## View/Export/Delete Multiple Variables

* \plugins\dataviewer\gui\datalist\**DataListTable.java**
  + Multiple select
  + Deselect when mouse click elsewhere
* \plugins\dataviewer\gui\datalist\**DataListModel.java**
  + Get data of all selected rows
* **DeleteDataMenuItem.java** & **ViewDataMenuItem.java**
  + Perform action on the list of Data
* Export multiple files at once??

Keyboard navigate buttons in optionPane 🡪 use Tab

* Tried getting the buttons from OptionPane, but couldn’t
* With buttons and their indices, I can navigate through
* \test\**ButtonPane.java**
* \plugin\dataviewer\gui\**NavigateButtons.java, DataUtils.java**

[DONE] ToolTip in Table Cell

* plugins.dataviewer.gui.datalist\**DataListPanel.java**
* <https://stackoverflow.com/questions/9467093/how-to-add-a-tooltip-to-a-cell-in-a-jtable>
* Which is a better implementation? (e.g. cleaner coding style)
* How does the second method actually work?!

[DONE] Default name for imported variable

* plugins\jprobe\gui\filemenu\**ImportMenu.java**
* plugins\jprobe\gui\**ExportImportUtil.java** 
  + String fileName = f.getName();
* jprobe\**CoreDataManager.java**

Default name for generated variable

Default name for to-be-export file

* plugins\jprobe\gui\**ExportImportUtil.java**
* always .txt extension?

[DONE] Make col width in ViewTab adjustable

* \plugins\dataviewer\gui\datalist\**DataListTable.java**

[DONE] Close tab with Ctrl+W

* DataViewerSplitPane, DataTabPane, DataTabLabel, DataTab
* Read tutorials carefully!

1. Remove errors and warnings

* Replaced old libraries with updated ones
* Unresolved: illegal reflective access warnings

|  |  |
| --- | --- |
| **Asset to Equal** | **Download Junit and add to Eclipse** |
| **import** com.sun.org.apache.xerces.internal.parsers.DOMParser; | **Download xerces (a library that was removed from the standard library of Java)**  [**http://xerces.apache.org/mirrors.cgi**](http://xerces.apache.org/mirrors.cgi)  **Replace import statement:**  [**https://xerces.apache.org/xerces2-j/javadocs/xerces2/org/apache/xerces/parsers/DOMParser.html**](https://xerces.apache.org/xerces2-j/javadocs/xerces2/org/apache/xerces/parsers/DOMParser.html) |
| **Errors in “old” directory** | **Deleted old** |

1. Building the project

* Wrote new DistributableBuilder.xml file
* Move junit.jar and xerceslmpl.jar to lib folder
* Installed Apache Ant

~~C:\Users\th184\Documents\JProbe\JProbe\distributable~~

~~Run command: ant -f ../builders/DistributableBuilder.xml makejar~~

8/14 (Tue): Tristan

* Was building the wrong file this whole time
  + Modified build.xml files: added ${basedir} to find MANIFEST file
  + Learned to use *ls –la* to check whether .jar files were built successfully
  + *Build All* in Eclipse: added library .jar file to eclipse’s build path (did it before, but didn’t work that time)

8/15 (Wed): Vincentius

* .jar files were not updated; realized that the built file was not put into *\distributable*, which is where jprobe.jar is located 🡪 modified path in build.xml
* suspected that \*distributable* was created when he was done with the entire project (confirmed by Tristan afterwards)
* Tristan helped me understand plugins
* What I learned:

1. Backend skeleton of the software
2. build.xml 🡪 MANIFEST.MF 🡪 Activator.java

8/16 (Thur)

* Stuck on one of the plugins: .jar was not built, didn’t know why; suddenly worked after going home… (sigh)

Summary to Build and Run:

* Build All in Eclipse
* Pwd: \JProbe\JProbe
* Ant build: ant -f builders\[relevant build file]
* java –jar distributable\jprobe.jar –g
* Note: to ensure that modifications made in text files (e.g. functions.txt) are reflected, discard all build results before rebuilding to clear the buffer.
  + Eclipse: Project 🡪 Clean

1. Full screen on the first opening of the application

* \plugins\jprobe\gui\**Constants.java**
* Subsequent openings: user’s previous state; **GUIConfig.java**
* What I learned: configuration file, saved only on Close (not when being killed by Ctrl+C)

1. Reasonable screen size and position when users [un-maximize](https://stackoverflow.com/questions/13912692/can-i-set-jframes-normal-size-while-it-is-maximized) the screen

* **GUIActivator.java**

1. Tooltip

* \plugins\functions\gui\**FunctionMenuItem.java**, added **JMultiLineToolTip.java**
* Modifications: multiline, padding around text, font, stay as long as mouse is over it

1. Changed displayed function names on the menu

* Removed “mutation profiler”

1. Tab

* Tab focus: \plugins\dataviewer\gui\**DataTabPane.java**
* Traversal keys: **DataViewerSplitPane.java**
* Goal: move tabs around, close icon (only on current tab), rounded corner
* <https://github.com/bobbylight/FifeCommon/blob/master/src/main/java/org/fife/ui/TabbedPaneTransferHandler.java>
* Tab listener: <http://forums.devshed.com/java-help-9/jtabbedpane-determine-currently-selected-tab-panel-399212.html>
* <http://www.java2s.com/Tutorial/Java/0240__Swing/ListeningforSelectedTabChanges.htm>

1. Function ArgsPanel

* **package** plugins.functions.gui.dialog: ArgsPanel
* chiptools\**Resources.java**: how args files are read in

1. Function input/output data tabs

* Sorting: enable users to sort by Type or Name
  + Src\plugins\dataviewer\gui\datalist\**DataListTable.java**

1. Metadata

* Plugins\functions\gui\dialog\**ArgumentsPanel.java**

1. Accelerator align to the right:

* Plugins\jprobe\gui\filemenu\**formatMI.java**
* <https://java-swing-tips.blogspot.com/2014/10/jmenuitem-accelerator-text-alignment.html>

1. Import file: default directory should be in current folder
2. Imported file: column width should automatically fit the name
3. Check and X icon
4. **validLabel**
5. \src\plugins\functions\gui\dialog\ArgumentPanel.java
   1. getCheckIcon()
   2. JCheckBox()
6. X: means mandatory but not defined (indicated by asterisk)
   1. Put red border around entry box
7. Output: plot E-score profiles, PWM profiles
8. JFreeChart: <http://www.jfree.org/jfreechart/>
9. Show all text files when loading peaks (not defined by extension)

* Help is missing
* Mandated fields: probes (give hint if not defined), number of binding sites (min= 2, max=4)
* Spell out words in all UI elements
* Put boundaries on elements (ex: number of binding sites cannot be less to 2)
* Min\_distance apart: can be 0, let default be 2
* Max distance: must be >= min distance, default = 16

Probe\_generator

1. Specify PWM size and provide a note on size restriction
   1. “Error: PWM length must be <= (binding site +2\*window)

Agilent Formatter is missing [later]

1. Ask users for: number of replicates, direction

When does jprobeActivator.java get called?

Questions

* Negative Control Generator:
  + IncludePeak 🡪 included region?
  + ExcludePeak 🡪 excluded region?
  + Summit 🡪 threshold?
  + Escore 🡪 E-score? E-value? E-Value? K-mer/K-Mer?
* Peak Filter
  + IncludeChroms 🡪
  + ExcludeChroms 🡪
  + QValue, PValue
* Probe Filter
  + Reads a file containing line separated sequences and keeps only probes containing an OR over the sequences.

**Input Dialog Design**

Original Design

1. Do you have what you need for the mandatory fields? V vs X

* Optional fields default to V

1. Do you intend to fill out the optional fields? Check Box

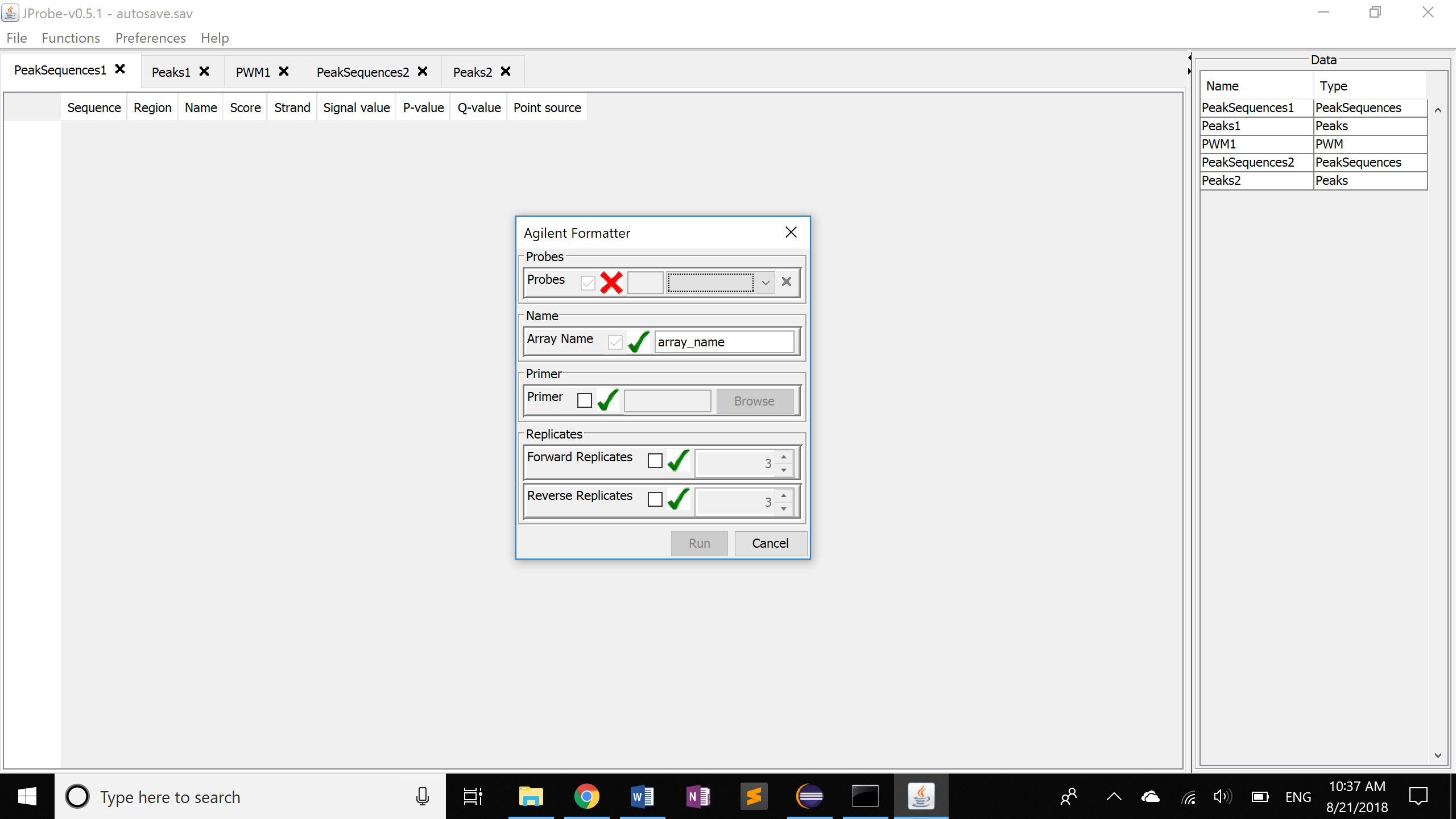
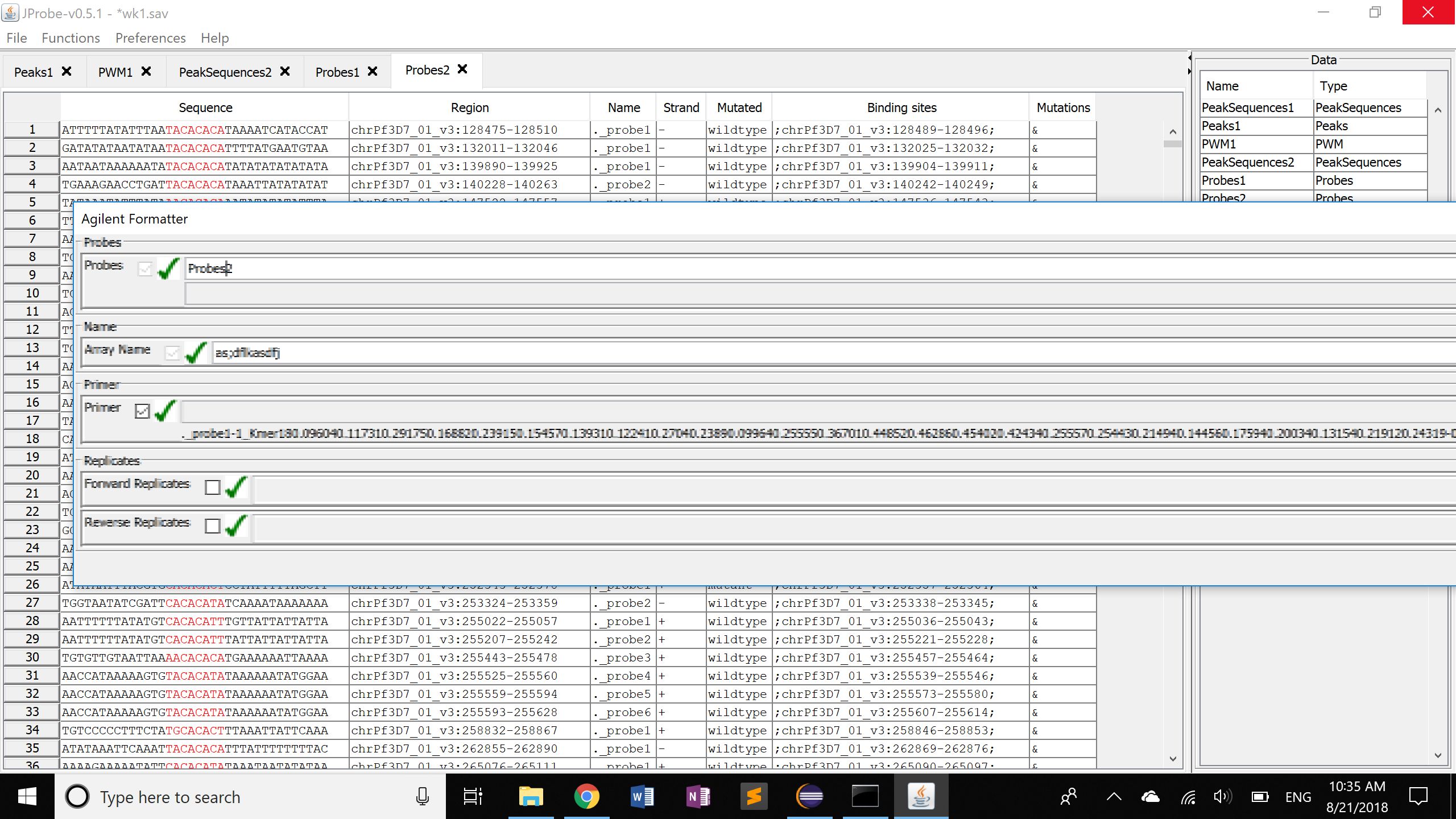
* If checked:
  + grayed field turns white/editable
  + If file not imported: V turns to X

1. Is the input valid?

* Out-of-range argument: X

Analysis:

* **Check box** was used to tell users whether a field is optional or not. But it is unnecessary.
  + Un-checkable boxes in already filled fields 🡪 frustration, irritation
  + Optional field: check the box before I can input file or change the default value 🡪 extra/unnecessary step
* Solution: indicate optionality using a “hint” that does not obstruct user’s work (e.g. does not force users to perform extra steps)
* File **import inconsistency**: Import and Browse (to distinguish optionality)
* **Red X** stresses users out… big fat X generally makes people uneasy. It is used to tell users that actions are needed:
  + Mandatory files have not been imported
  + Optional file needs to be imported (Ex: Probe Filter)

* Wrong primer file 🡪 froze; reopening the dialog clear the field; there’s no way for the users to get out other than restarting the program
* Solution: check file format before importing

Modified Design

Goals:

* indicate (1) whether a field is optional and (2) whether a needed file has been imported
* be able to import multiple files at once
* be able to close import windows, dialog windows with Ctrl+W

Ideas:

* optional fields: use a “hint” on the side. Ex: (Optional) in gray

8/21/18 (Tue): Discuss Dialog GUI with Dr. Gordan

* get rid of V, X, and check box
* allow to [import] files while [function] dialog is open
* indicate mandatory fields with \*
* default values show up in black
* if user input is out of range, box the field in red
  + validation happens right after the input to the field is completed
* alignment:
  + field name and input field next to each other; left-aligned and right-aligned respectively
  + categorize input fields into text and number field,
* panel on the right-hand side: input and output tab 🡪 allow to sort file by type
  + Create metadata for each file, both input and output; access metadata through
    - Right click: Get Info
    - Export: recursive print
  + right click: Get Info
  + for each file

# GUI Design Resources

Form Design: <https://uxplanet.org/10-rules-for-efficient-form-design-e13dc1fb0e03>

Label Placement: <https://www.uxmatters.com/mt/archives/2006/07/label-placement-in-forms.php>

What is a Bundle?

In OSGi, software is distributed in the form of a *bundle*. A bundle consists of Java classes and other resources that deliver functions to device owners, as well as providing services and packages to other bundles.

<https://www.javaworld.com/article/2077837/application-development/java-se-hello-osgi-part-1-bundles-for-beginners.html?page=2>

By default, none of the classes in a bundle are visible from any other bundle; inside bundles they follow the normal rules of the Java language. So, what do you do if you want to access the classes of one bundle from another bundle? The solution is to export packages from the source bundle and then import them into the target bundle.

Where is the entry point in Activator.java?

*Ask it to look for MANIFEST and bin relative to where the project is by adding “basedir”*

Ant –f builders\FunctionGUIBuild.xml makejar

ls –la builders\FunctionGUIBuild.xml

*Check timestamp after modifying FunctionGUIBuild.xml*

ls –la plugins

*Check the effect in plugins*

java –jar distributable\jprobe.jar –g

ls \Users\th184\jprobe\plugins

ls \Users\th184\jprobe\logs

Cat \Users\th184\jprobe\logs\jprobe.log

Ls –la ..\bin\plugins\functions\gui

Basedir: directory where the build file is

.java, .class, .jar and binary files --> can you help me understand they relationships?

.java is a source file. That's the code you write. .class is the compiled code. That's the output when you compile with javac. .jar files contains packaged code, usually for an app or a library. Jar files are actually ZIP files, and contain .class files and other files used for the app or library.

[Tristan]

each plugin has it's own builder that needs to be run if you modify the code of that plugin

a plugin is a group of code that can be loaded independently

without rebuilding any other plugins or the main executable

activator is the entry point for a plugin

main is only run by the main exectuable

that central system uses the activator of the plugin

to load the plugin

Full Workflow

DistributableBuilder.xml --> Launcher.java, where main() is called --> creates jprobe.jar

Build all the other plugins

Run jprobe.jar

if you want to rebuild the entire system yes

but the point of the organization into plugins

is that you don't need to do that

in build (not builders), there's a dir called "classes"; does it contain all the .class files? Like "\bin" contains all the binary files?

but the files in \bin are also .class files....

i'm not sure what the build directory is

my guess is that's something made by eclipse

i'm not sure how it relates to the bin directory

The whole library is called FifeCommon

That's just one class inside the library

Typically you can't just yank out code from a library bc it might depend on other classes in the library

So you download the whole library (which is packaged in a jar)

After you add the jar to your project, you can then access the target class in your code

Git

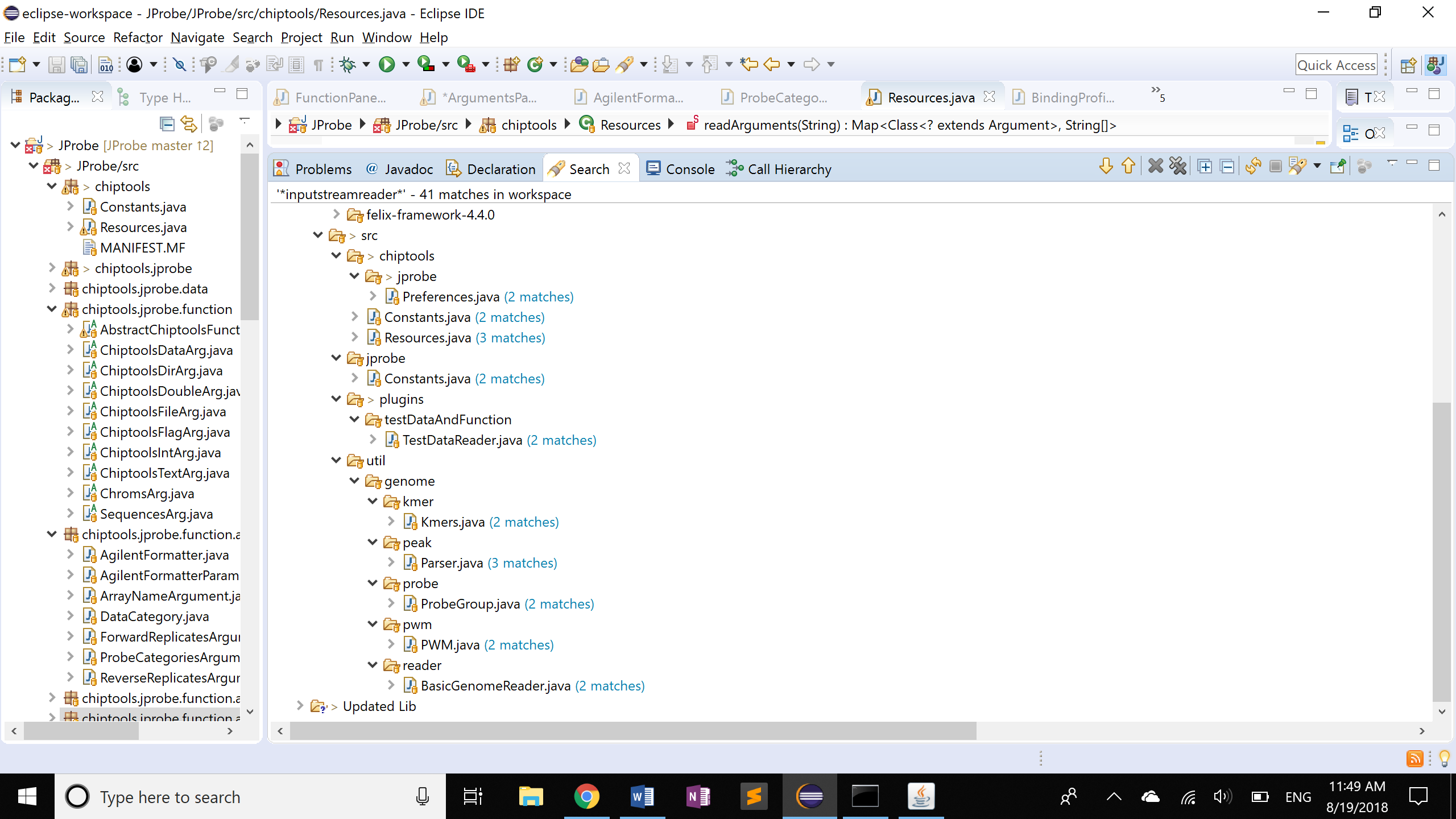
You use "git status" to see what files you've modified and what files aren't tracked by git

You can use "git diff" to view your changes

"git add FILE" will prepare a file for committing (moves file to what's known as the staging area)

You can use git add -u to add all changed files

And then "git commit" will commit your files. Be sure to type a useful message, in the editor that pops up!



# Code Structure

## Reformat Argument Dialog

FunctionMenuItem 🡪 FunctionPanel 🡪 ArgumentsPanel

* FunctionMenuItem
  + doFunction()
  + display(FunctionPanel)
* FunctionPanel
  + new ArgumentsPanel(function)
* ArgumentsPanel
  + Function.getArguments()

**package** chiptools.jprobe.function.agilentformatter;

* Agilentformatter: Add arguments

args.add(**new** ProbeCategoriesArgument(**this**.getClass(), **false**));

**package** chiptools.jprobe.function.args;

* + Arguments common to multiple FUNCTIONS
* ProCategoriesArgument:

Resources.*getArgument…*(funcClass, argClass)

**package** chiptools.jprobe.function;

* Abstract classes

**package** chiptools;

* Read functions.txt and argsFile.txt

**package** plugins.functions.gui.dialog;

**Outside to Inside**: FunctionDialogHandler 🡪 ArgumentsPanel 🡪 ArgumentPanel

ArgumentsPanel.java

layoutColumns(panels, targetAspect)

* getBestLayout(panels, targetAspect): fit categories into cols
  + input: list of categories
  + output: list of categories groups by columns on the dialog window
* for each col, create a JPanel, add the categories of the col
  + call categoryPanelConstraints() to layout the categories/rows within the col
* layout the cols of ArgumentsPanel

## Separate Function Input and Output

**package** plugins.functions.gui.dialog;

FunctionPanel.java

* create arguments panel, add Run and Cancel button with ActionListener
* actionPerformed(ActionEvent e): m\_ArgsPanel.run(m\_Core, m\_Bundle);

ArgumentsPanel.java

* run(core, bundle)
  + executor = new SwingFunctionExecutor()
  + executor.execute()

**package** plugins.functions.gui;

SwingFunctionExecutor.java

* execute(): mThread.start()
* run():
  + **final** Data d = m\_Function.execute(**this**, m\_Params);
  + done(d);
* done()
  + SwingUtilities.*invokeLater*(**new** Runnable(){

**public** **void** run() {m\_DataManager.addData(d, m\_Bundle);}

**package** plugins.jprobe.gui;

ExportImportUtil.java

* run(): core.getDataManager().addData(in, b);

**package** jprobe;

CoreDataManager.java

* **addData**: add data to a hashmap

How the Data strucutre is created:

**package** chiptools.jprobe.function.agilentformatter;

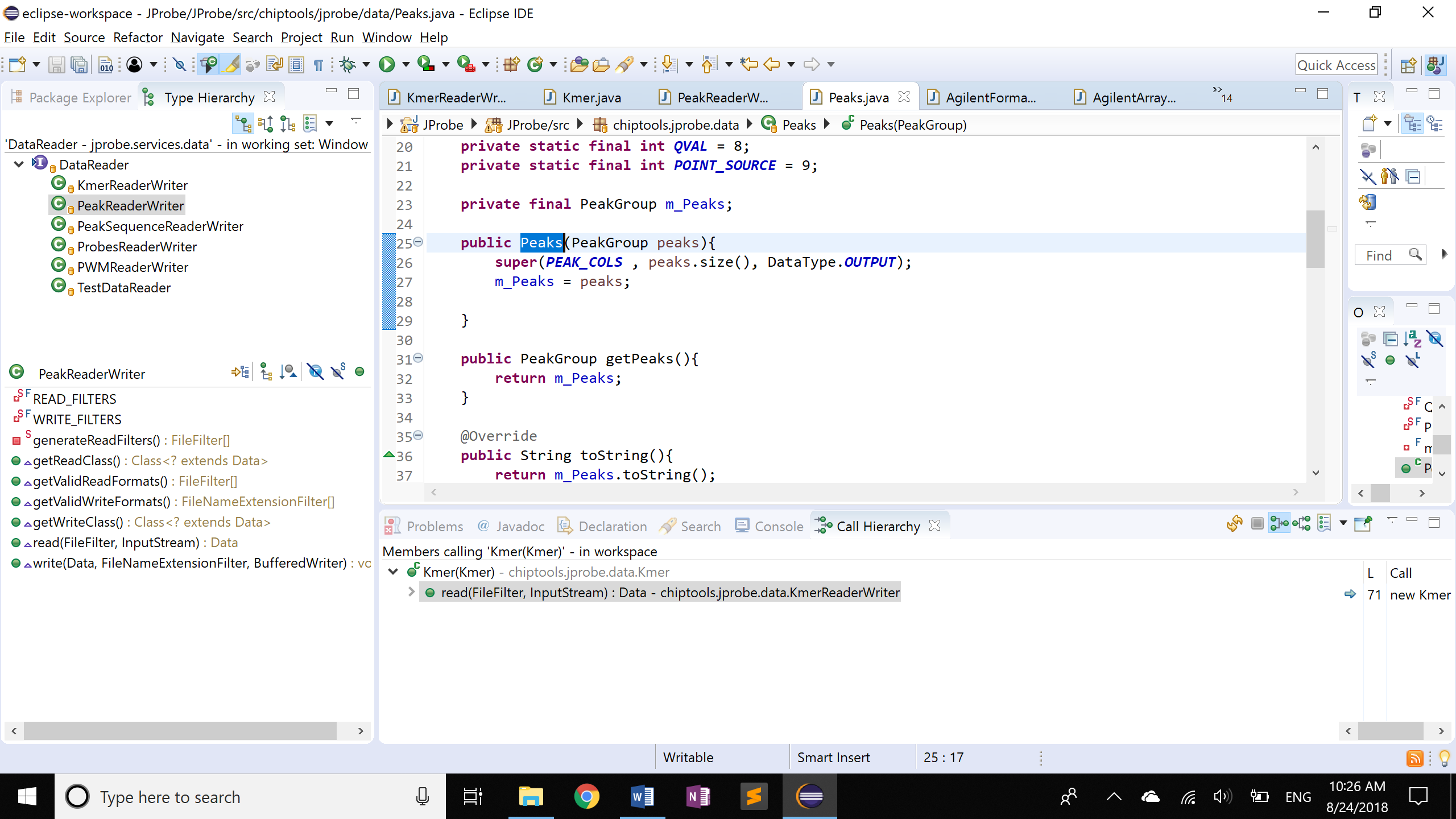
* execute(): **return** **new** AgilentArray(params.ARRAY\_NAME, agilentProbes);

**package** chiptools.jprobe.data;

* **class** XXX **extends** AbstractFinalData:

**super**(***NUM\_COLS***, probes.size(), DataType.***OUTPUT***);

* Type Hierarchy of AbstractFinalData 🡪 modify the third arg of the child classes
  + GenericTable.java 🡪 not sure what this is for; temporarily plug in DataType.OUTPUT for the third argument



ExportImportUtil.java

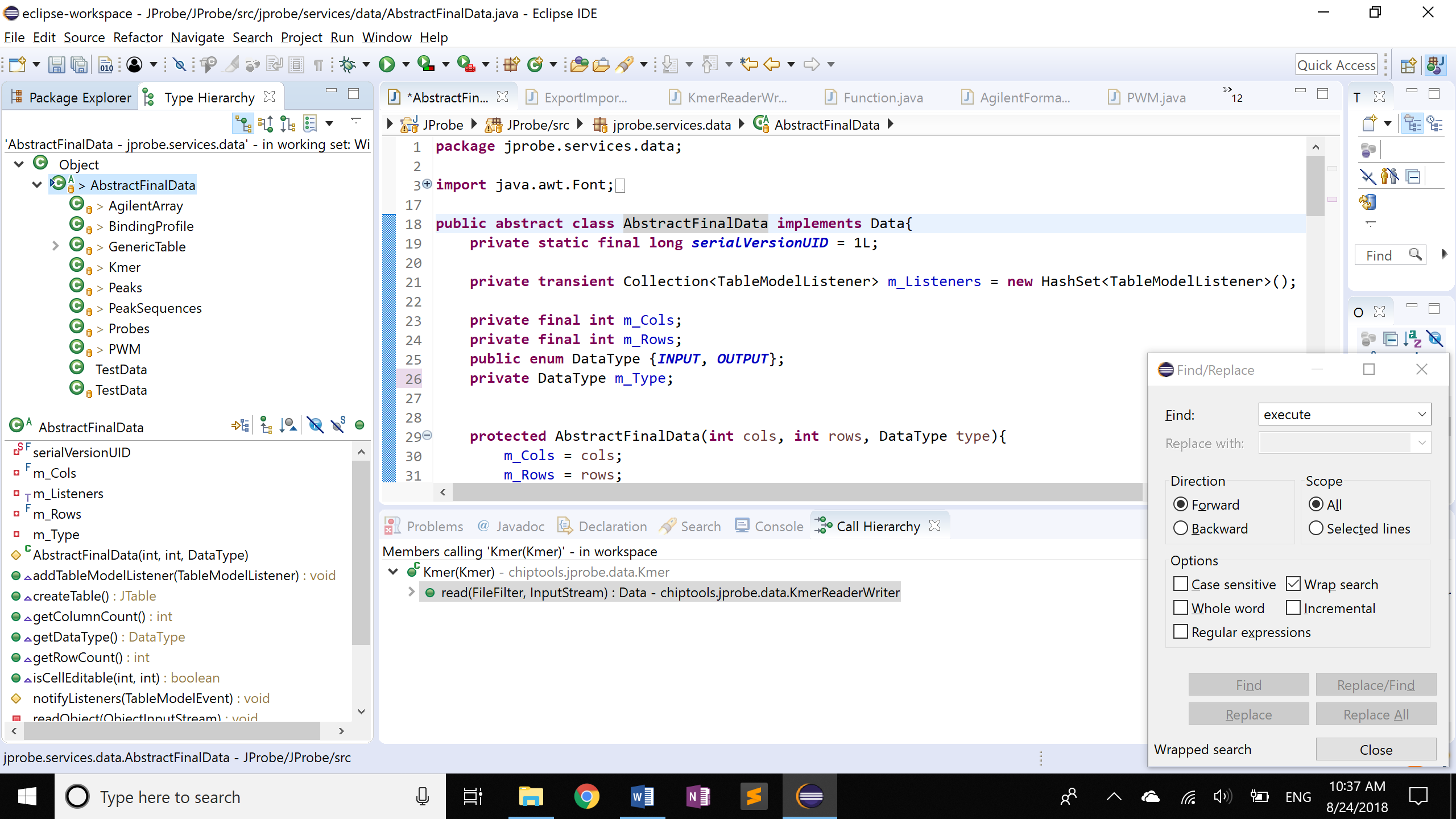
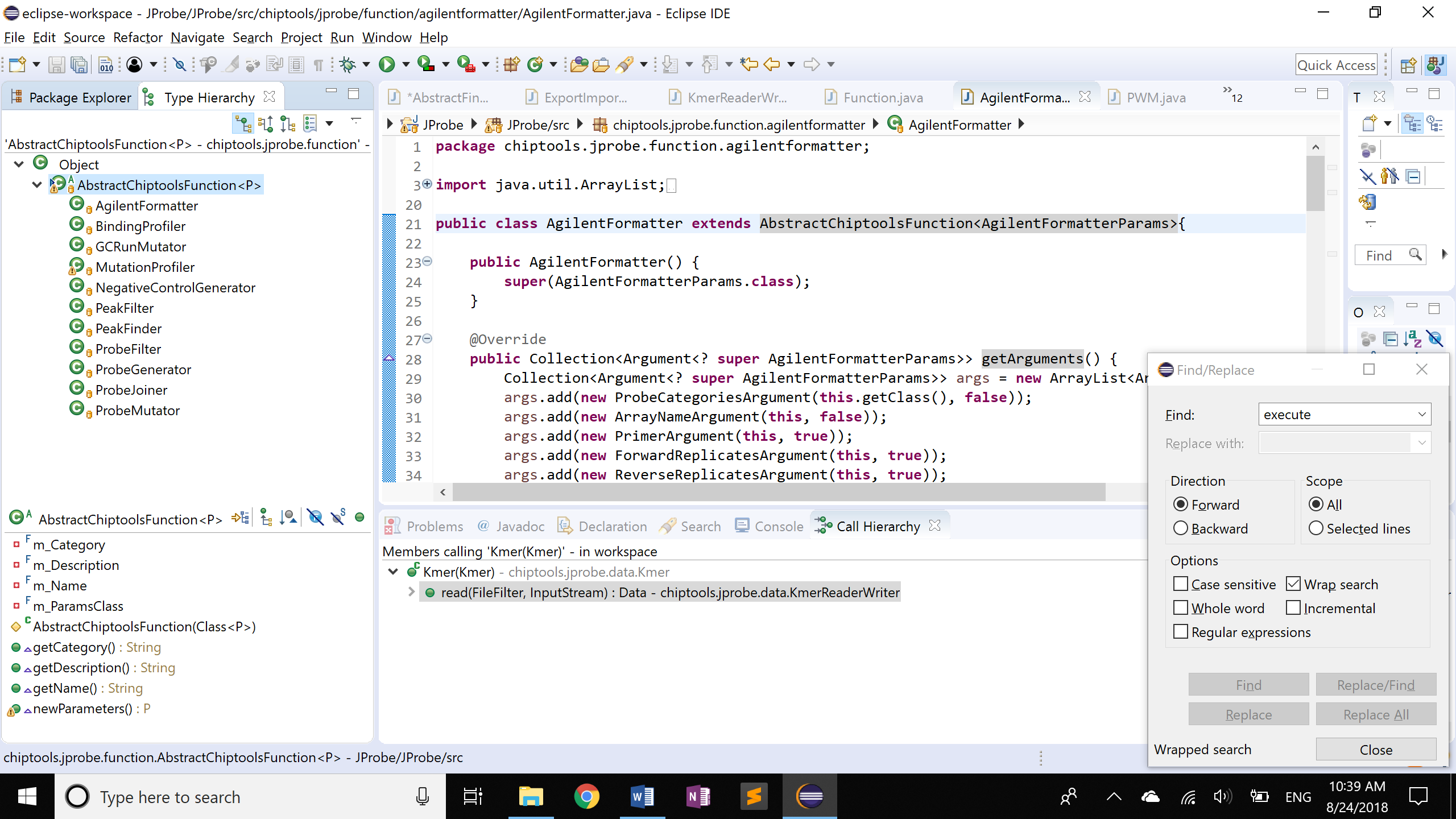
* importData():
  + Data in = reader.read(format, stream);
  + core.getDataManager().addData(in, b);

SwingFunctionExecutor.java

* run()
  + Data d = m\_Function.execute(**this**, m\_Params);

|  |  |
| --- | --- |
| **INPUT** | **Data read(){return new…}** |
| KmerReaderWriter 🡪 in \data | Kmer() 🡪 Kmer.java in \data |
| PeakReaderWriter | Peaks() |
| PeakSequenceReaderWriter | PeakSequences() |
| ProbesReaderWriter | Probes() |
| PWMReaderWriter | PWM() |
| **OUTPUT** | **Data execute(){return new…}** |
| AgilentFormatter | AgilentArray()🡪 AgilentArray.java in \data |
| BindingProfiler | BindingProfile |
| PeakFilter | Peaks |
| PeakFinder | PeakSequences |
| GCRunMutator, NegativeControlGenerator, ProbeFilter, ProbeGenerator, ProbeJoiner, ProbeMutator | Probes |
| Specify DataType | add (DataType type) as a constructor arg |

* Light blue: need to specify INPUT or OUTPUT in those constructors



Modifications

* created plugins\dataviewer\gui\**ViewTabPane.java**
* jprobe\services\data\**AbstractFinalData.java:** created field m\_Type
* **ExportImportUtil.java** and **SwingFunctionExecutor.java**: set DataType

## Parameter Object Creation

**package** chiptools.jprobe.function.bindingprofiler;

**public** **class** BindingProfiler **extends** AbstractChiptoolsFunction<BindingProfileParams>{

1. **public** BindingProfiler() {**super**(BindingProfileParams.**class**);}

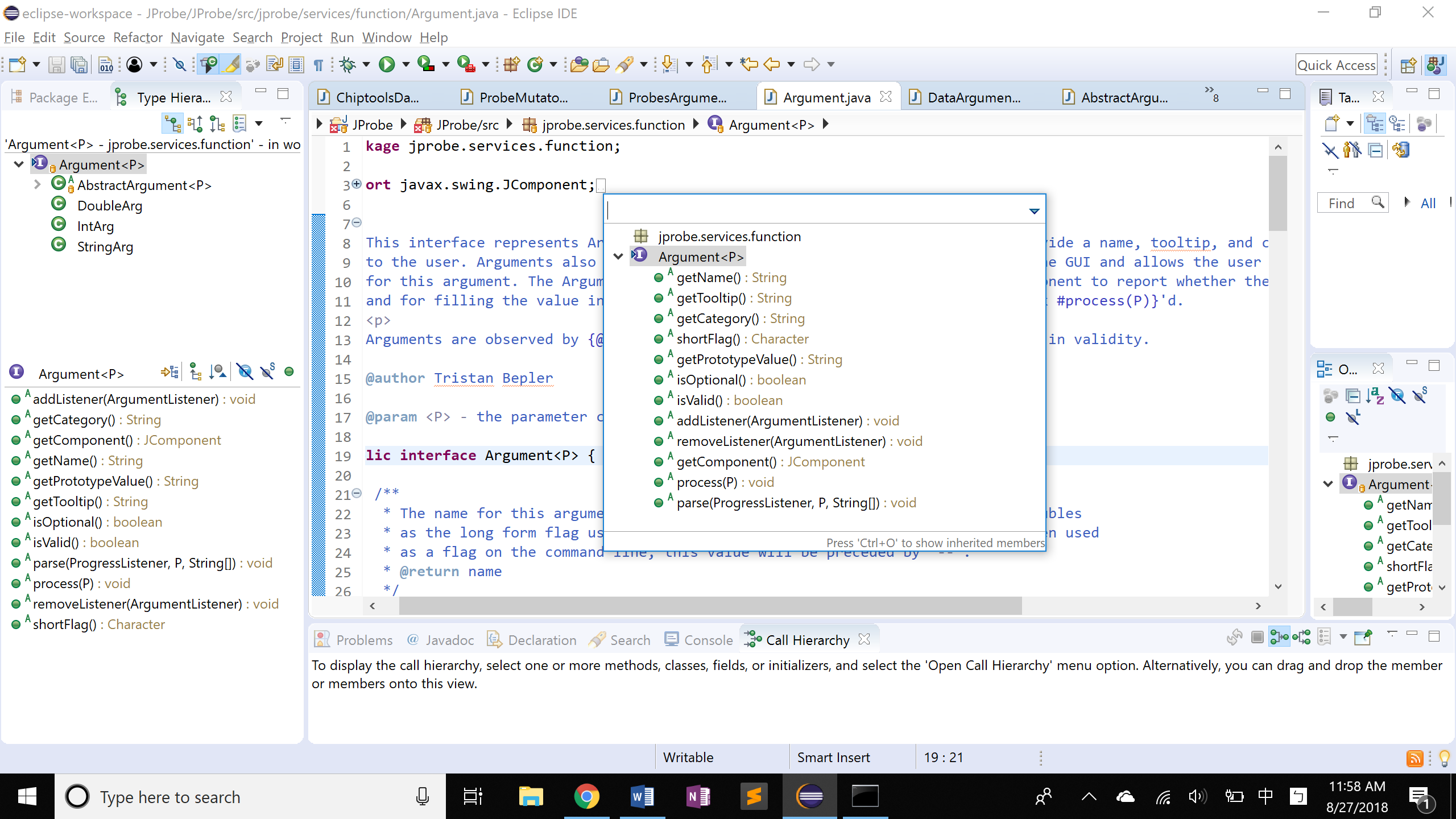
**package** chiptools.jprobe.function.bindingprofiler;

**class** BindingProfileParams **implements** ProbesParam, KmerListParam, PWMListParam

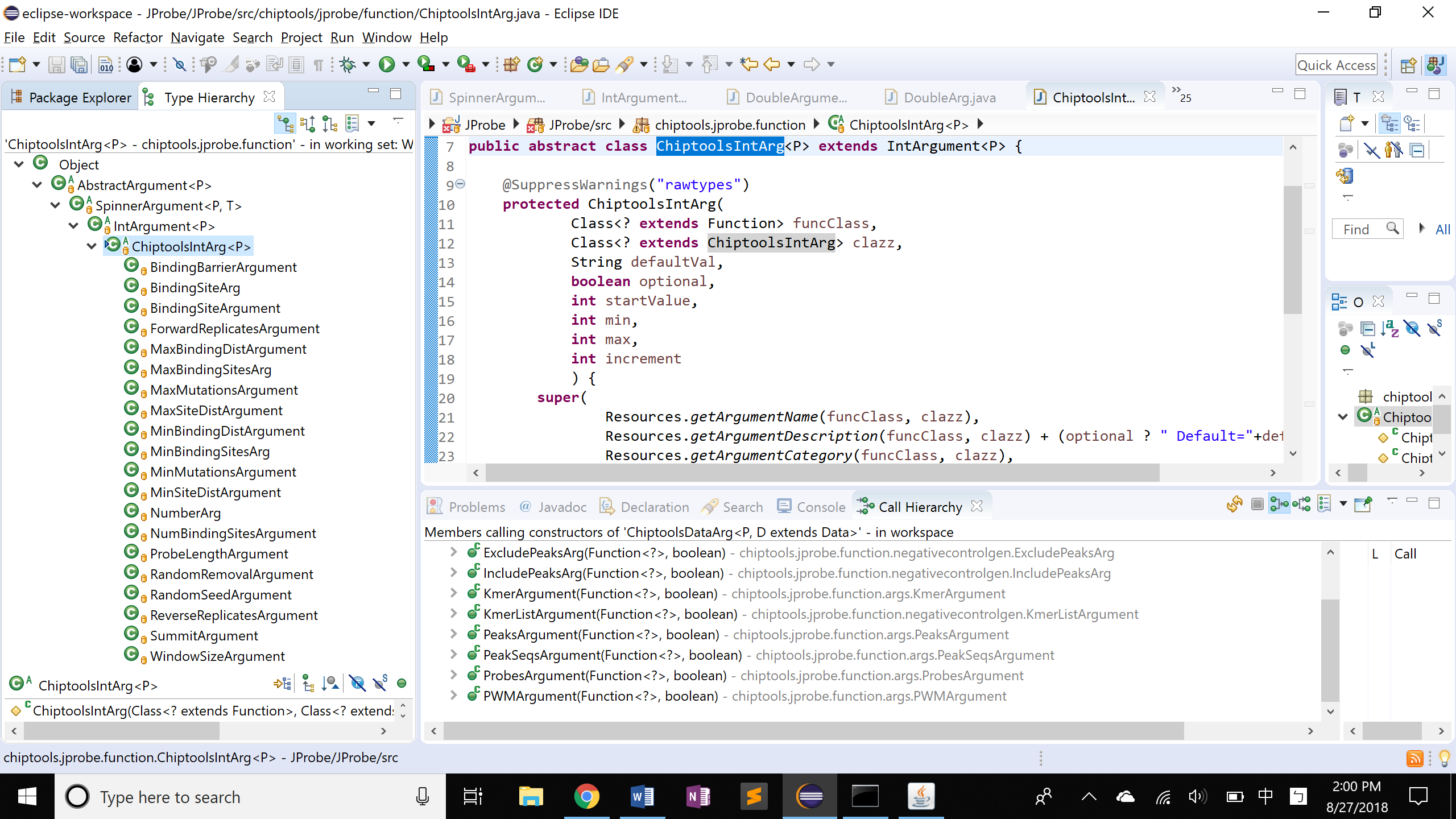
* + Creates empty parameter object for the function to be filled

1. getArguments()

## Argument Structure



**ChiptoolsIntArg**



## Default Generated Variable Name

|  |  |
| --- | --- |
| Function | Generated Variable Name |
| Agilent Formatter | N/A (file exported) |
| G-Runs Mutator | ProbeSetName\_GRun\_mut |
| Probe Generator | GenProbes\_n |
| Probe Mutator | ProbeSetName\_mut |
| Probe Joiner | JoinedProbes\_n |
| Peak Filter | PeakSetName\_filtered |
| Probe Filter | ProbeSetName\_filtered |
| Negative Control Generator | NegCtrl\_n |
| Peak Retriever | PeakSeqs\_PeakName |
| Binding Profiler | BindingProfile\_ProbeSetName |

Blue: input variable name

Metadata of the object Data

1. Metadata: a field in jprobe\services\data\**AbstractFinalData.java**
2. Data type:

Newly imported data OR

Function generation history

Issue: file rename

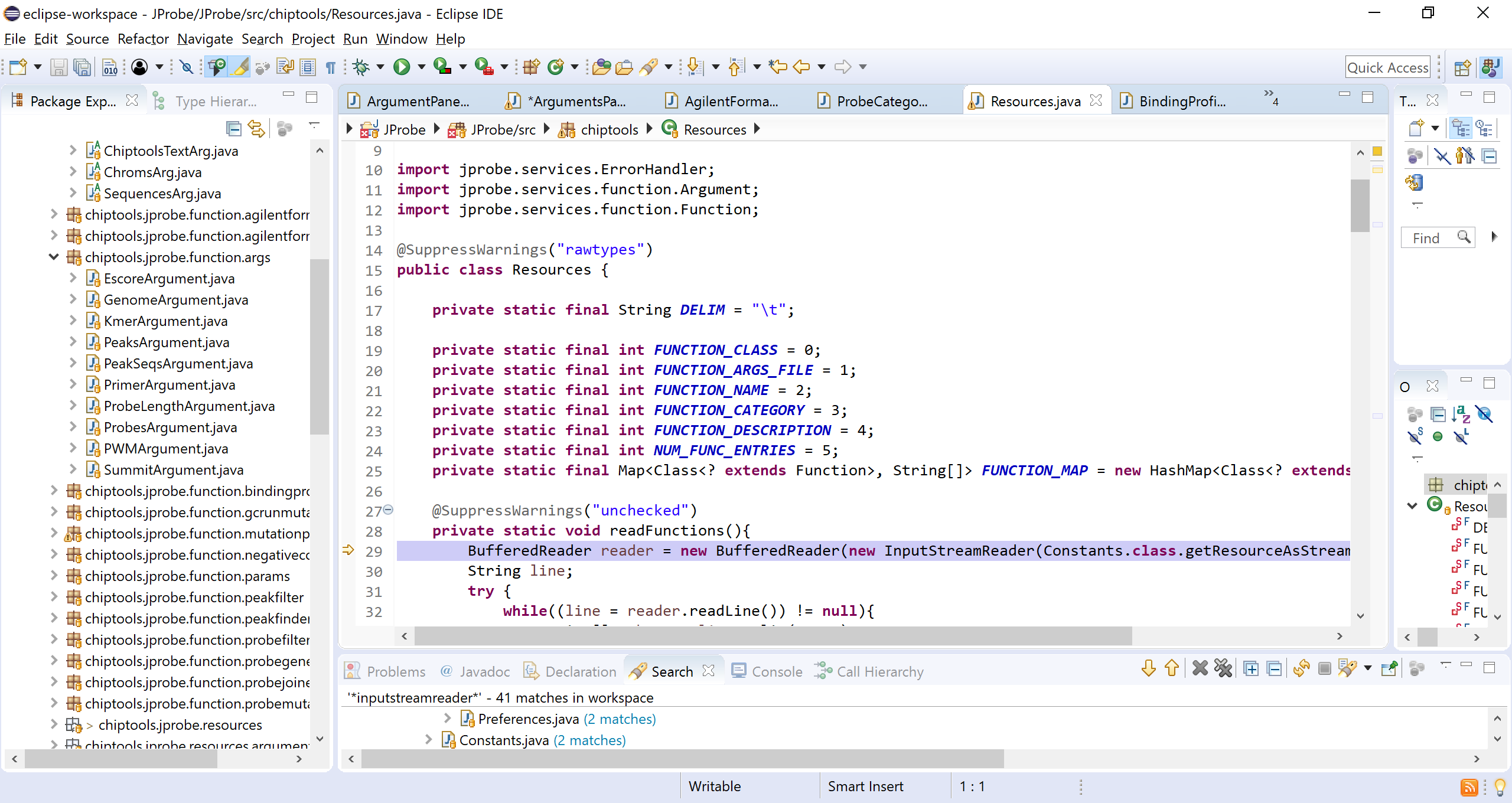
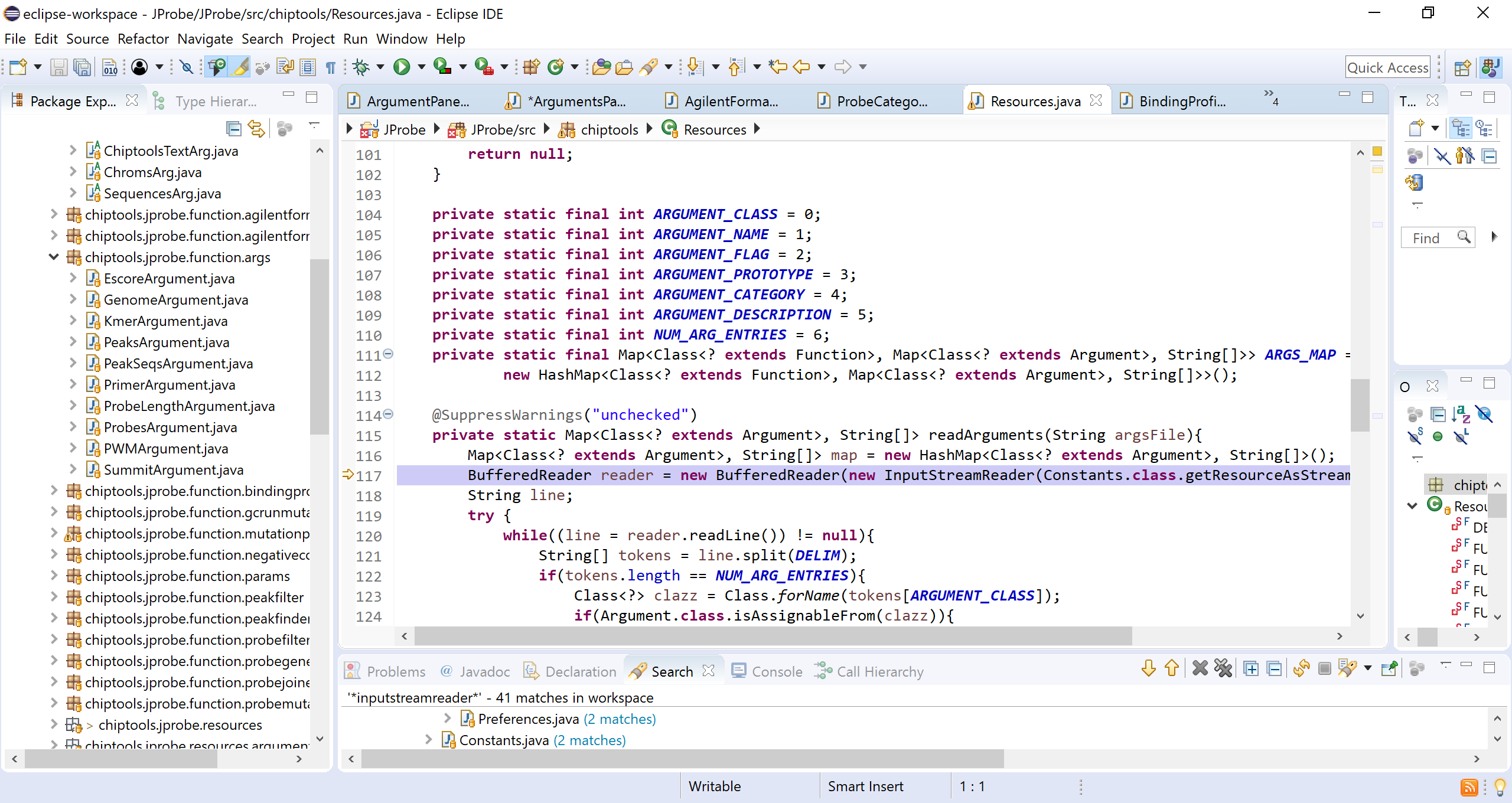
Can we get rid of [default = ] in description?

Export multiple files at once?

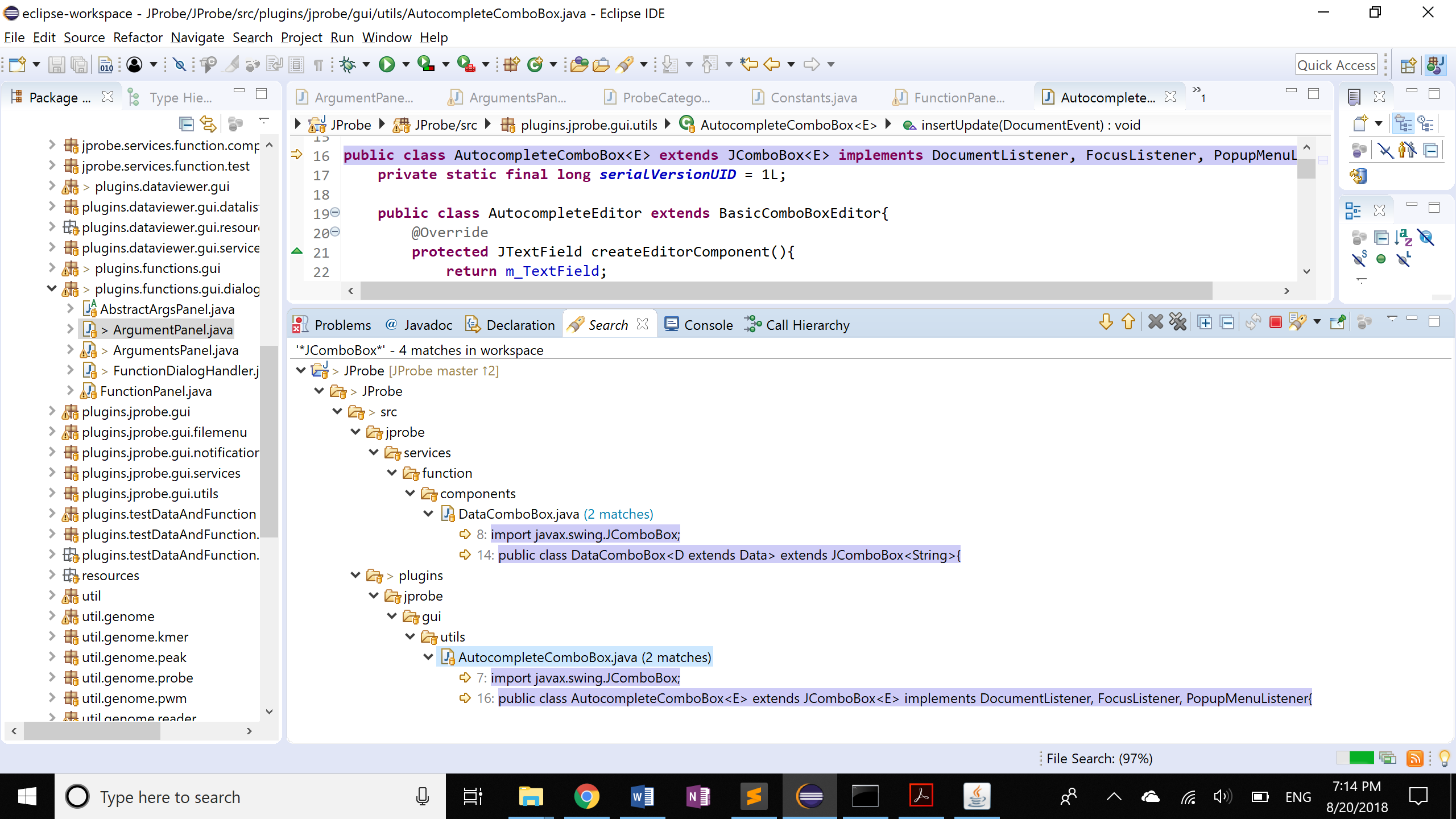
Need to prevent users from giving multiple input/output variables the same name; otherwise the map fails to work on Delete

To Do:

1. Create metadata
2. Complete getOutputName()

Check Box



# Other Ideas

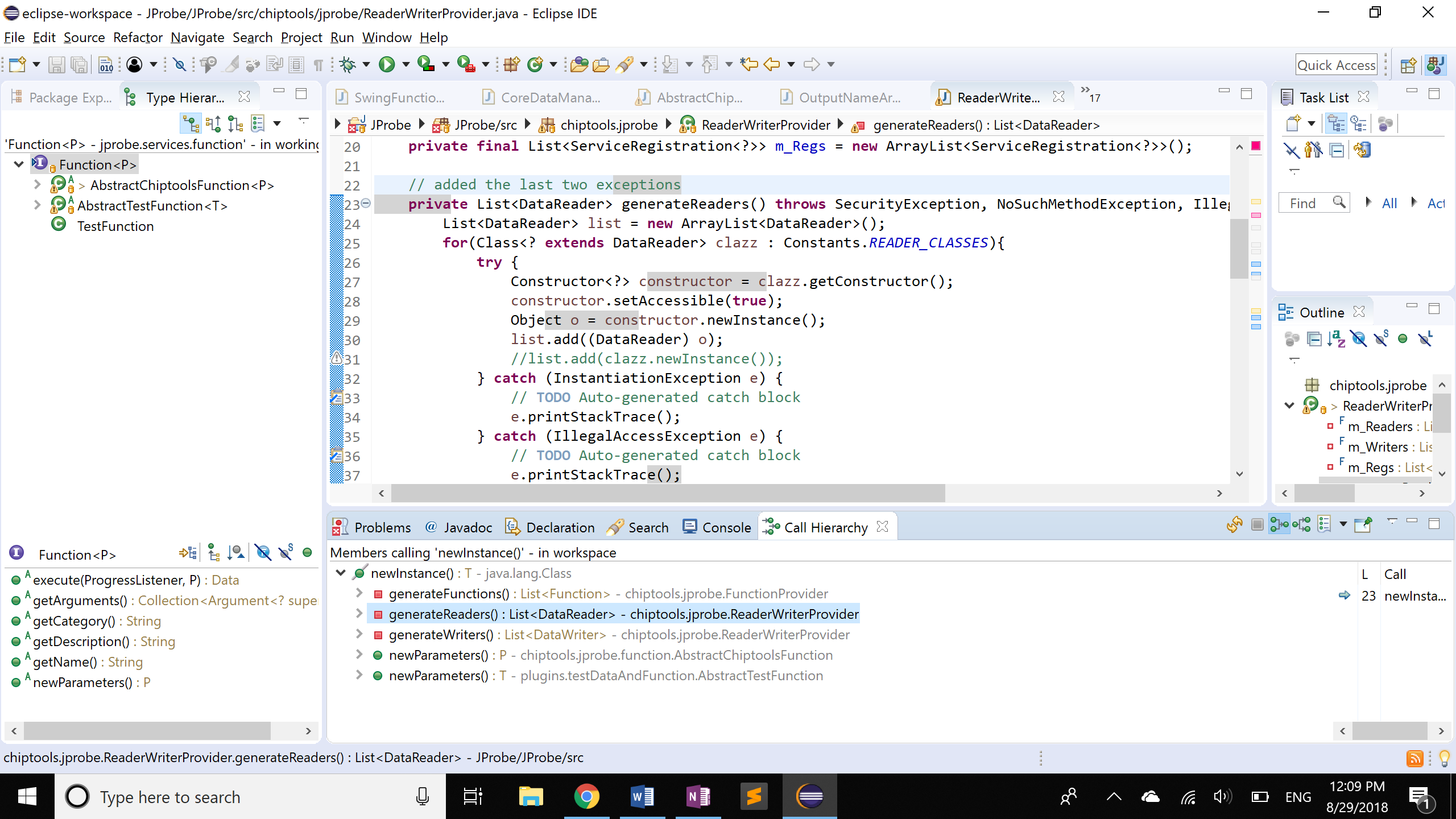
Class.newInstance() throws IllegalAccessException

<https://stackoverflow.com/questions/27764018/why-does-class-newinstance-always-throw-an-exception>

<https://stackoverflow.com/questions/29195039/java-getconstructor-throws-nosuchmethodexception>

* Class.newInstance():

Chiptools\jprobe\function\**AbstractChiptoolsFunction.java**



Pick back from: Abstract Chiptools Function