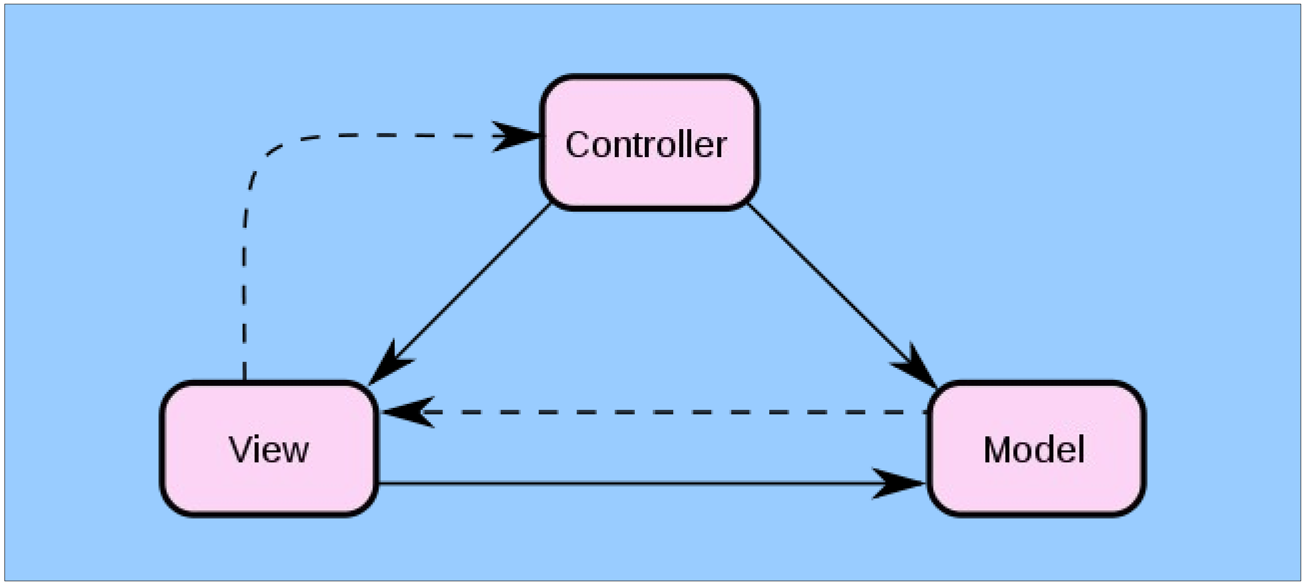
Goals

- Overall:
  - Refactoring existing software
    - Increase Maintainability, Reliability, Efficiency
  - Adding new functionality
    - So the others can focus on their work, not on coding
- Software should work and help, not fail and waste time

Current Work

- Java GUI
  - Applying MVC pattern
    - Split Code in 3 parts: Model, View, Controller
  - Refactoring code to use common coding and naming standards
    - Increases readability (a lot)
  - Adding Unit Tests for automatic testing, SoC
    - Helps to prevent a bad feeling for code changes
  - Adding code comments where necessary
  - Small tasks
    - Add this functionality, decrease loading time, decrease memory usage, ...

MVC Pattern

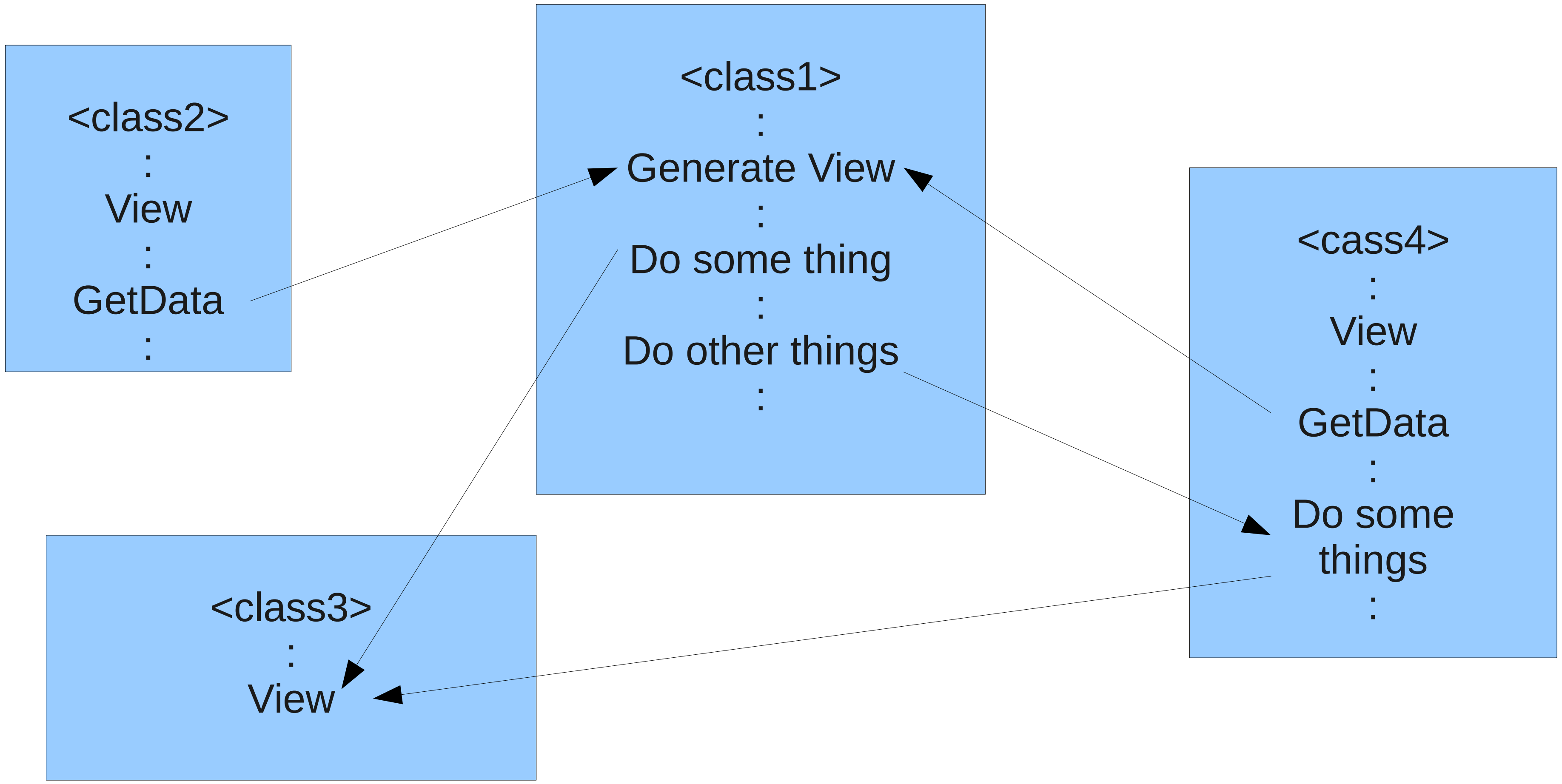


View: Different classes with JFrames

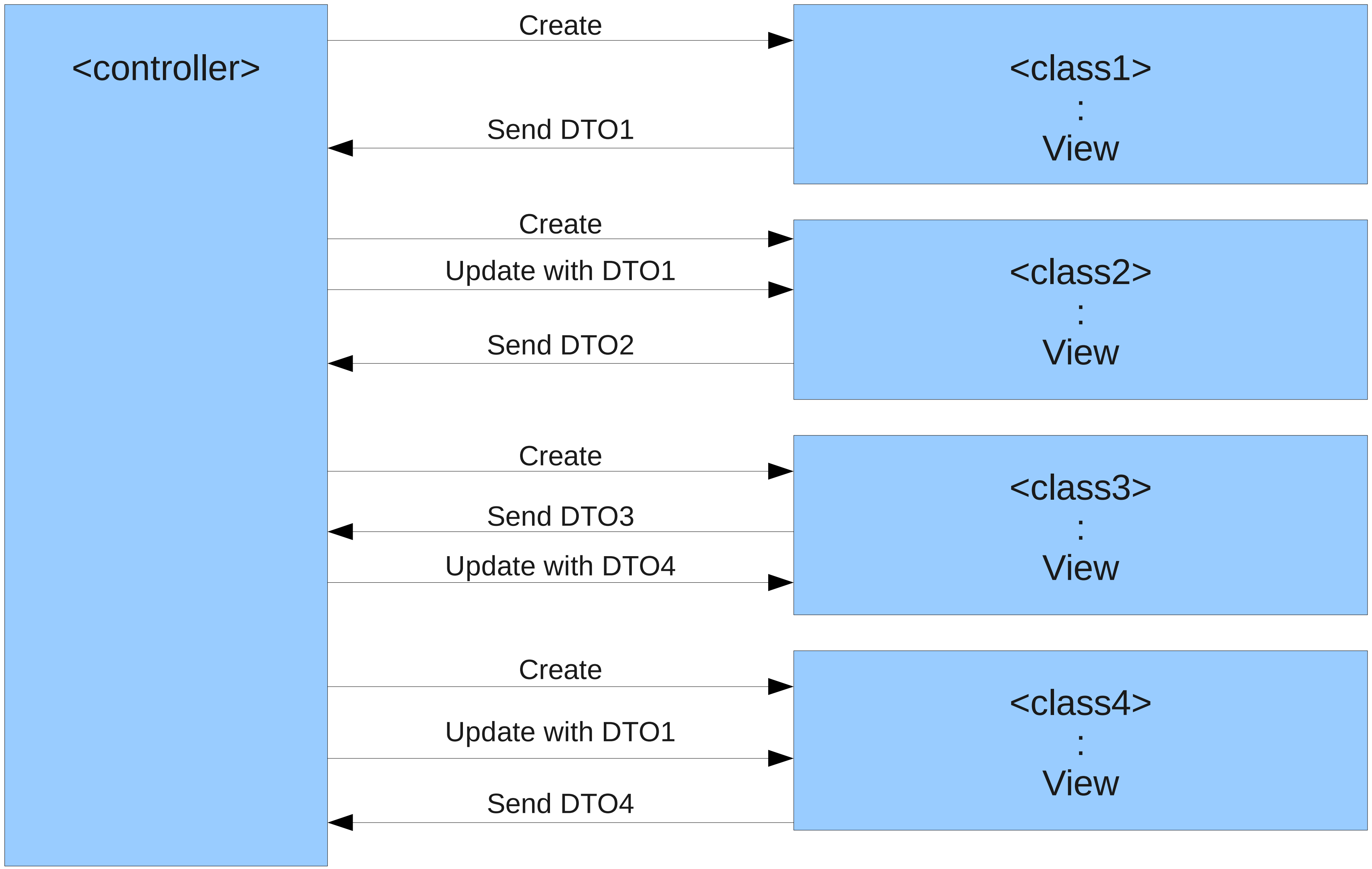
Controller: A single controller class

Model: Different data classes

Before MVC


<class2>
:
View
:
GetData
:
<class1>
:
Generate View
:
Do some thing
:
Do other things
:
<cass4>
:
View
:
GetData
:
Do some things
:
<class3>
:
View

MVC



*DTO = Data Transer Object

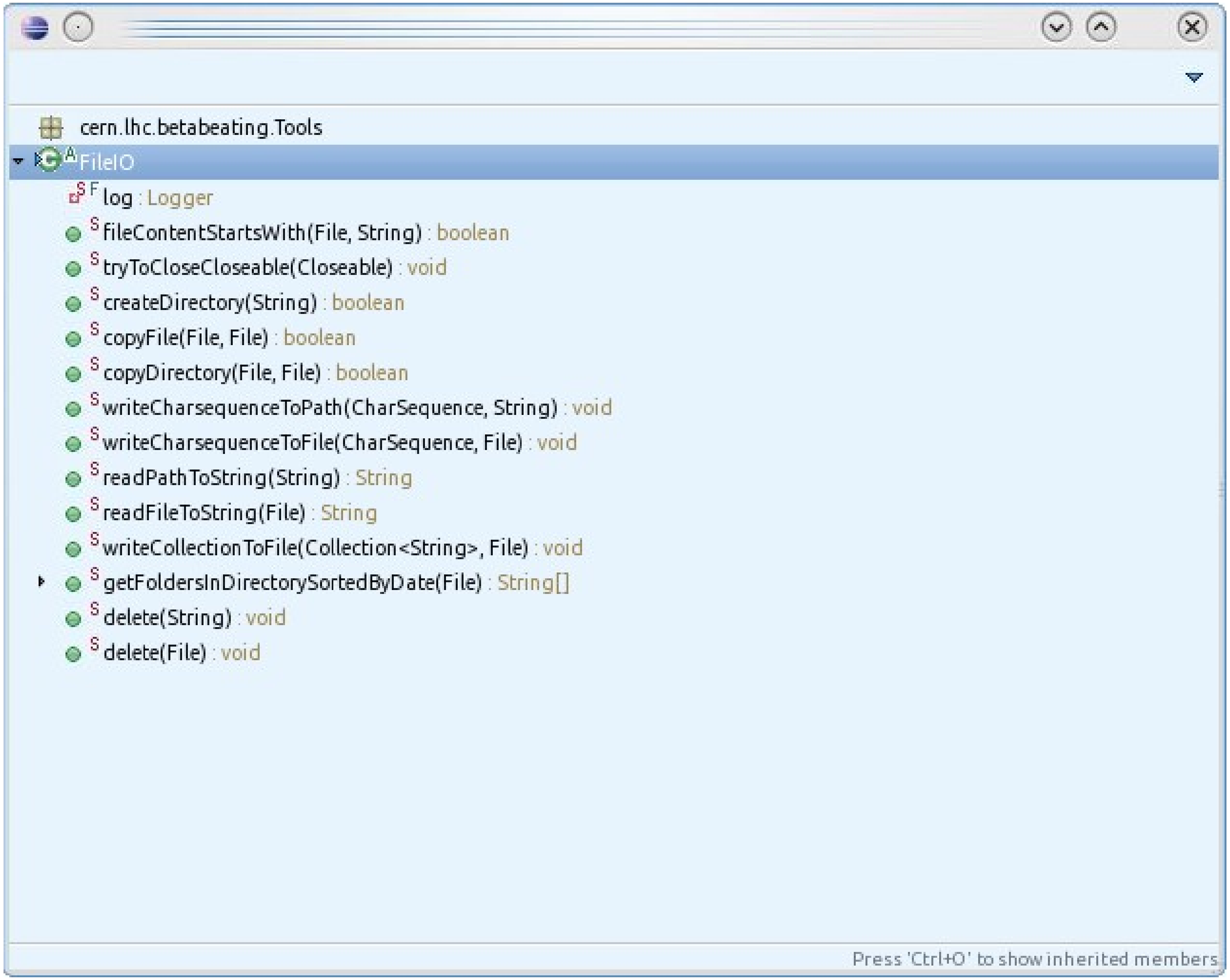
Coding / naming standards

- Code formatting
- Intention revealing names

```
public void createstructure(String accel,String input,String output,String
optics,String date,String bbdir){
 accelglo=accel;
 bbdirglo=bbdir;
 try {
            Thread.sleep(2000);
            label2.setText("In the next minute the data structure will be created ...");
            Thread.sleep(4000);
            label2.setText("Soooo sit back and relax :-)");
            Thread.sleep(2000);
            modelpath=output+"/"+date+"/models/"+accel+"/";
            // creating model dir
            if(!optics.equals("External")){
                        File4dir= new File(output+"/"+date+"/models/"+accel+"/"+optics);
                        file4optics=output+"/"+date+"/models/"+accel+"/"+optics;
                        File4dir.mkdirs();
                        if(File4dir.exists()){
                                    label2.setText("INFO: model dir created");
                        }else{
                                    label2.setText("ERROR: failed to create model dir ... contact expert");
                                    Thread.sleep(4000);
                                    System.exit(0);
                        }
            }else{
                        File4dir= new File(output+"/"+date+"/models/"+accel);
                        file4optics=output+"/"+date+"/models/"+accel;
                        File4dir.mkdirs();
                        if(File4dir.exists()){
                                    label2.setText("INFO: model dir created");
                        }else{
                                    label2.setText("ERROR: failed to create model dir ... contact expert");
                                    Thread.sleep(4000);
                                    System.exit(0);
                        }
            }
            Thread.sleep(2000);
            // creating measurements
            File4dir= new File(output+"/"+date+"/"+accel+"/Measurements");
            File4dir.mkdirs();
            if(File4dir.exists()){
                        label2.setText("INFO: measurements dir created");
            }else{
                        label2.setText("ERROR: failed to create measurements dir ... contact expert");
                        Thread.sleep(4000);
                        System.exit(0);
            }
            Thread.sleep(2000);
            // creating results
            File4dir= new File(output+"/"+date+"/"+accel+"/Results");
            File4dir.mkdirs();
            if(File4dir.exists()){
                        label2.setText("INFO: results dir created");
            }else{
                        label2.setText("ERROR: failed to create results dir ... contact expert");
                        Thread.sleep(4000);
                        System.exit(0);
            }
            Thread.sleep(2000);
            label2.setText("INFO: Finished creating dir... will create models");
            createoptics(optics);
            //creating models
            label2.setText("INFO: Model dir created! Will load main window");
            Thread.sleep(2000);
            setVisible(false);
 } catch (InterruptedException e) {
                       // TODO Auto-generated catch block
                       e.printStackTrace();
           }
}
... private methods
```

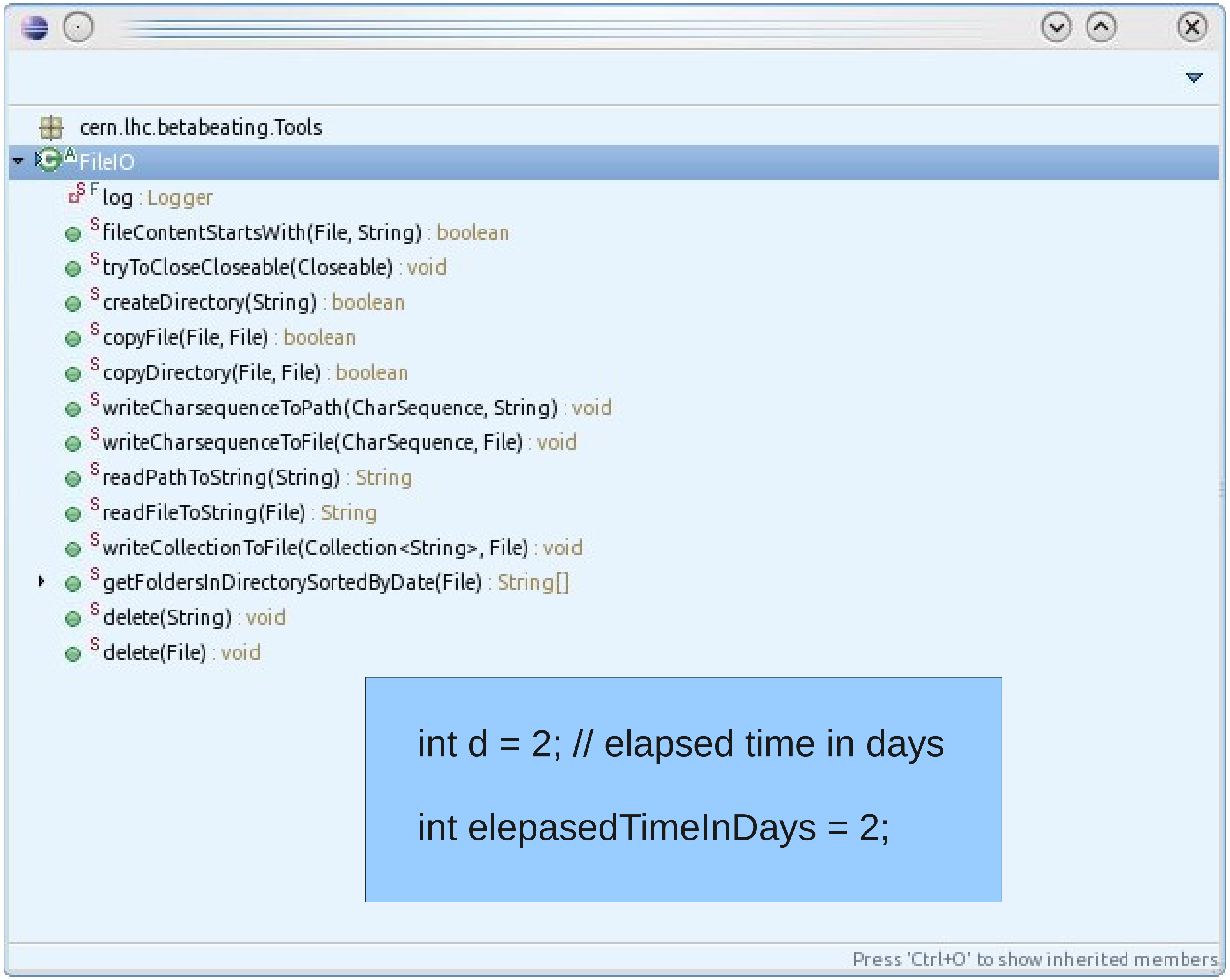
```
    public FileSystemStructureCreator(final BeamSelectionData
            beamSelectionData) {
        this.beamSelectionData = beamSelectionData;
    }

    public FileSystemStructureData createStructureAndGetData() {
        fileSystemStructureData = new FileSystemStructureData();
        createRoot();
        createOptics();
        createMeasurements();
        createResults();
        return fileSystemStructureData;
    }
... private methods
```

Intention revealing names



```
cern.lhc.betabeating.Tools
FileIO
    log : Logger
    fileContentStartsWith(File, String) : boolean
    tryToCloseCloseable(Closeable) : void
    createDirectory(String) : boolean
    copyFile(File, File) : boolean
    copyDirectory(File, File) : boolean
    writeCharsequenceToPath(CharSequence, String) : void
    writeCharsequenceToFile(CharSequence, File) : void
    readPathToString(String) : String
    readFileToString(File) : String
    writeCollectionToFile(Collection<String>, File) : void
    getFoldersInDirectorySortedByDate(File) : String[]
    delete(String) : void
    delete(File) : void
```

Intention revealing names

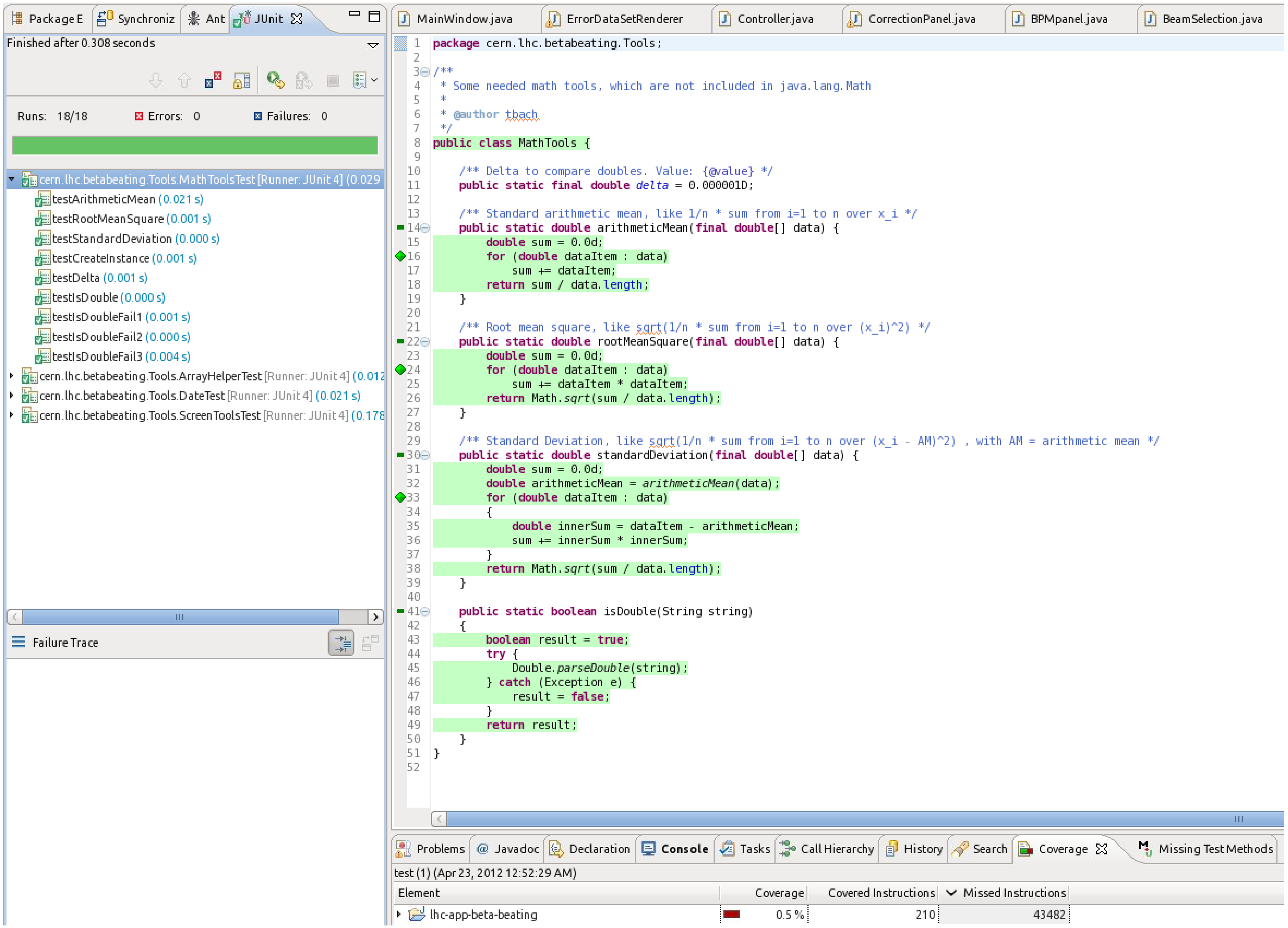


```
cern.lhc.betabeating.Tools
FileIO
  log : Logger
  fileContentStartsWith(File, String) : boolean
  tryToCloseCloseable(Closeable) : void
  createDirectory(String) : boolean
  copyFile(File, File) : boolean
  copyDirectory(File, File) : boolean
  writeCharsequenceToPath(CharSequence, String) : void
  writeCharsequenceToFile(CharSequence, File) : void
  readPathToString(String) : String
  readFileToString(File) : String
  writeCollectionToFile(Collection<String>, File) : void
  getFoldersInDirectorySortedByDate(File) : String[]
  delete(String) : void
  delete(File) : void
```

Press 'Ctrl+O' to show inherited members

```
int d = 2; // elapsed time in days

int elepasedTimeInDays = 2;
```

Junit Tests



Package E | Synchroniz | Ant | JUnit

Finished after 0.308 seconds

Runs: 18/18 | Errors: 0 | Failures: 0

```
cern.lhc.betabeating.Tools.MathToolsTest [Runner: JUnit 4] (0.029
    testArithmeticMean (0.021 s)
    testRootMeanSquare (0.001 s)
    testStandardDeviation (0.000 s)
    testCreateInstance (0.001 s)
    testDelta (0.001 s)
    testIsDouble (0.000 s)
    testIsDoubleFail1 (0.001 s)
    testIsDoubleFail2 (0.000 s)
    testIsDoubleFail3 (0.004 s)
cern.lhc.betabeating.Tools.ArrayHelperTest [Runner: JUnit 4] (0.012
cern.lhc.betabeating.Tools.DateTest [Runner: JUnit 4] (0.021 s)
cern.lhc.betabeating.Tools.ScreenToolsTest [Runner: JUnit 4] (0.178
```

Failure Trace

MainWindow.java | ErrorDataSetRenderer | Controller.java | CorrectionPanel.java | BPMpanel.java | BeamSelection.java

```java
package cern.lhc.betabeating.Tools;

/**
 * Some needed math tools, which are not included in java.lang.Math
 *
 * @author tbach
 */
public class MathTools {

    /** Delta to compare doubles. Value: {@value} */
    public static final double delta = 0.000001D;

    /** Standard arithmetic mean, like 1/n * sum from i=1 to n over x_i */
    public static double arithmeticMean(final double[] data) {
        double sum = 0.0d;
        for (double dataItem : data)
            sum += dataItem;
        return sum / data.length;
    }

    /** Root mean square, like sqrt(1/n * sum from i=1 to n over (x_i)^2) */
    public static double rootMeanSquare(final double[] data) {
        double sum = 0.0d;
        for (double dataItem : data)
            sum += dataItem * dataItem;
        return Math.sqrt(sum / data.length);
    }

    /** Standard Deviation, like sqrt(1/n * sum from i=1 to n over (x_i - AM)^2) , with AM = arithmetic mean */
    public static double standardDeviation(final double[] data) {
        double sum = 0.0d;
        double arithmeticMean = arithmeticMean(data);
        for (double dataItem : data)
        {
            double innerSum = dataItem - arithmeticMean;
            sum += innerSum * innerSum;
        }
        return Math.sqrt(sum / data.length);
    }

    public static boolean isDouble(String string)
    {
        boolean result = true;
        try {
            Double.parseDouble(string);
        } catch (Exception e) {
            result = false;
        }
        return result;
    }
}
```

Problems | Javadoc | Declaration | Console | Tasks | Call Hierarchy | History | Search | Coverage | Missing Test Methods

test (1) (Apr 23, 2012 12:52:29 AM)

Element	Coverage	Covered Instructions	Missed Instructions
lhc-app-beta-beating	🟥 0.5 %	210	43482

Separation of Concerns

- One class should do one big thing
  - Manages the settings
  - Displays the results, ...
  - Not: Loads the settings and creates the model and loads files and calculates some results and displays them
- One function should do one small thing
  - Save a string to a file
  - Calculate square root, ...

Separation of Concerns

- One class should do one big thing
  - Manages the settings
  - Displays the results, ...
  - Not: Loads the settings and creates the model and loads files and calculates some results and displays them
- One function should do one small thing
  - Save a string to a file
  - Calculate square root, ...
- Best Indicator: If you need an “and” in the description, you should think about separation.

Code comments

- Only when necessary
  - A lot of comments is a good sign for bad code. Make better code, not more comments.
  - Noone updates comments if code changes
  - Delete unused functions or variables. You have a version control system
- JavaDoc for public usage
  - API, “what a function does”
- Normal comments for internal usage
  - Algorithm, main idea, “how a function works”

Small tasks

- Functionality: Small things like the frame should remember the last used values, other values here or some labels there

- Improve loading time
  - Removed this while loop: and the loading screen. Startup from ~30s to <1s

```
int signall=0;
while(signall==0){
    System.out.println("Waiting");
    signall=bsel.signal;
}
```

Small things

- Reduced memory usage

```
0 BPMSW.1L1.B1     23497.87662  -0.08441  -0.08857  -0.13662 ... some 1000 more
... for all bpms
```

```
String[] splits=line.split("\\s+");

bpmsh.add(splits[1]);
```

Small things

- Reduced memory usage

```
0 BPMSW.1L1.B1     23497.87662  -0.08441  -0.08857  -0.13662 ... some 1000 more
... for all bpms
```

```
String[] splits=line.split("\\s+");

bpmsh.add(splits[1]);
```

Garbage Collector cannot work.
~40-80mb wasted for every file.

Small things

- Reduced memory usage

```
0 BPMSW.1L1.B1     23497.87662  -0.08441  -0.08857  -0.13662 ... some 1000 more
... for all bpms
```

```
String[] splits=line.split("\\s+");

bpmsh.add(splits[1]);
```

Garbage Collector cannot work.
~40-80mb wasted for every file.

```
String[] splits = line.split("\\s+");
String bpmName = new String(splits[1]);
```

No reference! Garbage Collector can work

=> Now: more than 50 files for less then 1gb RAM

To do for the GUI

- Finish refactoring
- Finish JUnit tests
- Write an integration test plan
- Testing framework for external python scripts
- Functionality ?
  - Improve file loading time
  - Wanalysis
  - ...
- Estimated time: Depends on the code, perhaps 4 to 8 weeks