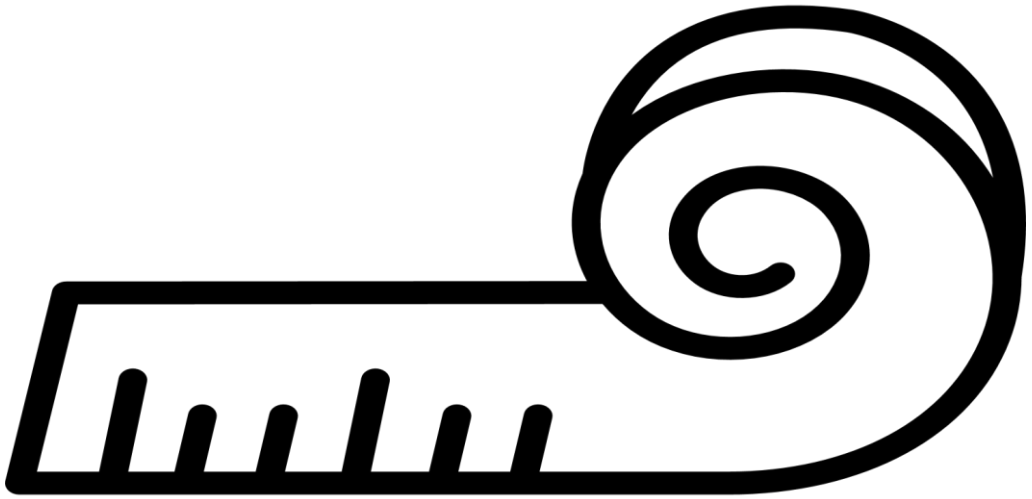
CODE COVERAGE



KEVIN DÍAZ MARRERO – ALU0100880625@ULL.EDU.ES

PEDRO MIGUEL LAGÜERA CABRERA – ALU0100891485@ULL.EDU.ES

INDEX

- What Is It?
- How is it measured?
- Coverage Criteria
- Types of Lines after Coverage
- What Code Coverage is and isn't
- Why Code Coverage?
- EclEmma

WHAT IS IT?

- It measures how much a program is covered by its tests.
- Its unit is always a percentage of code covered.

+ Code Coverage → - Chance of Bugs

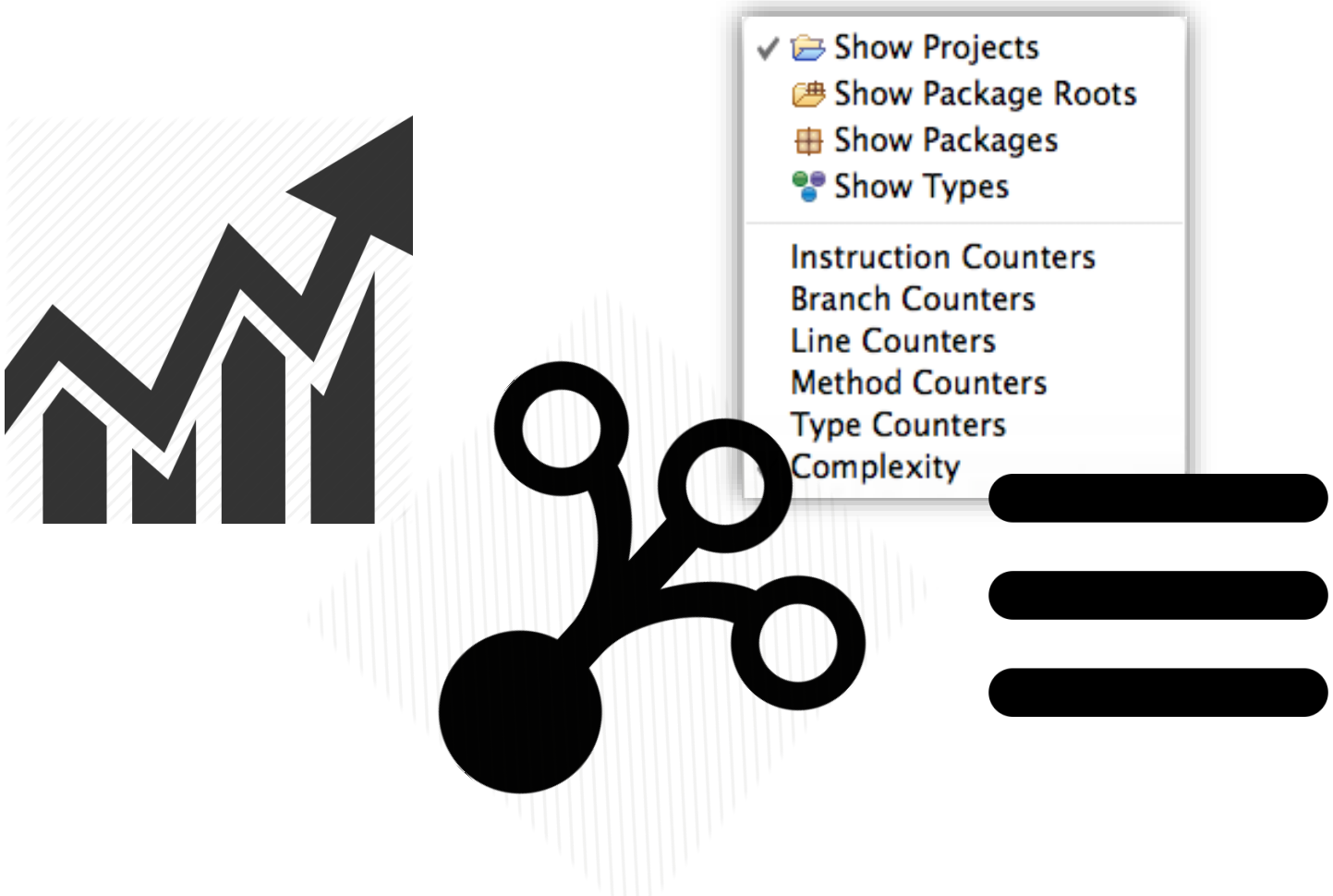
JaCoCo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed
org.jacoco.agent.rt		81%		86%	32	106	57	262	24	70	7
jacoco-maven-plugin		90%		83%	18	114	23	276	2	66	0
org.jacoco.core		99%		99%	25	864	30	2,022	17	506	0
org.jacoco.report		99%		98%	7	446	8	1,107	2	306	0
org.jacoco.ant		99%		99%	4	137	8	385	3	96	0
org.jacoco.agent		85%		75%	3	11	5	30	1	7	0
Total	514 of 16,926	97%	42 of 1,151	96%	89	1,678	131	4,082	49	1,051	7

HOW IS IT MEASURED?

$$CodeCoverage = \frac{NumberOfLinesOfCodeExercised}{TotalNumberOfLinesOfCode} * 100$$

- Instructions
- Branches
- Lines
- Methods
- Types
- Complexity



COVERAGE CRITERIA

- Instruction Coverage
- Branch Coverage
- Method Coverage
- Condition Coverage
- Parameter Value Coverage
- Loop Coverage

```
public static void main(String[] args){
  while(i != 0){
    if(i == 4)
      System.out.println("Four !!");
  }
}
```

COVERAGE CRITERIA

- Instruction
- Branch
- Method
- Condition
- Parameter Value

```
int foo (int x, int y)
{
    int z = 0;
    if ((x>0) && (y>0))
    {
        z = x;
    }
    return z;
}
```

TYPES OF LINES AFTER COVERAGE

- Covered Lines
- Partly Covered Lines
- Not executed lines

```
public boolean addAll(int index, Collection c) {
    if(c.isEmpty()) {
        return false;
    } else if(_size == index || _size == 0) {
        return addAll(c);
    } else {
        Listable succ = getListableAt(index);
        Listable pred = (null == succ) ? null : succ.prev();
        Iterator it = c.iterator();
        while(it.hasNext()) {
            pred = insertListable(pred,succ,it.next());
        }
        return true;
    }
}
```

1 of 2 branches missed.

Press 'F2' for focus

WHAT CODE COVERAGE IS AND ISN'T

- 100% Code Coverage ≠ bug-free program.
- Working tests ≠ 100% coverage.
- **No** additional code.
- Starts **alongside** the developement.
- Strives to find **untested code**.
- 100% coverage **isn't necessary**.

WHY CODE COVERAGE?

- To know:
  - **How well** our tests actually test our code.
  - Whether we have **enough testing** in place.
- Identify **dead** code.
- To maintain the test **quality**.
- **Early** detection of flaws.

	Clover	Cobertura	JaCoCo
Licensing Model	Commercial	Open Source	Open Source
Google Trend	Downward	Rising	Rising
Github star rating	n/a	358 stars	415 stars
Instrumentation	Runtime	Runtime	Runtime & Compile time
Ease of setup	Comparatively difficult	Easy	Easy
Reporting	Best	OK	OK
Eclipse IDE Integration	None	None	Good. Eclemma Plugin.
Multiple Session Coverage	None	None	YES

ECLEMMA

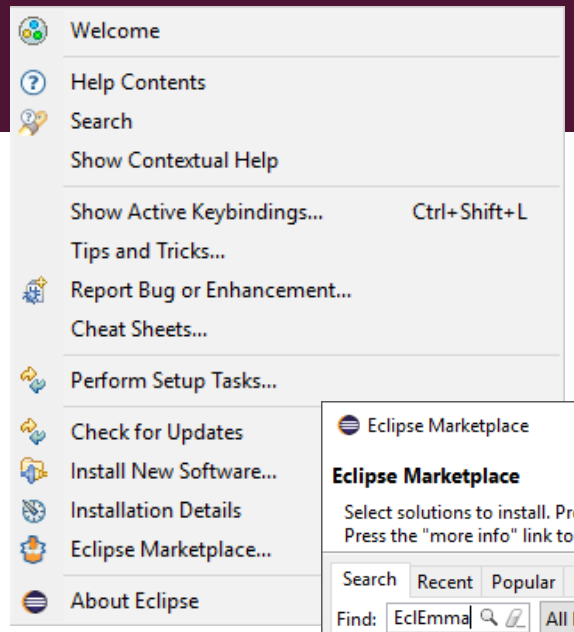
FREE JAVA TOOL WHICH IMPLEMENTS CODE COVERAGE INTO THE ECLIPSE WORKBENCH.

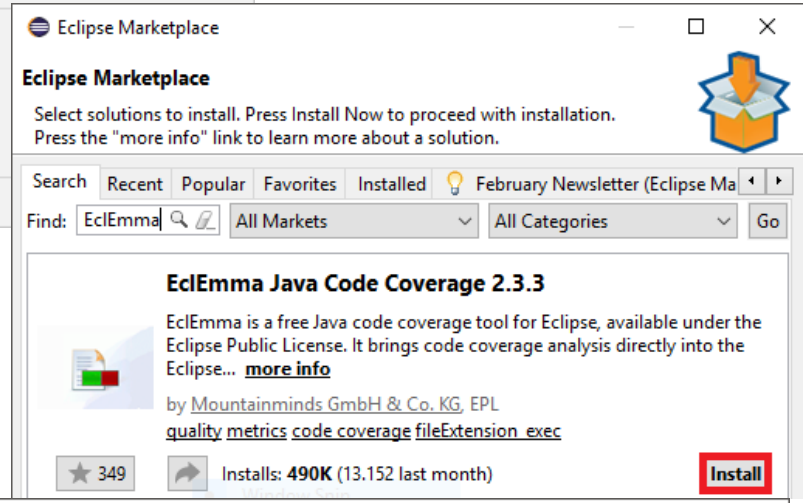
SUBINDEX - ECLEMMA

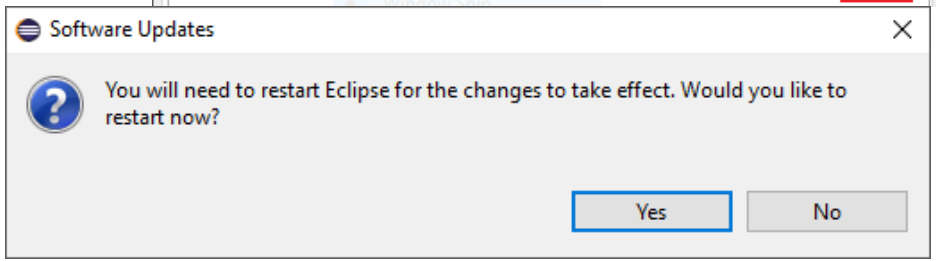
- Installation
- What's new?
- Using the Coverage View
- Toolbar
- Filters
- Source Code Annotation
- Coverage Properties
- Coverage Decorators
- Coverage Sessions
- Session Import and Export

INSTALLATION

- About → Eclipse Marketplace…
- Search for ‘EclEmma’.
- Press ‘Install’.
- Agree to the terms and conditions.
- Restart Eclipse

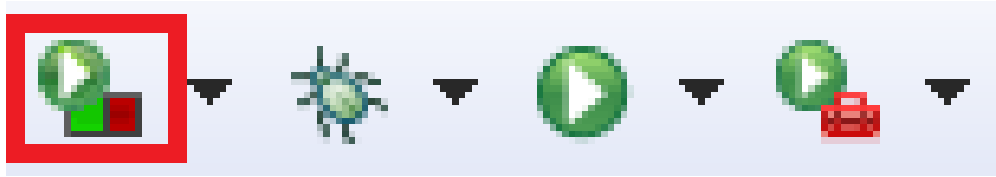






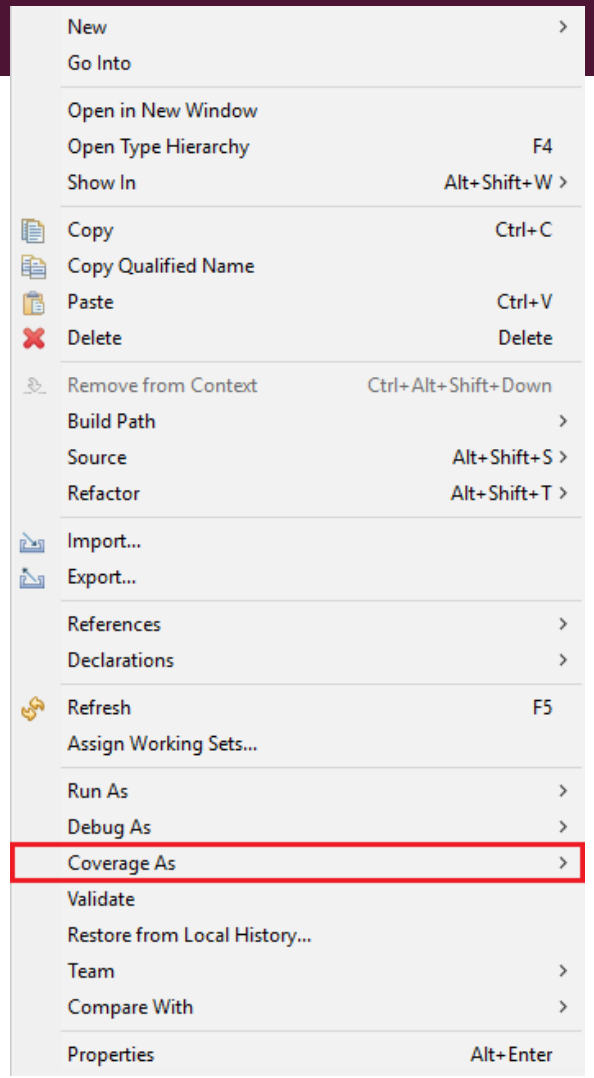
WHAT'S NEW?

- New **coverage button** next to the run and debug buttons.
- Clicking it will run the current context's code coverage.
- Dropdown menu with recent files and coverage configurations.



WHAT'S NEW?

- Right-click a file, package, folder or Project.
  - There's a new option called '**Coverage As**'.
- Depending on the type of file(s):
  - Junit Test
  - Java Application



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
CoCoTests	85,6 %	989	166	1.155
src	85,6 %	989	166	1.155
fecha	85,6 %	989	166	1.155
Fecha.java	85,2 %	717	125	842
TestFecha.java	86,4 %	242	38	280
MainFecha.java	90,9 %	30	3	33

USING THE COVERAGE VIEW

TOOLBAR

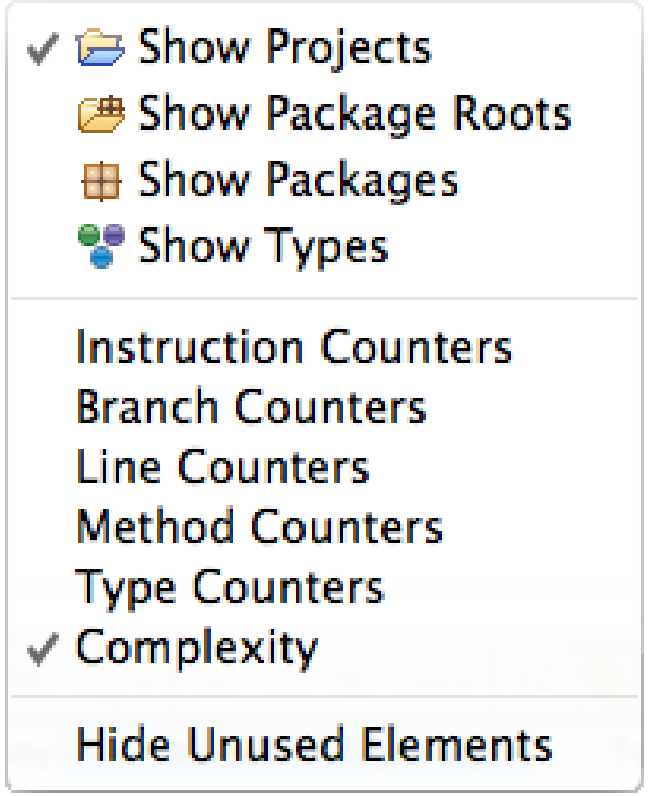
1. Coverage Last Launched
2. Dump Execution Data
3. Remove Active Session
4. Remove All Sessions
5. Merge Sessions
6. Select Session
7. Collapse All
8. Link with Current Selection

Coverage

TestAllPackages (Feb 13, 2012 9:52:22 AM)

Element	Coverage	Covered Lines	Missed Lines	Total Lines
commons-collections	80.7 %	11092	2646	13738
src	80.7 %	11092	2646	13738
org.apache.commons.collections	77.1 %	3991	1188	5179
org.apache.commons.collections.bag	66.9 %	234	116	350
org.apache.commons.collections.bidimap	91.2 %	964	93	1057
AbstractBidiMapDecorator.java	85.7 %	6	1	7
AbstractBidiMapDecorator	85.7 %	6	1	7
AbstractBidiMapDecorator(BidiMap)	100.0 %	2	0	2
getBidiMap()	100.0 %	1	0	1
getKey(Object)	100.0 %	1	0	1
inverseBidiMap()	0.0 %	0	1	1

FILTERS



- Show Elements
  - Projects
  - Package Roots
  - Packages
  - Types
- Counter Mode
- Hide Unused Elements

SOURCE CODE ANNOTATION

- Source lines containing executable code:
  - **Fully covered lines.**
  - **Partly covered lines.**
  - **Lines not executed at all.**
- Colored diamonds represent lines with decision branches.
  - **Fully covered branches.**
  - **Partly covered branches.**
  - **No branches executed.**

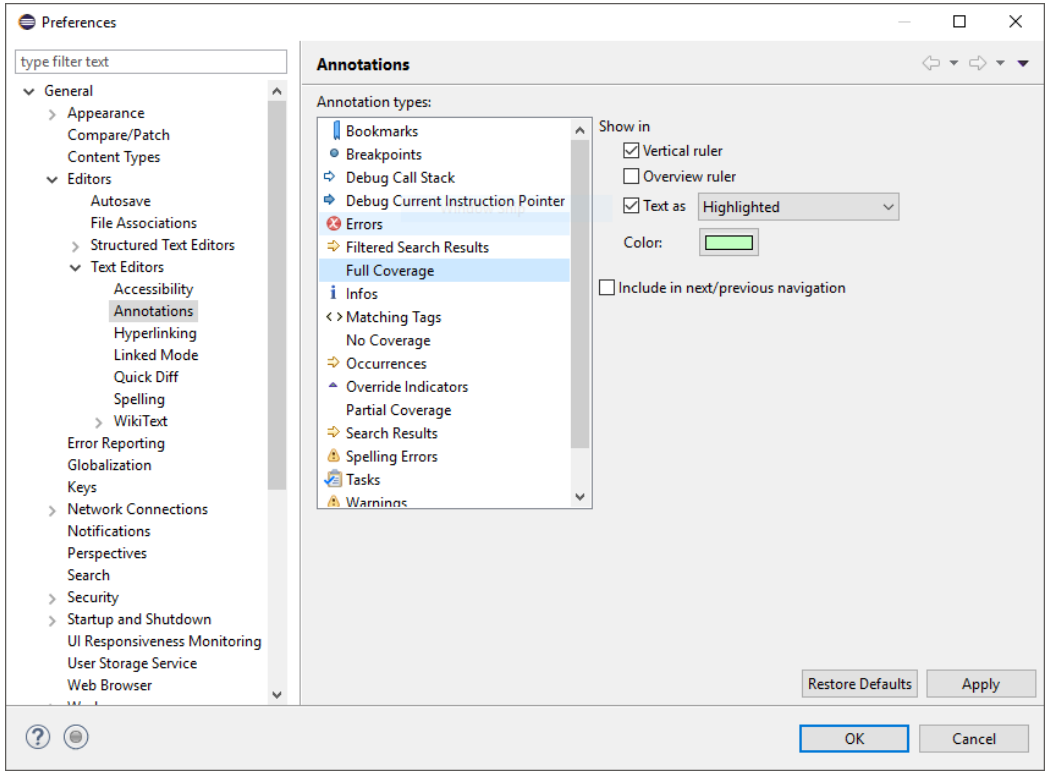
```
6   public static boolean isValid(String string){
7       if(string==null){
8           throw new IllegalArgumentException();
9       }
10      if(string.length()>2&&string.length()<9){
11          return true;
12      }
13      return false;
14  }
15
```

```java
public boolean addAll(int index, Collection c) {
    if(c.isEmpty()) {
        return false;
    } else if(_size == index || _size == 0) {
        return addAll(c);
    } else {
        Listable succ = getListableAt(index);
        Listable pred = (null == succ) ? null : succ.prev();
        Iterator it = c.iterator();
        while(it.hasNext()) {
            pred = insertListable(pred, succ, it.next());
        }
        return true;
    }
}
```

1 of 2 branches missed.

Press 'F2' for focus

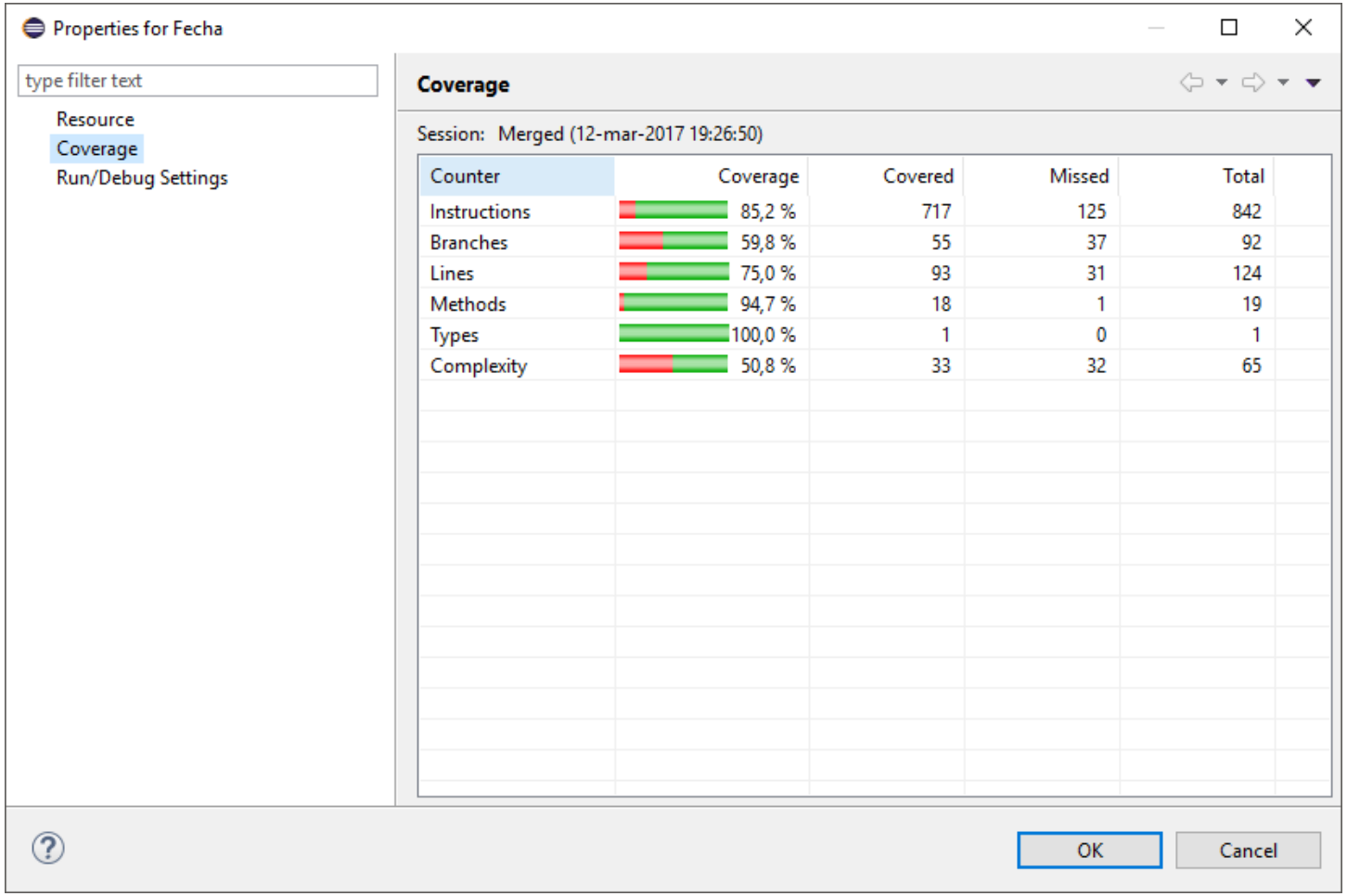
SOURCE CODE ANNOTATION

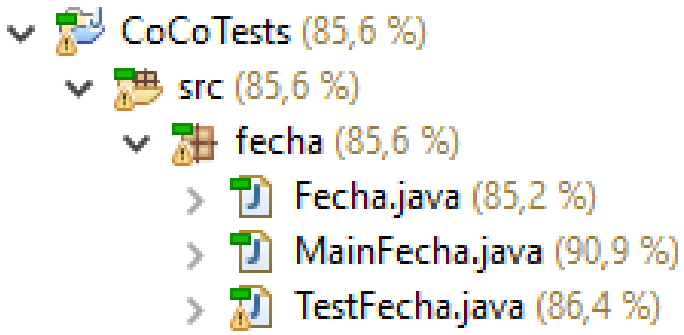


- To change these colors, go to:
- **Preferences → General →**
  - **Editors → Text Editors → Annotations**
- The corresponding entries are:
  - Full Coverage
  - Partial Coverage
  - No Coverage

COVERAGE PROPERTIES

- Summary of all coverage counters.
- Right-click file, open **Properties**, 'Coverage'.



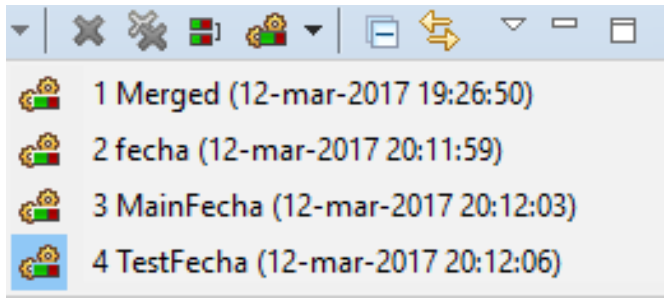


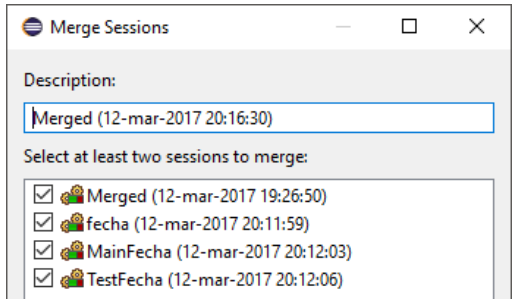
COVERAGE DECORATORS

- Decorators are available for the active session.
  - A little green/red bar on the elements' icons
  - Percentage value next to the names
- To enable this feature go to:
  - **Window** → **Preferences** → **General** → **Appearance** → **Label Decorators**
- Select *Java Code Coverage* and press **OK**.

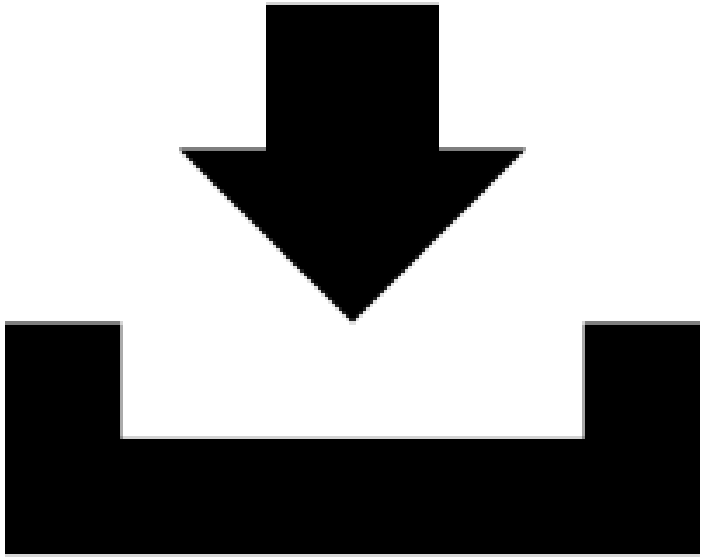
COVERAGE SESSIONS

- Session is the code coverage of a particular program run.
- Automatically created after each coverage launch.
- Can be imported, exported and removed.
- All coverage sessions are deleted when the workbench is closed.
- Multiple test launches result in multiple different coverage sessions.
  - **Merge** selects existing sessions and combines them in a single session.





SESSION IMPORT AND EXPORT



- **Import**
  - ***File → Import... → Other → Coverage Session***
  - The file needs to be in ***.exec*** format.
- **Export**
  - ***File → Export... → Java → Coverage Report***
  - **Zipped HTML** is highly recommended.

REFERENCES

- https://www.youtube.com/watch?v=sc0PW5KOXnY
- http://www.eclemma.org/jacoco/trunk/doc/counters.html
- https://confluence.atlassian.com/display/CLOVER/About+Code+Coverage
- http://www.bullseye.com/minimum.html
- http://www.eclemma.org/index.html
- http://www.eclemma.org/jacoco/trunk/doc/counters.html
- https://en.wikipedia.org/wiki/Java_Code_Coverage_Tools#Clover
- https://confluence.atlassian.com/display/CLOVER/Clover-for-Eclipse
- https://confluence.atlassian.com/display/CLOVER/Clover-for-Eclipse+Installation+Guide