INTRODUCTION

Thanks for taking time in participating in this study!

This survey aims to investigate how developers perform refactoring.

The objective audience is developers and software maintainers experienced in Java.

Refactoring is a software maintenance activity that aims to improve code design, while preserving behavior.
Anti-patterns are poor design choices to recurring design problems, and they are opposite to design patterns. Typically, they are introduced by inexperienced developers, and represent common pitfalls in software development.

Your contribution will help to develop better tools to support developers in refactoring and sorftware maintenance activities.
Your objective is to remove an instance of anti-pattern previously identified in a class from an open-source system. You could use any of the refactorings proposed by the catalog of Martin Fowler (https://refactoring.com/catalog/), and these have to **be manually applied**.

For a detailed description of anti-patterns, and refactorings, as well as online resources you can visit http://www.swat.polymtl.ca/rmorales/Antipatterns_definitions.html

Warm-up exercise

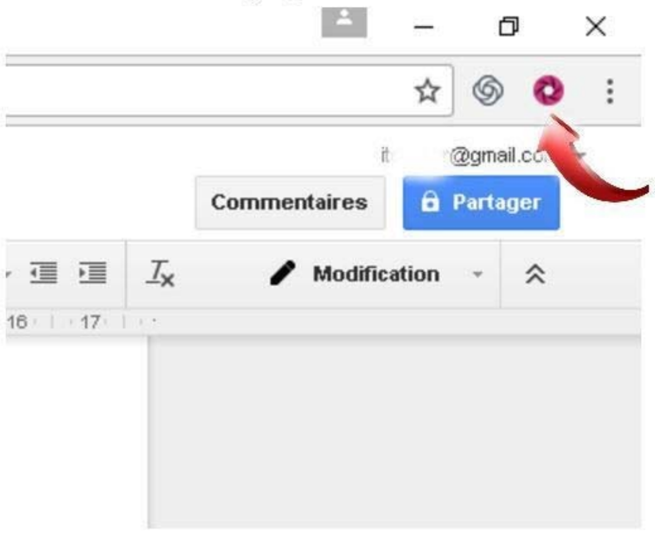
Requirements:
Java 1.8 SDK
Eclipse Neon.3 (4.6.3)
Git (command line)
Google chrome with cattura plug-in installed (download for free from https://www.catturavideo.com/)

Setup

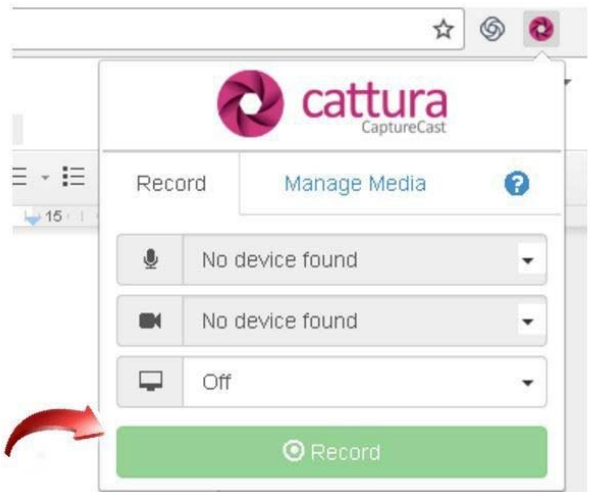
0- Video Recording:
You need to to make a screencast of your tasks by using "cattura".
Note: it is very important to record a screencast of your task, such that we can analyze your answers.

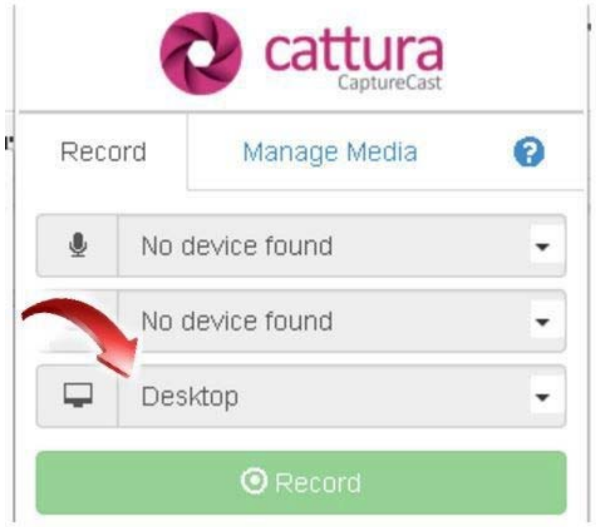
After installing cattura, please use the following steps to record a screencast:

- Open Google Chrome
- Click on the cattura icon in the top right of the Chrome window:

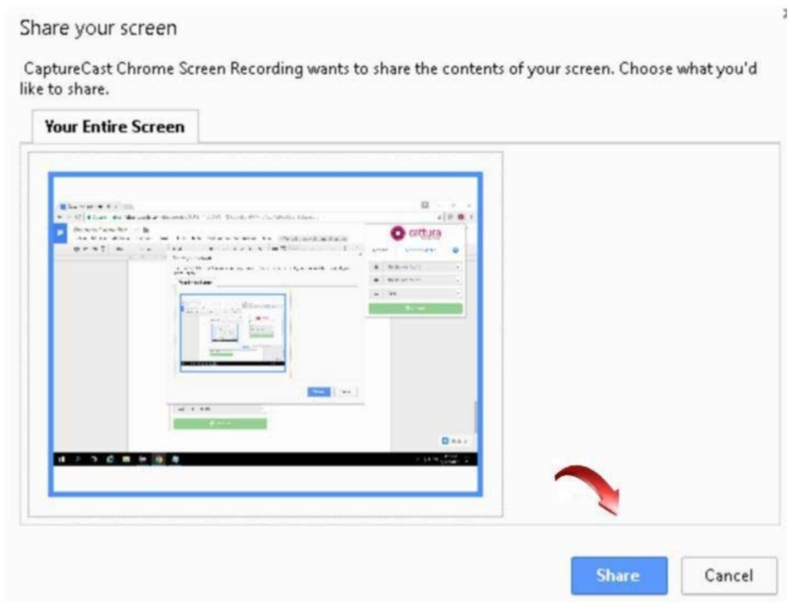


- then click on OFF to replace it by DESKTOP



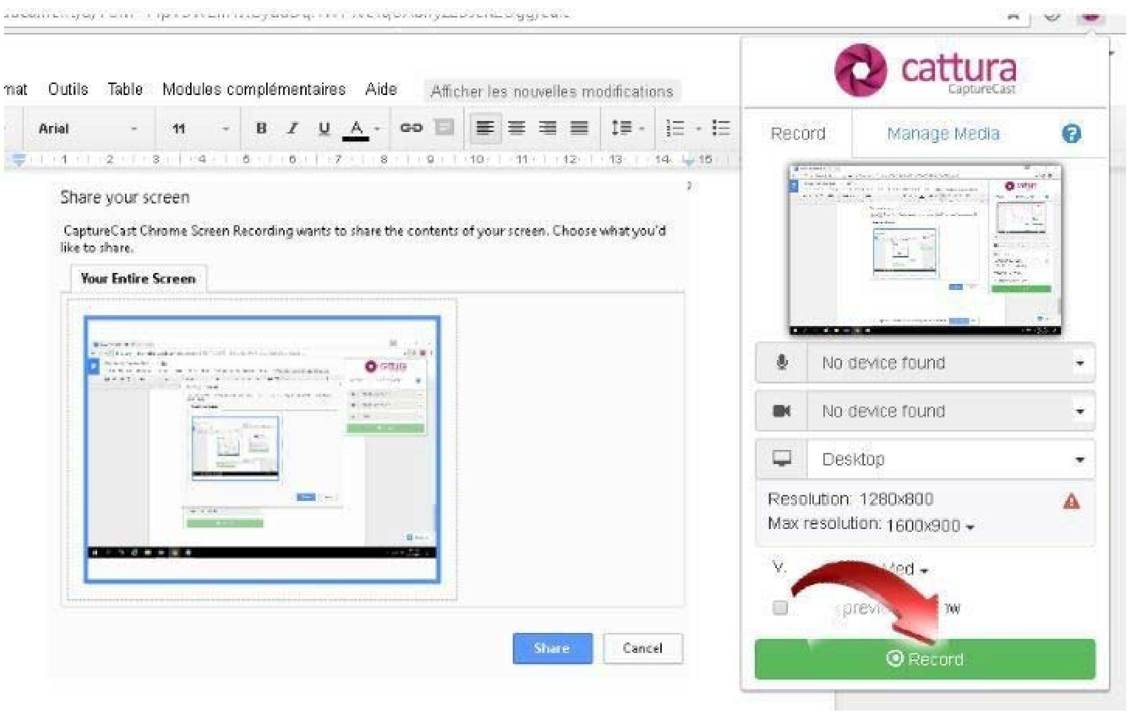


- A window will open, select “YOUR ENTIRE SCREEN” and click on “SHARE”


Share your screen

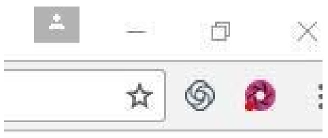
CaptureCast Chrome Screen Recording wants to share the contents of your screen. Choose what you'd like to share.

- Click record again



CaptureCast Chrome Screen Recording wants to share the contents of your screen. Choose what you'd like to share.

- A red circle will indicate that the recording is running.



Start.

System Id: 92
Checkout the following project from bitbucket

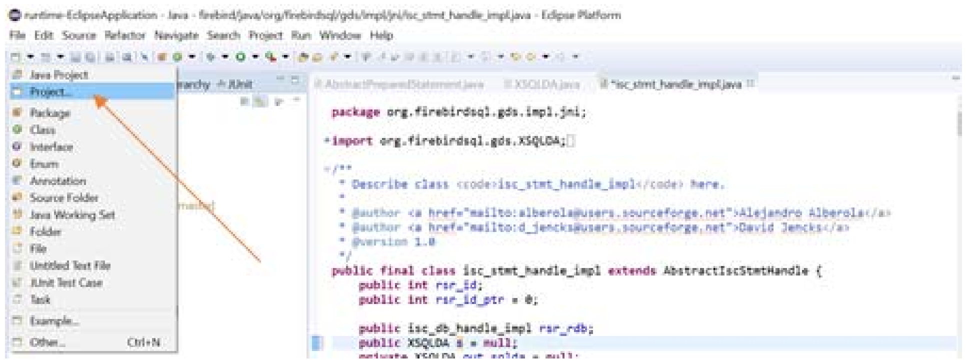
Repo: https://bitbucket.org/moar82/92.git

1. Clone the repository of the project containing the anti-pattern

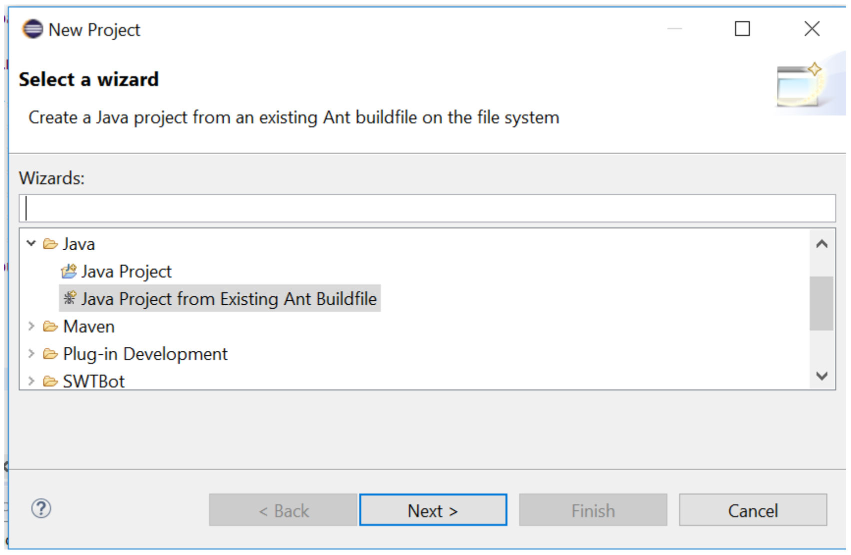
```
git clone https://bitbucket.org/moar82/92.git
```

2. Open Eclipse Neon

3. Import the project in the workspace


runtime-EclipseApplication - Java - firebird/java/org/firebirdsql/gds/impl/jni/isc_stmt_handle_impl.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Task
Example...
Other... Ctrl+N
AbstractPreparedStatement.java
XSQLDA.java
*isc_stmt_handle_impl.java

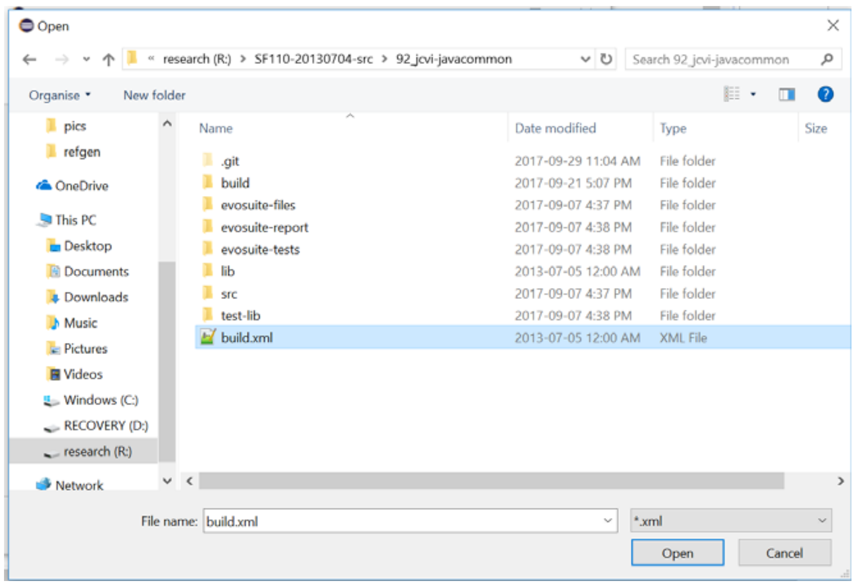
```
package org.firebirdsql.gds.impl.jni;

import org.firebirdsql.gds.XSQLDA;

/**
 * Describe class <code>isc_stmt_handle_impl</code> here.
 *
 * @author <a href="mailto:alberola@users.sourceforge.net">Alejandro Alberola</a>
 * @author <a href="mailto:d_jencks@users.sourceforge.net">David Jencks</a>
 * @version 1.0
 */
public final class isc_stmt_handle_impl extends AbstractIscStmtHandle {
    public int rsr_id;
    public int rsr_id_ptr = 0;

    public isc_db_handle_impl rsr_rdb;
    public XSQLDA s = null;
```



Select a wizard

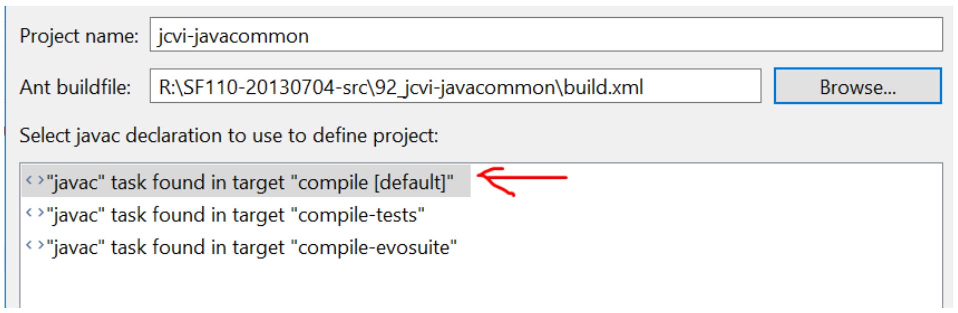
Create a Java project from an existing Ant buildfile on the file system

Wizards:



Name	Date modified	Type	Size
.git	2017-09-29 11:04 AM	File folder	
build	2017-09-21 5:07 PM	File folder	
evosuite-files	2017-09-07 4:37 PM	File folder	
evosuite-report	2017-09-07 4:38 PM	File folder	
evosuite-tests	2017-09-07 4:38 PM	File folder	
lib	2013-07-05 12:00 AM	File folder	
src	2017-09-07 4:37 PM	File folder	
test-lib	2017-09-07 4:38 PM	File folder	
build.xml	2013-07-05 12:00 AM	XML File	

File name: build.xml
*.xml



Project name: jcvi-javacommon

Ant buildfile: R:\SF110-20130704-src\92_jcvi-javacommon\build.xml

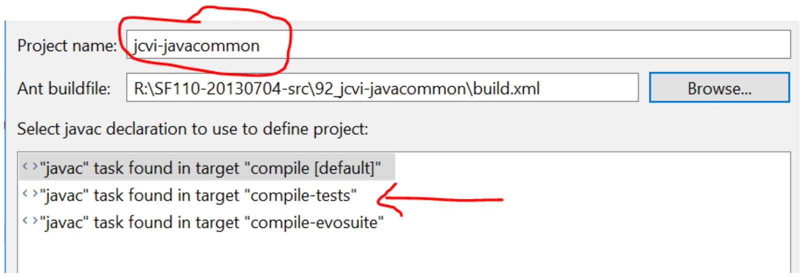
Select javac declaration to use to define project:

"javac" task found in target "compile [default]"
"javac" task found in target "compile-tests"
"javac" task found in target "compile-evosuite"

4. Import the unit tests to your Eclipse workspace

Most of the projects contains a directory with a set of unit tests developed by a developer, that you can use as a reference the correctness of your work.

To import the unit tests, you can repeat step 3, **modifying the project name (as you cannot have 2 projects with the same name!),** and selecting compile-tests in “Select javac declaration to use to define project:”



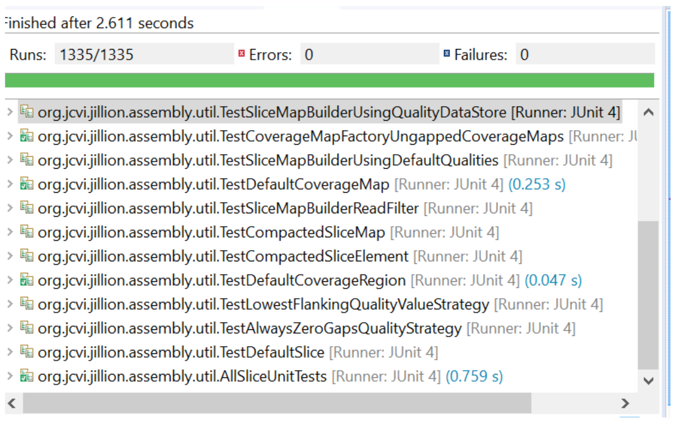
Project name: jcvi-javacommon

Ant buildfile: R:\SF110-20130704-src\92_jcvi-javacommon\build.xml

Select javac declaration to use to define project:

"javac" task found in target "compile [default]"
"javac" task found in target "compile-tests"
"javac" task found in target "compile-evosuite"

Remember to execute the tests using **Run as Junit** from the test project.



```
inished after 2.611 seconds
Runs: 1335/1335    Errors: 0    Failures: 0

> org.jcvi.jillion.assembly.util.TestSliceMapBuilderUsingQualityDataStore [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestCoverageMapFactoryUngappedCoverageMaps [Runner: Jl
> org.jcvi.jillion.assembly.util.TestSliceMapBuilderUsingDefaultQualities [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestDefaultCoverageMap [Runner: JUnit 4] (0.253 s)
> org.jcvi.jillion.assembly.util.TestSliceMapBuilderReadFilter [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestCompactedSliceMap [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestCompactedSliceElement [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestDefaultCoverageRegion [Runner: JUnit 4] (0.047 s)
> org.jcvi.jillion.assembly.util.TestLowestFlankingQualityValueStrategy [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestAlwaysZeroGapsQualityStrategy [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestDefaultSlice [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.AllSliceUnitTests [Runner: JUnit 4] (0.759 s)
```

Note that in some cases the unit tests do not pass, even before refactoring. In such a case, you need to at least validate that the same tests that pass before refactoring are the same tests that pass after refactoring.

4. Refactor the anti-pattern **manually.**

Class containing anti-pattern: **org.jcvi.jillion.internal.assembly.util.AbstractSliceMap**

Anti-pattern type: Speculative Generality (http://www.swat.polymtl.ca/rmorales/Antipatterns_definitions.html)

Read carefully the definition provided in the link above, as you need to identify which refactorings you need to apply. You can also check external web resources provided in the same address.

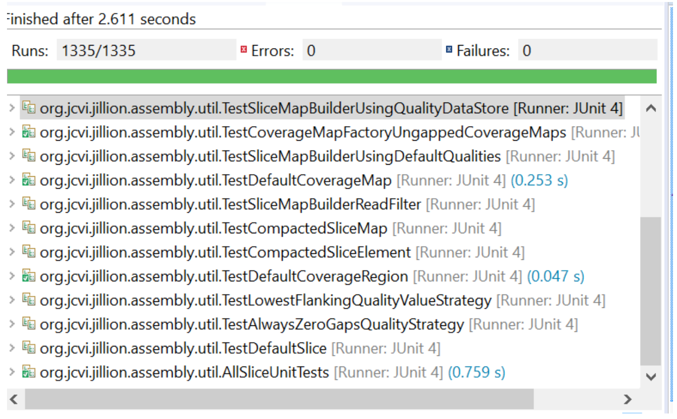
Because this is a warm-up task, we will tell you what you have to do. In the real tasks, we cannot tell you the procedure to follow as it will bias our experiments. However, you are always free to consult the link above. At the end you validate your answer by validating that the code behaviour is not altered and that the symptoms of the anti-patterns are corrected.

The procedure to correct speculative generality is **Collapse Hierarchy.**

1. So, your work consist of moving all the methods and features of class **org.jcvi.jillion.internal.assembly.util.DefaultSliceMap (children class) to class org.jcvi.jillion.internal.assembly.util.AbstractSliceMap.**
2. Update existence references from class DefaultSliceMap to AbstractSliceMap, and finally delete AbstractSliceMap.

5. It is very important that you validate that your code compiles after applying the refactoring(s) required to remove the anti-pattern.
Remember to execute the tests using **Run as Junit** from the test project. You also need to update the references in the unit tests. For this warm-up task, you need to update the reference of class **org.jcvi.jillion.assembly.util.TestDefaultSliceMap** to class AbstractSlice.

Then run the test cases to validate that the code is functional.



```
Finished after 2.611 seconds
Runs: 1335/1335    Errors: 0    Failures: 0

> org.jcvi.jillion.assembly.util.TestSliceMapBuilderUsingQualityDataStore [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestCoverageMapFactoryUngappedCoverageMaps [Runner: Jl
> org.jcvi.jillion.assembly.util.TestSliceMapBuilderUsingDefaultQualities [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestDefaultCoverageMap [Runner: JUnit 4] (0.253 s)
> org.jcvi.jillion.assembly.util.TestSliceMapBuilderReadFilter [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestCompactedSliceMap [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestCompactedSliceElement [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestDefaultCoverageRegion [Runner: JUnit 4] (0.047 s)
> org.jcvi.jillion.assembly.util.TestLowestFlankingQualityValueStrategy [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestAlwaysZeroGapsQualityStrategy [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.TestDefaultSlice [Runner: JUnit 4]
> org.jcvi.jillion.assembly.util.AllSliceUnitTests [Runner: JUnit 4] (0.759 s)
```

Note that in some cases the unit tests do not pass, even before refactoring. In such a case, you need to at least validate that the same tests that pass before refactoring are the same tests that pass after refactoring.

Once you are satisfied with your code, you can check the classes that you modified from the command line, in the folder where you clone the repo, by typing:
>git status

NOTE: **if you need to clean the repo, to discard your current changes**, you can use the following command:

```
git fetch --all
git reset --hard
```

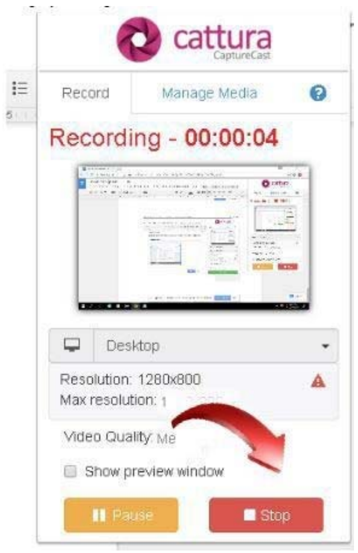
6. Add all the changes related to java classes that you modify after refactored the anti-pattern
>git add *.java

7. Generate the patch with your changes
`git diff --cached >` *RR_XX_Y_SourceClassName*`.patch`

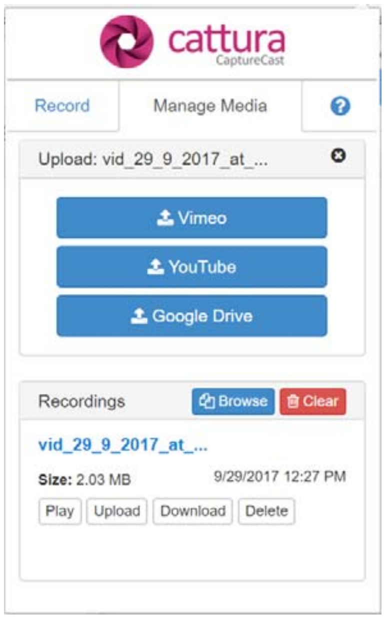
where RR is an abbreviation of the type of refactoring applied (see Anti-pattern defintions table, column 3 at http://www.swat.polymtl.ca/rmorales/Antipatterns_definitions.html)
XX is the name of the repo
and Y is always 1.

For this warm-up task you will save it as:
ch_92_1_AbstractSliceMap.patch

8. At the end of Task 1 (as well as the real exercise), we ask you to stop the recording by clicking on "STOP"



9. Download the video file **with the same name o**f the patch that you generated.:



10. Sent me an email with the links for each individual file generated in the task to rodrigomorales2@acm.org, that includes:
Patch file, video of task

As the video probably will be large, you can use https://testpilot.firefox.com/experiments/send .