Code Coverage

Outlines

- Code Coverage.
- EMMA
- Installing EMMA
- Running EMMA
- View Result

Code Coverage

- Is a measure used in software testing. It describes the degree to which the source code of a program has been tested.
- coverage analysis attempts to address questions about when to stop testing, or the amount of testing that is enough for a given program.

Code Coverage

- Code Coverage analysis allows determining the completeness of the test cases, and the percentage of the code exercised by executing the test cases.

Coverage Criteria

- To measure how well the program is exercised by a test suite, one or more *coverage criteria* are used.
  - Statement Coverage
  - Branch Coverage
  - Condition Coverage
  - Path Coverage
  - Function Coverage

EMMA

- EMMA is a free code coverage tool.
- Supported coverage types: *class*, *method*, *line*, *basic block*.
- EMMA can detect when a single source code line is covered only partially.

Netbeans Plugin for EMMA

- The functionality provided by the plugin helps to visually (and quickly) identify the portions of java code with low coverage and helps in targeted tests development.

Feature

- Java sources coloring according to the coverage information from the latest unit tests execution.
- Automated java code markup updated after running unit tests or reopening file

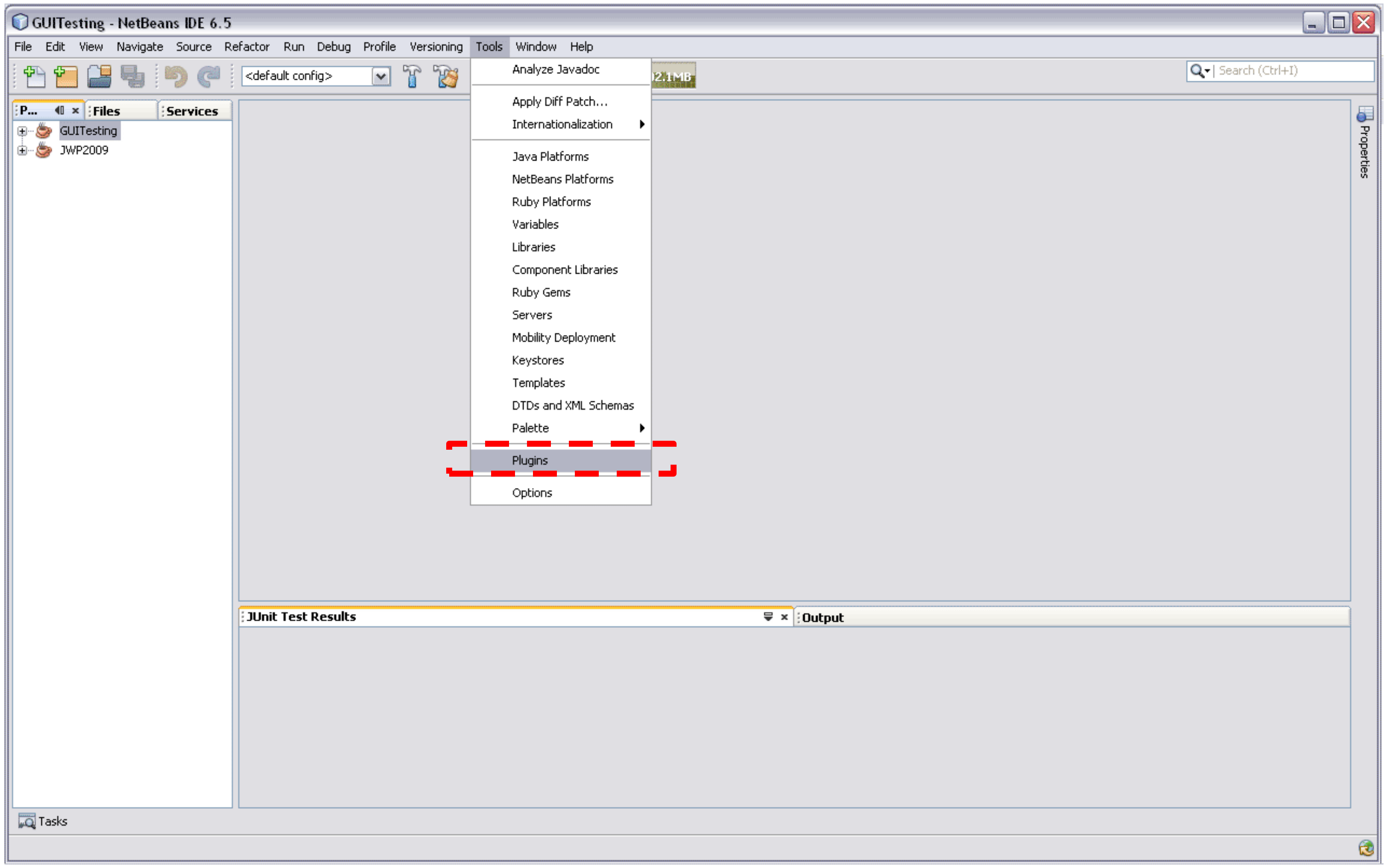
Feature

- Code coverage markup info displayed at the java editor sidebar
- Currently Java Application, Java Library, Java Project with Existing Sources and NetBeans module projects are supported.

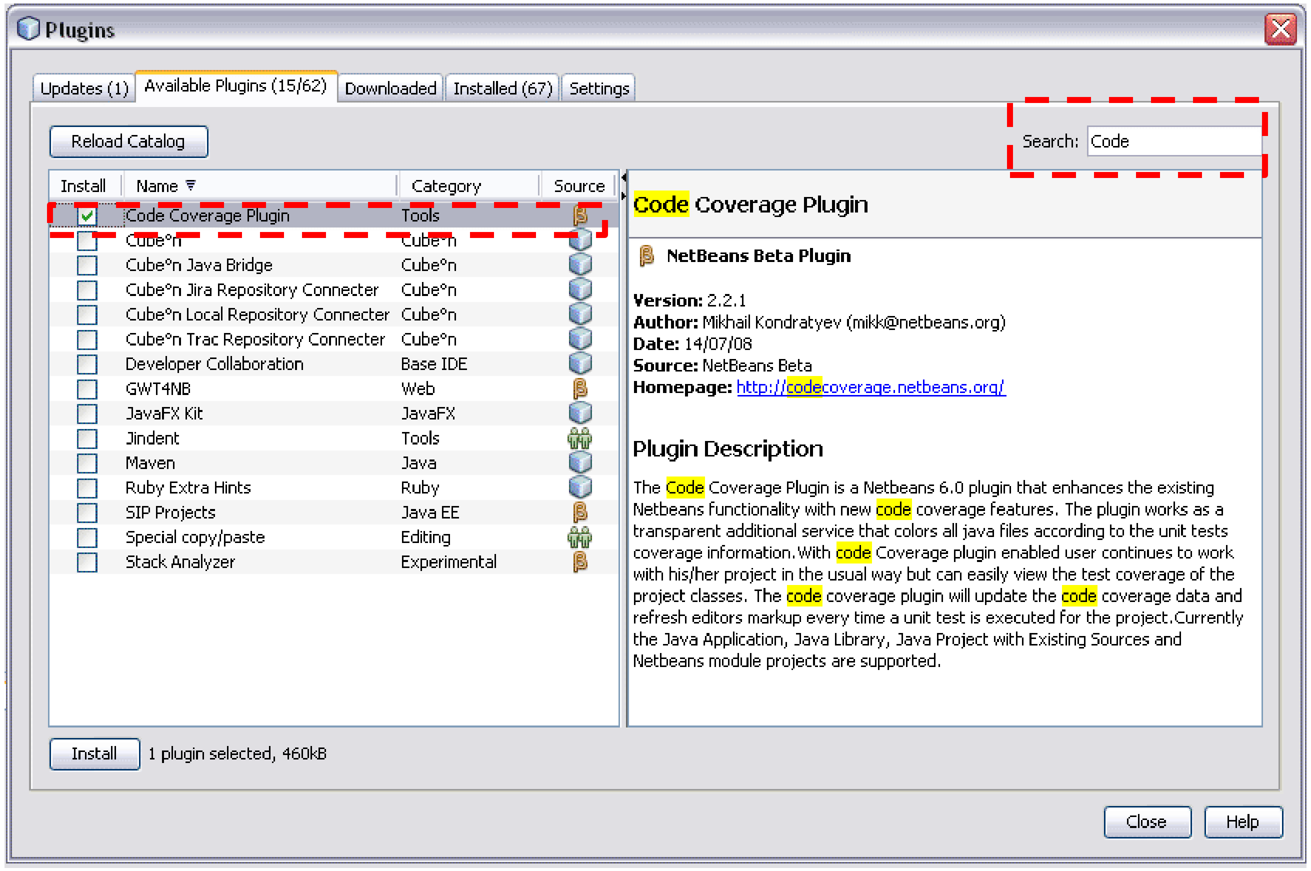
EMMA Plugin in Use

- You need to develop your JUnit Test.
- Make sure that Coverage Plugin is installed.
- Activate the coverage collection action.
- view coverage reports.

Install Code Coverage Plugin



Install Code Coverage Plugin



Search: Code

Install	Name	Category	Source
✓	Code Coverage Plugin	Tools	β
☐	Cube°n	Cube°n	
☐	Cube°n Java Bridge	Cube°n	
☐	Cube°n Jira Repository Connecter	Cube°n	
☐	Cube°n Local Repository Connecter	Cube°n	
☐	Cube°n Trac Repository Connecter	Cube°n	
☐	Developer Collaboration	Base IDE	
☐	GWT4NB	Web	β
☐	JavaFX Kit	JavaFX	
☐	Jindent	Tools	
☐	Maven	Java	
☐	Ruby Extra Hints	Ruby	
☐	SIP Projects	Java EE	β
☐	Special copy/paste	Editing	
☐	Stack Analyzer	Experimental	β

1 plugin selected, 460kB

Code Coverage Plugin

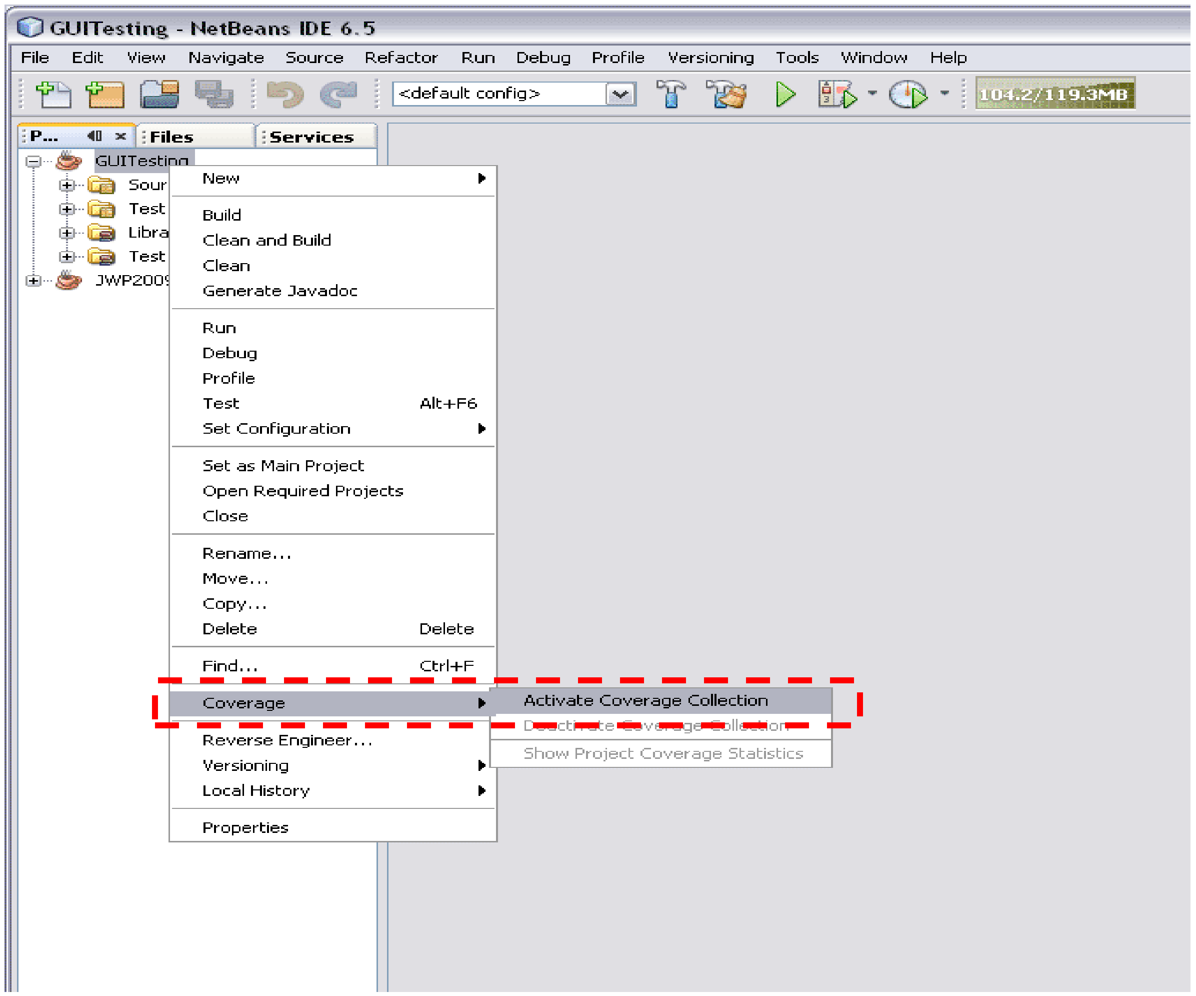
NetBeans Beta Plugin

Version: 2.2.1
Author: Mikhail Kondratyev (mikk@netbeans.org)
Date: 14/07/08
Source: NetBeans Beta
Homepage: http://codecoverage.netbeans.org/

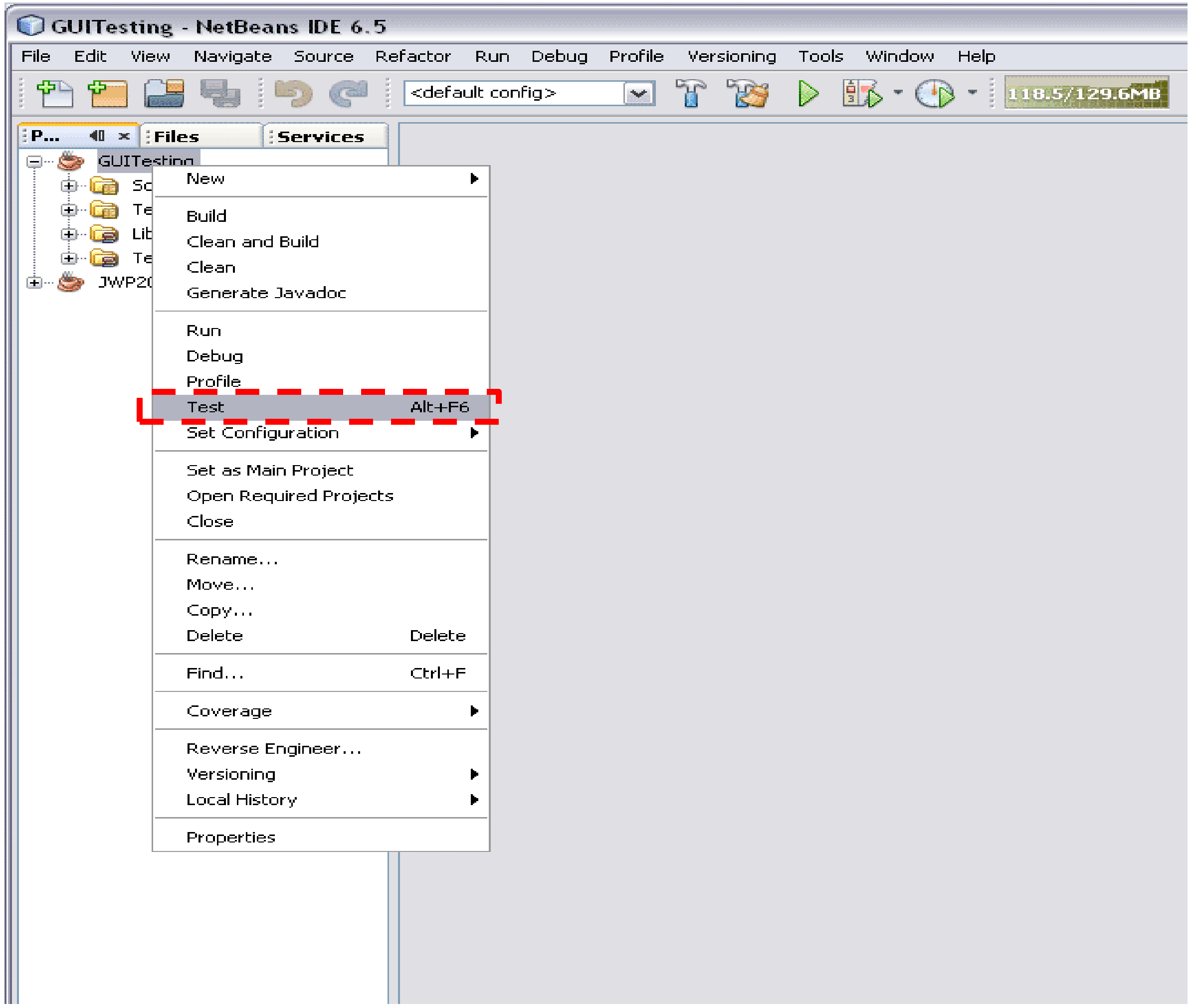
Plugin Description

The Code Coverage Plugin is a Netbeans 6.0 plugin that enhances the existing Netbeans functionality with new code coverage features. The plugin works as a transparent additional service that colors all java files according to the unit tests coverage information. With code Coverage plugin enabled user continues to work with his/her project in the usual way but can easily view the test coverage of the project classes. The code coverage plugin will update the code coverage data and refresh editors markup every time a unit test is executed for the project. Currently the Java Application, Java Library, Java Project with Existing Sources and Netbeans module projects are supported.

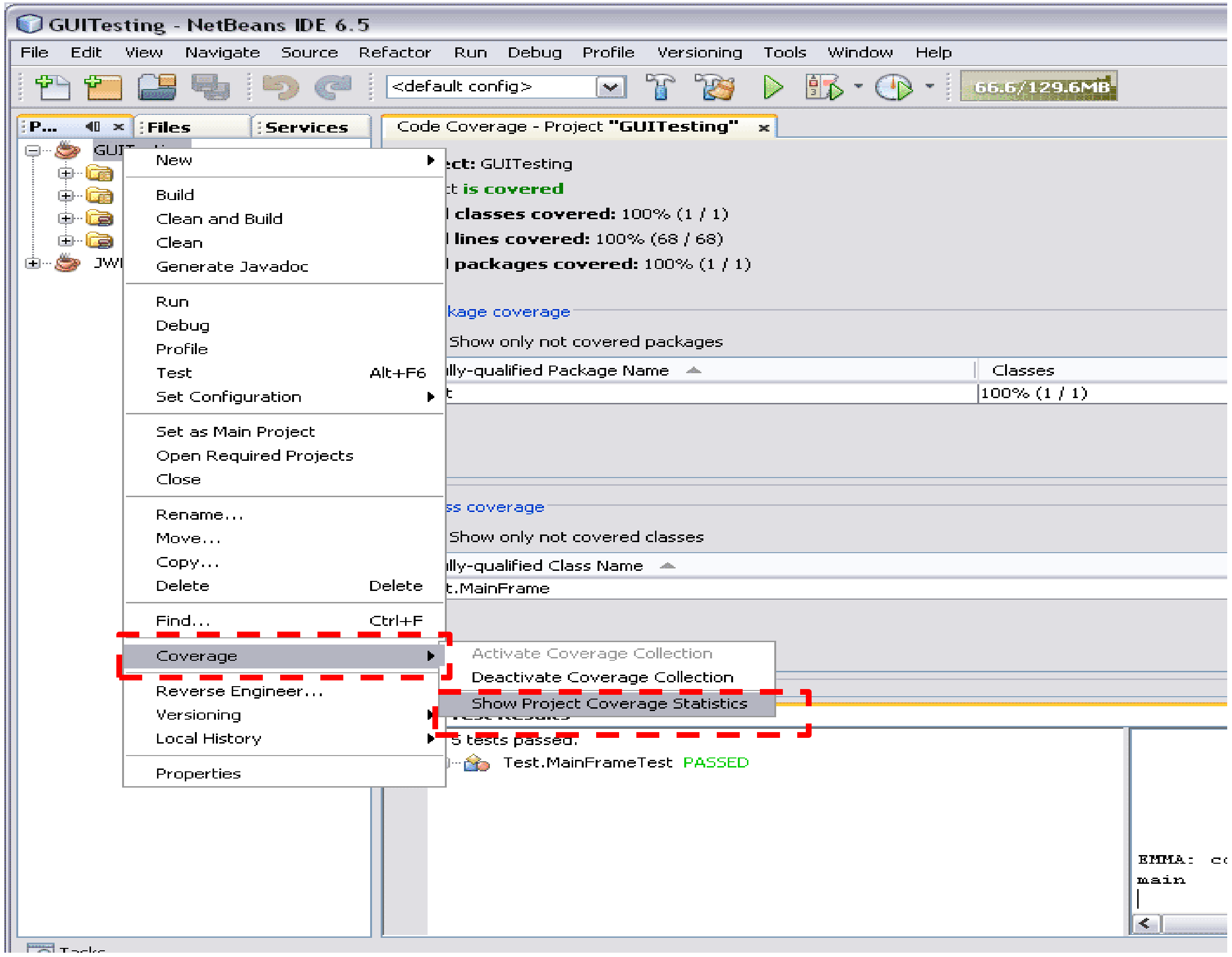
Activate Coverage Collection



Execute Your JUnit Test Code



View Coverage Reports



ct: GUITesting
t is covered
l classes covered: 100% (1 / 1)
l lines covered: 100% (68 / 68)
l packages covered: 100% (1 / 1)

kage coverage
Show only not covered packages

lly-qualified Package Name	Classes
t	100% (1 / 1)

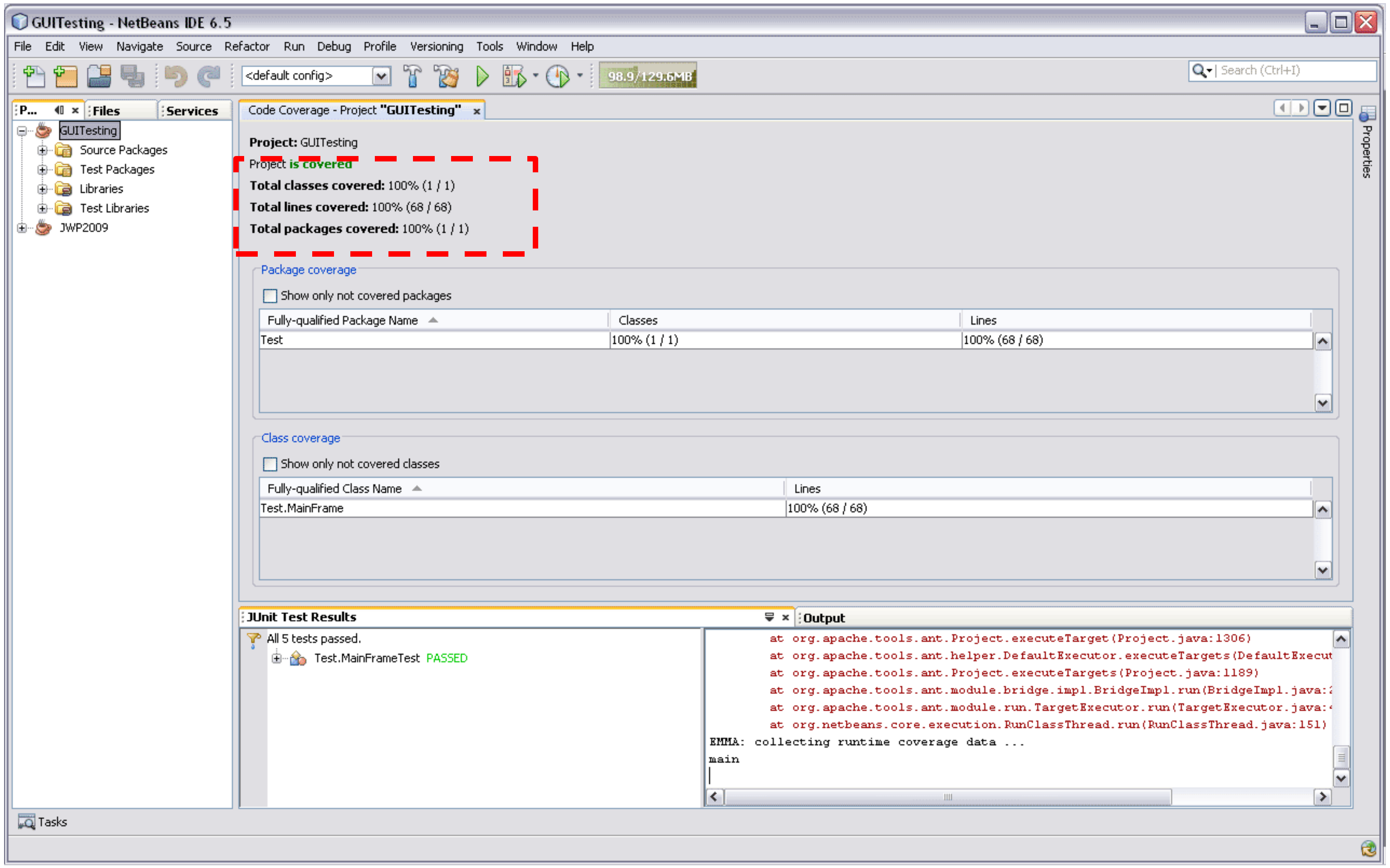
ss coverage
Show only not covered classes

lly-qualified Class Name
t.MainFrame

5 tests passed.
Test.MainFrameTest PASSED

```
EMMA: co
main
```

Coverage Report


GUITesting - NetBeans IDE 6.5
File Edit View Navigate Source Refactor Run Debug Profile Versioning Tools Window Help
<default config>
98.9/129.6MB
Search (Ctrl+I)
P... Files Services
GUITesting
Source Packages
Test Packages
Libraries
Test Libraries
JWP2009
Code Coverage - Project "GUITesting"
Properties
Tasks

Project: GUITesting
Project **is covered**
Total classes covered: 100% (1 / 1)
Total lines covered: 100% (68 / 68)
Total packages covered: 100% (1 / 1)

Package coverage

☐ Show only not covered packages

Fully-qualified Package Name	Classes	Lines
Test	100% (1 / 1)	100% (68 / 68)

Class coverage

☐ Show only not covered classes

Fully-qualified Class Name	Lines
Test.MainFrame	100% (68 / 68)

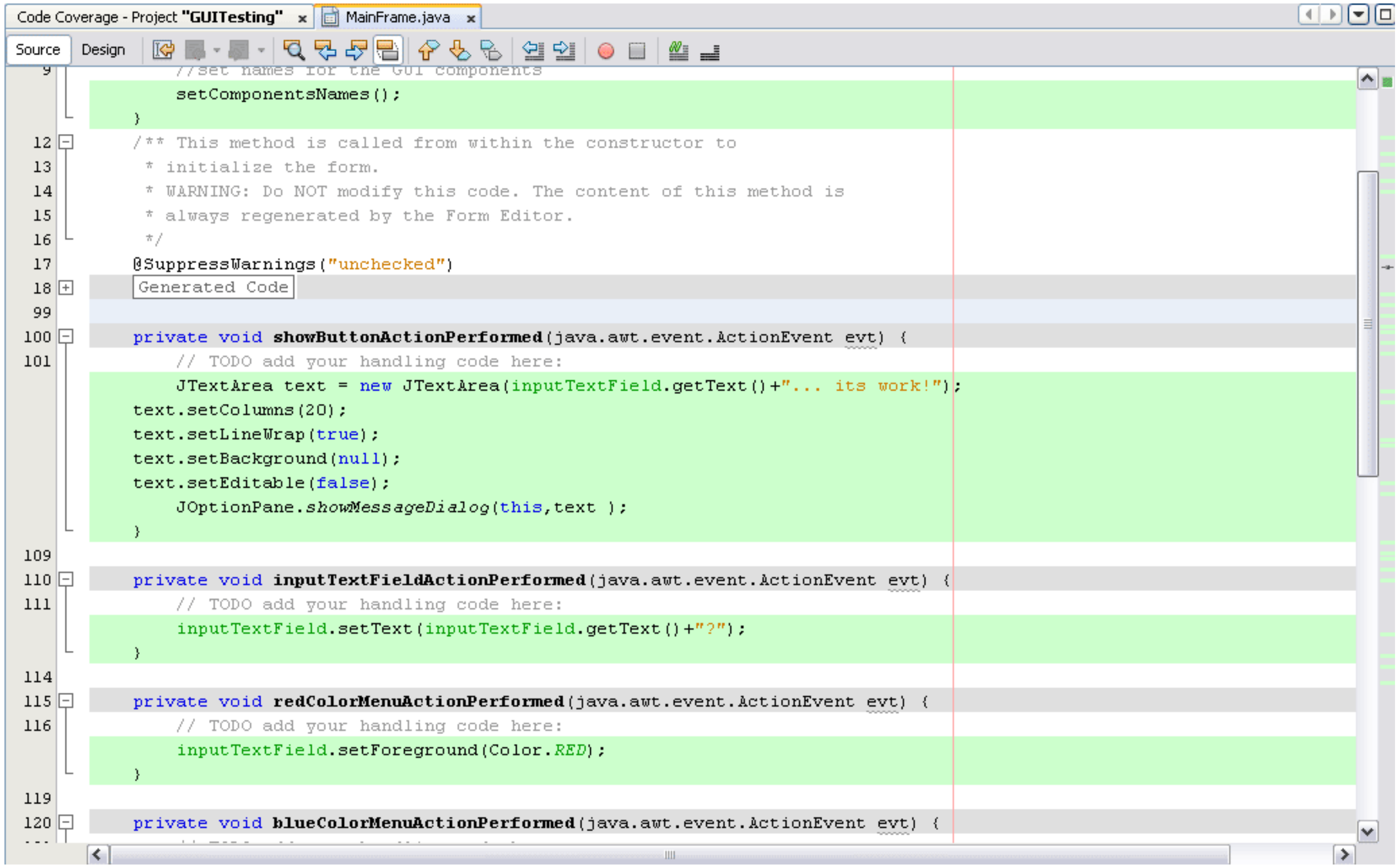
JUnit Test Results

All 5 tests passed.
Test.MainFrameTest PASSED

Output

```
        at org.apache.tools.ant.Project.executeTarget(Project.java:1306)
        at org.apache.tools.ant.helper.DefaultExecutor.executeTargets(DefaultExecut
        at org.apache.tools.ant.Project.executeTargets(Project.java:1189)
        at org.apache.tools.ant.module.bridge.impl.BridgeImpl.run(BridgeImpl.java:2
        at org.apache.tools.ant.module.run.TargetExecutor.run(TargetExecutor.java:
        at org.netbeans.core.execution.RunClassThread.run(RunClassThread.java:151)
EMMA: collecting runtime coverage data ...
main
```

Java sources coloring according to the coverage information

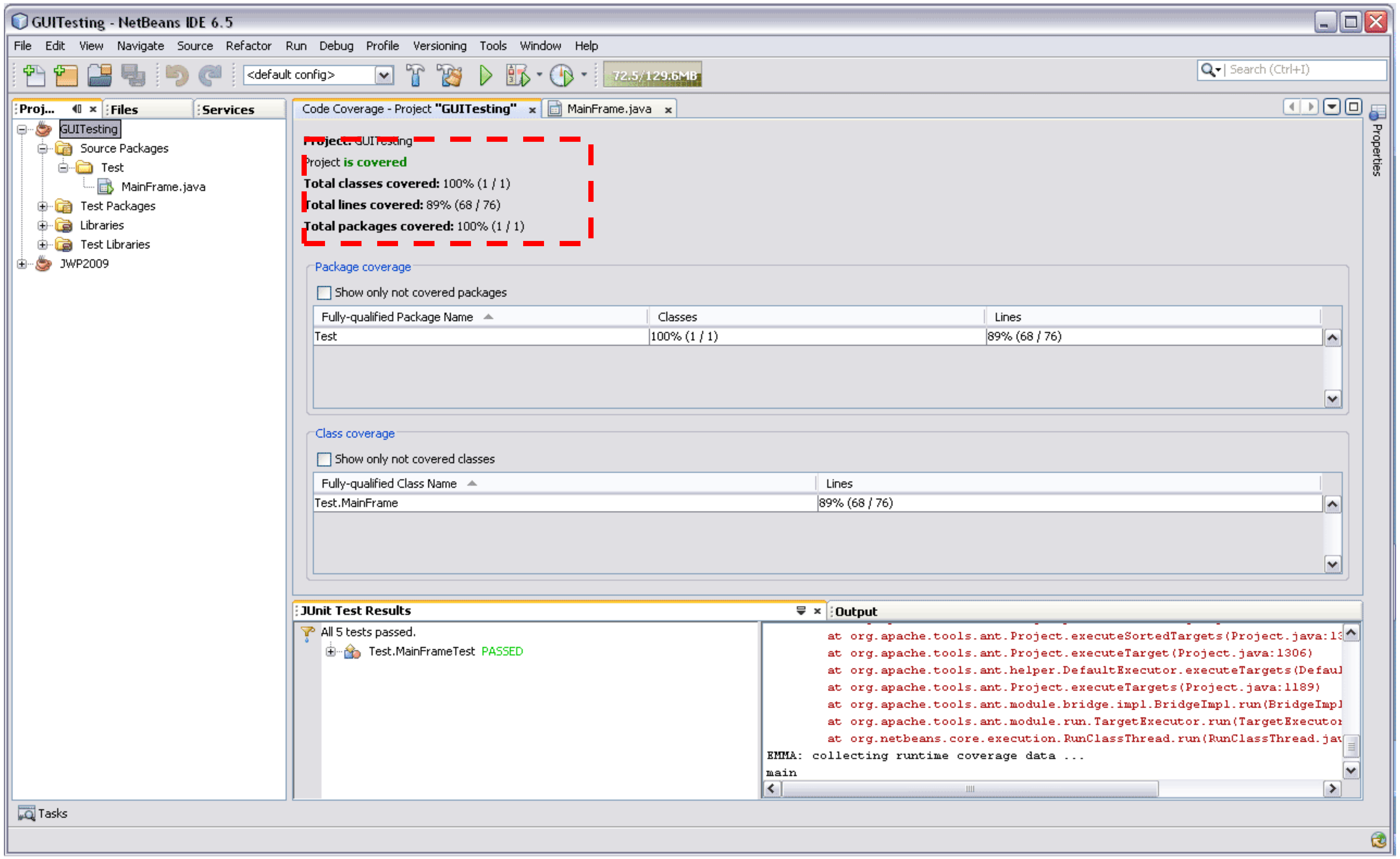


```
        setComponentsNames();
    }
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void showButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        JTextArea text = new JTextArea(inputTextField.getText()+"... its work!");
    text.setColumns(20);
    text.setLineWrap(true);
    text.setBackground(null);
    text.setEditable(false);
        JOptionPane.showMessageDialog(this,text );
    }

    private void inputTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        inputTextField.setText(inputTextField.getText()+"?");
    }

    private void redColorMenuActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        inputTextField.setForeground(Color.RED);
    }

    private void blueColorMenuActionPerformed(java.awt.event.ActionEvent evt) {
```

Improve Your JWP Coverage

- Avoid (remove) unreachable code (function, block, statement).
- Implement JUnit that cover GUI components.
- Modify JWP classes if needed to improve your test.

Code Coverage with Unreachable Code



Project: GUITesting
Project is covered
Total classes covered: 100% (1 / 1)
Total lines covered: 89% (68 / 76)
Total packages covered: 100% (1 / 1)

Package coverage

☐ Show only not covered packages

Fully-qualified Package Name	Classes	Lines
Test	100% (1 / 1)	89% (68 / 76)

Class coverage

☐ Show only not covered classes

Fully-qualified Class Name	Lines
Test.MainFrame	89% (68 / 76)

JUnit Test Results

All 5 tests passed.
Test.MainFrameTest PASSED

Output

```
        at org.apache.tools.ant.Project.executeSortedTargets(Project.java:13
        at org.apache.tools.ant.Project.executeTarget(Project.java:1306)
        at org.apache.tools.ant.helper.DefaultExecutor.executeTargets(Defaul
        at org.apache.tools.ant.Project.executeTargets(Project.java:1189)
        at org.apache.tools.ant.module.bridge.impl.BridgeImpl.run(BridgeImpl
        at org.apache.tools.ant.module.run.TargetExecutor.run(TargetExecutor
        at org.netbeans.core.execution.RunClassThread.run(RunClassThread.jav
EMMA: collecting runtime coverage data ...
main
```

Netbeans version 7.0

- Check the following plugin if you are using netbeans version 7.0
  - Unit Tests Code Coverage Plugin for NetBeans 7.0)

Next Lab

- Lab Assignment 3
- Project Part 4