynamic polymorphism detector.

As we can see in the Static Detection Output (see Appendix – JXPath Static) we have no polymorphic fields guessed, since no field has more than one type assigned when we parse the external project and apply our heuristics.
The first field which has a field assigned, is the field `nodes`. As we can see in the source code segment 1 below, the field `nodes` is declared as a `Java.util.List`. And the field has the value type `Java.util.List` (return type of method unmodifiableList).

```
73    public synchronized List getNodes() {
74        if (nodes == null) {
75            nodes = new ArrayList();
76            for (int i = 0; i < pointers.size(); i++) {
77                Pointer pointer = (Pointer) pointers.get(i);
78                nodes.add(pointer.getNode());
79            }
80            nodes = Collections.unmodifiableList(nodes);
81        }
82        return nodes;
83    }
```

Code segment 1: Project JXPath - Method BasicNodeSet.getNodes

But as already mentioned, we do not get the assignment at line 75, since the value of nodes is a new `ArrayList()` object and not a field. Only field-writers and field-readers at the same line are merged.
The field `nodes` is monomorphic because it has only one value type.

The fields `readOnlyPointers` and `values` (see static output table lines 2&3) has the same value type as the field `node`.

In the package `axes` (see static output table lines 4-14) we got several field accesses. The class `AncestorContext` has the field `currentNodePointer` of type `org.apache.commons.jxpath.ri.model.NodePointer` and value type `org.apache.commons.jxpath.ri.model.NodePointer`.

```
77      public boolean nextNode() {
78          if (!setStarted) {
79              setStarted = true;
80              currentNodePointer = parentContext.getCurrentNodePointer();
81              if (includeSelf && currentNodePointer.testNode(nodeTest)) {
82                  position++;
83                  return true;
84              }
85          }
86
87          while (true) {
88              currentNodePointer = currentNodePointer.getImmediateParentPointer();
89
90              if (currentNodePointer == null) {
91                  return false;
92              }
93
94              if (currentNodePointer.testNode(nodeTest)) {
95                  position++;
96                  return true;
97              }
98          }
99      }
.00 }
```

Code segment 2: Project JXPath - Method AncestorContext.nextNode

In the Apache source code we can confirm that the field `currentNodePointer` has the value `parentContext.getCurrentNodePointer()` at code segment 2 - line 80 and at line 88 the value `currentNodePointer.getImmediateParentPointer()`. So the field has two values, but the value types are the same, since the method `getCurrentNodePointer()` at line 80 and `getImmediateParentPointer()` at line 88 both return the type `NodePointer`. So again, the field `currentNodePointer` is not polymorphic.

In the class `AttributeContext` we have the fields: `iterator` and `currentNodePointer`.

```
75      public boolean nextNode() {
76          super.setPosition(getCurrentPosition() + 1);
77          if (!setStarted) {
78              setStarted = true;
79              QName name;
80              if (nodeTest instanceof NodeNameTest) {
81                  name = ((NodeNameTest) nodeTest).getNodeName();
82              }
83              else {
84                  if (nodeTest instanceof NodeTypeTest
85                          && ((NodeTypeTest) nodeTest).getNodeType() == Compiler.NODE_TYPE_NODE) {
86                      name = WILDCARD;
87                  }
88                  else {
89                      iterator = null;
90                      return false;
91                  }
92              }
93              iterator = parentContext.getCurrentNodePointer().attributeIterator(
94                      name);
95          }
96          if (iterator == null) {
97              return false;
98          }
99          if (!iterator.setPosition(iterator.getPosition() + 1)) {
100             return false;
101         }
102         currentNodePointer = iterator.getNodePointer();
103         return true;
104     }
105 }
```

Code segment 3: Project JXPath - Method AttributeContext.nextNode

The field `iterator` (see static output table line 5) is declared as a `NodeIterator` and has the value `parentContext.getCurrentNodePointer().attributeIterator(name)` at source code line 93. Since the method `attributeIterator(name)` has the return type `NodeIterator`, the value type of `iterator` is `NodeIterator`.

At code segment 3 line 102 `currentNodePointer` (see static output table line 6) has the value type `NodePointer`, because the method `getNodePointer()` has the return value `NodePointer`.

The same is the case for the fields at static output table line 7-11.

At static output table line 12 we have a field called `context` inside the class `RecursiveAxesTest`.
Like the name says, this field occurs in a unit test class.

```
26 public class RecursiveAxesTest extends JXPathTestCase {
27
28     private RecursiveBean bean;
29     private JXPathContext context;
30
31     protected void setUp() throws Exception {
32         bean = new RecursiveBean("zero");
33         RecursiveBean bean1 = new RecursiveBean("one");
34         RecursiveBean bean2 = new RecursiveBean("two");
35         RecursiveBean bean3 = new RecursiveBean("three");
36         bean.setFirst(bean1);
37         bean1.setFirst(bean2);
38         bean2.setFirst(bean1);
39         bean2.setSecond(bean3);
40
41         context = JXPathContext.newContext(null, bean);
42     }
```

Code segment 4: Project JXPath - Unit test method RecursiveAxesTest.setUp

As we can see at the source code segment 4 line 29 the type of the field `context` is `JXPathContext` and the value type is `JXPathContext` as well (code segment 4 line 41).
Pretty the same is the case for fields at the static output table line 14-15.

At static output table line 16 we have inside class `EvalContext` the field `rootContext` of type `RootContext`:

```
282     public RootContext getRootContext() {
283         if (rootContext == null) {
284             rootContext = parentContext.getRootContext();
285         }
286         return rootContext;
287     }
```

Code segment 5: Project JXPath - Method EvalContext.getRootContext

The return type of `getRootContext()` is `RootContext` so the field value has this type at code segment 5 line 284.

For all other fields received from our Static Detector we have the same model.
So instead of confirming all other fields of our Static Detector output, we will have a closer look at the Dynamic Detection result:

As we can see every field has only one field type so we have no polymorphic fields at runtime in the external project Apache Commons JXPath.

If we have a look at the first field detected by the Dynamic Detector, we see that the field `name` of class `NestedTestBean` has the value type `Java.lang.String` at code segment 6 lines 26, 33, and 37.

```
25 public class NestedTestBean {
26     private String name = "Name 0";
27     private int integer = 1;
28
29     public NestedTestBean() {
30     }
31
32     public NestedTestBean(String name) {
33         this.name = name;
34     }
35
36     public void setName(String name) {
37         this.name = name;
```

Code segment 6: Project JXPath - Constructor NestedTestBean and method NestedTestBean.setName

Note: The field `name` has not been detected by the Static Detector, since the value of the field `name` is not a field (here the value is a local variable (code segment 6 line 33 and 37) or a character sequence (code segment 6 line 26)). As already mentioned there is no way in the Javassist API to get the value type of a field if the value is not a field. That is the reason why this field does not appear in the Static Detector. To remember, the Static Detector only merges field-writers (here `this.name`) with field-readers (here local `name` and "`Name 0`") if both are fields and occur at the same line.

In the same class we get another field called `strings`, whose type is a string-array.

```
65     private String[] strings = new String[] { "String 1", "String 2", "String 3" };
66
67     public String[] getStrings() {
68         return strings;
69     }
70
71     public void setStrings(String[] array) {
72         strings = array;
73     }
```

Code segment 7: Project JXPath - NestedTestBean methods: getStrings and setStrings

At code segment 7 lines 65 and 72 the value type of field `strings` is `Java.lang.String[]`. The "[L" in the result table stands for a reference to a one dimension array and is equal to `Java.lang.String[]`. The same counts for the base types B (`byte`), C (`char`), D (`double`), F (`float`), I (`int`), J (`long`), S (`short`) and Z (`boolean`).

And there are a lot of other fields with only one value type. But they are not of interest at this point, since we are heading for polymorphic fields.

If we compare the Static and Dynamic Detection result we can observe that we have polymorphic fields neither in the Static nor the Dynamic case. So the guessing matches 100%. But we have to consider that the Dynamic Detection result depends on the test coverage of the project. The better the coverage the more LOC's are actually touched by the simulative dynamic run.
Remember, the static detection parses the whole code of the external project. In other words we cover every LOC of the project. In contrast the Dynamic Detection algorithm only covers LOC's which are actually reached at runtime.

Since unit tests only cover a specific and small part of the code, it is very atypical to have a lot of polymorphism while running the JUnit tests.

If we compare the fields detected by the two Detectors, it is conspicuous that we have fewer dynamic detected fields than static detected fields. As already mentioned the reason lies in the test coverage of the project itself.
Another noticeable thing is that the most of the dynamic detected fields do not appear in the static result, because the Dynamic Detector saves more field accesses than the Static one. As remarked, the Dynamic Detector is able to save field accesses whose value is not a field. For lack of the Javassist API it is not possible to get a field's value, if the value itself is not a field in the Static Detector. It would be very interesting to use another more powerful library for code manipulation to see the difference.

A further conspicuous point is that in the dynamic output there are a lot of fields which are members of the unit tests and not the code itself. To see why that happens, we will track down the Dynamic Detector algorithm for one specific unit test class.

If we only run the unit test class: `org.apache.commons.jxpath.ri.model.EmbeddedColonMapKeysTest().run()`

We get the following three field accesses at runtime (monomorphic):

```
KEY: org.apache.commons.jxpath.ri.QName:
     Java.lang.String:name
 VALUE(S): Java.lang.String
```
(see dynamic result table line 14)

```
KEY:org.apache.commons.jxpath.ri.QName:
     Java.lang.String:qualifiedName
 VALUE(S): Java.lang.String
```
(see dynamic result table line 15)

```
KEY: org.apache.commons.jxpath.PackageFunctions:
     Java.lang.String:classPrefix
 VALUE(S): Java.lang.String
```
(see dynamic result table line 3)

And the call stack looks like that:

```
MainClass at localhost:50989
  Thread [main] (Suspended)
    QName.<init>(String, String) line: 55
    JXPathContextReferenceImpl.<init>(JXPathContext, Object, Pointer) line: 195
    JXPathContextReferenceImpl.<init>(JXPathContext, Object) line: 167
    JXPathContextFactoryReferenceImpl.newContext(JXPathContext, Object) line: 39
    JXPathContext.newContext(Object) line: 416
    EmbeddedColonMapKeysTest.setUp() line: 37
    EmbeddedColonMapKeysTest(TestCase).runBare() line: 128
    TestResult$1.protect() line: 106
    TestResult.runProtected(Test, Protectable) line: 124
    TestResult.run(TestCase) line: 109
    EmbeddedColonMapKeysTest(TestCase).run(TestResult) line: 120
    EmbeddedColonMapKeysTest(TestCase).run() line: 113
    MainClass.main(String[]) line: 31
C:\Program Files (x86)\Java\jre7\bin\javaw.exe (13.07.2013 14:20:40)
```

Listing 1: Project JXPath - Stack trace of MainClass

Now, we will have a look at the called methods inside the red rectangles. First the `MainClass` (our self written test runner) calls the unit test `EmbeddedColonMapKeysTest()`.

```
30 public class EmbeddedColonMapKeysTest extends JXPathTestCase {
31     private JXPathContext context;
32
33     protected void setUp() throws Exception {
34         super.setUp();
35         HashMap m = new HashMap();
36         m.put("foo:key", "value");
37         context = JXPathContext.newContext(m);
38         context.setLenient(true);
39     }
```

Code segment 8: Project JXPath - Unit test method EmbeddedColonMapKeysTest.setUp

Inside the class `EmbeddedColonMapKeysTest` we got a field `context` which is not saved at the dynamic run. Because the field `context` appears more than 737 occurrences in different packages and as mentioned our fields have to be made public before being able to inspect them. So the reason is that the variable `context` of type `JXPathContext` is shared.

If we search in our Eclipse IDE the field name `context` we get this:

```
'context' - 737 occurrences in project 'commons-jxpath-1.3-src' (no JRE) (0 matches filtered from view)
    org.apache.commons.jxpath - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.axes - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.compiler - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.model - src/java - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.model - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.model.beans - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.model.dom - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.model.dynamic - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.ri.model.jdom - src/test - commons-jxpath-1.3-src
    org.apache.commons.jxpath.servlet - src/java - commons-jxpath-1.3-src
```

Listing 2: Project JXPath - Occurrences of field context

Later at runtime we get to the class `JXPathContext` and inside the method `getContextFactory()` we get an access but since it is declared as static its ignored by the Dynamic Detector.

```
415     public static JXPathContext newContext(Object contextBean) {
416         return getContextFactory().newContext(null, contextBean);
417     }
```

Code segment 9: Project JXPath - Method JXPathContext:newContext

```
437     private static JXPathContextFactory getContextFactory () {
438         if (contextFactory == null) {
439             contextFactory = JXPathContextFactory.newInstance();
440         }
441         return contextFactory;
442     }
```

Code segment 10: Project JXPath - Method JXPathContext.getContextFactory

Inside the class `PackageFunctions` the field `classPrefix` appears which is a member of our Static Detection output.
The field namespace is not saved since the field has the value null.

Note: If you have a look at the `MetaClass`. `FieldwriteTraps` which have a value null are skipped. Otherwise we would get a `NullPointerException` if we want to `getClass()` of a null object.

```
70 public class PackageFunctions implements Functions {
71     private String classPrefix;
72     private String namespace;
73     private static final Object[] EMPTY_ARRAY = new Object[0];
74
75     /**
76      * Create a new PackageFunctions.
77      * @param classPrefix class prefix
78      * @param namespace namespace String
79      */
80     public PackageFunctions(String classPrefix, String namespace) {
81         this.classPrefix = classPrefix;
82         this.namespace = namespace;
83     }
```

Code segment 11: Project JXPath - Constructor PackageFunctions

Last but not least we reach the second constructor of class `QName` (at the top of the stacktrace) which contains the last two detected fields: `name` and `qualifiedName`.

```
29 public class QName implements Serializable {
30     private static final long serialVersionUID = 7616199282015091496L;
31
32     private String prefix;
33     private String name;
34     private String qualifiedName;
35
36     /**
37      * Create a new QName.
38      * @param qualifiedName value
39      */
40     public QName(String qualifiedName) {
41         this.qualifiedName = qualifiedName;
42         int index = qualifiedName.indexOf(':');
43         prefix = index < 0 ? null : qualifiedName.substring(0, index);
44         name = index < 0 ? qualifiedName : qualifiedName.substring(index + 1);
45     }
46
47     /**
48      * Create a new QName.
49      * @param prefix ns
50      * @param localName String
51      */
52     public QName(String prefix, String localName) {
53         this.prefix = prefix;
54         this.name = localName;
55         this.qualifiedName = prefix == null ? localName : prefix + ':' + localName;
56     }
```

Code segment 12: Project JXPath - Constructors QName

The field called `prefix` has the value null when we have a look at the debugger's field value and is therefore ignored by the Dynamic Detector.

3.3.2 Apache Commons Collections

To get some polymorphism we have to analyze a much larger project with good test coverage. The Apache Commons Collection library contains 39'143 unit tests.

After analyzing the Static Detector output, we see that we have guessed two fields as polymorphic at compile time:

```
KEY:
org.apache.commons.collections.map.AbstractHashedMap$HashEntry:org.a
pache.commons.collections.map.AbstractHashedMap$HashEntry:next
VALUE(S):
org.apache.commons.collections.map.AbstractHashedMap$HashEntry
VALUE(S):
org.apache.commons.collections.map.AbstractHashedMap$HashEntry[]
```

(see static output table line 82)

```
KEY:
org.apache.commons.collections.ReferenceMap$Entry:org.apache.commons
.collections.ReferenceMap$Entry:next
VALUE(S): org.apache.commons.collections.ReferenceMap$Entry
VALUE(S): org.apache.commons.collections.ReferenceMap$Entry[]
```

(see static output table line 122)

Note: The $ symbol stands for a separator between public class and private class inside the same .class file.

The fields have both two values and are guessed as polymorphic by our implemented Static Detector.

The first field called `next` of class `AbstractHashedMap$HashEntry` has the value types `HashEntry` and `HashEntry[]`. But if we have a closer look at the code where the values of `next` appear, we can see that the value of the field `entry.next` is field `data` (at code segment 13 - line 472). `Data` is declared as a `HashEntry[]` (HashEntry Array). But at code segment 13 - line 472 we do not assign the Array `data` to the field `next`, but an element of `data`, which is a `HashEntry`.
This case is not detected by our implementation of the Static Detector. Javassist does not offer an API, which can detect array element accesses at source level.
So the as polymorphic guessed field next is not polymorphic because it has only one value type in both methods, which is `HashEntry`.

```
470     protected void reuseEntry(HashEntry entry, int hashIndex,
471                               int hashCode, Object key, Object value) {
472         entry.next = data[hashIndex];
473         entry.hashCode = hashCode;
474         entry.key = key;
475         entry.value = value;
476     }
```

Code segment 13: Project Collections - Method AbstractHashedMap.reuseEntry

```
559     protected void removeEntry
560     (HashEntry entry, int hashIndex, HashEntry previous) {
561         if (previous == null) {
562             data[hashIndex] = entry.next;
563         } else {
564             previous.next = entry.next;
565         }
566     }
```

Code segment 14: Project Collections - Method AbstractHashedMap.removeEntry

Further we have the problem that the field `previous.next` at code segment 14 - line 564 inside method `removeEntry()` is not the same instance as `entry.next`. But due to our field renaming problem of Javassist, we can not distinct them. Our static implementation detects the field `next` of the instances `previous` and `entry` as the same field instance.

Pretty the same is the case with the second polymorphic guessed field:

`org.apache.commons.collections.ReferenceMap$Entry:`
`org.apache.commons.collections.ReferenceMap$Entry:`**`next`**

The field `next` at code segment 15 - line 423 of method `resize()`, has the value type `Entry` and the value is of type `Entry` and not `Entry[]`.

```
413     private void resize() {
414         Entry[] old = table;
415         table = new Entry[old.length * 2];
416
417         for (int i = 0; i < old.length; i++) {
418             Entry next = old[i];
419             while (next != null) {
420                 Entry entry = next;
421                 next = next.next;
422                 int index = indexFor(entry.hash);
423                 entry.next = table[index];
424                 table[index] = entry;
425             }
426             old[i] = null;
427         }
428         threshold = (int)(table.length * loadFactor);
429     }
```

Code segment 15: Project Collections - Method ReferenceMap.resize

And in the method body of `purge()` at code segment 16 - line 465 we get the same access as in the class `AbstractHashedMap$HashEntry`.

```
454     private void purge(Reference ref) {
455         // The hashCode of the reference is the hashCode of the
456         // mapping key, even if the reference refers to the
457         // mapping value...
458         int hash = ref.hashCode();
459         int index = indexFor(hash);
460         Entry previous = null;
461         Entry entry = table[index];
462         while (entry != null) {
463             if (entry.purge(ref)) {
464                 if (previous == null) table[index] = entry.next;
465                 else previous.next = entry.next;
466                 this.size--;
467                 return;
468             }
469             previous = entry;
470             entry = entry.next;
471         }
472
473     }
```

Code segment 16: Project Collections - Method ReferenceMap.purge

Again we have no polymorphic fields at compile time. The two fields we assumed to be polymorphic are monomorphic due to detecting the type of an array access to an element as an Array.

We will now compare our guessing result with the runtime result.

The Dynamic Detector detects two fields (`sortedKeys` and `sortedValues`) as polymorphic at runtime:

`org.apache.commons.collections.bidimap.TestDualTreeBidiMap:Java.util.List:`**`sortedKeys`**

`org.apache.commons.collections.bidimap.TestDualTreeBidiMap:Java.util.List:`**`sortedValues`**

`org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:Java.util.List:`**`sortedKeys`**

`org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:Java.util.List:`**`sortedValues`**

`org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:Java.util.List:`**`sortedKeys`**

`org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:Java.util.List:`**`sortedValues`**

The classes `TestDualTreeBidiMap`, `TestDualTreeBidiMap2`, `TestUnmodifiableSortedBidiMap` are derived classes of `AbstractTestSortedBidiMap`.

The parent class `AbstractTestSortedBidiMap` has the `protected` fields `sortedKeys` and `sortedValues`, so the fields are shared with the subclasses.

The fields themselves are declared in the parent class `AbstractTestSortedBidiMap`. But the fields are detected several times as polymorphic because every constructor of the derived classes calls via “super” the parent constructor. Inside the constructor of the parent class `AbstractTestSortedBidiMap`, the fields `sortedKeys` and `sortedValues` are assigned.
In the following screenshot we can see that the fields: `sortedKeys` and `sortedValues` of type `Java.util.List` have the value `Java.util.ArrayList` (at code segment 17 - lines 45 and 46).
Inside the constructor `AbstractTestSortedBidiMap()` the fields have the new value `Collection.unmodifableList()`. This method returns the value type `List`.

```
43 public abstract class AbstractTestSortedBidiMap extends AbstractTestOrderedBidiMap {
44
45     protected List sortedKeys = new ArrayList();
46     protected List sortedValues = new ArrayList();
47     protected SortedSet sortedNewValues = new TreeSet();
48
49     public AbstractTestSortedBidiMap(String testName) {
50         super(testName);
51         sortedKeys.addAll(Arrays.asList(getSampleKeys()));
52         Collections.sort(sortedKeys);
53         sortedKeys = Collections.unmodifiableList(sortedKeys);
54
55         Map map = new TreeMap();
56         for (int i = 0; i < getSampleKeys().length; i++) {
57             map.put(getSampleKeys()[i], getSampleValues()[i]);
58         }
59         sortedValues.addAll(map.values());
60         sortedValues = Collections.unmodifiableList(sortedValues);
61
62         sortedNewValues.addAll(Arrays.asList(getNewSampleValues()));
63     }
```

Code segment 17: Project Collections - Constructor AbstractTestSortedBidiMap

The object `Java.util.ArrayList` is a derived class of `List`. So the fields: `sortedKeys` and `sortedValues` are polymorphic via inheritance, since they both have two value types `ArrayList` and `List`.

But why are these fields not detected as polymorphic via the Static Detector?
As already mentioned the access at code segment 17 - line 45 and 46 are not registered by the static detector, since the value at this line is not a field. If we could save this access with the Javassist API we would get these fields guessed as polymorphic.

Then the Static Detector saved one access:

```
KEY: org.apache.commons.collections.bidimap.
     AbstractTestSortedBidiMap:Java.util.List:sortedKeys
VALUE(S): Java.util.List
```
(see static result table line 10)

```
KEY: org.apache.commons.collections.bidimap.
     AbstractTestSortedBidiMap:Java.util.List:sortedValues
VALUE(S): Java.util.List
```
(see static result table line 11)

So we only need one more value type to guess these fields as polymorphic at compile-time. That would be the case if we could save the accesses at code segment 17 - line 45 and 46.

3.3.3 P2 Snakes and Ladders

We choose this project as source for our detectors, since the program "Snakes and Ladders" runs the game itself via unit tests. So we have very good unit test coverage, because not only small units of the application are tested, but the whole game itself.

Static Detector Output:

	A	B
1	KEY: snakes.Player:snakes.ISquare:square	VALUE(S): snakes.ISquare
2		
3	***Polymorphic Fields:***	
4	***Polymorphic Fields end***	
5		

Table 1: Static Detector output (Excel table)

The field `square` of type `ISquare` has only one assigned value and is therefore monomorphic at development time.

	A	B
1	KEY: snakes.FirstSquare:java.util.List:players	VALUE(S): java.util.ArrayList
2	KEY: snakes.FirstSquare:snakes.Game:game	VALUE(S): snakes.Game
3	KEY: snakes.Game:java.util.List:squares	VALUE(S): java.util.ArrayList
4	KEY: snakes.Game:java.util.Queue:players	VALUE(S): java.util.LinkedList
5	KEY: snakes.Game:snakes.Player:winner	VALUE(S): snakes.Player
6	KEY: snakes.Ladder:snakes.Game:game	VALUE(S): snakes.Game
7	KEY: snakes.LastSquare:snakes.Game:game	VALUE(S): snakes.Game
8	KEY: snakes.LastSquare:snakes.Player:player	VALUE(S): snakes.Player
9	KEY: snakes.Player:java.lang.String:name	VALUE(S): java.lang.String
10	KEY: snakes.Player:snakes.ISquare:square	VALUE(S): snakes.FirstSquare
11		VALUE(S): snakes.Square
12		VALUE(S): snakes.LastSquare
13	KEY: snakes.SimpleGameTest:snakes.Player:jack	VALUE(S): snakes.Player
14	KEY: snakes.SimpleGameTest:snakes.Player:jill	VALUE(S): snakes.Player
15	KEY: snakes.Snake:snakes.Game:game	VALUE(S): snakes.Game
16	KEY: snakes.Square:snakes.Game:game	VALUE(S): snakes.Game
17	KEY: snakes.Square:snakes.Player:player	VALUE(S): snakes.Player
18		
19	***Polymorphic Fields:***	
20	snakes.Player:snakes.ISquare:square	
21	***Polymorphic Fields end***	
22		

Table 2: Dynamic Detector output (Excel table)

At runtime we get three different value types for the fields `square`: `FirstSquare`, `Square` and `LastSquare`. The field `square` is polymorphic via interface and inheritance. The classes `FirstSquare` and `LastSquare` are subclasses of the class `Square`. And the class `Square` implements the interface `ISquare`.
Inside the class `snakes.Player` where the polymorphic field appears we have the field `square` as a writer-access at code segment 18 - line 19 and 32. The methods `firstSquare()` at line 19 and `moveAndLand()` at line 32 return both a instance of `ISquare` and can therefore be the type `LastSquare`, `Square` or `FirstSquare`.

```
1  package snakes;
2
3  public class Player {
4
5      private String name;
6      private ISquare square;
7
8      private boolean invariant() {
9          return name != null
10             && square != null;
11     }
12
13     public Player(String name) {
14         this.name = name;
15         // invariant holds only after joining a game
16     }
17
18     public void joinGame(Game game) {
19         square = game.firstSquare();
20         square.enter(this);
21         assert invariant();
22     }
23
24     public int position() {
25         assert invariant();
26         return square.position();
27     }
28
29     public void moveForward(int moves) {
30         assert moves > 0;
31         square.leave(this);
32         square = square.moveAndLand(moves);
33         square.enter(this);
34     }
35
```

Code segment 18: Project S&L - Class Player

If we would count the heuristic “fields with interface type are polymorphic” we have cancelled, we would get in this very project a better coverage of guessed and detected polymorphic fields.

3.4 Further Projects

Other Apache Commons Libraries has been analyzed but they had no polymorphic fields whether in the Static nor in the Dynamic case.

Project name	**#Unittests**	**#Errors**	**#Failures**	**Runner**
Codec 1.8	616	52	8	Unittests only
Pool 1.6	266			Unittests only
Daemon 1.0.15	None	X	X	Generate a simpleDaemon instance
CLI 1.2	187	0	0	Unittests only

For static and dynamic output see Appendix - Outputs of the projects listed above.

4 Conclusion

Project	# polymorphic fields at compile time (static detection)	# polymorphic fields at runtime (dynamic detection)
Apache Commons JXPath 1.3	0	0
Apache Commons Collections 3.2.1	0	2
Apache Commons Codec 1.8	0	0
Apache Commons Pool 1.6	0	0
Apache Commons Daemon 1.0.15	0	0
Apache Commons CLI 1.2	0	0
P2 Snakes and Ladders	0	1

Most projects we analyzed, have no polymorphism whether at compile time or at runtime. This is the case if we run the unit tests of a project to get the runtime. We used the unit tests for the dynamic case because they have no interaction with the user. The problem of running unit tests to get the dynamic detection results is that the result depends on the test coverage. But that's not all. A Java unit test typically covers a single method or small procedure. So if we run our detection on this unit test we simply run a method which holds often more local variables than fields in the body. So the chance to catch a field access in unit tests is very small and a lot smaller if we are looking for polymorphic fields. It would be better to run the main of the project directly. But if we run the projects without the unit tests we have the problem that the simulative run of Javassist is in a separate JVM. Because of this fact it is not possible to run a system like for example an Editor where a lot of interactions occur between system and user with Javassist. More we would have to define specific use-cases to run the system. I have tested to run the projects directly and without the tests but as already mentioned the interaction between user and system crashed the simulative Javassist run.
The Javassist library is also not able to rename all field-name occurrences. So that does not allow us to eliminate field duplications through renaming and avoid field sharing between parent and subclass.
The Javassist library also does not offer an API which makes it possible to get the assigned value of a field access if the access value is not a field itself.

As a conclusion we can say that with Javassist we can get a satisfying result of polymorphism at compile time and runtime, despite the APIs constraint. We gain a pretty good match of static and dynamic polymorphism. But the results are inconclusive as we did not have many projects with demonstrable polymorphism.
But to eventually catch more polymorphism at runtime we should run the system directly via the main class and not the unit tests. But to do so we would have to choose a more powerful bytecode manipulation library than Javassist, which is able to run a simulative run with user interactions.

5 Threats to validity

In our implementation we concentrated on field accesses. But as we know polymorphism can occur in different ways and not only via accesses. Our achieved findings are correct concerning polymorphism via field accesses. Polymorphism can also occure via aliasing. For example if we have a class A and subclass B. Both can be passed via parameter to an external class where internally no field access occurs. So that is indeed a threat to validity. More we save the return value of an method at an access line via the code definition of the method. We do not track down the actually method return type. So we get the abstract return type of the method. For example if we have a method which returns an interface it is possible that the method returns at runtime a class which implements this interface and not the interface itself. This case is not detected by our Static Detector implementation. In our case we just get the return type: interface. If we want to rely on our evaluation we have to consider these threats to validity.

More we run our project analysis via the unit tests. This further threat does not represent the result of the project if we run it via the main method. The actually run of an application via the main method will cover other parts of the source code at runtime than unit tests which often only cover a single method or small procedures. This constraint can produce a wrong estimation of polymorphic fields at runtime.

6 Sources

	Source for Apache Commons Project (freeware):
JXPath	http://projects.apache.org/projects/commons_jxpath.html
Collections	http://projects.apache.org/projects/commons_collections.html
Codec	http://projects.apache.org/projects/commons_codec.html
Pool	http://projects.apache.org/projects/commons_pool.html
Daemon	http://projects.apache.org/projects/commons_daemon.html
CLI	http://projects.apache.org/projects/commons_cli.html
	Project at GitHub
Polymorphism Detector	https://github.com/mmorelli/PolymorphismDetection
External Projects	https://github.com/mmorelli/External-Projects

7 Appendix – Outputs

7.1 JXPath Static

	A	B
1	KEY: org.apache.commons.jxpath.BasicNodeSet:java.util.List:nodes	VALUE(S): java.util.List
2	KEY: org.apache.commons.jxpath.BasicNodeSet:java.util.List:readOnlyPointers	VALUE(S): java.util.List
3	KEY: org.apache.commons.jxpath.BasicNodeSet:java.util.List:values	VALUE(S): java.util.List
4	KEY: org.apache.commons.jxpath.ri.axes.AncestorContext:org.apache.commons.jxpath.ri.model.NodePointer:currentNodePointer	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
5	KEY: org.apache.commons.jxpath.ri.axes.AttributeContext:org.apache.commons.jxpath.ri.model.NodeIterator:iterator	VALUE(S): org.apache.commons.jxpath.ri.model.NodeIterator
6	KEY: org.apache.commons.jxpath.ri.axes.AttributeContext:org.apache.commons.jxpath.ri.model.NodePointer:currentNodePointer	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
7	KEY: org.apache.commons.jxpath.ri.axes.ChildContext:org.apache.commons.jxpath.ri.model.NodeIterator:iterator	VALUE(S): org.apache.commons.jxpath.ri.model.NodeIterator
8	KEY: org.apache.commons.jxpath.ri.axes.DescendantContext:org.apache.commons.jxpath.ri.model.NodePointer:currentNodePointer	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
9	KEY: org.apache.commons.jxpath.ri.axes.NamespaceContext:org.apache.commons.jxpath.ri.model.NodePointer:currentNodePointer	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
10	KEY: org.apache.commons.jxpath.ri.axes.ParentContext:org.apache.commons.jxpath.ri.model.NodePointer:currentNodePointer	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
11	KEY: org.apache.commons.jxpath.ri.axes.PrecedingOrFollowingContext:org.apache.commons.jxpath.ri.model.NodePointer:currentRootLocation	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
12	KEY: org.apache.commons.jxpath.ri.axes.RecursiveAxesTest:org.apache.commons.jxpath.JXPathContext:context	VALUE(S): org.apache.commons.jxpath.JXPathContext
13	KEY: org.apache.commons.jxpath.ri.axes.SelfContext:org.apache.commons.jxpath.ri.model.NodePointer:nodePointer	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
14	KEY: org.apache.commons.jxpath.ri.axes.SimplePathInterpreterTest:org.apache.commons.jxpath.JXPathContext:context	VALUE(S): org.apache.commons.jxpath.JXPathContext
15	KEY: org.apache.commons.jxpath.ri.compiler.ExtensionFunctionTest:org.apache.commons.jxpath.JXPathContext:context	VALUE(S): org.apache.commons.jxpath.JXPathContext
16	KEY: org.apache.commons.jxpath.ri.EvalContext:org.apache.commons.jxpath.ri.axes.RootContext:rootContext	VALUE(S): org.apache.commons.jxpath.ri.axes.RootContext
17	KEY: org.apache.commons.jxpath.ri.JXPathContextReferenceImpl:org.apache.commons.jxpath.Pointer:rootPointer	VALUE(S): org.apache.commons.jxpath.Pointer
18	KEY: org.apache.commons.jxpath.ri.JXPathContextReferenceImpl:org.apache.commons.jxpath.ri.NamespaceResolver:namespaceResolver	VALUE(S): org.apache.commons.jxpath.ri.NamespaceResolver
19	KEY: org.apache.commons.jxpath.ri.model.beans.BeanPropertyPointer:java.beans.PropertyDescriptor[]:propertyDescriptors	VALUE(S): java.beans.PropertyDescriptor[]
20	KEY: org.apache.commons.jxpath.ri.model.dom.DOMNodeIterator:org.w3c.dom.Node:child	VALUE(S): org.w3c.dom.Node
21	KEY: org.apache.commons.jxpath.ri.model.dom.NamespacePointer:java.lang.String:namespaceURI	VALUE(S): java.lang.String
22	KEY: org.apache.commons.jxpath.ri.model.jdom.JDOMNamespacePointer:java.lang.String:namespaceURI	VALUE(S): java.lang.String
23	KEY: org.apache.commons.jxpath.ri.model.jdom.JDOMNodeIterator:java.lang.Object:child	VALUE(S): java.lang.Object
24	KEY: org.apache.commons.jxpath.ri.model.NodePointer:java.lang.Object:rootNode	VALUE(S): java.lang.Object
25	KEY: org.apache.commons.jxpath.ri.model.NodePointer:java.util.Locale:locale	VALUE(S): java.util.Locale
26	KEY: org.apache.commons.jxpath.ri.model.NodePointer:org.apache.commons.jxpath.ri.model.NodePointer:parent	VALUE(S): org.apache.commons.jxpath.ri.model.NodePointer
27	KEY: org.apache.commons.jxpath.ri.model.NodePointer:org.apache.commons.jxpath.ri.NamespaceResolver:namespaceResolver	VALUE(S): org.apache.commons.jxpath.ri.NamespaceResolver
28	KEY: org.apache.commons.jxpath.ri.parser.Token:java.lang.String:image	VALUE(S): java.lang.String
29	KEY: org.apache.commons.jxpath.ri.parser.Token:org.apache.commons.jxpath.ri.parser.Token:next	VALUE(S): org.apache.commons.jxpath.ri.parser.Token
30	KEY: org.apache.commons.jxpath.ri.parser.XPathParser$JJCalls:org.apache.commons.jxpath.ri.parser.Token:first	VALUE(S): org.apache.commons.jxpath.ri.parser.Token
31	KEY: org.apache.commons.jxpath.ri.parser.XPathParser$JJCalls:org.apache.commons.jxpath.ri.parser.XPathParser$JJCalls:next	VALUE(S): org.apache.commons.jxpath.ri.parser.XPathParser$JJCalls
32	KEY: org.apache.commons.jxpath.ri.parser.XPathParser:org.apache.commons.jxpath.ri.parser.Token:jj_lastpos	VALUE(S): org.apache.commons.jxpath.ri.parser.Token
33	KEY: org.apache.commons.jxpath.ri.parser.XPathParser:org.apache.commons.jxpath.ri.parser.Token:jj_nt	VALUE(S): org.apache.commons.jxpath.ri.parser.Token
34	KEY: org.apache.commons.jxpath.ri.parser.XPathParser:org.apache.commons.jxpath.ri.parser.Token:jj_scanpos	VALUE(S): org.apache.commons.jxpath.ri.parser.Token
35	KEY: org.apache.commons.jxpath.ri.parser.XPathParser:org.apache.commons.jxpath.ri.parser.XPathParserTokenManager:token_source	VALUE(S): org.apache.commons.jxpath.ri.parser.SimpleCharStream
36	KEY: org.apache.commons.jxpath.TestBean:int[]:array	VALUE(S): int[]
37	KEY: org.apache.commons.jxpath.TestMixedModelBean:org.w3c.dom.Document:document	VALUE(S): org.w3c.dom.Document
38	KEY: org.apache.commons.jxpath.TestMixedModelBean:org.w3c.dom.Element:element	VALUE(S): org.w3c.dom.Element
39	KEY: org.apache.commons.jxpath.util.BasicTypeConverter$ValueNodeSet:java.util.List:pointers	VALUE(S): java.util.List
40	KEY: org.apache.commons.jxpath.XMLDocumentContainer:java.lang.Object:document	VALUE(S): java.lang.Object

7.2 JXPath Dynamic

	A	B
1	KEY: org.apache.commons.jxpath.NestedTestBean:java.lang.String:name	VALUE(S): java.lang.String
2	KEY: org.apache.commons.jxpath.NestedTestBean:java.lang.String[]:strings	VALUE(S): [Ljava.lang.String;
3	KEY: org.apache.commons.jxpath.PackageFunctions:java.lang.String:classPrefix	VALUE(S): java.lang.String
4	KEY: org.apache.commons.jxpath.ri.axes.RecursiveAxesTest:org.apache.commons.jxpath.ri.axes.RecursiveBean:bean	VALUE(S): org.apache.commons.jxpath.ri.axes.RecursiveBean
5	KEY: org.apache.commons.jxpath.ri.axes.RecursiveBean:java.lang.String:name	VALUE(S): java.lang.String
6	KEY: org.apache.commons.jxpath.ri.axes.RecursiveBean:org.apache.commons.jxpath.ri.axes.RecursiveBean:first	VALUE(S): org.apache.commons.jxpath.ri.axes.RecursiveBean
7	KEY: org.apache.commons.jxpath.ri.axes.RecursiveBean:org.apache.commons.jxpath.ri.axes.RecursiveBean:second	VALUE(S): org.apache.commons.jxpath.ri.axes.RecursiveBean
8	KEY: org.apache.commons.jxpath.ri.axes.TestBeanWithNode:int[]:array	VALUE(S): [I
9	KEY: org.apache.commons.jxpath.ri.axes.TestBeanWithNode:java.lang.Object:object	VALUE(S): org.apache.commons.jxpath.NestedTestBean
10	KEY: org.apache.commons.jxpath.ri.axes.TestBeanWithNode:java.util.HashMap:map	VALUE(S): java.util.HashMap
11	KEY: org.apache.commons.jxpath.ri.axes.TestBeanWithNode:org.apache.commons.jxpath.NestedTestBean:nestedBean	VALUE(S): org.apache.commons.jxpath.NestedTestBean
12	KEY: org.apache.commons.jxpath.ri.axes.TestBeanWithNode:org.apache.commons.jxpath.NestedTestBean[]:beans	VALUE(S): [Lorg.apache.commons.jxpath.NestedTestBean;
13	KEY: org.apache.commons.jxpath.ri.compiler.ExtensionFunctionTest:org.apache.commons.jxpath.TestBean:testBean	VALUE(S): org.apache.commons.jxpath.TestBean
14	KEY: org.apache.commons.jxpath.ri.QName:java.lang.String:name	VALUE(S): java.lang.String
15	KEY: org.apache.commons.jxpath.ri.QName:java.lang.String:qualifiedName	VALUE(S): java.lang.String
16	KEY: org.apache.commons.jxpath.TestBean:int[]:array	VALUE(S): [I
17	KEY: org.apache.commons.jxpath.TestBean:java.util.HashMap:map	VALUE(S): java.util.HashMap
18	KEY: org.apache.commons.jxpath.TestBean:org.apache.commons.jxpath.NestedTestBean:nestedBean	VALUE(S): org.apache.commons.jxpath.NestedTestBean
19	KEY: org.apache.commons.jxpath.TestBean:org.apache.commons.jxpath.NestedTestBean:object	VALUE(S): org.apache.commons.jxpath.NestedTestBean
20	KEY: org.apache.commons.jxpath.TestBean:org.apache.commons.jxpath.NestedTestBean[]:beans	VALUE(S): [Lorg.apache.commons.jxpath.NestedTestBean;
21	KEY: org.apache.commons.jxpath.TestMixedModelBean:java.lang.String:string	VALUE(S): java.lang.String
22	KEY: org.apache.commons.jxpath.TestMixedModelBean:java.util.List:list	VALUE(S): java.util.ArrayList
23	KEY: org.apache.commons.jxpath.TestMixedModelBean:java.util.Map:map	VALUE(S): java.util.HashMap
24	KEY: org.apache.commons.jxpath.TestMixedModelBean:org.apache.commons.jxpath.TestBean:bean	VALUE(S): org.apache.commons.jxpath.TestBean

7.3 Collections Static

	A	B
1	KEY: org.apache.commons.collections.bag.AbstractMapBag$BagIterator:java.util.Map$Entry:current	VALUE(S): java.util.Map$Entry
2	KEY: org.apache.commons.collections.bag.AbstractMapBag:java.util.Set:uniqueSet	VALUE(S): java.util.Set
3	KEY: org.apache.commons.collections.bag.TestTypedBag:java.lang.Class:objectClass	VALUE(S): java.lang.Class
4	KEY: org.apache.commons.collections.bag.TestTypedSortedBag:java.lang.Class:objectClass	VALUE(S): java.lang.Class
5	KEY: org.apache.commons.collections.BeanMap:java.util.HashMap:readMethods	VALUE(S): java.util.HashMap
6	KEY: org.apache.commons.collections.bidimap.AbstractDualBidiMap$BidiMapIterator:java.util.Iterator:iterator	VALUE(S): java.util.Iterator
7	KEY: org.apache.commons.collections.bidimap.AbstractDualBidiMap$BidiMapIterator:java.util.Map$Entry:last	VALUE(S): java.util.Map$Entry
8	KEY: org.apache.commons.collections.bidimap.AbstractDualBidiMap$EntrySetIterator:java.util.Map$Entry:last	VALUE(S): java.util.Map$Entry
9	KEY: org.apache.commons.collections.bidimap.AbstractDualBidiMap:org.apache.commons.collections.BidiMap:inverseBidiMap	VALUE(S): org.apache.commons.collections.BidiMap
10	KEY: org.apache.commons.collections.bidimap.AbstractTestSortedBidiMap:java.util.List:sortedKeys	VALUE(S): java.util.List
11	KEY: org.apache.commons.collections.bidimap.AbstractTestSortedBidiMap:java.util.List:sortedValues	VALUE(S): java.util.List
12	KEY: org.apache.commons.collections.bidimap.DualTreeBidiMap$BidiOrderedMapIterator:java.util.ListIterator:iterator	VALUE(S): java.util.ListIterator
13	KEY: org.apache.commons.collections.bidimap.DualTreeBidiMap$BidiOrderedMapIterator:java.util.Map$Entry:last	VALUE(S): java.util.Map$Entry
14	KEY: org.apache.commons.collections.bidimap.DualTreeBidiMap$ViewMap:org.apache.commons.collections.bidimap.DualTreeBidiMap:bidi	VALUE(S): org.apache.commons.collections.bidimap.DualTreeBidiMap
15	KEY: org.apache.commons.collections.bidimap.TestAbstractOrderedBidiMapDecorator$TestOrderedBidiMap:org.apache.commons.collections.bidimap.TestAbstrac	VALUE(S): org.apache.commons.collections.bidimap.TestAbstractOrderedBidiMapDecorator$TestOrderedBidiMap
16	KEY: org.apache.commons.collections.bidimap.TreeBidiMap$Inverse:java.util.Set:keySet	VALUE(S): org.apache.commons.collections.bidimap.TreeBidiMap
17	KEY: org.apache.commons.collections.bidimap.TreeBidiMap$Inverse:java.util.Set:valuesSet	VALUE(S): org.apache.commons.collections.bidimap.TreeBidiMap
18	KEY: org.apache.commons.collections.bidimap.TreeBidiMap$ViewIterator:org.apache.commons.collections.bidimap.TreeBidiMap$Node:lastReturnedNode	VALUE(S): org.apache.commons.collections.bidimap.TreeBidiMap$Node
19	KEY: org.apache.commons.collections.bidimap.TreeBidiMap$ViewIterator:org.apache.commons.collections.bidimap.TreeBidiMap$Node:nextNode	VALUE(S): org.apache.commons.collections.bidimap.TreeBidiMap$Node
20	KEY: org.apache.commons.collections.bidimap.TreeBidiMap$ViewIterator:org.apache.commons.collections.bidimap.TreeBidiMap$Node:previousNode	VALUE(S): org.apache.commons.collections.bidimap.TreeBidiMap$Node
21	KEY: org.apache.commons.collections.bidimap.TreeBidiMap:org.apache.commons.collections.bidimap.TreeBidiMap$Node[]:rootNode	VALUE(S): org.apache.commons.collections.bidimap.TreeBidiMap$Node[]
22	KEY: org.apache.commons.collections.bidimap.UnmodifiableBidiMap:org.apache.commons.collections.bidimap.UnmodifiableBidiMap:inverse	VALUE(S): org.apache.commons.collections.bidimap.UnmodifiableBidiMap
23	KEY: org.apache.commons.collections.bidimap.UnmodifiableOrderedBidiMap:org.apache.commons.collections.bidimap.UnmodifiableOrderedBidiMap:inverse	VALUE(S): org.apache.commons.collections.bidimap.UnmodifiableOrderedBidiMap
24	KEY: org.apache.commons.collections.bidimap.UnmodifiableSortedBidiMap:org.apache.commons.collections.bidimap.UnmodifiableSortedBidiMap:inverse	VALUE(S): org.apache.commons.collections.bidimap.UnmodifiableSortedBidiMap
25	KEY: org.apache.commons.collections.BinaryHeap:java.lang.Object[]:m_elements	VALUE(S): java.lang.Object[]
26	KEY: org.apache.commons.collections.buffer.PriorityBuffer:java.lang.Object[]:elements	VALUE(S): java.lang.Object[]
27	KEY: org.apache.commons.collections.BulkTest:java.lang.String:verboseName	VALUE(S): java.lang.String
28	KEY: org.apache.commons.collections.BulkTestSuiteMaker:java.lang.String:prefix	VALUE(S): java.lang.String
29	KEY: org.apache.commons.collections.CursorableLinkedList$ListIter:org.apache.commons.collections.CursorableLinkedList$Listable:_cur	VALUE(S): org.apache.commons.collections.CursorableLinkedList$Listable
30	KEY: org.apache.commons.collections.CursorableLinkedList$ListIter:org.apache.commons.collections.CursorableLinkedList$Listable:_lastReturned	VALUE(S): org.apache.commons.collections.CursorableLinkedList$Listable
31	KEY: org.apache.commons.collections.CursorableSubList:org.apache.commons.collections.CursorableLinkedList$Listable:_post	VALUE(S): org.apache.commons.collections.CursorableLinkedList$Listable
32	KEY: org.apache.commons.collections.CursorableSubList:org.apache.commons.collections.CursorableLinkedList$Listable:_pre	VALUE(S): org.apache.commons.collections.CursorableLinkedList$Listable
33	KEY: org.apache.commons.collections.DefaultMapBag$BagIterator:java.lang.Object:_current	VALUE(S): java.lang.Object
34	KEY: org.apache.commons.collections.DoubleOrderedMap$DoubleOrderedMapIterator:org.apache.commons.collections.DoubleOrderedMap$Node:lastReturnedNo	VALUE(S): org.apache.commons.collections.DoubleOrderedMap$Node
35	KEY: org.apache.commons.collections.DoubleOrderedMap$DoubleOrderedMapIterator:org.apache.commons.collections.DoubleOrderedMap$Node:nextNode	VALUE(S): org.apache.commons.collections.DoubleOrderedMap$Node
36	KEY: org.apache.commons.collections.DoubleOrderedMap:org.apache.commons.collections.DoubleOrderedMap$Node[]:rootNode	VALUE(S): org.apache.commons.collections.DoubleOrderedMap$Node[]
37	KEY: org.apache.commons.collections.ExtendedProperties:java.lang.String:basePath	VALUE(S): java.lang.String
38	KEY: org.apache.commons.collections.FastArrayList$ListIter:java.util.List:expected	VALUE(S): java.util.ArrayList
39	KEY: org.apache.commons.collections.FastArrayList$SubList$SubListIter:java.util.List:expected	VALUE(S): org.apache.commons.collections.FastArrayList
40	KEY: org.apache.commons.collections.FastArrayList$SubList$SubListIter:java.util.ListIterator:iter	VALUE(S): java.util.ListIterator
41	KEY: org.apache.commons.collections.FastArrayList$SubList:java.util.List:expected	VALUE(S): java.util.ArrayList
42	KEY: org.apache.commons.collections.FastArrayList:java.util.ArrayList:list	VALUE(S): org.apache.commons.collections.FastArrayList
43	KEY: org.apache.commons.collections.FastHashMap$CollectionView$CollectionViewIterator:java.util.Iterator:iterator	VALUE(S): java.util.Iterator
44	KEY: org.apache.commons.collections.FastHashMap$CollectionView$CollectionViewIterator:java.util.Map$Entry:lastReturned	VALUE(S): java.util.Map$Entry
45	KEY: org.apache.commons.collections.FastHashMap$CollectionView$CollectionViewIterator:java.util.Map:expected	VALUE(S): org.apache.commons.collections.FastHashMap
46	KEY: org.apache.commons.collections.FastHashMap:java.util.HashMap:map	VALUE(S): org.apache.commons.collections.FastHashMap
47	KEY: org.apache.commons.collections.FastTreeMap$CollectionView$CollectionViewIterator:java.util.Iterator:iterator	VALUE(S): java.util.Iterator
48	KEY: org.apache.commons.collections.FastTreeMap$CollectionView$CollectionViewIterator:java.util.Map$Entry:lastReturned	VALUE(S): java.util.Map$Entry
49	KEY: org.apache.commons.collections.FastTreeMap$CollectionView$CollectionViewIterator:java.util.Map:expected	VALUE(S): org.apache.commons.collections.FastTreeMap
50	KEY: org.apache.commons.collections.FastTreeMap:java.util.TreeMap:map	VALUE(S): org.apache.commons.collections.FastTreeMap

	A	B
51	KEY: org.apache.commons.collections.functors.InstantiateFactory:java.lang.reflect.Constructor:iConstructor	VALUE(S): java.lang.reflect.Constructor
52	KEY: org.apache.commons.collections.functors.PrototypeFactory$PrototypeCloneFactory:java.lang.reflect.Method:iCloneMethod	VALUE(S): java.lang.reflect.Method
53	KEY: org.apache.commons.collections.iterators.CollatingIterator:java.util.ArrayList:values	VALUE(S): java.util.ArrayList
54	KEY: org.apache.commons.collections.iterators.CollatingIterator:java.util.BitSet:valueSet	VALUE(S): java.util.ArrayList
55	KEY: org.apache.commons.collections.iterators.EntrySetMapIterator:java.util.Iterator:iterator	VALUE(S): java.util.Iterator
56	KEY: org.apache.commons.collections.iterators.EntrySetMapIterator:java.util.Map$Entry:last	VALUE(S): java.util.Map$Entry
57	KEY: org.apache.commons.collections.iterators.EnumerationIterator:java.lang.Object:last	VALUE(S): java.lang.Object
58	KEY: org.apache.commons.collections.iterators.IteratorChain:java.util.Iterator:currentIterator	VALUE(S): java.util.Iterator
59	KEY: org.apache.commons.collections.iterators.IteratorChain:java.util.Iterator:lastUsedIterator	VALUE(S): java.util.Iterator
60	KEY: org.apache.commons.collections.iterators.LoopingIterator:java.util.Iterator:iterator	VALUE(S): java.util.Iterator
61	KEY: org.apache.commons.collections.iterators.LoopingListIterator:java.util.ListIterator:iterator	VALUE(S): java.util.ListIterator
62	KEY: org.apache.commons.collections.iterators.ObjectGraphIterator:java.util.Iterator:currentIterator	VALUE(S): java.util.Iterator
63	KEY: org.apache.commons.collections.iterators.ObjectGraphIterator:java.util.Iterator:lastUsedIterator	VALUE(S): java.util.Iterator
64	KEY: org.apache.commons.collections.iterators.ReverseListIterator:java.util.ListIterator:iterator	VALUE(S): java.util.ListIterator
65	KEY: org.apache.commons.collections.iterators.TestFilterIterator:java.util.List:list	VALUE(S): java.util.List
66	KEY: org.apache.commons.collections.iterators.TestUnmodifiableIterator:java.util.List:testList	VALUE(S): java.util.List
67	KEY: org.apache.commons.collections.iterators.TestUnmodifiableListIterator:java.util.List:testList	VALUE(S): java.util.List
68	KEY: org.apache.commons.collections.list.AbstractLinkedList$LinkedListIterator:org.apache.commons.collections.list.AbstractLinkedList$Node:current	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
69	KEY: org.apache.commons.collections.list.AbstractLinkedList$LinkedListIterator:org.apache.commons.collections.list.AbstractLinkedList$Node:next	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
70	KEY: org.apache.commons.collections.list.AbstractLinkedList$Node:org.apache.commons.collections.list.AbstractLinkedList$Node:next	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
71	KEY: org.apache.commons.collections.list.AbstractLinkedList$Node:org.apache.commons.collections.list.AbstractLinkedList$Node:previous	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
72	KEY: org.apache.commons.collections.list.AbstractTestList$BulkTestSubList:java.util.Collection:collection	VALUE(S): java.util.List
73	KEY: org.apache.commons.collections.list.AbstractTestList$BulkTestSubList:java.util.Collection:confirmed	VALUE(S): java.util.List
74	KEY: org.apache.commons.collections.list.CursorableLinkedList$Cursor:org.apache.commons.collections.list.AbstractLinkedList$Node:next	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
75	KEY: org.apache.commons.collections.list.NodeCachingLinkedList:org.apache.commons.collections.list.AbstractLinkedList$Node:firstCachedNode	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
76	KEY: org.apache.commons.collections.list.TreeList$AVLNode:java.lang.Object:value	VALUE(S): java.lang.Object
77	KEY: org.apache.commons.collections.list.TreeList$AVLNode:org.apache.commons.collections.list.TreeList$AVLNode:left	VALUE(S): org.apache.commons.collections.list.TreeList$AVLNode
78	KEY: org.apache.commons.collections.list.TreeList$AVLNode:org.apache.commons.collections.list.TreeList$AVLNode:right	VALUE(S): org.apache.commons.collections.list.TreeList$AVLNode
79	KEY: org.apache.commons.collections.list.TreeList$TreeListIterator:org.apache.commons.collections.list.TreeList$AVLNode:current	VALUE(S): org.apache.commons.collections.list.TreeList$AVLNode
80	KEY: org.apache.commons.collections.list.TreeList$TreeListIterator:org.apache.commons.collections.list.TreeList$AVLNode:next	VALUE(S): org.apache.commons.collections.list.TreeList$AVLNode
81	KEY: org.apache.commons.collections.list.TreeList:org.apache.commons.collections.list.TreeList$AVLNode:root	VALUE(S): org.apache.commons.collections.list.TreeList$AVLNode
82	KEY: org.apache.commons.collections.map.AbstractHashedMap$HashEntry:org.apache.commons.collections.map.AbstractHashedMap$HashEntry:next	VALUE(S): org.apache.commons.collections.map.AbstractHashedMap$HashEntry[]
83		VALUE(S): org.apache.commons.collections.map.AbstractHashedMap$HashEntry
84	KEY: org.apache.commons.collections.map.AbstractHashedMap:org.apache.commons.collections.map.AbstractHashedMap$HashEntry[]:data	VALUE(S): org.apache.commons.collections.map.AbstractHashedMap$HashEntry[]
85	KEY: org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry:org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry:after	VALUE(S): org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry
86	KEY: org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry:org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry:before	VALUE(S): org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry
87	KEY: org.apache.commons.collections.map.AbstractLinkedMap$LinkIterator:org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry:last	VALUE(S): org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry
88	KEY: org.apache.commons.collections.map.AbstractLinkedMap$LinkIterator:org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry:next	VALUE(S): org.apache.commons.collections.map.AbstractLinkedMap$LinkEntry
89	KEY: org.apache.commons.collections.map.AbstractReferenceMap$ReferenceEntry:java.lang.Object:value	VALUE(S): java.lang.Object
90	KEY: org.apache.commons.collections.map.AbstractReferenceMap$ReferenceEntrySetIterator:java.lang.Object:currentKey	VALUE(S): java.lang.Object
91	KEY: org.apache.commons.collections.map.AbstractReferenceMap$ReferenceEntrySetIterator:java.lang.Object:currentValue	VALUE(S): java.lang.Object
92	KEY: org.apache.commons.collections.map.AbstractReferenceMap$ReferenceEntrySetIterator:org.apache.commons.collections.map.AbstractReferenceMap$Refe	VALUE(S): org.apache.commons.collections.map.AbstractReferenceMap$ReferenceEntry
93	KEY: org.apache.commons.collections.map.AbstractReferenceMap$ReferenceEntrySetIterator:org.apache.commons.collections.map.AbstractReferenceMap$Refe	VALUE(S): org.apache.commons.collections.map.AbstractReferenceMap$ReferenceEntry
94	KEY: org.apache.commons.collections.map.AbstractTestMap$TestMapEntrySet:java.util.Collection:collection	VALUE(S): java.util.Set
95	KEY: org.apache.commons.collections.map.AbstractTestMap$TestMapEntrySet:java.util.Collection:confirmed	VALUE(S): java.util.Set
96	KEY: org.apache.commons.collections.map.AbstractTestMap$TestMapKeySet:java.util.Collection:collection	VALUE(S): java.util.Set
97	KEY: org.apache.commons.collections.map.AbstractTestMap$TestMapKeySet:java.util.Collection:confirmed	VALUE(S): java.util.Set
98	KEY: org.apache.commons.collections.map.AbstractTestMap$TestMapValues:java.util.Collection:collection	VALUE(S): java.util.Collection
99	KEY: org.apache.commons.collections.map.AbstractTestMap$TestMapValues:java.util.Collection:confirmed	VALUE(S): java.util.Collection
100	KEY: org.apache.commons.collections.map.AbstractTestMap:java.util.Collection:values	VALUE(S): java.util.Collection

	A	B
101	KEY: org.apache.commons.collections.map.AbstractTestMap:java.util.Set:entrySet	VALUE(S): java.util.Set
102	KEY: org.apache.commons.collections.map.AbstractTestMap:java.util.Set:keySet	VALUE(S): java.util.Set
103	KEY: org.apache.commons.collections.map.AbstractTestSortedMap$TestHeadMap:java.lang.Object:toKey	VALUE(S): java.lang.Object
104	KEY: org.apache.commons.collections.map.AbstractTestSortedMap$TestSubMap:java.lang.Object:fromKey	VALUE(S): java.lang.Object
105	KEY: org.apache.commons.collections.map.AbstractTestSortedMap$TestSubMap:java.lang.Object:toKey	VALUE(S): java.lang.Object
106	KEY: org.apache.commons.collections.map.AbstractTestSortedMap$TestTailMap:java.lang.Object:fromKey	VALUE(S): java.lang.Object
107	KEY: org.apache.commons.collections.map.AbstractTestSortedMap$TestTailMap:java.lang.Object:invalidKey	VALUE(S): java.lang.Object
108	KEY: org.apache.commons.collections.map.Flat3Map:java.lang.Object:key1	VALUE(S): java.lang.Object
109	KEY: org.apache.commons.collections.map.Flat3Map:java.lang.Object:key2	VALUE(S): java.lang.Object
110	KEY: org.apache.commons.collections.map.Flat3Map:java.lang.Object:value1	VALUE(S): java.lang.Object
111	KEY: org.apache.commons.collections.map.Flat3Map:java.lang.Object:value2	VALUE(S): java.lang.Object
112	KEY: org.apache.commons.collections.map.Flat3Map:java.lang.Object:value3	VALUE(S): java.lang.Object
113	KEY: org.apache.commons.collections.map.Flat3Map:org.apache.commons.collections.map.AbstractHashedMap:delegateMap	VALUE(S): org.apache.commons.collections.map.HashedMap
114	KEY: org.apache.commons.collections.map.ListOrderedMap$EntrySetView:java.util.Set:entrySet	VALUE(S): java.util.Set
115	KEY: org.apache.commons.collections.map.ListOrderedMap$ListOrderedMapIterator:java.lang.Object:last	VALUE(S): java.lang.Object
116	KEY: org.apache.commons.collections.map.ListOrderedMap$ListOrderedMapIterator:java.util.ListIterator:iterator	VALUE(S): java.util.ListIterator
117	KEY: org.apache.commons.collections.map.MultiValueMap$ValuesIterator:java.util.Iterator:iterator	VALUE(S): java.util.Iterator
118	KEY: org.apache.commons.collections.map.StaticBucketMap$EntryIterator:java.util.Map$Entry:last	VALUE(S): java.util.Map$Entry
119	KEY: org.apache.commons.collections.map.StaticBucketMap$Node:org.apache.commons.collections.map.StaticBucketMap$Node:next	VALUE(S): org.apache.commons.collections.map.StaticBucketMap$Node
120	KEY: org.apache.commons.collections.MultiHashMap$ValueIterator:java.util.Iterator:tempIterator	VALUE(S): java.util.Collection
121	KEY: org.apache.commons.collections.ReferenceMap$Entry:java.lang.Object:value	VALUE(S): java.lang.Object
122	KEY: org.apache.commons.collections.ReferenceMap$Entry:org.apache.commons.collections.ReferenceMap$Entry:next	VALUE(S): org.apache.commons.collections.ReferenceMap$Entry
123		VALUE(S): org.apache.commons.collections.ReferenceMap$Entry[]
124	KEY: org.apache.commons.collections.ReferenceMap$EntryIterator:java.lang.Object:currentKey	VALUE(S): java.lang.Object
125	KEY: org.apache.commons.collections.ReferenceMap$EntryIterator:java.lang.Object:currentValue	VALUE(S): java.lang.Object
126	KEY: org.apache.commons.collections.ReferenceMap$EntryIterator:org.apache.commons.collections.ReferenceMap$Entry:entry	VALUE(S): org.apache.commons.collections.ReferenceMap$Entry
127	KEY: org.apache.commons.collections.ReferenceMap$EntryIterator:org.apache.commons.collections.ReferenceMap$Entry:previous	VALUE(S): org.apache.commons.collections.ReferenceMap$Entry
128	KEY: org.apache.commons.collections.SequencedHashMap$Entry:org.apache.commons.collections.SequencedHashMap$Entry:next	VALUE(S): org.apache.commons.collections.SequencedHashMap$Entry
129	KEY: org.apache.commons.collections.SequencedHashMap$Entry:org.apache.commons.collections.SequencedHashMap$Entry:prev	VALUE(S): org.apache.commons.collections.SequencedHashMap$Entry
130	KEY: org.apache.commons.collections.SequencedHashMap$OrderedIterator:org.apache.commons.collections.SequencedHashMap$Entry:pos	VALUE(S): org.apache.commons.collections.SequencedHashMap$Entry
131	KEY: org.apache.commons.collections.set.ListOrderedSet$OrderedSetIterator:java.lang.Object:last	VALUE(S): java.lang.Object
132	KEY: org.apache.commons.collections.StaticBucketMap$EntryIterator:java.util.Map$Entry:last	VALUE(S): java.util.Map$Entry
133	KEY: org.apache.commons.collections.StaticBucketMap$Node:org.apache.commons.collections.StaticBucketMap$Node:next	VALUE(S): org.apache.commons.collections.StaticBucketMap$Node
134	KEY: org.apache.commons.collections.TestArrayStack:java.util.ArrayList:list	VALUE(S): org.apache.commons.collections.ArrayStack
135	KEY: org.apache.commons.collections.TestBeanMap:java.util.Collection:values	VALUE(S): java.util.Collection
136	KEY: org.apache.commons.collections.TestListUtils:java.util.List:fullList	VALUE(S): java.util.List
137	KEY: org.apache.commons.collections.TestMultiHashMap:java.util.Collection:values	VALUE(S): java.util.Collection
138		
139		
140	***Polymorphic Fields:***	
141	org.apache.commons.collections.ReferenceMap$Entry:org.apache.commons.collections.ReferenceMap$Entry:next	
142	org.apache.commons.collections.map.AbstractHashedMap$HashEntry:org.apache.commons.collections.map.AbstractHashedMap$HashEntry:next	
143	***Polymorphic Fields end***	
144		

7.4 Collections Dynamic

	A	B
1	KEY: org.apache.commons.collections.bag.TestHashBag:java.lang.String:verboseName	VALUE(S): java.lang.String
2	KEY: org.apache.commons.collections.bag.TestPredicatedBag:java.lang.String:verboseName	VALUE(S): java.lang.String
3	KEY: org.apache.commons.collections.bag.TestPredicatedBag:org.apache.commons.collections.Predicate:truePredicate	VALUE(S): org.apache.commons.collections.functors.TruePredicate
4	KEY: org.apache.commons.collections.bag.TestPredicatedSortedBag:java.lang.String:verboseName	VALUE(S): java.lang.String
5	KEY: org.apache.commons.collections.bag.TestPredicatedSortedBag:org.apache.commons.collections.Predicate:truePredicate	VALUE(S): org.apache.commons.collections.functors.TruePredicate
6	KEY: org.apache.commons.collections.bag.TestTransformedBag:java.lang.String:verboseName	VALUE(S): java.lang.String
7	KEY: org.apache.commons.collections.bag.TestTransformedSortedBag:java.lang.String:verboseName	VALUE(S): java.lang.String
8	KEY: org.apache.commons.collections.bag.TestTreeBag:java.lang.String:verboseName	VALUE(S): java.lang.String
9	KEY: org.apache.commons.collections.bag.TestTypedBag:java.lang.Class:objectClass	VALUE(S): java.lang.Class
10	KEY: org.apache.commons.collections.bag.TestTypedBag:java.lang.Class:stringClass	VALUE(S): java.lang.Class
11	KEY: org.apache.commons.collections.bag.TestTypedBag:java.lang.Object:obj	VALUE(S): java.lang.Object
12	KEY: org.apache.commons.collections.bag.TestTypedBag:java.lang.String:verboseName	VALUE(S): java.lang.String
13	KEY: org.apache.commons.collections.bag.TestTypedSortedBag:java.lang.Class:objectClass	VALUE(S): java.lang.Class
14	KEY: org.apache.commons.collections.bag.TestTypedSortedBag:java.lang.Class:stringClass	VALUE(S): java.lang.Class
15	KEY: org.apache.commons.collections.bag.TestTypedSortedBag:java.lang.Object:obj	VALUE(S): java.lang.Object
16	KEY: org.apache.commons.collections.bag.TestTypedSortedBag:java.lang.String:verboseName	VALUE(S): java.lang.String
17	KEY: org.apache.commons.collections.bidimap.TestAbstractOrderedBidiMapDecorator:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
18	KEY: org.apache.commons.collections.bidimap.TestAbstractOrderedBidiMapDecorator:java.lang.String:verboseName	VALUE(S): java.lang.String
19	KEY: org.apache.commons.collections.bidimap.TestDualHashBidiMap:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
20	KEY: org.apache.commons.collections.bidimap.TestDualHashBidiMap:java.lang.String:verboseName	VALUE(S): java.lang.String
21	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
22	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap:java.lang.String:verboseName	VALUE(S): java.lang.String
23	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap:java.util.List:sortedKeys	VALUE(S): java.util.ArrayList
24		VALUE(S): java.util.Collections$UnmodifiableRandomAccessList
25	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap:java.util.List:sortedValues	VALUE(S): java.util.ArrayList
26		VALUE(S): java.util.Collections$UnmodifiableRandomAccessList
27	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap:java.util.SortedSet:sortedNewValues	VALUE(S): java.util.TreeSet
28	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
29	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:java.lang.String:verboseName	VALUE(S): java.lang.String
30	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:java.util.List:sortedKeys	VALUE(S): java.util.ArrayList
31		VALUE(S): java.util.Collections$UnmodifiableRandomAccessList
32	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:java.util.List:sortedValues	VALUE(S): java.util.ArrayList
33		VALUE(S): java.util.Collections$UnmodifiableRandomAccessList
34	KEY: org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:java.util.SortedSet:sortedNewValues	VALUE(S): java.util.TreeSet
35	KEY: org.apache.commons.collections.bidimap.TestTreeBidiMap:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
36	KEY: org.apache.commons.collections.bidimap.TestTreeBidiMap:java.lang.String:verboseName	VALUE(S): java.lang.String
37	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableBidiMap:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
38	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableBidiMap:java.lang.String:verboseName	VALUE(S): java.lang.String
39	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableOrderedBidiMap:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
40	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableOrderedBidiMap:java.lang.String:verboseName	VALUE(S): java.lang.String
41	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:java.lang.Object[][]:entries	VALUE(S): [[Ljava.lang.Object;
42	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:java.lang.String:verboseName	VALUE(S): java.lang.String
43	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:java.util.List:sortedKeys	VALUE(S): java.util.ArrayList
44		VALUE(S): java.util.Collections$UnmodifiableRandomAccessList
45	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:java.util.List:sortedValues	VALUE(S): java.util.ArrayList
46		VALUE(S): java.util.Collections$UnmodifiableRandomAccessList
47	KEY: org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:java.util.SortedSet:sortedNewValues	VALUE(S): java.util.TreeSet
48	KEY: org.apache.commons.collections.buffer.TestBlockingBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
49	KEY: org.apache.commons.collections.buffer.TestBoundedBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
50	KEY: org.apache.commons.collections.buffer.TestBoundedFifoBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String

	A	B
51	KEY: org.apache.commons.collections.buffer.TestBoundedFifoBuffer2:java.lang.String:verboseName	VALUE(S): java.lang.String
52	KEY: org.apache.commons.collections.buffer.TestCircularFifoBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
53	KEY: org.apache.commons.collections.buffer.TestPriorityBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
54	KEY: org.apache.commons.collections.buffer.TestSynchronizedBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
55	KEY: org.apache.commons.collections.buffer.TestUnboundedFifoBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
56	KEY: org.apache.commons.collections.buffer.TestUnmodifiableBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
57	KEY: org.apache.commons.collections.BulkTest:java.lang.String:verboseName	VALUE(S): java.lang.String
58	KEY: org.apache.commons.collections.collection.TestCompositeCollection:java.lang.String:verboseName	VALUE(S): java.lang.String
59	KEY: org.apache.commons.collections.collection.TestSynchronizedCollection:java.lang.String:verboseName	VALUE(S): java.lang.String
60	KEY: org.apache.commons.collections.collection.TestTransformedCollection:java.lang.String:verboseName	VALUE(S): java.lang.String
61	KEY: org.apache.commons.collections.collection.TestUnmodifiableCollection:java.lang.String:verboseName	VALUE(S): java.lang.String
62	KEY: org.apache.commons.collections.comparators.TestBooleanComparator:java.lang.String:verboseName	VALUE(S): java.lang.String
63	KEY: org.apache.commons.collections.comparators.TestComparableComparator:java.lang.String:verboseName	VALUE(S): java.lang.String
64	KEY: org.apache.commons.collections.comparators.TestComparatorChain:java.lang.String:verboseName	VALUE(S): java.lang.String
65	KEY: org.apache.commons.collections.comparators.TestReverseComparator:java.lang.String:verboseName	VALUE(S): java.lang.String
66	KEY: org.apache.commons.collections.ExtendedProperties:java.lang.String:fileSeparator	VALUE(S): java.lang.String
67	KEY: org.apache.commons.collections.ExtendedProperties:java.util.ArrayList:keysAsListed	VALUE(S): java.util.ArrayList
68	KEY: org.apache.commons.collections.FastArrayList:java.util.ArrayList:list	VALUE(S): java.util.ArrayList
69	KEY: org.apache.commons.collections.FastHashMap:java.util.HashMap:map	VALUE(S): java.util.HashMap
70	KEY: org.apache.commons.collections.FastTreeMap:java.util.TreeMap:map	VALUE(S): java.util.TreeMap
71	KEY: org.apache.commons.collections.functors.ConstantFactory:java.lang.Object:iConstant	VALUE(S): java.lang.String
72	KEY: org.apache.commons.collections.iterators.TestArrayIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
73	KEY: org.apache.commons.collections.iterators.TestArrayIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
74	KEY: org.apache.commons.collections.iterators.TestArrayIterator2:int[]:testArray	VALUE(S): [I
75	KEY: org.apache.commons.collections.iterators.TestArrayIterator2:java.lang.String:verboseName	VALUE(S): java.lang.String
76	KEY: org.apache.commons.collections.iterators.TestArrayListIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
77	KEY: org.apache.commons.collections.iterators.TestArrayListIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
78	KEY: org.apache.commons.collections.iterators.TestArrayListIterator2:int[]:testArray	VALUE(S): [I
79	KEY: org.apache.commons.collections.iterators.TestArrayListIterator2:java.lang.String:verboseName	VALUE(S): java.lang.String
80	KEY: org.apache.commons.collections.iterators.TestCollatingIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
81	KEY: org.apache.commons.collections.iterators.TestCollatingIterator:java.util.ArrayList:evens	VALUE(S): java.util.ArrayList
82	KEY: org.apache.commons.collections.iterators.TestCollatingIterator:java.util.ArrayList:fib	VALUE(S): java.util.ArrayList
83	KEY: org.apache.commons.collections.iterators.TestCollatingIterator:java.util.ArrayList:odds	VALUE(S): java.util.ArrayList
84	KEY: org.apache.commons.collections.iterators.TestCollatingIterator:java.util.Comparator:comparator	VALUE(S): org.apache.commons.collections.comparators.ComparableComparator
85	KEY: org.apache.commons.collections.iterators.TestFilterIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
86	KEY: org.apache.commons.collections.iterators.TestFilterIterator:java.lang.String[]:array	VALUE(S): [Ljava.lang.String;
87	KEY: org.apache.commons.collections.iterators.TestFilterIterator:java.util.List:list	VALUE(S): java.util.ArrayList
88	KEY: org.apache.commons.collections.iterators.TestFilterListIterator:java.util.ArrayList:evens	VALUE(S): java.util.ArrayList
89	KEY: org.apache.commons.collections.iterators.TestFilterListIterator:java.util.ArrayList:fours	VALUE(S): java.util.ArrayList
90	KEY: org.apache.commons.collections.iterators.TestFilterListIterator:java.util.ArrayList:list	VALUE(S): java.util.ArrayList
91	KEY: org.apache.commons.collections.iterators.TestFilterListIterator:java.util.ArrayList:odds	VALUE(S): java.util.ArrayList
92	KEY: org.apache.commons.collections.iterators.TestFilterListIterator:java.util.ArrayList:sixes	VALUE(S): java.util.ArrayList
93	KEY: org.apache.commons.collections.iterators.TestFilterListIterator:java.util.ArrayList:threes	VALUE(S): java.util.ArrayList
94	KEY: org.apache.commons.collections.iterators.TestFilterListIterator:java.util.Random:random	VALUE(S): java.util.Random
95	KEY: org.apache.commons.collections.iterators.TestIteratorChain:java.lang.String:verboseName	VALUE(S): java.lang.String
96	KEY: org.apache.commons.collections.iterators.TestIteratorChain:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
97	KEY: org.apache.commons.collections.iterators.TestIteratorChain:java.util.List:list1	VALUE(S): java.util.ArrayList
98	KEY: org.apache.commons.collections.iterators.TestIteratorChain:java.util.List:list2	VALUE(S): java.util.ArrayList
99	KEY: org.apache.commons.collections.iterators.TestIteratorChain:java.util.List:list3	VALUE(S): java.util.ArrayList
100	KEY: org.apache.commons.collections.iterators.TestListIteratorWrapper:java.lang.String:verboseName	VALUE(S): java.lang.String

	A	B
101	KEY: org.apache.commons.collections.iterators.TestListIteratorWrapper:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
102	KEY: org.apache.commons.collections.iterators.TestListIteratorWrapper:java.util.List:list1	VALUE(S): java.util.ArrayList
103	KEY: org.apache.commons.collections.iterators.TestObjectArrayIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
104	KEY: org.apache.commons.collections.iterators.TestObjectArrayIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
105	KEY: org.apache.commons.collections.iterators.TestObjectArrayListIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
106	KEY: org.apache.commons.collections.iterators.TestObjectArrayListIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
107	KEY: org.apache.commons.collections.iterators.TestObjectArrayListIterator2:java.lang.String:verboseName	VALUE(S): java.lang.String
108	KEY: org.apache.commons.collections.iterators.TestObjectArrayListIterator2:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
109	KEY: org.apache.commons.collections.iterators.TestObjectGraphIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
110	KEY: org.apache.commons.collections.iterators.TestObjectGraphIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
111	KEY: org.apache.commons.collections.iterators.TestObjectGraphIterator:java.util.List:iteratorList	VALUE(S): java.util.ArrayList
112	KEY: org.apache.commons.collections.iterators.TestObjectGraphIterator:java.util.List:list1	VALUE(S): java.util.ArrayList
113	KEY: org.apache.commons.collections.iterators.TestObjectGraphIterator:java.util.List:list2	VALUE(S): java.util.ArrayList
114	KEY: org.apache.commons.collections.iterators.TestObjectGraphIterator:java.util.List:list3	VALUE(S): java.util.ArrayList
115	KEY: org.apache.commons.collections.iterators.TestReverseListIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
116	KEY: org.apache.commons.collections.iterators.TestReverseListIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
117	KEY: org.apache.commons.collections.iterators.TestSingletonIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
118	KEY: org.apache.commons.collections.iterators.TestSingletonIterator2:java.lang.String:verboseName	VALUE(S): java.lang.String
119	KEY: org.apache.commons.collections.iterators.TestSingletonListIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
120	KEY: org.apache.commons.collections.iterators.TestUniqueFilterIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
121	KEY: org.apache.commons.collections.iterators.TestUniqueFilterIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
122	KEY: org.apache.commons.collections.iterators.TestUniqueFilterIterator:java.util.List:list1	VALUE(S): java.util.ArrayList
123	KEY: org.apache.commons.collections.iterators.TestUnmodifiableIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
124	KEY: org.apache.commons.collections.iterators.TestUnmodifiableIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
125	KEY: org.apache.commons.collections.iterators.TestUnmodifiableIterator:java.util.List:testList	VALUE(S): java.util.ArrayList
126	KEY: org.apache.commons.collections.iterators.TestUnmodifiableListIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
127	KEY: org.apache.commons.collections.iterators.TestUnmodifiableListIterator:java.lang.String[]:testArray	VALUE(S): [Ljava.lang.String;
128	KEY: org.apache.commons.collections.iterators.TestUnmodifiableListIterator:java.util.List:testList	VALUE(S): java.util.ArrayList
129	KEY: org.apache.commons.collections.iterators.TestUnmodifiableMapIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
130	KEY: org.apache.commons.collections.iterators.TestUnmodifiableOrderedMapIterator:java.lang.String:verboseName	VALUE(S): java.lang.String
131	KEY: org.apache.commons.collections.keyvalue.TestDefaultKeyValue:java.lang.String:key	VALUE(S): java.lang.String
132	KEY: org.apache.commons.collections.keyvalue.TestDefaultKeyValue:java.lang.String:value	VALUE(S): java.lang.String
133	KEY: org.apache.commons.collections.keyvalue.TestDefaultMapEntry:java.lang.String:key	VALUE(S): java.lang.String
134	KEY: org.apache.commons.collections.keyvalue.TestDefaultMapEntry:java.lang.String:value	VALUE(S): java.lang.String
135	KEY: org.apache.commons.collections.keyvalue.TestMultiKey:java.lang.Integer:FIVE	VALUE(S): java.lang.Integer
136	KEY: org.apache.commons.collections.keyvalue.TestMultiKey:java.lang.Integer:FOUR	VALUE(S): java.lang.Integer
137	KEY: org.apache.commons.collections.keyvalue.TestMultiKey:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
138	KEY: org.apache.commons.collections.keyvalue.TestMultiKey:java.lang.Integer:THREE	VALUE(S): java.lang.Integer
139	KEY: org.apache.commons.collections.keyvalue.TestMultiKey:java.lang.Integer:TWO	VALUE(S): java.lang.Integer
140	KEY: org.apache.commons.collections.keyvalue.TestTiedMapEntry:java.lang.String:key	VALUE(S): java.lang.String
141	KEY: org.apache.commons.collections.keyvalue.TestTiedMapEntry:java.lang.String:value	VALUE(S): java.lang.String
142	KEY: org.apache.commons.collections.keyvalue.TestUnmodifiableMapEntry:java.lang.String:key	VALUE(S): java.lang.String
143	KEY: org.apache.commons.collections.keyvalue.TestUnmodifiableMapEntry:java.lang.String:value	VALUE(S): java.lang.String
144	KEY: org.apache.commons.collections.list.AbstractLinkedList$Node:org.apache.commons.collections.list.AbstractLinkedList$Node:next	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
145	KEY: org.apache.commons.collections.list.AbstractLinkedList$Node:org.apache.commons.collections.list.AbstractLinkedList$Node:previous	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
146	KEY: org.apache.commons.collections.list.CursorableLinkedList:java.util.List:cursors	VALUE(S): java.util.ArrayList
147	KEY: org.apache.commons.collections.list.CursorableLinkedList:org.apache.commons.collections.list.AbstractLinkedList$Node:header	VALUE(S): org.apache.commons.collections.list.AbstractLinkedList$Node
148	KEY: org.apache.commons.collections.list.TestCursorableLinkedList:java.lang.String:verboseName	VALUE(S): java.lang.String
149	KEY: org.apache.commons.collections.list.TestCursorableLinkedList:org.apache.commons.collections.list.CursorableLinkedList:list	VALUE(S): org.apache.commons.collections.list.CursorableLinkedList
150	KEY: org.apache.commons.collections.list.TestFixedSizeList:java.lang.String:verboseName	VALUE(S): java.lang.String

	A	B
151	KEY: org.apache.commons.collections.list.TestGrowthList:java.lang.String:verboseName	VALUE(S): java.lang.String
152	KEY: org.apache.commons.collections.list.TestNodeCachingLinkedList:java.lang.String:verboseName	VALUE(S): java.lang.String
153	KEY: org.apache.commons.collections.list.TestSetUniqueList:java.lang.String:verboseName	VALUE(S): java.lang.String
154	KEY: org.apache.commons.collections.list.TestSynchronizedList:java.lang.String:verboseName	VALUE(S): java.lang.String
155	KEY: org.apache.commons.collections.list.TestTransformedList:java.lang.String:verboseName	VALUE(S): java.lang.String
156	KEY: org.apache.commons.collections.list.TestTreeList:java.lang.String:verboseName	VALUE(S): java.lang.String
157	KEY: org.apache.commons.collections.list.TestTypedList:java.lang.String:verboseName	VALUE(S): java.lang.String
158	KEY: org.apache.commons.collections.list.TestUnmodifiableList:java.lang.String:verboseName	VALUE(S): java.lang.String
159	KEY: org.apache.commons.collections.map.TestCaseInsensitiveMap:java.lang.String:verboseName	VALUE(S): java.lang.String
160	KEY: org.apache.commons.collections.map.TestCompositeMap:java.lang.String:verboseName	VALUE(S): java.lang.String
161	KEY: org.apache.commons.collections.map.TestDefaultedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
162	KEY: org.apache.commons.collections.map.TestFixedSizeMap:java.lang.String:verboseName	VALUE(S): java.lang.String
163	KEY: org.apache.commons.collections.map.TestFixedSizeSortedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
164	KEY: org.apache.commons.collections.map.TestFlat3Map:java.lang.String:verboseName	VALUE(S): java.lang.String
165	KEY: org.apache.commons.collections.map.TestHashedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
166	KEY: org.apache.commons.collections.map.TestIdentityMap:java.lang.String:verboseName	VALUE(S): java.lang.String
167	KEY: org.apache.commons.collections.map.TestLazyMap:java.lang.String:verboseName	VALUE(S): java.lang.String
168	KEY: org.apache.commons.collections.map.TestLazySortedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
169	KEY: org.apache.commons.collections.map.TestLinkedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
170	KEY: org.apache.commons.collections.map.TestListOrderedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
171	KEY: org.apache.commons.collections.map.TestListOrderedMap2:java.lang.String:verboseName	VALUE(S): java.lang.String
172	KEY: org.apache.commons.collections.map.TestLRUMap:java.lang.String:verboseName	VALUE(S): java.lang.String
173	KEY: org.apache.commons.collections.map.TestMultiKeyMap:java.lang.String:verboseName	VALUE(S): java.lang.String
174	KEY: org.apache.commons.collections.map.TestReferenceIdentityMap:java.lang.String:verboseName	VALUE(S): java.lang.String
175	KEY: org.apache.commons.collections.map.TestReferenceMap:java.lang.String:verboseName	VALUE(S): java.lang.String
176	KEY: org.apache.commons.collections.map.TestSingletonMap:java.lang.String:verboseName	VALUE(S): java.lang.String
177	KEY: org.apache.commons.collections.map.TestStaticBucketMap:java.lang.String:verboseName	VALUE(S): java.lang.String
178	KEY: org.apache.commons.collections.map.TestTransformedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
179	KEY: org.apache.commons.collections.map.TestTransformedSortedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
180	KEY: org.apache.commons.collections.map.TestUnmodifiableMap:java.lang.String:verboseName	VALUE(S): java.lang.String
181	KEY: org.apache.commons.collections.map.TestUnmodifiableOrderedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
182	KEY: org.apache.commons.collections.map.TestUnmodifiableSortedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
183	KEY: org.apache.commons.collections.set.TestCompositeSet:java.lang.String:verboseName	VALUE(S): java.lang.String
184	KEY: org.apache.commons.collections.set.TestListOrderedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
185	KEY: org.apache.commons.collections.set.TestListOrderedSet2:java.lang.String:verboseName	VALUE(S): java.lang.String
186	KEY: org.apache.commons.collections.set.TestMapBackedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
187	KEY: org.apache.commons.collections.set.TestMapBackedSet2:java.lang.String:verboseName	VALUE(S): java.lang.String
188	KEY: org.apache.commons.collections.set.TestSynchronizedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
189	KEY: org.apache.commons.collections.set.TestSynchronizedSortedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
190	KEY: org.apache.commons.collections.set.TestTransformedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
191	KEY: org.apache.commons.collections.set.TestTransformedSortedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
192	KEY: org.apache.commons.collections.set.TestTypedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
193	KEY: org.apache.commons.collections.set.TestTypedSortedSet:java.lang.Class:integerType	VALUE(S): java.lang.Class
194	KEY: org.apache.commons.collections.set.TestTypedSortedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
195	KEY: org.apache.commons.collections.set.TestUnmodifiableSet:java.lang.String:verboseName	VALUE(S): java.lang.String
196	KEY: org.apache.commons.collections.set.TestUnmodifiableSortedSet:java.lang.String:verboseName	VALUE(S): java.lang.String
197	KEY: org.apache.commons.collections.TestArrayStack:java.lang.String:verboseName	VALUE(S): java.lang.String
198	KEY: org.apache.commons.collections.TestArrayStack:java.util.ArrayList:list	VALUE(S): org.apache.commons.collections.ArrayStack
199	KEY: org.apache.commons.collections.TestArrayStack:org.apache.commons.collections.ArrayStack:stack	VALUE(S): org.apache.commons.collections.ArrayStack
200	KEY: org.apache.commons.collections.TestBagUtils:java.lang.Class:stringClass	VALUE(S): java.lang.Class

	A	B
201	KEY: org.apache.commons.collections.TestBagUtils:java.lang.String:verboseName	VALUE(S): java.lang.String
202	KEY: org.apache.commons.collections.TestBagUtils:org.apache.commons.collections.Predicate:truePredicate	VALUE(S): org.apache.commons.collections.functors.TruePredicate
203	KEY: org.apache.commons.collections.TestBagUtils:org.apache.commons.collections.Transformer:nopTransformer	VALUE(S): org.apache.commons.collections.functors.NOPTransformer
204	KEY: org.apache.commons.collections.TestBeanMap:java.lang.Object:objectInFullMap	VALUE(S): java.lang.Object
205	KEY: org.apache.commons.collections.TestBeanMap:java.lang.String:verboseName	VALUE(S): java.lang.String
206	KEY: org.apache.commons.collections.TestBinaryHeap:java.lang.String:verboseName	VALUE(S): java.lang.String
207	KEY: org.apache.commons.collections.TestBoundedFifoBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
208	KEY: org.apache.commons.collections.TestBoundedFifoBuffer2:java.lang.String:verboseName	VALUE(S): java.lang.String
209	KEY: org.apache.commons.collections.TestBufferUtils:java.lang.String:verboseName	VALUE(S): java.lang.String
210	KEY: org.apache.commons.collections.TestCursorableLinkedList:java.lang.String:verboseName	VALUE(S): java.lang.String
211	KEY: org.apache.commons.collections.TestDoubleOrderedMap:java.lang.String:verboseName	VALUE(S): java.lang.String
212	KEY: org.apache.commons.collections.TestEnumerationUtils:java.lang.String:verboseName	VALUE(S): java.lang.String
213	KEY: org.apache.commons.collections.TestExtendedProperties:org.apache.commons.collections.ExtendedProperties:eprop	VALUE(S): org.apache.commons.collections.ExtendedProperties
214	KEY: org.apache.commons.collections.TestFastArrayList:java.lang.String:verboseName	VALUE(S): java.lang.String
215	KEY: org.apache.commons.collections.TestFastArrayList:java.util.ArrayList:list	VALUE(S): org.apache.commons.collections.FastArrayList
216	KEY: org.apache.commons.collections.TestFastArrayList1:java.lang.String:verboseName	VALUE(S): java.lang.String
217	KEY: org.apache.commons.collections.TestFastArrayList1:java.util.ArrayList:list	VALUE(S): org.apache.commons.collections.FastArrayList
218	KEY: org.apache.commons.collections.TestFastHashMap:java.lang.String:verboseName	VALUE(S): java.lang.String
219	KEY: org.apache.commons.collections.TestFastHashMap1:java.lang.String:verboseName	VALUE(S): java.lang.String
220	KEY: org.apache.commons.collections.TestFastHashMap1:java.util.Map:map	VALUE(S): org.apache.commons.collections.FastHashMap
221	KEY: org.apache.commons.collections.TestFastTreeMap:java.lang.String:verboseName	VALUE(S): java.lang.String
222	KEY: org.apache.commons.collections.TestFastTreeMap:java.util.TreeMap:map	VALUE(S): org.apache.commons.collections.FastTreeMap
223	KEY: org.apache.commons.collections.TestFastTreeMap1:java.lang.String:verboseName	VALUE(S): java.lang.String
224	KEY: org.apache.commons.collections.TestFastTreeMap1:java.util.TreeMap:map	VALUE(S): org.apache.commons.collections.FastTreeMap
225	KEY: org.apache.commons.collections.TestHashBag:java.lang.String:verboseName	VALUE(S): java.lang.String
226	KEY: org.apache.commons.collections.TestIteratorUtils:java.lang.String:verboseName	VALUE(S): java.lang.String
227	KEY: org.apache.commons.collections.TestListUtils:java.lang.String:verboseName	VALUE(S): java.lang.String
228	KEY: org.apache.commons.collections.TestListUtils:java.lang.String[]:fullArray	VALUE(S): [Ljava.lang.String;
229	KEY: org.apache.commons.collections.TestListUtils:java.util.List:fullList	VALUE(S): java.util.ArrayList
230	KEY: org.apache.commons.collections.TestLRUMap:java.lang.String:verboseName	VALUE(S): java.lang.String
231	KEY: org.apache.commons.collections.TestMapUtils:java.lang.String:verboseName	VALUE(S): java.lang.String
232	KEY: org.apache.commons.collections.TestReferenceMap:java.lang.String:verboseName	VALUE(S): java.lang.String
233	KEY: org.apache.commons.collections.TestSequencedHashMap:java.lang.String:verboseName	VALUE(S): java.lang.String
234	KEY: org.apache.commons.collections.TestSetUtils:java.lang.String:verboseName	VALUE(S): java.lang.String
235	KEY: org.apache.commons.collections.TestStaticBucketMap:java.lang.String:verboseName	VALUE(S): java.lang.String
236	KEY: org.apache.commons.collections.TestTreeBag:java.lang.String:verboseName	VALUE(S): java.lang.String
237	KEY: org.apache.commons.collections.TestUnboundedFifoBuffer:java.lang.String:verboseName	VALUE(S): java.lang.String
238		
239		
240	***Dump end***	
241	***Polymorphic Fields:***	
242	org.apache.commons.collections.bidimap.TestDualTreeBidiMap:java.util.List:sortedKeys	
243	org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:java.util.List:sortedValues	
244	org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:java.util.List:sortedValues	
245	org.apache.commons.collections.bidimap.TestUnmodifiableSortedBidiMap:java.util.List:sortedKeys	
246	org.apache.commons.collections.bidimap.TestDualTreeBidiMap2:java.util.List:sortedKeys	
247	org.apache.commons.collections.bidimap.TestDualTreeBidiMap:java.util.List:sortedValues	
248	***Polymorphic Fields end***	
249		

7.5 Codec 1.8 Static

	A	B
1	KEY: org.apache.commons.codec.language.bm.BeiderMorseEncoder: org.apache.commons.codec.language.bm.PhoneticEngine:engine	VALUE(S): org.apache.commons.codec.language.bm.NameType
2	KEY: org.apache.commons.codec.language.bm.PhoneticEngine$RulesApplication: org.apache.commons.codec.language.bm.PhoneticEngine$PhonemeBuilder:phonemeBuilder	VALUE(S): org.apache.commons.codec.language.bm.PhoneticEngine$PhonemeBuilder
3		
4	***Polymorphic Fields:***	
5	***Polymorphic Fields end***	
6		

7.6 Codec 1.8 Dynamic

	A	B
1	KEY: org.apache.commons.codec.binary.Base64Test:java.util.Random:random	VALUE(S): java.util.Random
2	KEY: org.apache.commons.codec.digest.DigestUtilsTest:byte[]:testData	VALUE(S): [B
3	KEY: org.apache.commons.codec.language.Caverphone1Test: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.Caverphone1
4	KEY: org.apache.commons.codec.language.Caverphone2Test: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.Caverphone2
5	KEY: org.apache.commons.codec.language.ColognePhoneticTest: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.ColognePhonetic
6	KEY: org.apache.commons.codec.language.DoubleMetaphone2Test: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.DoubleMetaphone
7	KEY: org.apache.commons.codec.language.DoubleMetaphoneTest: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.DoubleMetaphone
8	KEY: org.apache.commons.codec.language.MatchRatingApproachEncoderTest: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.MatchRatingApproachEncoder
9	KEY: org.apache.commons.codec.language.MetaphoneTest: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.Metaphone
10	KEY: org.apache.commons.codec.language.NysiisTest: org.apache.commons.codec.language.Nysiis:fullNysiis	VALUE(S): org.apache.commons.codec.language.Nysiis
11	KEY: org.apache.commons.codec.language.NysiisTest: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.Nysiis
12	KEY: org.apache.commons.codec.language.RefinedSoundex:char[]:soundexMapping	VALUE(S): [C
13	KEY: org.apache.commons.codec.language.RefinedSoundexTest: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.RefinedSoundex
14	KEY: org.apache.commons.codec.language.Soundex:char[]:soundexMapping	VALUE(S): [C
15	KEY: org.apache.commons.codec.language.SoundexTest: org.apache.commons.codec.StringEncoder:stringEncoder	VALUE(S): org.apache.commons.codec.language.Soundex
16		
17	***Polymorphic Fields:***	
18	***Polymorphic Fields end***	
19		

7.7 Pool 1.6 Static

	A	B
1	KEY: org.apache.commons.pool.impl.CursorableLinkedList$ListIter: org.apache.commons.pool.impl.CursorableLinkedList$Listable:_cur	VALUE(S): org.apache.commons.pool.impl.CursorableLinkedList$Listable
2	KEY: org.apache.commons.pool.impl.CursorableLinkedList$ListIter: org.apache.commons.pool.impl.CursorableLinkedList$Listable:_lastReturned	VALUE(S): org.apache.commons.pool.impl.CursorableLinkedList$Listable
3	KEY: org.apache.commons.pool.impl.CursorableSubList: org.apache.commons.pool.impl.CursorableLinkedList$Listable:_post	VALUE(S): org.apache.commons.pool.impl.CursorableLinkedList$Listable
4	KEY: org.apache.commons.pool.impl.CursorableSubList: org.apache.commons.pool.impl.CursorableLinkedList$Listable:_pre	VALUE(S): org.apache.commons.pool.impl.CursorableLinkedList$Listable
5	KEY: org.apache.commons.pool.impl.GenericKeyedObjectPool$ObjectQueue: org.apache.commons.pool.impl.CursorableLinkedList:queue	VALUE(S): org.apache.commons.pool.impl.CursorableLinkedList
6	KEY: org.apache.commons.pool.impl.GenericKeyedObjectPool: org.apache.commons.pool.impl.CursorableLinkedList$Cursor:_evictionKeyCursor	VALUE(S): org.apache.commons.pool.impl.CursorableLinkedList$Cursor
7	KEY: org.apache.commons.pool.impl.GenericObjectPool: org.apache.commons.pool.impl.CursorableLinkedList$Cursor:_evictionCursor	VALUE(S): org.apache.commons.pool.impl.CursorableLinkedList$Cursor
8	KEY:org.apache.commons.pool.impl.TestGenericObjectPool$ConcurrentBorrowAndEvictThread: java.lang.String:obj	VALUE(S): java.lang.String
9	KEY: org.apache.commons.pool.impl.TestGenericObjectPool$TestThread: java.lang.Throwable:_error	VALUE(S): java.lang.String
10		
11	***Polymorphic Fields:***	
12	***Polymorphic Fields end***	
13		

7.8 Pool 1.6 Dynamic

	A	B
1	KEY: org.apache.commons.pool.impl.TestGenericKeyedObjectPool:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
2	KEY: org.apache.commons.pool.impl.TestGenericKeyedObjectPool:java.lang.Integer:one	VALUE(S): java.lang.Integer
3	KEY: org.apache.commons.pool.impl.TestGenericKeyedObjectPool:java.lang.Integer:two	VALUE(S): java.lang.Integer
4	KEY: org.apache.commons.pool.impl.TestGenericKeyedObjectPool:java.lang.Integer:ZERO	VALUE(S): java.lang.Integer
5	KEY: org.apache.commons.pool.impl.TestGenericKeyedObjectPool:java.lang.Integer:zero	VALUE(S): java.lang.Integer
6	KEY: org.apache.commons.pool.impl.TestGenericKeyedObjectPool:java.lang.String:KEY	VALUE(S): java.lang.String
7	KEY: org.apache.commons.pool.impl.TestGenericObjectPool:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
8	KEY: org.apache.commons.pool.impl.TestGenericObjectPool:java.lang.Integer:ZERO	VALUE(S): java.lang.Integer
9	KEY: org.apache.commons.pool.impl.TestSoftReferenceObjectPool:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
10	KEY: org.apache.commons.pool.impl.TestSoftReferenceObjectPool:java.lang.Integer:ZERO	VALUE(S): java.lang.Integer
11	KEY: org.apache.commons.pool.impl.TestStackKeyedObjectPool:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
12	KEY: org.apache.commons.pool.impl.TestStackKeyedObjectPool:java.lang.Integer:ZERO	VALUE(S): java.lang.Integer
13	KEY: org.apache.commons.pool.impl.TestStackKeyedObjectPool:java.lang.String:KEY	VALUE(S): java.lang.String
14	KEY: org.apache.commons.pool.impl.TestStackObjectPool:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
15	KEY: org.apache.commons.pool.impl.TestStackObjectPool:java.lang.Integer:ZERO	VALUE(S): java.lang.Integer
16	KEY: org.apache.commons.pool.MethodCall:java.lang.String:name	VALUE(S): java.lang.String
17	KEY: org.apache.commons.pool.MethodCall:java.util.List:params	VALUE(S): java.util.Collections$EmptyList
18	KEY: org.apache.commons.pool.MethodCallPoolableObjectFactory:java.util.List:methodCalls	VALUE(S): java.util.ArrayList
19	KEY: org.apache.commons.pool.TestBaseKeyedObjectPool:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
20	KEY: org.apache.commons.pool.TestBaseKeyedObjectPool:java.lang.Integer:ZERO	VALUE(S): java.lang.Integer
21	KEY: org.apache.commons.pool.TestBaseKeyedObjectPool:java.lang.String:KEY	VALUE(S): java.lang.String
22	KEY: org.apache.commons.pool.TestBaseObjectPool:java.lang.Integer:ONE	VALUE(S): java.lang.Integer
23	KEY: org.apache.commons.pool.TestBaseObjectPool:java.lang.Integer:ZERO	VALUE(S): java.lang.Integer
24		
25	***Polymorphic Fields:***	
26	***Polymorphic Fields end***	
27		

7.9 Daemon 1.0.15 Static

	A	B
1	KEY:org.apache.commons.daemon.support.DaemonWrapper$Invoker:java.lang.Class:main	VALUE(S): java.lang.Class
2	KEY:org.apache.commons.daemon.support.DaemonWrapper$Invoker: java.lang.reflect.Method:inst	VALUE(S): java.lang.reflect.Method
3		
4	***Polymorphic Fields:***	
5	***Polymorphic Fields end***	
6		

7.10 Daemon 1.0.15 Dynamic

	A	B
1	KEY: org.apache.commons.daemon.SimpleDaemon:java.util.Vector:handlers	VALUE(S): java.util.Vector
2		
3	***Polymorphic Fields:***	
4	***Polymorphic Fields end***	
5		

7.11 CLI 1.2 Static

	A	B
1	KEY: org.apache.commons.cli.Option:java.util.List:values	VALUE(S): java.util.List
2	KEY: org.apache.commons.cli.PosixParser:org.apache.commons.cli.Option:currentOption	VALUE(S): org.apache.commons.cli.Option
3	KEY: org.apache.commons.cli.ValueTest:org.apache.commons.cli.CommandLine:_cl	VALUE(S): org.apache.commons.cli.CommandLine
4		
5	***Polymorphic Fields:***	
6	***Polymorphic Fields end***	
7		

7.12CLI 1.2 Dynamic

	A	B
28	KEY:org.apache.commons.cli.ArgumentIsOptionTest: org.apache.commons.cli.CommandLineParser:parser	VALUE(S): org.apache.commons.cli.PosixParser
29	KEY:org.apache.commons.cli.bug.BugCLI71Test: org.apache.commons.cli.CommandLineParser:parser	VALUE(S): org.apache.commons.cli.PosixParser
30	KEY:org.apache.commons.cli.bug.BugCLI71Test: org.apache.commons.cli.Options:options	VALUE(S): org.apache.commons.cli.Options
31	KEY: org.apache.commons.cli.Option:java.lang.String:longOpt	VALUE(S): java.lang.String
32	KEY: org.apache.commons.cli.Option:java.util.List:values	VALUE(S): java.util.ArrayList
33	KEY: org.apache.commons.cli.OptionGroup:java.util.Map:optionMap	VALUE(S): java.util.HashMap
34	KEY:org.apache.commons.cli.OptionGroupTest: org.apache.commons.cli.CommandLineParser:parser	VALUE(S): org.apache.commons.cli.PosixParser
35	KEY: org.apache.commons.cli.OptionGroupTest:org.apache.commons.cli.Options:_options	VALUE(S): org.apache.commons.cli.Options
36	KEY: org.apache.commons.cli.Options:java.util.List:requiredOpts	VALUE(S): java.util.ArrayList
37	KEY: org.apache.commons.cli.Options:java.util.Map:shortOpts	VALUE(S): java.util.HashMap
38	KEY: org.apache.commons.cli.ParseRequiredTest:org.apache.commons.cli.Options:_options	VALUE(S): org.apache.commons.cli.Options
39	KEY: org.apache.commons.cli.PosixParser:java.util.List:requiredOptions	VALUE(S): java.util.ArrayList
40	KEY: org.apache.commons.cli.PosixParser:org.apache.commons.cli.Option:currentOption	VALUE(S): org.apache.commons.cli.Option
41	KEY: org.apache.commons.cli.PosixParser:org.apache.commons.cli.Options:options	VALUE(S): org.apache.commons.cli.Options
42	KEY: org.apache.commons.cli.ValueTest:org.apache.commons.cli.Options:opts	VALUE(S): org.apache.commons.cli.Options
43		
44	***Polymorphic Fields:***	
45	***Polymorphic Fields end***	
46		