Lightweight Syntactic API Usage Analysis with UCov

32nd IEEE/ACM International Conference on Program Comprehension (ICPC '24)

April 15–16, 2024

Lisbon, Portugal

Gustave Monce

Thomas Couturou

Yasmine Hamdaoui

Thomas Degueule

Jean-Rémy Falleri

Software Libraries

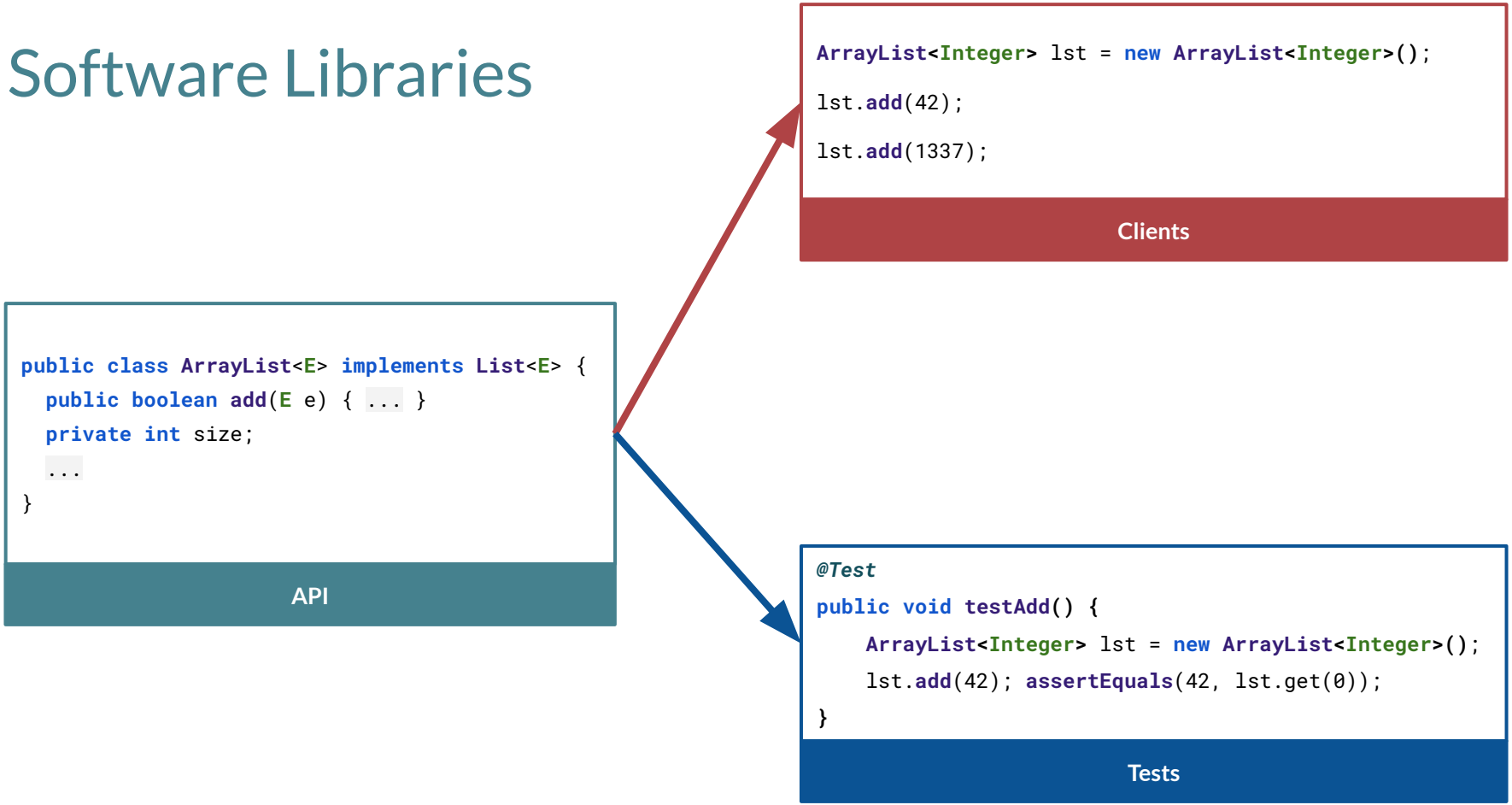
```
public class ArrayList<E> implements List<E> {
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

API

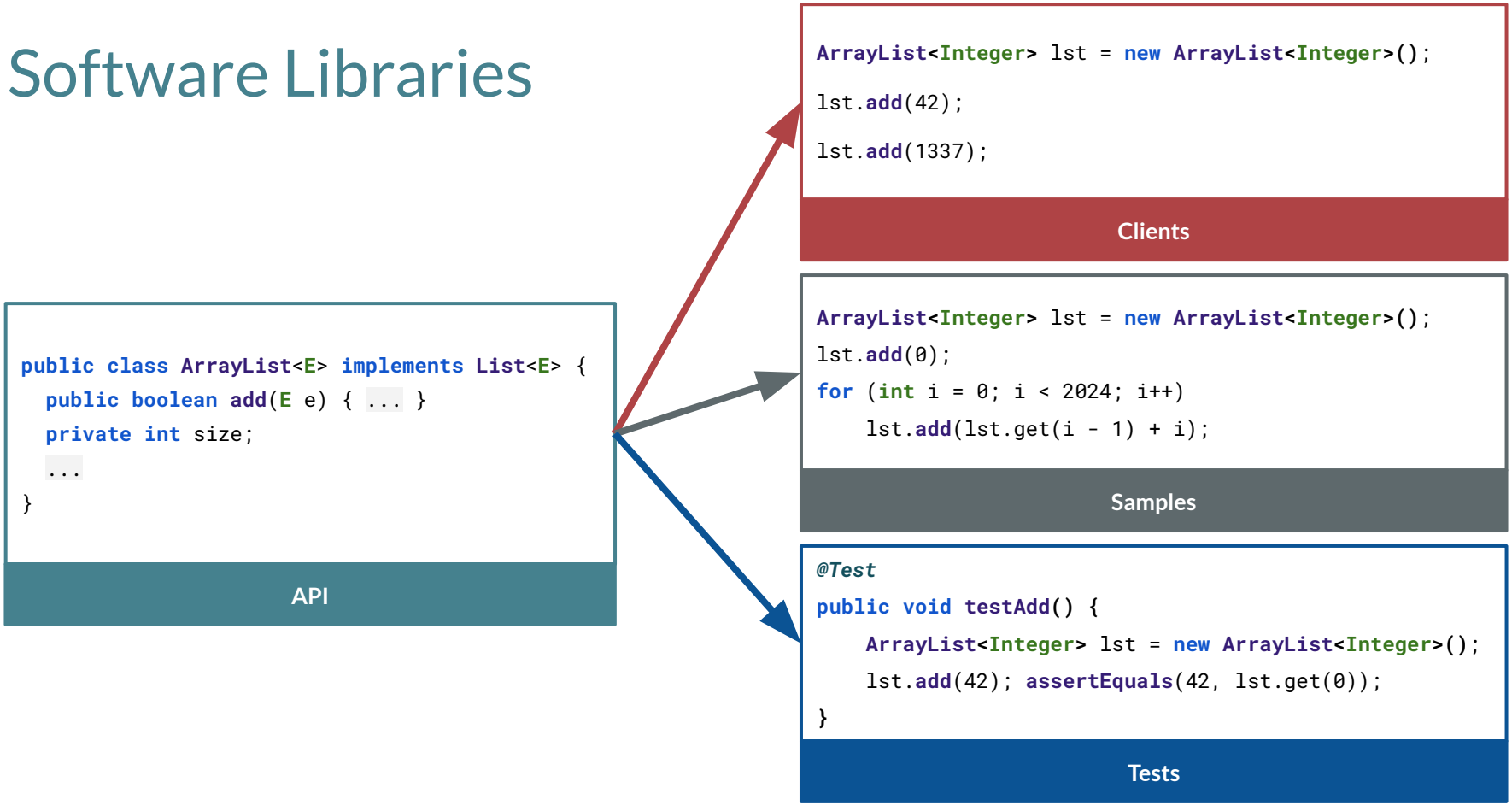
```
ArrayList<Integer> lst = new ArrayList<Integer>();
lst.add(42);
lst.add(1337);
```

Clients

Software Libraries

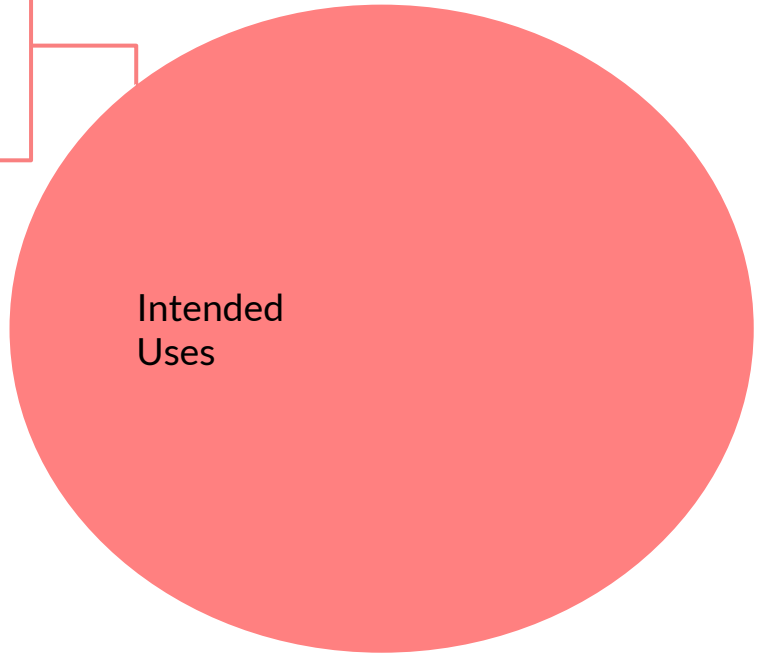


Software Libraries



API Design & Usage

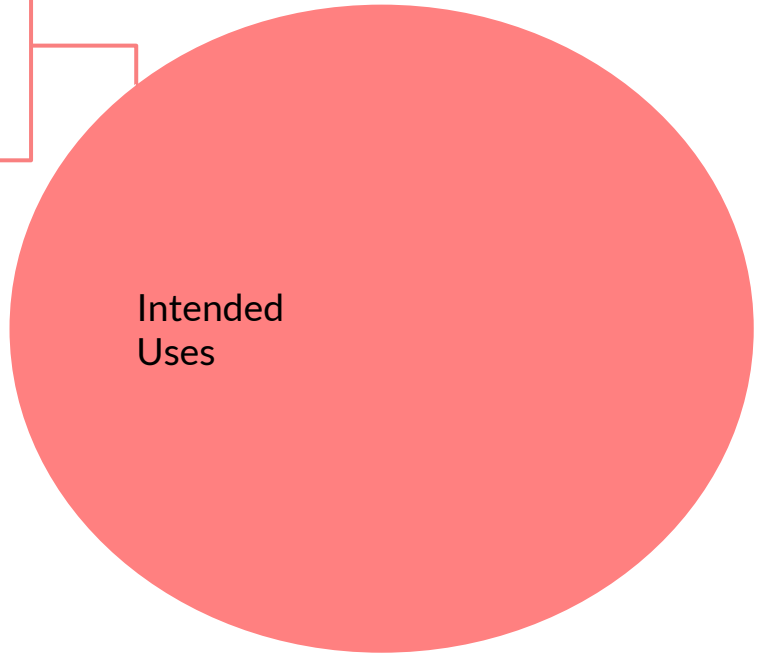
```
new ArrayList<Integer>();
lst.add(0)
lst.add(2)
lst.add(6 * 2)
lst.get(0)
```



API Design & Usage

```
public class ArrayList<E> implements List<E> {
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

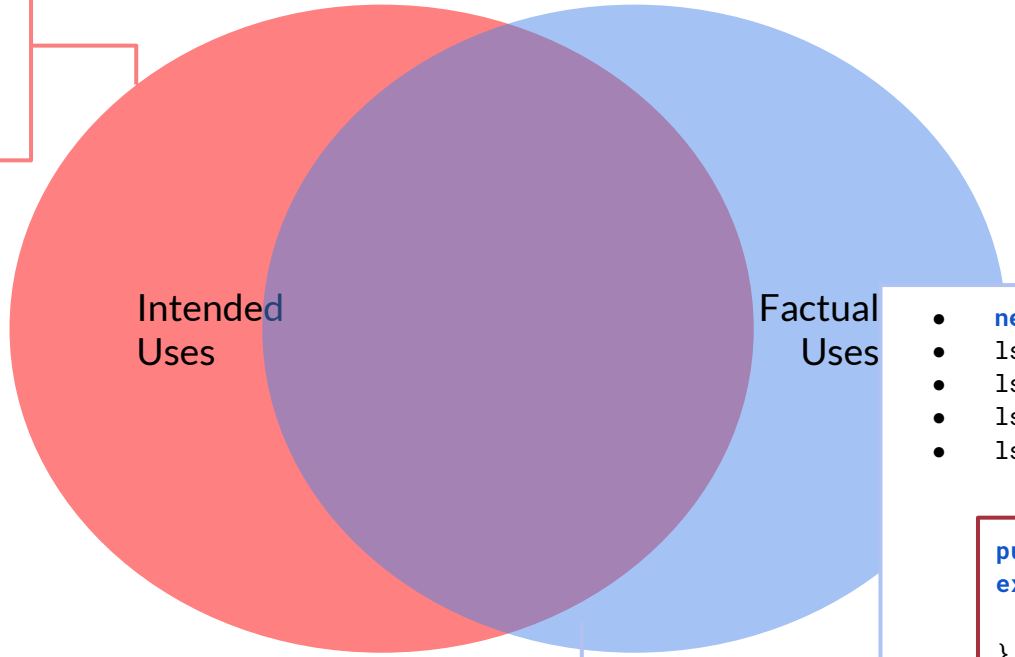
- `new ArrayList<Integer>();`
- `lst.add(0)`
- `lst.add(2)`
- `lst.add(6 * 2)`
- `lst.get(0)`



API Design & Usage

```
public class ArrayList<E> implements List<E> {
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

- `new ArrayList<Integer>();`
- `lst.add(0)`
- `lst.add(2)`
- `lst.add(6 * 2)`
- `lst.get(0)`



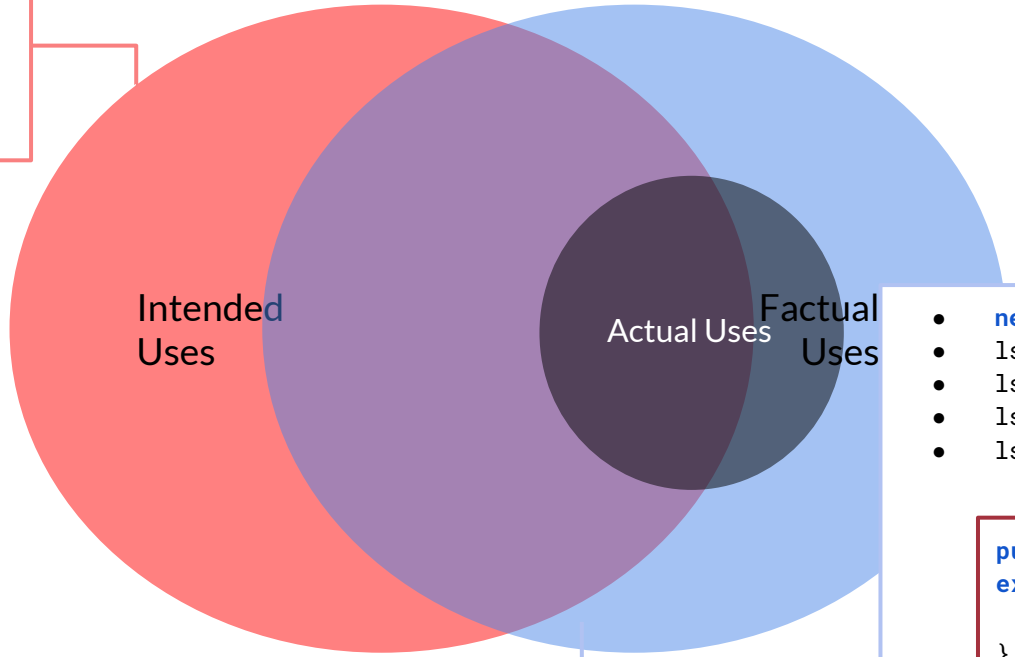
- `new ArrayList<Integer>();`
- `lst.add(0)`
- `lst.add(2)`
- `lst.add(6 * 2)`
- `lst.get(0)`

```
public class NewArrayList<E>
extends ArrayList<E> {
   @Override { ... }
}
```

API Design & Usage

```
public class ArrayList<E> implements List<E> {
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

```
• new ArrayList<Integer>();
• lst.add(0)
• lst.add(2)
• lst.add(6 * 2)
• lst.get(0)
```



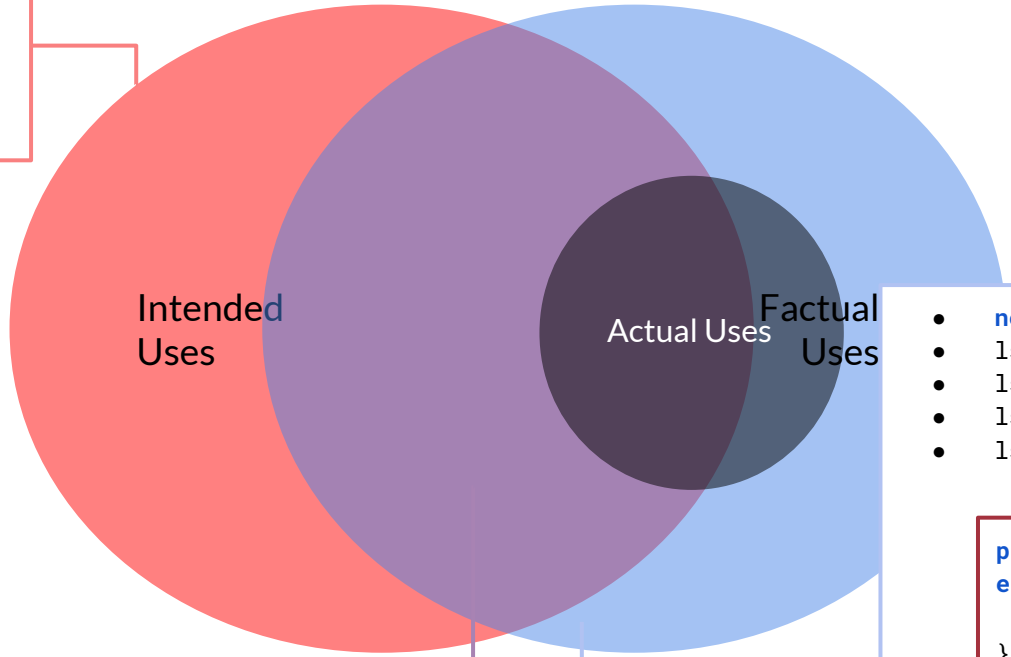

```
• new ArrayList<Integer>();
• lst.add(0)
• lst.add(2)
• lst.add(6 * 2)
• lst.get(0)
```

```
public class NewArrayList<E>
extends ArrayList<E> {
   @Override { ... }
}
```

API Design & Usage

```
public class ArrayList<E> implements List<E> {
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

```
• new ArrayList<Integer>();
• lst.add(0)
• lst.add(2)
• lst.add(6 * 2)
• lst.get(0)
```



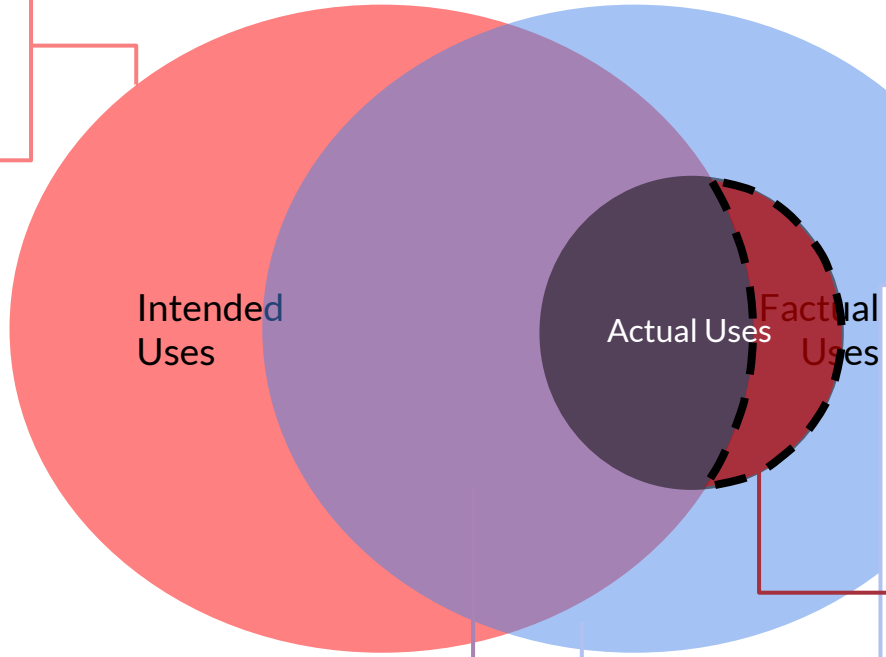

```
• new ArrayList<Integer>();
• lst.add(0)
• lst.add(2)
• lst.add(6 * 2)
• lst.get(0)
```

```
public class NewArrayList<E>
extends ArrayList<E> {
   @Override { ... }
}
```

API Design & Usage

```
public class ArrayList<E> implements List<E> {
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

- `new ArrayList<Integer>();`
- `lst.add(0)`
- `lst.add(2)`
- `lst.add(6 * 2)`
- `lst.get(0)`



- `new ArrayList<Integer>();`
- `lst.add(0)`
- `lst.add(2)`
- `lst.add(6 * 2)`
- `lst.get(0)`

```
public class NewArrayList<E>
extends ArrayList<E> {
   @Override { ... }
}
```

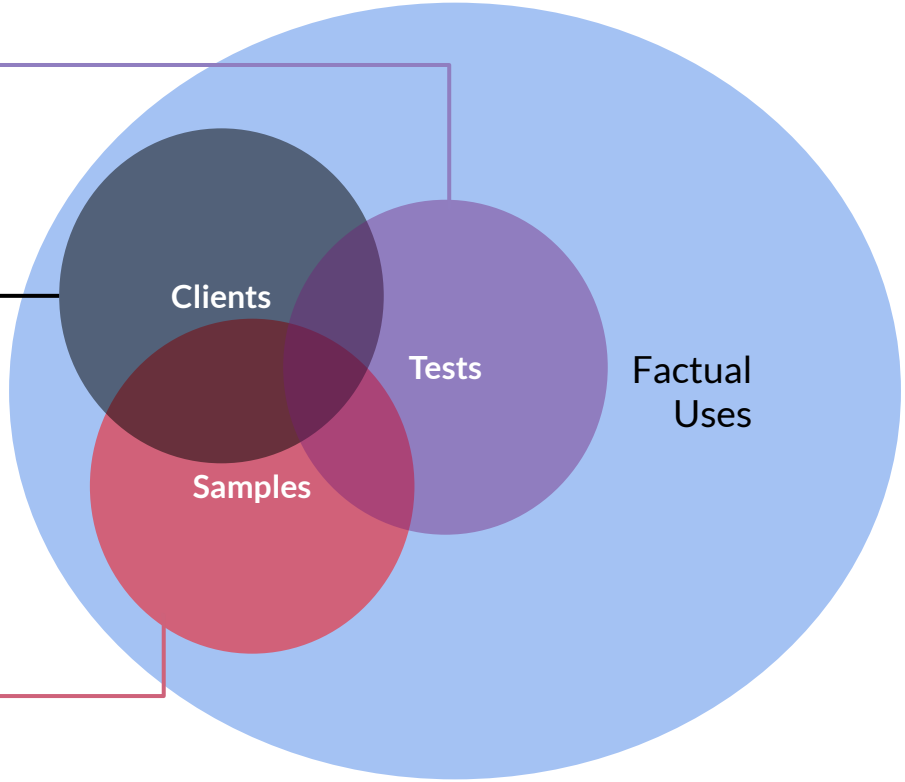
API Design & Usage

```
assert lst.size() == 2
...
```

```
new ArrayList<Integer>();

lst.add(0)
lst.get(0)

public class
NewArrayList<E>
     extends ArrayList<E>
```




```
new ArrayList<Integer>();

lst.add(0)
lst.get(0)
```

Symbol-based modeling of uses (Existing Work)

```
public class ArrayList<E> implements List<E>
{

  public boolean add(E e) { ... }

  private int size;

  ...

}
```

API

Symbol
`ArrayList<E>`
`ArrayList<E>.ArrayList()`
`ArrayList<E>.add(E e)`

Symbols

Symbol-based modeling of uses (Existing Work)

1
3
3

```
ArrayList<Integer> lst = new ArrayList<Integer>();
lst.add(42);
lst.add(1337);
```

2

Symbol
1 `ArrayList<E>`
2 `ArrayList<E>.ArrayList()`
3 `ArrayList<E>.add(E e)`

Symbols

Symbol-based modeling of uses (Existing Work)

1 3 3

```
ArrayList<Integer> lst = new ArrayList<Integer>();
lst.add(42);
lst.add(1337);
```

2

Symbol
1 `ArrayList<E>`
2 `ArrayList<E>.ArrayList()`
3 `ArrayList<E>.add(E e)`

Symbols

```
public class MyArrayList<E> extends ArrayList<E> {
    @Override
    public boolean add(E e) { ... }
}
```

1 2

Symbol
`ArrayList<E>` 1
`ArrayList<E>.add(E e)` 2

Symbols

Syntactic Usage Model (SUM) (Contribution)

```
public class ArrayList<E> implements List<E>
{

  public boolean add(E e) { ... }

  private int size;

  ...

}
```

API

Exported Symbol
`public class ArrayList<E>`
`public ArrayList<E>()`
`public boolean add(E e)`

Syntactic Usage Model (SUM) (Contribution)

New!

```
public class ArrayList<E> implements List<E>
{

  public boolean add(E e) { ... }

  private int size;

  ...

}
```

API

Exported Symbol	Interaction (Use)
public class ArrayList<E>	*Referenced*
	Instantiated
	Extended
public ArrayList<E>()	*Invoked*
public boolean add(E e)	*Invoked*
	Overridden

SUM

Definition of each symbol and allowed interactions

Syntactic Usage Footprint (SUF) (Contribution)

1 2
```
ArrayList<Integer> lst = new ArrayList<Integer>();
```
3
```
lst.add(42);
```
3
```
lst.add(1337);
```

Symbol	Interaction
1 `public class ArrayList<E>`	*Referenced*
2 `public class ArrayList<E>`	*Instantiated*
2 `public ArrayList<E>()`	*Invoked*
3 `public boolean add(E e)`	*Invoked*

SUF 1

Syntactic Usage Footprint (SUF) (Contribution)

1 3 3

```
ArrayList<Integer> lst = new ArrayList<Integer>();   2
lst.add(42);
lst.add(1337);
```

```
public class MyArrayList<E> extends ArrayList<E> {          1
    @Override
    public boolean add(E e) { ... }   2
}
```

Symbol	Interaction
1 `public class ArrayList<E>`	*Referenced*
2 `public class ArrayList<E>`	*Instantiated*
2 `public ArrayList<E>()`	*Invoked*
3 `public boolean add(E e)`	*Invoked*

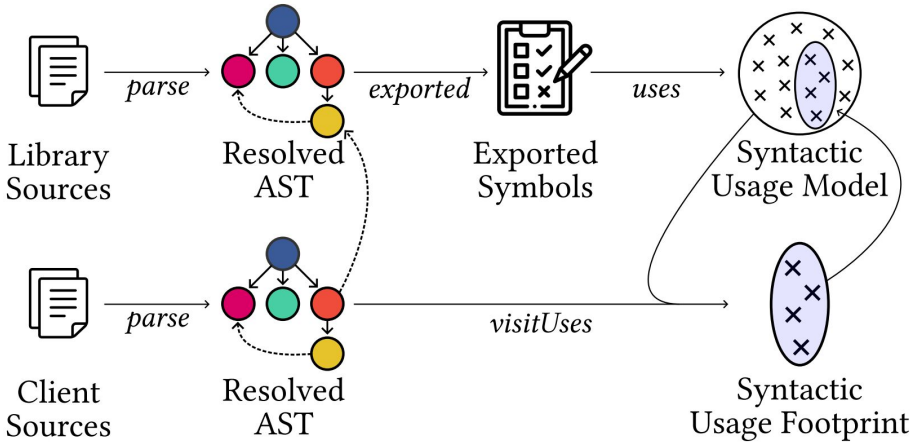
SUF 1

Symbol	Interaction
`public class ArrayList<E>`	*Extended* 1
`public boolean add(E e)`	*Overridden* 2

SUF 2

UCov: SUMs & SUFs for Java

github.com/
alien-tools/ucov



Symbol Type	Use	Example	Resolved API Symbol

Type	Reference	`String s`
		`void f(Integer i)`
		`Integer f()`
		`void f()throws IOException`
		`catch (IOException e)`
		`List<String> l`
		etc.
			`java.lang.String`
			`java.lang.Integer`
			`java.lang.Integer`
			`java.io.IOException`
			`java.io.IOException`
			`java.lang.String`
			etc.

Class	Instantiation Inheritance	`new Integer(42)` `class T extends Thread`	`java.lang.Integer` `java.lang.Thread`

Interface	Implementation
	
	Extension
		`class R implements Runnable`
		`Runnable r = ()→{...}`
		`interface R extends Runnable`
			`java.lang.Runnable`
			`java.lang.Runnable`
			`java.lang.Runnable`

Constructor	Invocation	`new Integer(42)`	`java.lang.Integer(int)`

Method	Invocation
	Static invocation
	Overriding
		`"a".length()`
		`String.valueOf(42)`
		`@Override void run()`
		`Runnable r = ()→{...}`
			`java.lang.String.length()`
			`java.lang.String.valueOf(int)`
			`java.lang.Thread.run()`
			`java.lang.Runnable.run()`

Field	Field read Field write	`Integer.MAX_VALUE` `Point.x = 2`	`java.lang.Integer.MAX_VALUE` `java.awt.Point.x`

Evaluation: Selected Projects

zenodo.org/records/
10571867

Classical

```
//commons.apache.org
//proper/commons-cli/usage.html


CommandLineParser parser = new
DefaultParser();
Options options = new Options();
options.addOption("a", "all",
  false, "do not hide entries");
options.addOption("C", false,
  "list entries by columns");
```

commons-cli

Framework

```
//sparkjava.com
//documentation#routes


get("/", (rq,rs) → { ... });
put("/", (rq,rs) → { ... });
post("/", (rq,rs) → { ... });
```

sparkjava

Fluent

```
//jsoup.org/cookbook
//input/load-document-from-url


Document doc = Jsoup
  .connect("http://example.com")
  .data("query", "Java")
  .userAgent("Mozilla")
  .cookie("auth", "token")
  .timeout(3000)
  .post();
```

jsoup

Clients + **Samples** + **Tests**

Unexpected

Library

```
// src/main/java/org/jsoup/internal/StringUtil.java

/**
 A minimal String utility class. Designed for <b>internal</b>
 jsoup use only - the API and outcome may change without
 notice.
 */
public final class StringUtil {
    ...
}
```

Sample (& Clients too!)

```
// src/main/java/org/jsoup/examples/HtmlToPlainText.java

package org.jsoup.examples;

public class HtmlToPlainText {
    ...
    private static class FormattingVisitor
                        implements NodeVisitor {
        ...
        public void tail(Node node, int depth) {
            String name = node.nodeName();
            if (StringUtil.in(name, ...))
                append("\n");
            else if (name.equals("a"))
                append(String.format(" <%s>",
                        node.absUrl("href")));
        }
        ...
    }
}
```

Unexpected

Library

```
// src/main/java/spark/Route.java

package spark;

/**
This is a functional interface and can therefore be used as
the assignment target for a lambda expression or method
reference.
*/

@FunctionalInterface
public interface Route {

    Object handle(Request req, Response res) throws Exception;

}
```

```
get("/", (req, res) → { ... });
```

Sample

```
// src/main/java/TodoList.java

import model.*;
import spark.*;
import spark.template.velocity.*;
import java.util.*;
import static spark.Spark.*;

/**
 * This class uses the ICRoute interface to create
 * void routes. The response for an ICRoute is
 * rendered in an after-filter.
 */
public class TodoList {
    ...
    @FunctionalInterface
    private interface ICRoute extends Route {
        default Object handle(Request request,
                Response response) throws Exception {
            handle(request);
            return "";
        }
        void handle(Request request)
                                    throws Exception;
    }
}
```

Unexpected

Library

```
// src/java/org/apache/commons/cli/Parser.java

public class PosixParser extends Parser {

    /**
     * @param stopAtNonOption specifies whether to stop interpreting
     * the arguments when a non option has been encountered and to add
     * them to the CommandLines args list.
     */
     public CommandLine parse(Options options, String[] arguments,
                boolean stopAtNonOption) throws ParseException { ... }
}
```

```
 // java -jar ./main.jar -d -t

 PosixParser p = new PosixParser();
 Options o = new Options();
 o.addOption("t", false, "display current time");
 CommandLine c = p.parse(o, args, false);
```

Client

```
package org.obolibrary.robot;

/**
 * A custom CommandLineParser that ignores
 * unrecognized options without throwing an
 * exception. See
 * http://stackoverflow.com/a/8613949
 */
public class ExtPosixParser extends PosixParser {
  ...
  @Override
  protected void processOption(final String arg,
                  final ListIterator iter)
       throws ParseException { ... }
}
```

Conclusion

- Interaction-based modeling of uses
- Support for discussing API design
- Support for discussing alignment with in-the-wild uses

github.com/
alien-tools/ucov

zenodo.org/records/
10571867

UCov
Git
Repository

Exploratory
Case Study
Artifacts

The End

Appendix

API

```
public class ArrayList<E> implements List<E> {
  public boolean add(E e) { ... }
  private int size;
  ...
}
```

SUM

Symbol	Interaction
`public class ArrayList<E>`	*Referenced*
	Instantiated
	Extended
`public ArrayList<E>()`	*Invoked*
`public boolean add(E e)`	*Invoked*
	Overridden

SUF 1

Symbol	Interaction
`public class ArrayList<E>`	*Referenced*
	Instantiated
`public ArrayList<E>()`	*Invoked*
`public boolean add(E e)`	*Invoked*

SUF 2

Symbol	Interaction
`public class ArrayList<E>`	*Extended*
`public boolean add(E e)`	*Overridden*

Usage model in UCov

Symbol Type	Use	Example	Resolved API Symbol
Type	Reference	`String s`	`java.lang.String`
		`void f(Integer i)`	`java.lang.Integer`
		`Integer f()`	`java.lang.Integer`
		`void f()throws IOException`	`java.io.IOException`
		`catch (IOException e)`	`java.io.IOException`
		`List<String> l`	`java.lang.String`
		etc.	*etc.*
Class	Instantiation	`new Integer(42)`	`java.lang.Integer`
	Inheritance	`class T extends Thread`	`java.lang.Thread`
Interface	Implementation	`class R implements Runnable`	`java.lang.Runnable`
		`Runnable r = ()→{...}`	`java.lang.Runnable`
	Extension	`interface R extends Runnable`	`java.lang.Runnable`
Constructor	Invocation	`new Integer(42)`	`java.lang.Integer(int)`
Method	Invocation	`"a".length()`	`java.lang.String.length()`
	Static invocation	`String.valueOf(42)`	`java.lang.String.valueOf(int)`
	Overriding	`@Override void run()`	`java.lang.Thread.run()`
		`Runnable r = ()→{...}`	`java.lang.Runnable.run()`
Field	Field read	`Integer.MAX_VALUE`	`java.lang.Integer.MAX_VALUE`
	Field write	`Point.x = 2`	`java.awt.Point.x`

Inspired and refined from the work of Qiu et al.

Dong Qiu, Bixin Li, and Hareton Leung. Understanding the API usage in Java. Information and Software Technology, 73:81–100, May 2016. ISSN 09505849. doi:10.1016/j.infsof.2016.01.011.

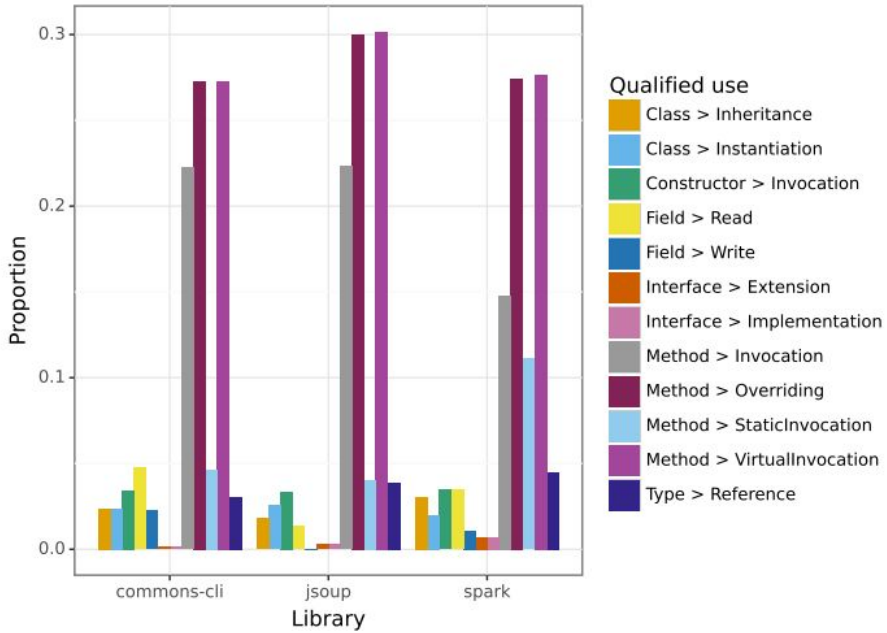
Breakdown per lib

			CLI	JSOUP	SPARK
SUM		API symbols	291	1,138	771
		Legal uses	755	2,941	1,960
SUF	Symbols used	All	205 (70%)	608 (53%)	301 (39%)
		Clients	153 (53%)	282 (25%)	179 (23%)
		Tests	179 (62%)	586 (51%)	248 (32%)
		Samples	24 (8%)	47 (4%)	134 (17%)
	Unique uses	All	367 (49%)	1,028 (35%)	476 (24%)
		Clients	266 (35%)	456 (16%)	278 (14%)
		Tests	315 (42%)	967 (33%)	400 (20%)
		Samples	38 (5%)	69 (2%)	211 (11%)
	Total uses	All	79,284	29,979	23,882
		Clients	74,453	15,246	20,827
		Tests	4,690	14,511	1,605
		Samples	141	222	1450

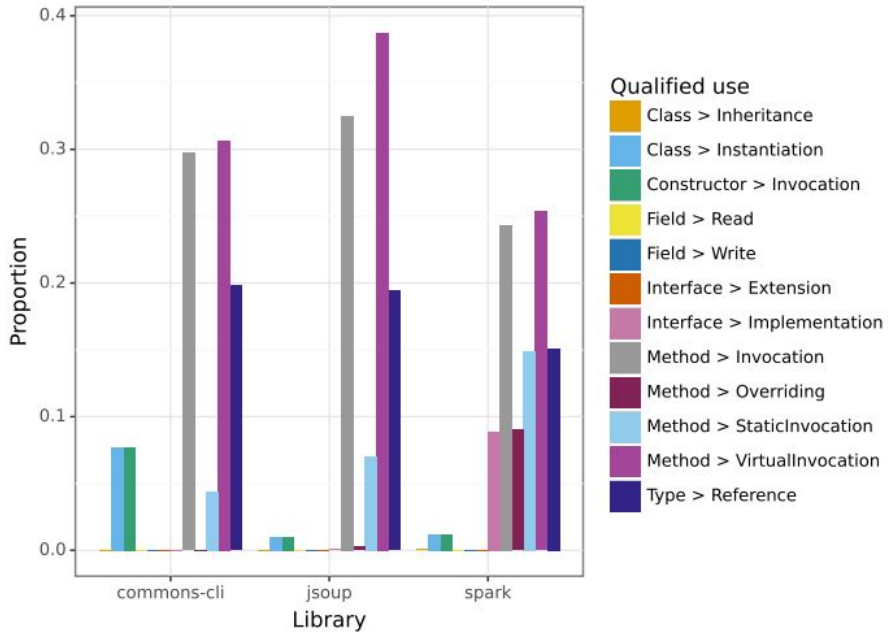
Descriptive statistics of the subject libraries, extracted from GitHub on October 30, 2023

	COMMONS-CLI	JSOUP	SPARK
Last release date	2021/10/29	2023/04/29	2020/10/08
Version	1.5.0	1.16.1	2.9.3
Since	2002	2010	2011
Stars	309	10.4k	9.5k
Commits	1,374	1,889	1,067
Size (LoC)	6,307	27,817	11,298
Contributors	46	97	100
Clients	49k	131k	30k

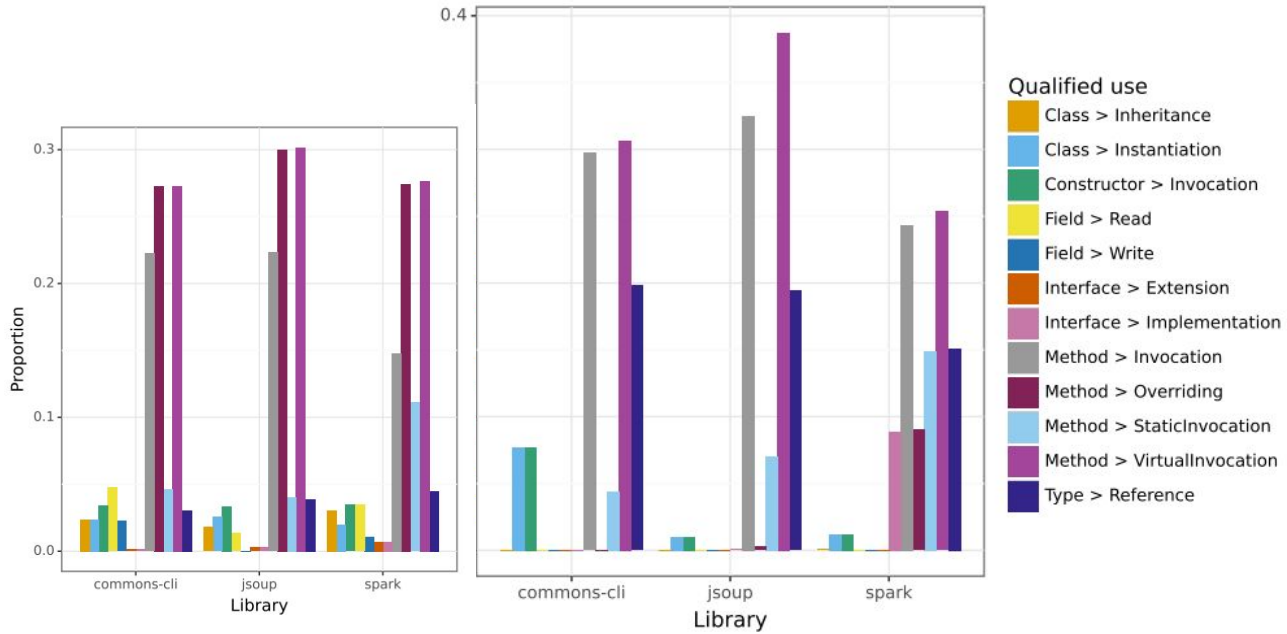
Syntactic Usage Models of the three libraries commons-cli, jsoup, and spark



Syntactic Usage Footprints of the three libraries commons-cli, jsoup, and spark

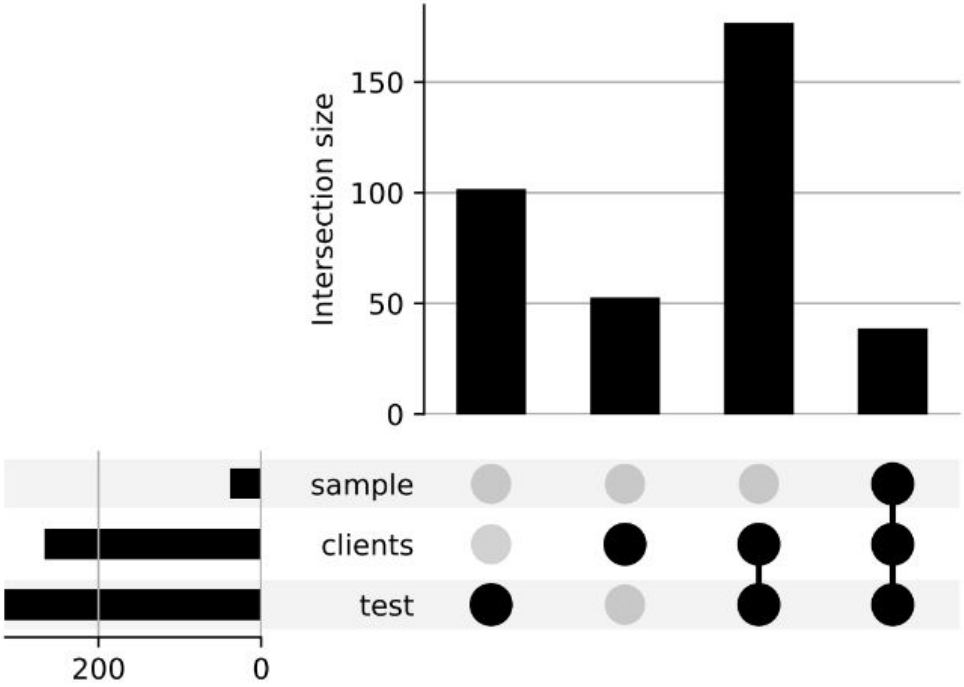


Usage Profiles (Side by Side) of the three libraries commons-cli, jsoup, and spark

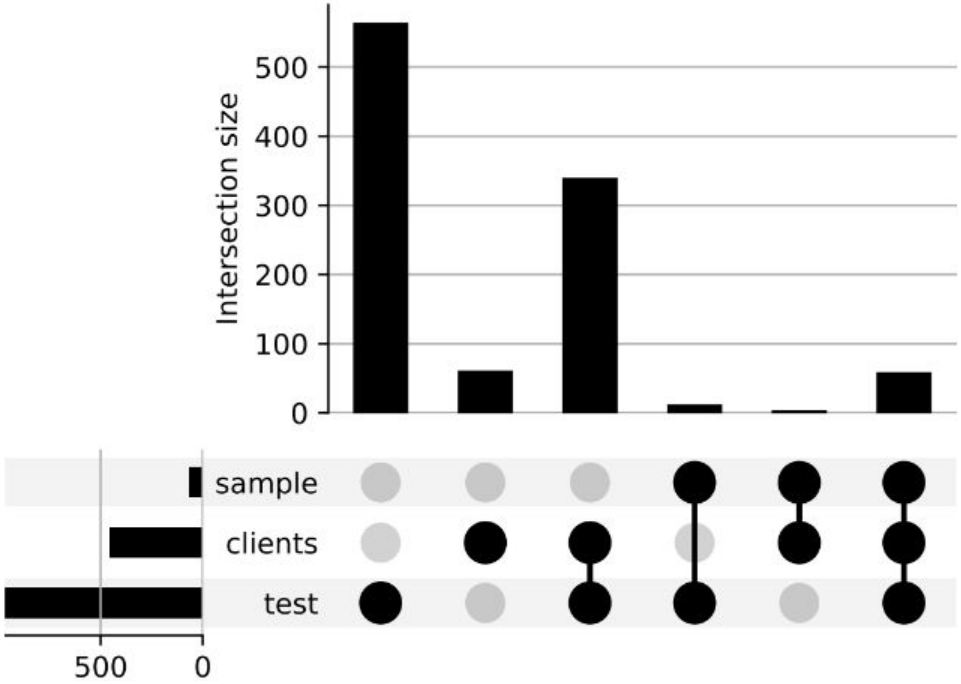


SUM SUF

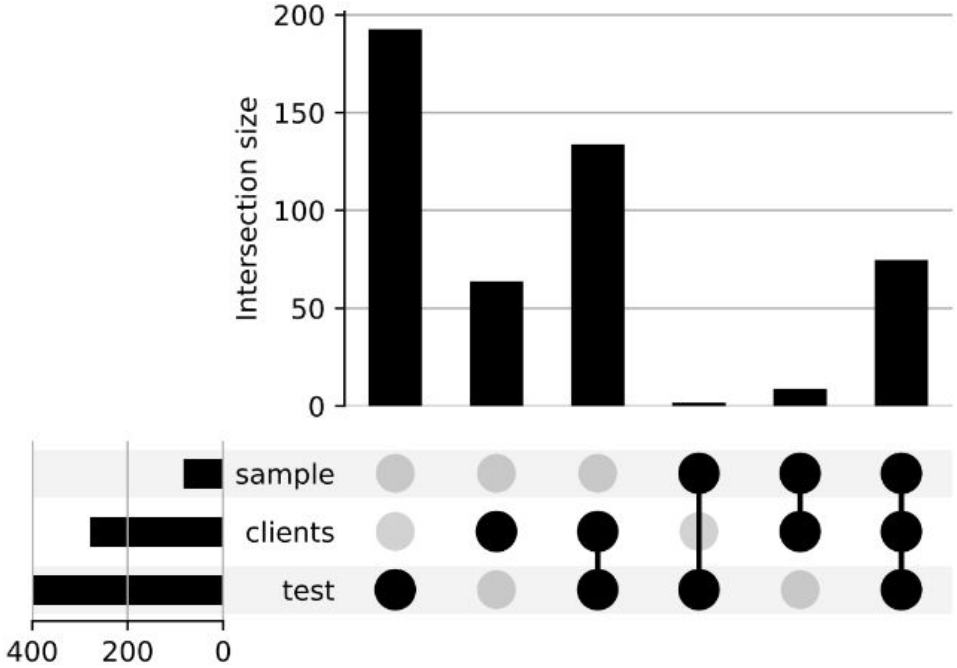
Syntactic Usage Footprint intersections (Side by Side) of commons-cli



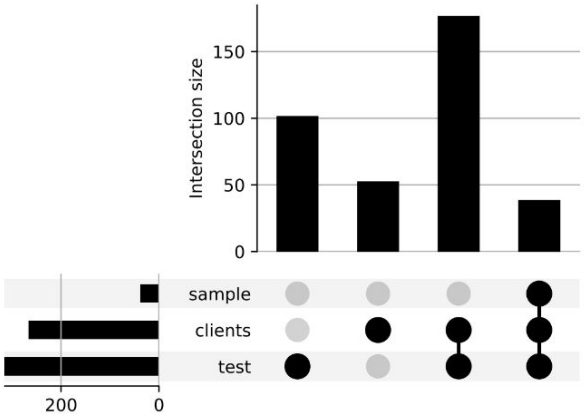
Syntactic Usage Footprint intersections (Side by Side) of jsoup



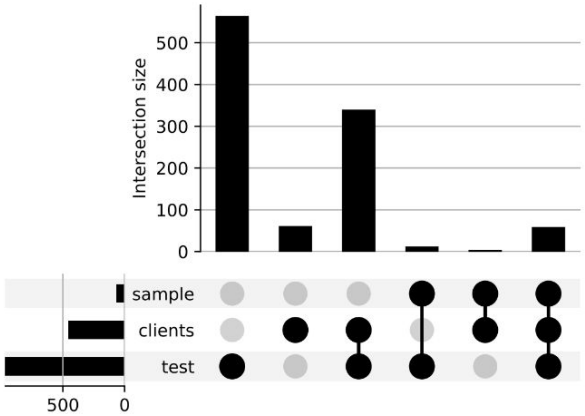
Syntactic Usage Footprint intersections (Side by Side) of spark



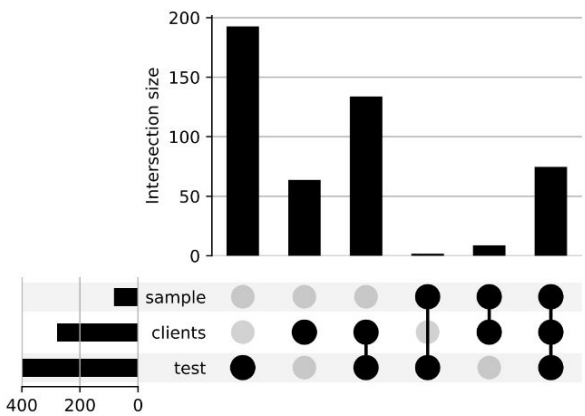
Syntactic Usage Footprint intersections (Side by Side) of the three libraries commons-cli, jsoup, and spark



commons-cli



jsoup



spark (sparkjava)

EOF