ImageJ Legacy

How to build mixed IJ1-IJ2 pipelines

Max Planck Institute of Molecular Cell Biology and Genetics

Robert Haase

MPI-CBG, Scientific Computing Facility

- **How to switch between coding ImageJ1 and ImageJ2 with facility.**

Outline

- Introduction
  - What?
  - Why?
- How to?
  - run ImageJ1 and ImageJ2 from within the IDE
  - run ImageJ2 plugins the ImageJ1/2 ways
  - convert/wrap/visualise ImageJ1/2 images in(to) ImageJ2/1

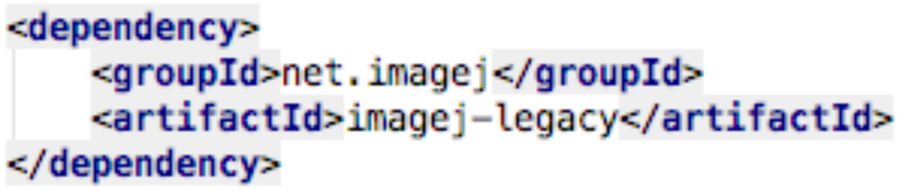
    exercise
  - convert/wrap/visualise ImageJ1/2 regions in(to) ImageJ2/1

    exercise
  - convert/visualise ImageJ1/2 tables in(to) ImageJ2/1

    exercise

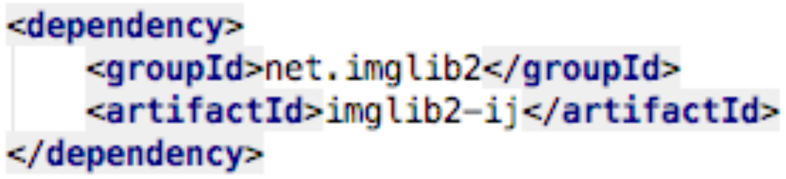
What is ImageJ-Legacy?

- `ImageJ-Legacy` is a dependency allowing to build ImageJ1 pipelines with ImageJ2 modules/commands conveniently



```
<dependency>
    <groupId>net.imagej</groupId>
    <artifactId>imagej-legacy</artifactId>
</dependency>
```

- More legacy stuff is available in `ImageJFunctions` and `ImagePlusAdapter` in `imglib2-ij`



```
<dependency>
    <groupId>net.imglib2</groupId>
    <artifactId>imglib2-ij</artifactId>
</dependency>
```

CBG
Max Planck Institute
of Molecular Cell Biology
and Genetics

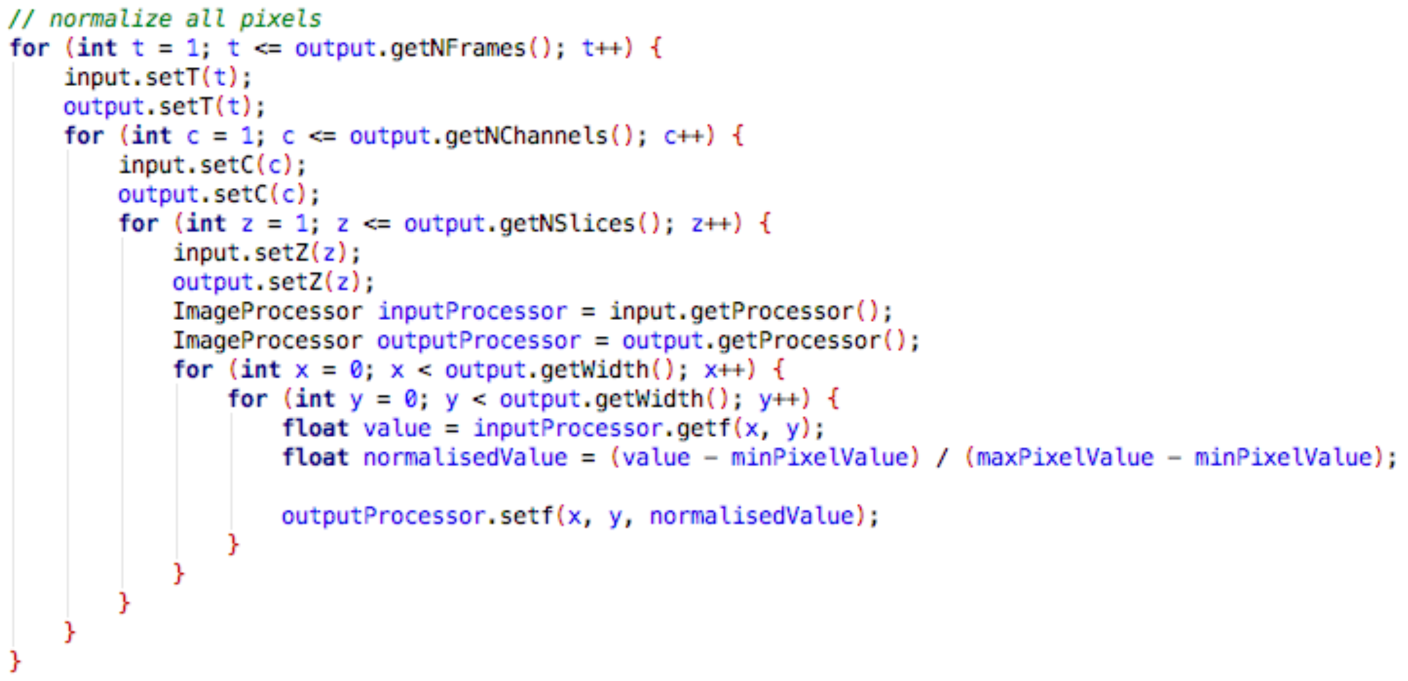
Why shall we upgrade code?

- To use state-of-the-art libraries, modules and plugins
- To be potentially faster
- To be future-save
- To ensure maintainability

ImageJ legacy

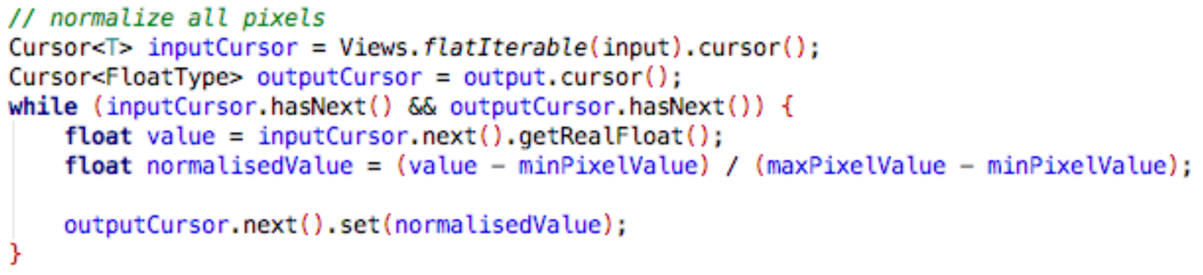
Why shall we upgrade code?

- ImageJ1



```
// normalize all pixels
for (int t = 1; t <= output.getNFrames(); t++) {
    input.setT(t);
    output.setT(t);
    for (int c = 1; c <= output.getNChannels(); c++) {
        input.setC(c);
        output.setC(c);
        for (int z = 1; z <= output.getNSlices(); z++) {
            input.setZ(z);
            output.setZ(z);
            ImageProcessor inputProcessor = input.getProcessor();
            ImageProcessor outputProcessor = output.getProcessor();
            for (int x = 0; x < output.getWidth(); x++) {
                for (int y = 0; y < output.getWidth(); y++) {
                    float value = inputProcessor.getf(x, y);
                    float normalisedValue = (value - minPixelValue) / (maxPixelValue - minPixelValue);

                    outputProcessor.setf(x, y, normalisedValue);
                }
            }
        }
    }
}
```

- ImageJ2



```
// normalize all pixels
Cursor<T> inputCursor = Views.flatIterable(input).cursor();
Cursor<FloatType> outputCursor = output.cursor();
while (inputCursor.hasNext() && outputCursor.hasNext()) {
    float value = inputCursor.next().getRealFloat();
    float normalisedValue = (value - minPixelValue) / (maxPixelValue - minPixelValue);

    outputCursor.next().set(normalisedValue);
}
```

ImageJ legacy

Session goal

- Replace parts of ImageJ1 code by ImageJ2 code
- Step-by-step upgrading existing code instead of
- ~~total re-coding of entire pipelines~~

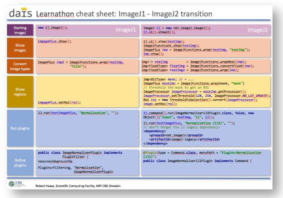
- Therefore you need to learn how to switch between ImageJ1 and ImageJ2 within your pipelines.
  - Data conversion

- Exercises (interactive coding sessions)
  - ImagePlus versus Img
  - ROIs versus Regions
  - ResultsTable versus GenericTable

- Take home: code snippets making your life easier

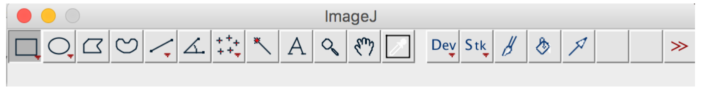
ImageJ legacy

How to start ImageJ UI via code from within the IDE



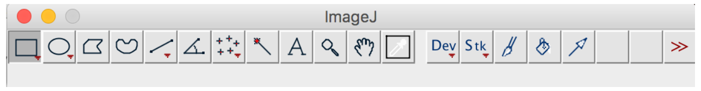
- The constructor of ImageJ1 also opens the user interface

```
new ij.ImageJ();
```



- In ImageJ2, you need to do this on your own

```
ImageJ ij = new net.imagej.ImageJ();

ij.ui().showUI();
```



- Furthermore, you can use the `ij` variable to do all the awesome stuff

```
ij.op().run("fancyAlgorithm”, image);
```

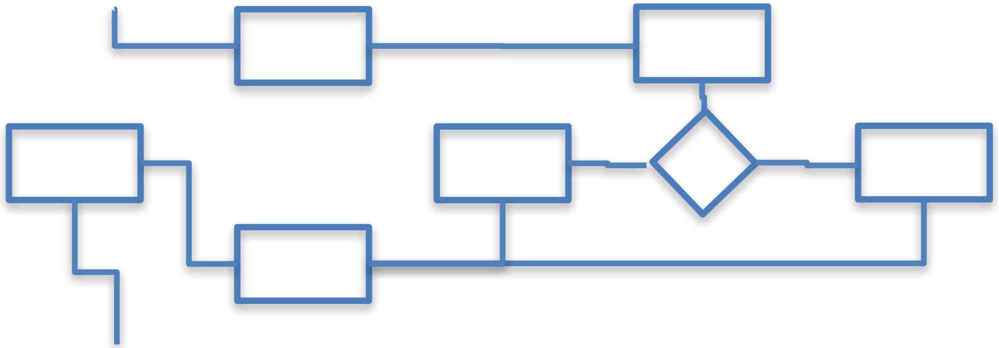
- Never loose this `ij` variable! DON’T use global variables!

ImageJ legacy

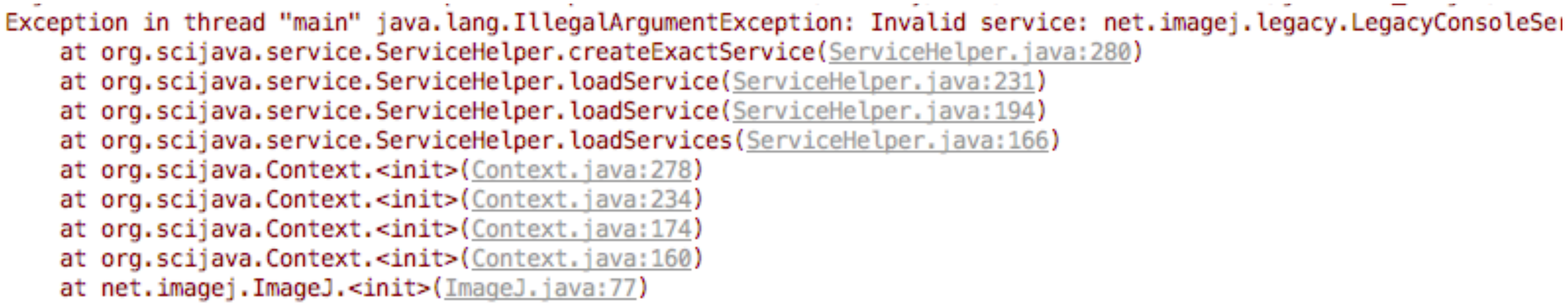
Troubleshooting: starting ImageJ2 from IDE

- At some point in your pipeline, you need an ImageJ2 ij varable (e.g. to access Ops)

```
line 15:  new ij.ImageJ();
```



```
line 390: ImageJ ij = new net.imagej.ImageJ();
```

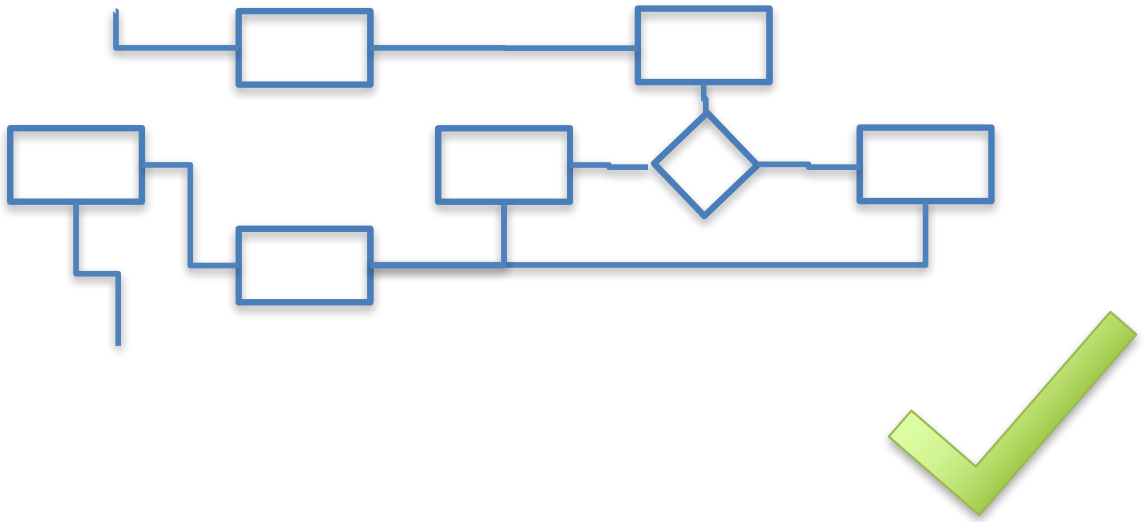


```
Exception in thread "main" java.lang.IllegalArgumentException: Invalid service: net.imagej.legacy.LegacyConsoleSe
    at org.scijava.service.ServiceHelper.createExactService(ServiceHelper.java:280)
    at org.scijava.service.ServiceHelper.loadService(ServiceHelper.java:231)
    at org.scijava.service.ServiceHelper.loadService(ServiceHelper.java:194)
    at org.scijava.service.ServiceHelper.loadServices(ServiceHelper.java:166)
    at org.scijava.Context.<init>(Context.java:278)
    at org.scijava.Context.<init>(Context.java:234)
    at org.scijava.Context.<init>(Context.java:174)
    at org.scijava.Context.<init>(Context.java:160)
    at net.imagej.ImageJ.<init>(ImageJ.java:77)
```

Troubleshooting: starting ImageJ2 from IDE

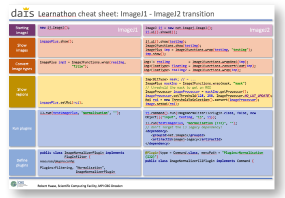
- At some point in your pipeline, you need an ImageJ2 ij variable (e.g. to access Ops)

```
line 14: ImageJ ij = new net.imagej.ImageJ();
line 15: new ij.ImageJ();
```



ImageJ legacy

How to run plugins



dais Learnathon cheat sheet: ImageJ1 - ImageJ2 transition

	ImageJ1	ImageJ2

Starting ImageJ	`new ij.ImageJ();`	`ImageJ ij = new net.imagej.ImageJ();`
		`ij.ui().showUI();`

Show images	`imagePlus.show();`	`ij.ui().show(testImg);`
		`ImageJFunctions.show(testImg);`
		`ImagePlus imp = ImageJFunctions.wrap(testImg, "testImg");`
		`imp.show();`

Convert image types	`ImagePlus imp2 = ImageJFunctions.wrap(realImg, "Title");`	`Img<T> realImg = ImageJFunctions.wrapReal(imp);`
		`Img<FloatType> floatImg = ImageJFunctions.convertFloat(imp);`
		`Img<FloatType> realImg2 = ImageJFunctions.wrap(imp);`

Show regions	`imagePlus.setRoi(roi);`	`Img<BitType> mask; // = ...`
		`ImagePlus maskImp = ImageJFunctions.wrap(mask, "mask")`
		`// threshold the mask to get an ROI`
		`ImageProcessor imageProcessor = maskImp.getProcessor();`
		`imageProcessor.setThreshold(128, 258, ImageProcessor.NO_LUT_UPDATE);`
		`Roi roi = new ThresholdToSelection().convert(imageProcessor);`
		`image.setRoi(roi);`

Run plugins	`IJ.run(testImagePlus, "Normalisation", "");`	`ij.command().run(ImageNormalizerIJ2Plugin.class, false, new Object[]{"input", testImg, "ij", ij});`
		`IJ.run(testImagePlus, "Normalisation (IJ2)", "");`
		`// don't forget the IJ legacy dependency!`
		`<dependency>`
		`<groupId>net.imagej</groupId>`
		`<artifactId>imagej-legacy</artifactId>`
		`</dependency>`

Define plugins	`public class ImageNormalizerPlugin implements PlugInFilter {`
	resources/plugins.config:
	`Plugins>Filtering, "Normalisation", ImageNormalizerPlugin`
		`@Plugin(type = Command.class, menuPath = "Plugins>Normalisation (IJ2)")`
		`public class ImageNormalizerIJ2Plugin implements Command {`

Robert Haase, Scientific Computing Facility, MPI CBG Dresden

- In IJ1, you can call a plugin like this

```
IJ.run(testImagePlus, "Normalisation", "");
```

- If the string "`Normalisation`" is linked to the right class in the `resources/plugin.config` file

```
Plugins>Filtering, "Normalisation", the.full.classname.ImageNormalizerPlugin
```

- The class must implement `PlugInFilter` (or...)

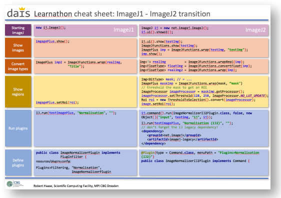
```
public class ImageNormalizerPlugin implements PlugInFilter {
```

- This does only work from within ImageJ, not from the IDE

CBG
Max Planck Institute
of Molecular Cell Biology
and Genetics

ImageJ legacy

How to run plugins



- In IJ2, you *can* call plugins the same way

```
IJ.run(testImagePlus, "Normalisation (IJ2)", "");
```

- If the class is marked with the right string

```
@Plugin(type = Command.class, menuPath = "Plugins>Normalisation (IJ2)")
public class ImageNormalizerIJ2Plugin implements Command {
```

- And `imagej-legacy` is part of your dependencies

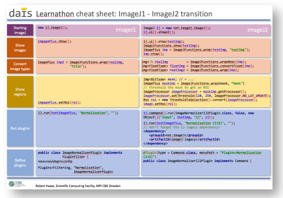
```
<dependency>
   <groupId>net.imagej</groupId>
   <artifactId>imagej-legacy</artifactId>
</dependency>
```

- The cool thing is: This works from the IDE!

CBG Max Planck Institute of Molecular Cell Biology and Genetics

ImageJ legacy

How to run plugins


Learnathon cheat sheet: ImageJ1 - ImageJ2 transition

- In IJ2, you *should* call plugins this way

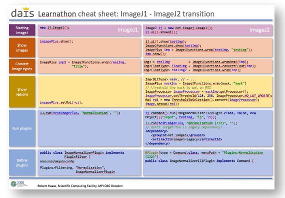
```
ij.command().run(ImageNormalizerIJ2Plugin.class, false,
                 new Object[]{"input", testImg, "ij", ij});
```

- You need to know the parameters of your plugin

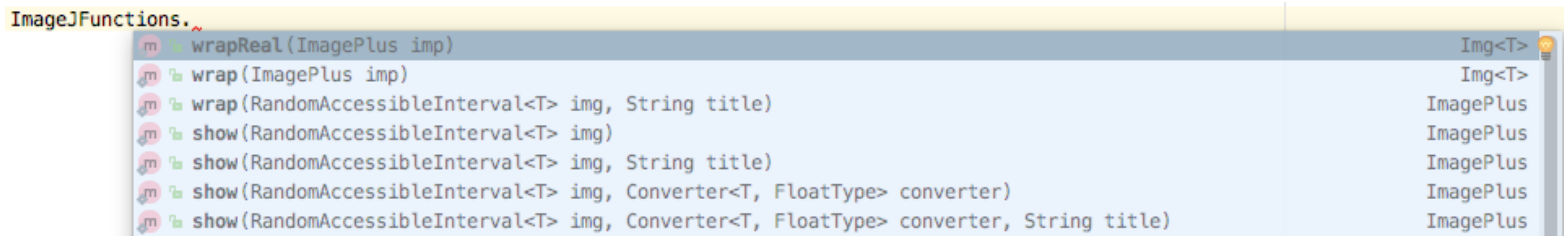
```
@Plugin(type = Command.class, menuPath = "Plugins>Filtering>Normalisation (IJ2)")
public class ImageNormalizerIJ2Plugin implements Command {
    @Parameter
    Img input;

    @Parameter
    ImageJ ij;

    @Override
    public void run() {
```

- This, of course, also works from the IDE

ImageJFunctions



- `ImageJFunctions` contains super useful convenience functions for switching between the ImageJ1 and ImageJ2 worlds.



```
ImageJFunctions.
  wrapReal(ImagePlus imp)                                                       Img<T>
  wrap(ImagePlus imp)                                                           Img<T>
  wrap(RandomAccessibleInterval<T> img, String title)                        ImagePlus
  show(RandomAccessibleInterval<T> img)                                      ImagePlus
  show(RandomAccessibleInterval<T> img, String title)                        ImagePlus
  show(RandomAccessibleInterval<T> img, Converter<T, FloatType> converter)   ImagePlus
  show(RandomAccessibleInterval<T> img, Converter<T, FloatType> converter, String title)   ImagePlus
```

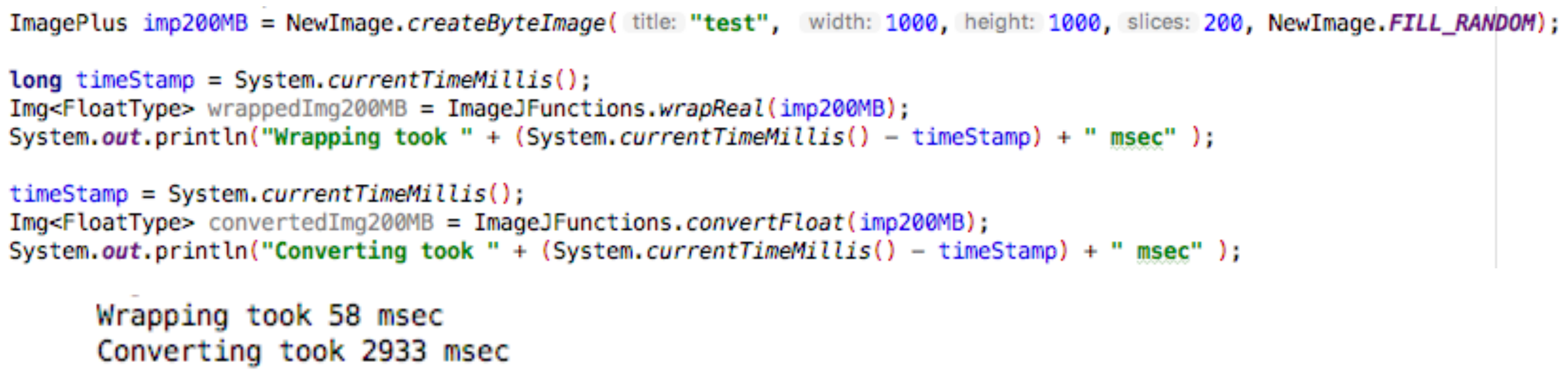
- These are my favorites:

```
Img<T> realImg              = ImageJFunctions.wrapReal(imp);

Img<FloatType> floatImg = ImageJFunctions.convertFloat(imp);

Img<FloatType> realImg2 = ImageJFunctions.wrap(imp);

ImagePlus imp2              = ImageJFunctions.wrap(realImg, "Title");
```

Max Planck Institute of Molecular Cell Biology and Genetics

ImageJ legacy

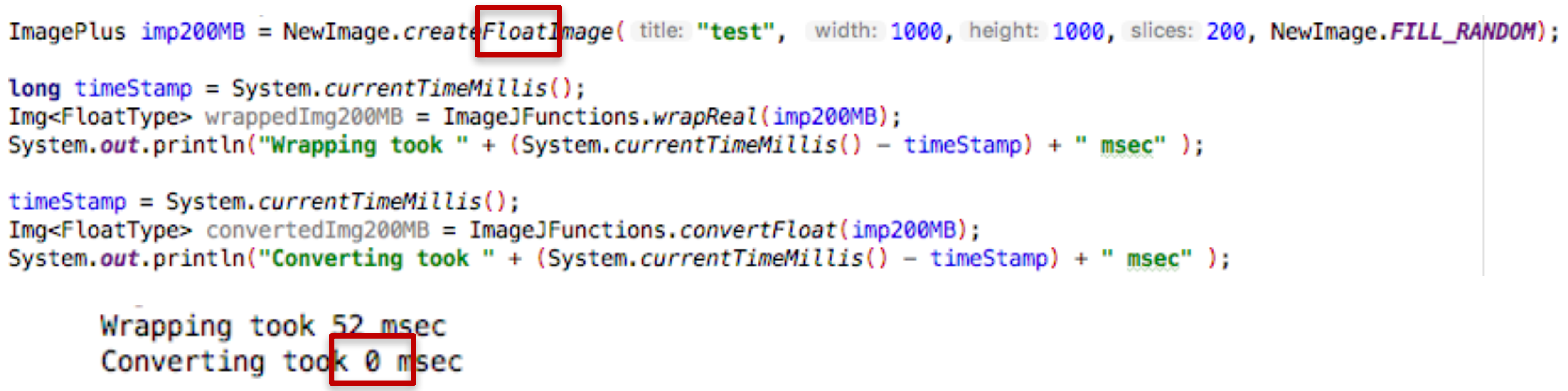
Pitfalls: ImageJFunctions



```
ImagePlus imp200MB = NewImage.createByteImage( title: "test",  width: 1000, height: 1000, slices: 200, NewImage.FILL_RANDOM);

long timeStamp = System.currentTimeMillis();
Img<FloatType> wrappedImg200MB = ImageJFunctions.wrapReal(imp200MB);
System.out.println("Wrapping took " + (System.currentTimeMillis() - timeStamp) + " msec" );

timeStamp = System.currentTimeMillis();
Img<FloatType> convertedImg200MB = ImageJFunctions.convertFloat(imp200MB);
System.out.println("Converting took " + (System.currentTimeMillis() - timeStamp) + " msec" );
```

```
Wrapping took 58 msec
Converting took 2933 msec
```

- However,

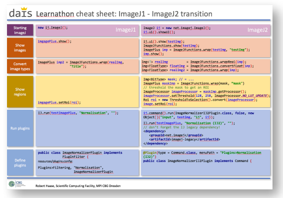


```
ImagePlus imp200MB = NewImage.createFloatImage( title: "test",  width: 1000, height: 1000, slices: 200, NewImage.FILL_RANDOM);

long timeStamp = System.currentTimeMillis();
Img<FloatType> wrappedImg200MB = ImageJFunctions.wrapReal(imp200MB);
System.out.println("Wrapping took " + (System.currentTimeMillis() - timeStamp) + " msec" );

timeStamp = System.currentTimeMillis();
Img<FloatType> convertedImg200MB = ImageJFunctions.convertFloat(imp200MB);
System.out.println("Converting took " + (System.currentTimeMillis() - timeStamp) + " msec" );
```

```
Wrapping took 52 msec
Converting took 0 msec
```

ImageJ legacy

How to show ImageJ2 images

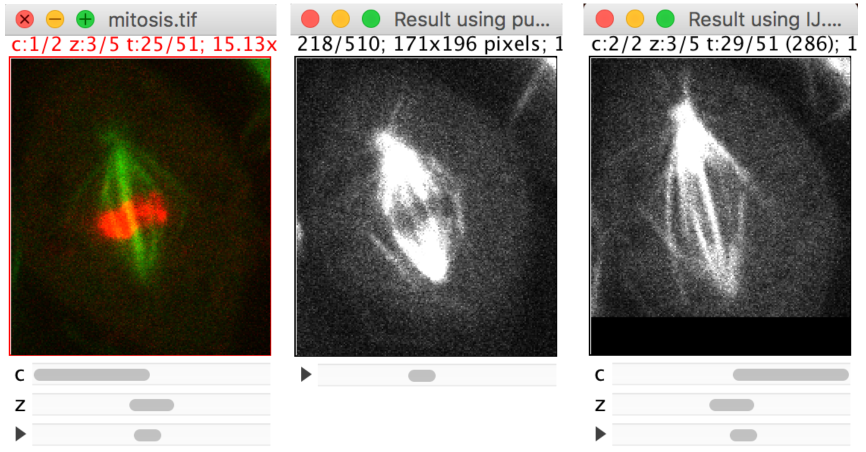


- Especially during developing/debugging showing images is useful
  - if you have an `ij` variable

```
ij.ui().show(testImg);
```

  - if not

```
ImageJFunctions.show(testImg);
```


mitosis.tif
c:1/2 z:3/5 t:25/51; 15.13x
Result using pu...
218/510; 171x196 pixels; 1
Result using IJ....
c:2/2 z:3/5 t:29/51 (286); 1
c
z
c
z

  - if everything is falling apart (e.g. if you want to display an ROI on top of the image, see exercise 2)

```
ImagePlus imp = ImageJFunctions.wrap(testImg, "testImg");
imp.show();
```

Max Planck Institute of Molecular Cell Biology and Genetics

ImageJ legacy

Intermediate summary

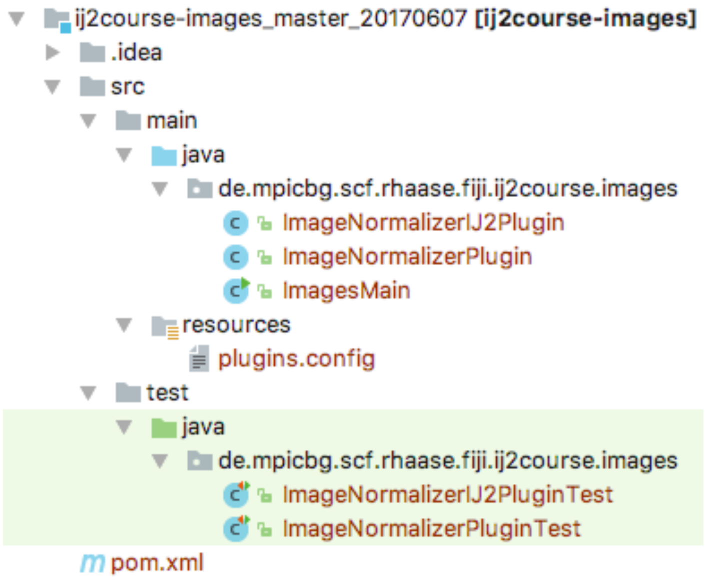
- We know now
    - How to run the ImageJ UI from within the IDE
    - How to run a plugin the ImageJ2 and the ImageJ1 way
    - How to convert ImagePlus to Img and back
    - How to show images

CBG Max Planck Institute of Molecular Cell Biology and Genetics

ImageJ legacy

Exercise 1

- Clone the repository
  https://github.com/mpicbg-scicomp/ij2course-images
- Inspect the code
  - Update `ImagesMain.main`
    - call `ImageNormaliserIJ2Plugin` instead of `ImageNormaliserPlugin`
    - Use `IJ.run();`
    - Use `ij.command.run();`
    - Use `ij.ui().show();`
    - Use `ImageJFunctions.show();`
  - Update `ImageNormaliserIJ2PluginTest`
    - write code to test the new plugin
    - be inspired by the tests in `ImageNormaliserPluginTest`
  - Optional: Write a test which proves that the output of both Plugins is equal!


```
▼ ij2course-images_master_20170607 [ij2course-images]
  ▶ .idea
  ▼ src
    ▼ main
      ▼ java
        ▼ de.mpicbg.scf.rhaase.fiji.ij2course.images
            ImageNormalizerIJ2Plugin
            ImageNormalizerPlugin
            ImagesMain
      ▼ resources
          plugins.config
    ▼ test
      ▼ java
        ▼ de.mpicbg.scf.rhaase.fiji.ij2course.images
            ImageNormalizerIJ2PluginTest
            ImageNormalizerPluginTest
  pom.xml
```

ImageJ legacy

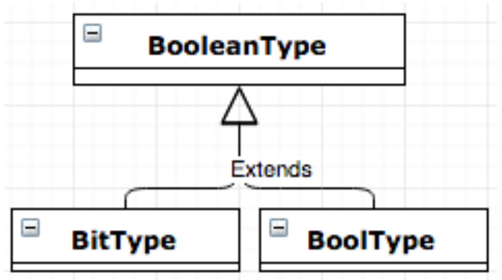
Regions - Visualisation

- What are regions in ImageJ1?

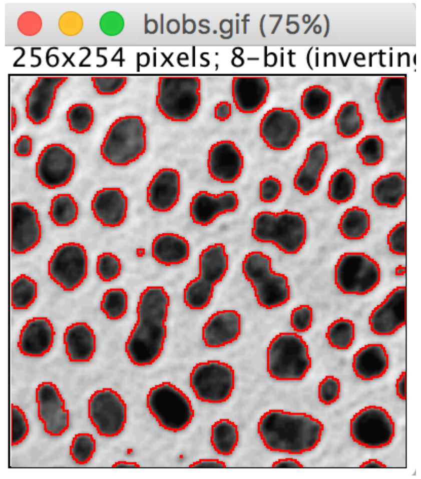
```
Roi, ShapeRoi, OvalRoi, TextRoi, PolygonRoi,...
```

- What are regions in ImageJ2?

```
RandomAccessibleInterval<B extends BooleanType<B>>
RealRandomAccessibleRealInterval<B extends BooleanType<B>>
```

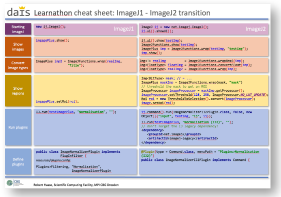


- Visualisation of ImageJ2-ROIs in ImageJ2 is not fully implemented yet. We need to convert the data to ImageJ1 in order to visualise it.



ImageJ legacy

How to visualise IJ2 Regions in ImageJ1


daIs Learnathon cheat sheet: ImageJ1 - ImageJ2 transition

- We start with an image and convert it to Img<T>

```
ImagePlus input = IJ.openImage("src/resources/blobs.gif");
Img<T> inputImg = ImageJFunctions.wrapReal(input);
```

- Then, we apply a threshold to it.

```
Img<BitType> mask = ThresholdMask.threshold(inputImg, 128);
```

- Visualisation using good old ImageJ `ThresholdToSelection` technique

```
ImageJFunctions.show(mask);
ImagePlus maskImp = IJ.getImage();

ImageProcessor imageProcessor = maskImp.getProcessor();
imageProcessor.setThreshold(128, 258, ImageProcessor.NO_LUT_UPDATE);

Roi roi = new ThresholdToSelection().convert(imageProcessor);
image.setRoi(roi);
```

ImageJ legacy

How does the way back work?

- To come from whatever kind of a ROI, we need a class implementing the interfaces
  - `RealRandomAccessibleRealInterval<BoolType>`
  - `Contains<RealLocalizable>`

```
public class RoiRealRandomAccessibleRealInterval implements
RealRandomAccessibleRealInterval<BoolType>, Contains<RealLocalizable> {
    Roi roi;
    RealInterval boundingBox;

    public RoiRealRandomAccessibleRealInterval(Roi roi) {
```

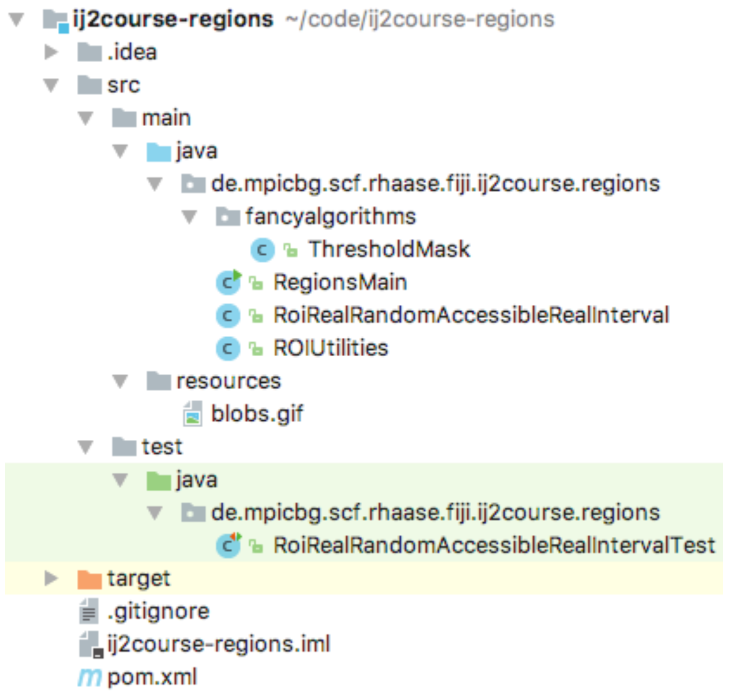
- Then, we can use our Region

```
RoiRealRandomAccessibleRealInterval rrrari =
                                         new RoiRealRandomAccessibleRealInterval(roi);

RandomAccessibleInterval<BoolType> binaryImage = ROIUtilities.raster(rrrari);
ImageJFunctions.show(binaryImage);
```

CBG Max Planck Institute of Molecular Cell Biology and Genetics

ImageJ legacy

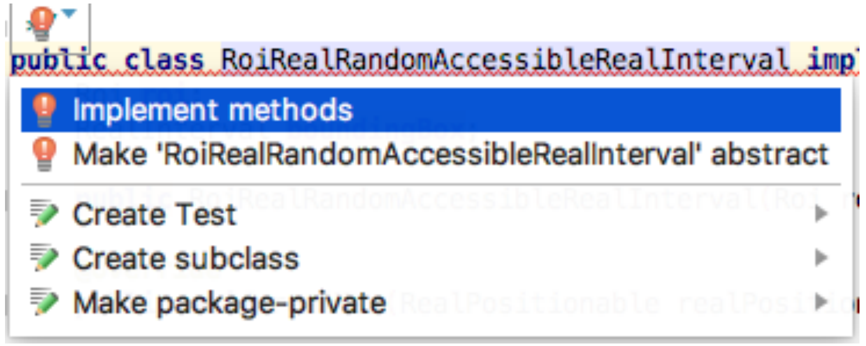
Exercise 2



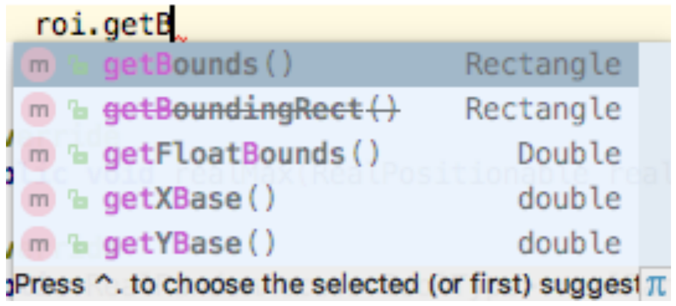
```
▼ ij2course-regions ~/code/ij2course-regions
  ▶ .idea
  ▼ src
    ▼ main
      ▼ java
        ▼ de.mpicbg.scf.rhaase.fiji.ij2course.regions
          ▼ fancyalgorithms
              ThresholdMask
            RegionsMain
            RoiRealRandomAccessibleRealInterval
            ROIUtilities
      ▼ resources
          blobs.gif
    ▼ test
      ▼ java
        ▼ de.mpicbg.scf.rhaase.fiji.ij2course.regions
            RoiRealRandomAccessibleRealIntervalTest
  ▶ target
    .gitignore
    ij2course-regions.iml
    pom.xml
```

- Clone the repository
  https://github.com/mpicbg-scicomp/ij2course-regions
- `RegionsMain.main` contains everything you need already to go from IJ2 to IJ1!

- Complete `RoiRealRandomAccessibleRealInterval`
- Let the IDE do the boring part!



```
public class RoiRealRandomAccessibleRealInterval imp
```



```
roi.getB
```

getBounds() Rectangle
~~getBoundingRect()~~ Rectangle
getFloatBounds() Double
getXBase() double
getYBase() double
Press ^. to choose the selected (or first) suggest

- After you are done, check if the *Test* is passed!

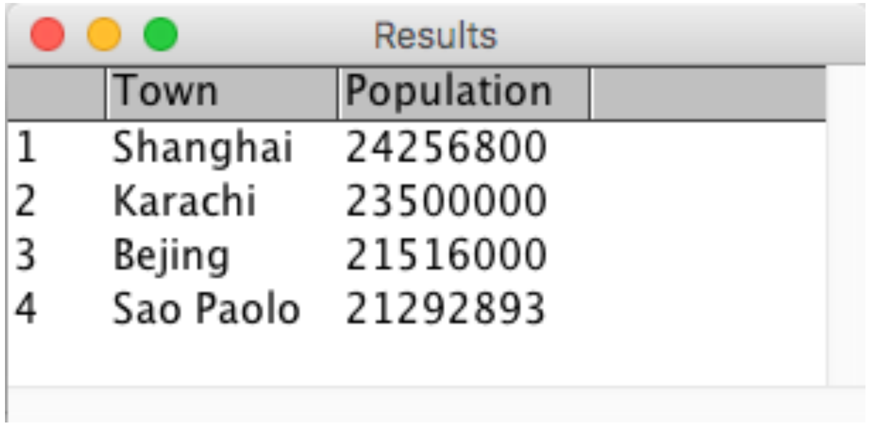
CBG
Max Planck Institute of Molecular Cell Biology and Genetics

ImageJ legacy

Tables

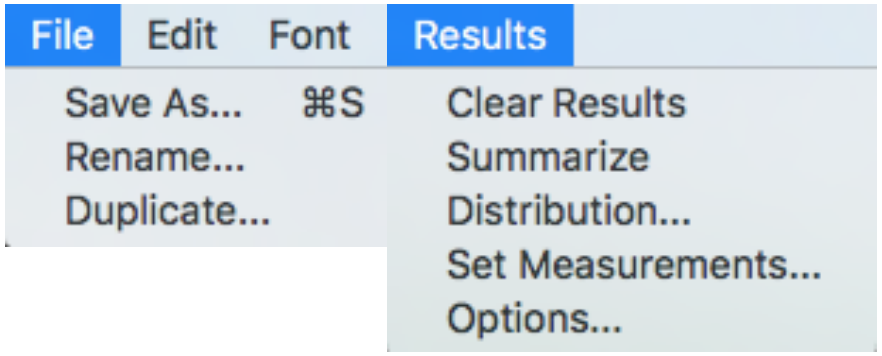
- For ImageJ2, new tables are coming...

- ImageJ1

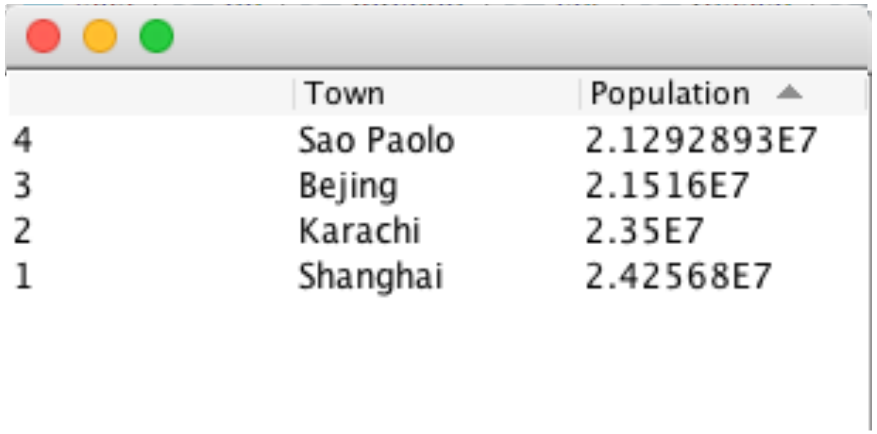


Results

	Town	Population
1	Shanghai	24256800
2	Karachi	23500000
3	Bejing	21516000
4	Sao Paolo	21292893



- ImageJ2

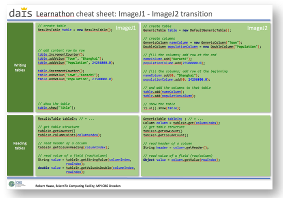


	Town	Population ▲
4	Sao Paolo	2.1292893E7
3	Bejing	2.1516E7
2	Karachi	2.35E7
1	Shanghai	2.42568E7

CBG
Max Planck Institute
of Molecular Cell Biology
and Genetics

ImageJ legacy

Write tables



- ImageJ1

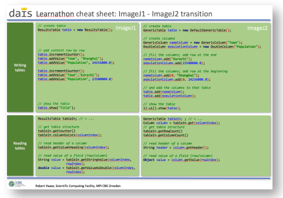
```
// create table
ResultsTable table = new ResultsTable();




// add content row by row
table.incrementCounter();
table.addValue("Town", "Shanghai");
table.addValue("Population", 24256800.0);

table.incrementCounter();
table.addValue("Town","Karachi");
table.addValue("Population", 23500000.0);





// show the table
table.show("Title");
```

- ImageJ2

```
// create table
GenericTable table = new DefaultGenericTable();

// create columns
GenericColumn nameColumn = new GenericColumn("Town");
DoubleColumn populationColumn = new DoubleColumn("Population");

// fill the columns; add row at the end
nameColumn.add("Karachi");
populationColumn.add(23500000.0);

// fill the columns; add row at the beginning
nameColumn.add(0, ”Shanghai");
populationColumn.add(0, 24256800.0);

// and add the columns to that table
table.add(nameColumn);
table.add(populationColumn);

// show the table
ij.ui().show(table);
```

ImageJ legacy

Read tables



daIs Learnathon cheat sheet: ImageJ1 - ImageJ2 transition

Writing tables

ImageJ1

```
// create table
ResultsTable table = new ResultsTable();

// add content row by row
table.incrementCounter();
table.addValue("Town", "Shanghai");
table.addValue("Population", 24256800.0);

table.incrementCounter();
table.addValue("Town","Karachi");
table.addValue("Population", 23500000.0)

// show the table
table.show("Title");
```

ImageJ2

```
// create table
GenericTable table = new DefaultGenericTable();

// create columns
GenericColumn nameColumn = new GenericColumn("Town");
DoubleColumn populationColumn = new DoubleColumn("Population");

// fill the columns; add row at the end
nameColumn.add("Karachi");
populationColumn.add(23500000.0);

// fill the columns; add row at the beginning
nameColumn.add(0, "Shanghai");
populationColumn.add(0, 24256800.0);

// and add the columns to that table
table.add(nameColumn);
table.add(populationColumn);

// show the table
ij.ui().show(table);
```

Reading tables

```
ResultsTable tableIn; // = ...

// get table structure
tableIn.getCounter()
tableIn.columnExists(columnIndex);

// read header of a column
tableIn.getColumnHeading(columnIndex);

// read value of a field (row/column)
String value = tableIn.getStringValue(columnIndex,
               rowIndex);
double value = tableIn.getValueAsDouble(columnIndex,
               rowIndex);
```

```
GenericTable tableIn; ; // = ...
Column column = tableIn.get(columnIndex);
// get table structure
tableIn.getRowCount()
tableIn.getColumnCount()

// read header of a column
String header = column.getHeader();

// read value of a field (row/column)
Object value = column.getValue(rowIndex);
```

Robert Haase, Scientific Computing Facility, MPI CBG Dresden

- ImageJ1

```
ResultsTable tableIn; // = ...


tableIn.getCounter()
tableIn.columnExists(columnIndex);

// read header of a column
tableIn.getColumnHeading(columnIndex);

// read value of a field (row/column)
String value = tableIn.getStringValue(columnIndex, rowIndex);
double value = tableIn.getValueAsDouble(columnIndex, rowIndex);
```

- ImageJ2

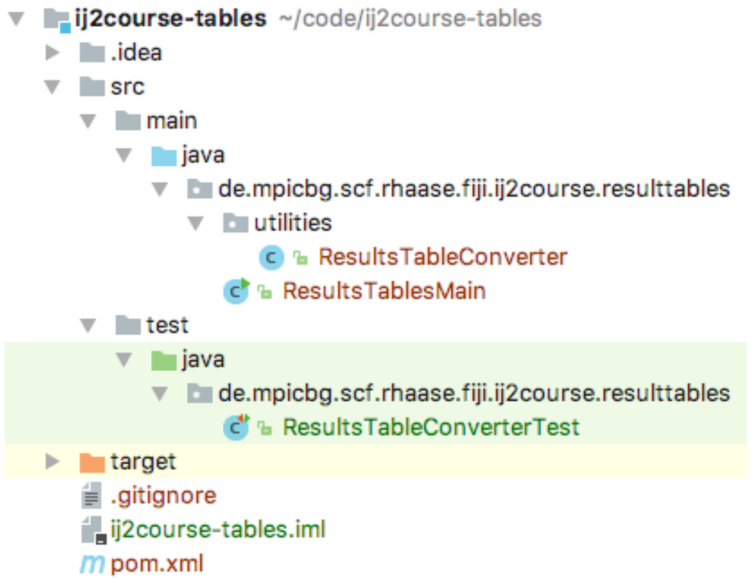
```
GenericTable tableIn; ; // = ...
Column column = tableIn.get(columnIndex);

tableIn.getRowCount()
tableIn.getColumnCount()

// read header of a column
String header = column.getHeader();

// read value of a field (row/column)
Object value = column.getValue(rowIndex);
```

Exercise 3

- Clone the repository

  https://github.com/mpicbg-scicomp/ij2course-tables.git
- `ResultsTableConverter` contains an IJ1-IJ2 converter
- Add an IJ2-IJ1 converter!

- Afterwards, check if the `ResultsTableConverterTest` is passed!

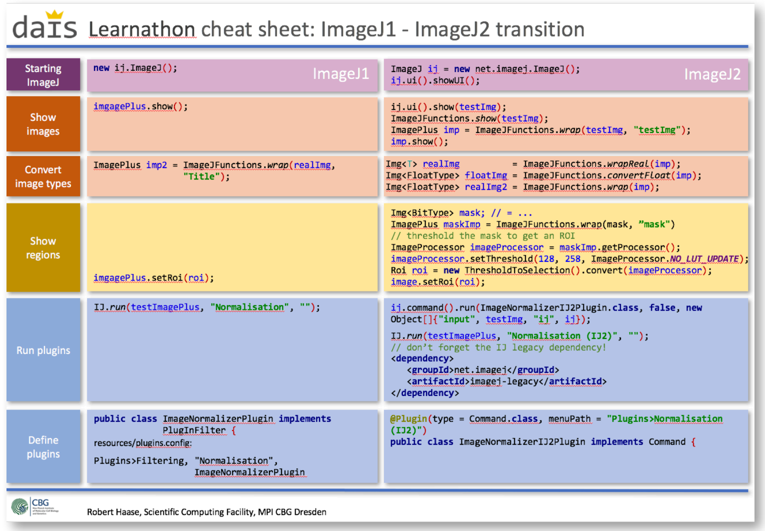

```
▼ ij2course-tables ~/code/ij2course-tables
  ▶ .idea
  ▼ src
    ▼ main
      ▼ java
        ▼ de.mpicbg.scf.rhaase.fiji.ij2course.resulttables
          ▼ utilities
              ResultsTableConverter
            ResultsTablesMain
    ▼ test
      ▼ java
        ▼ de.mpicbg.scf.rhaase.fiji.ij2course.resulttables
            ResultsTableConverterTest
  ▶ target
    .gitignore
    ij2course-tables.iml
    pom.xml
```

CBG Max Planck Institute of Molecular Cell Biology and Genetics

ImageJ legacy

Summary

- You just learned how to
  - run ImageJ2 and ImageJ1
  - show images
  - convert between image types
  - show/convert ROIs
  - convert tables



daIs Learnathon cheat sheet: ImageJ1 - ImageJ2 transition

Starting ImageJ

ImageJ1

```java
new ij.ImageJ();
```

ImageJ2

```java
ImageJ ij = new net.imagej.ImageJ();
ij.ui().showUI();
```

Show images

```java
imgagePlus.show();
```

```java
ij.ui().show(testImg);
ImageJFunctions.show(testImg);
ImagePlus imp = ImageJFunctions.wrap(testImg, "testImg");
imp.show();
```

Convert image types

```java
ImagePlus imp2 = ImageJFunctions.wrap(realImg,
                 "Title");
```

```java
Img<T> realImg           = ImageJFunctions.wrapReal(imp);
Img<FloatType> floatImg = ImageJFunctions.convertFloat(imp);
Img<FloatType> realImg2 = ImageJFunctions.wrap(imp);
```

Show regions

```java
imgagePlus.setRoi(roi);
```

```java
Img<BitType> mask; // = ...
ImagePlus maskImp = ImageJFunctions.wrap(mask, ”mask")
// threshold the mask to get an ROI
ImageProcessor imageProcessor = maskImp.getProcessor();
imageProcessor.setThreshold(128, 258, ImageProcessor.NO_LUT_UPDATE);
Roi roi = new ThresholdToSelection().convert(imageProcessor);
image.setRoi(roi);
```

Run plugins

```java
IJ.run(testImagePlus, "Normalisation", "");
```

```java
ij.command().run(ImageNormalizerIJ2Plugin.class, false, new
Object[]{"input", testImg, "ij", ij});

IJ.run(testImagePlus, "Normalisation (IJ2)", "");
// don’t forget the IJ legacy dependency!
<dependency>
   <groupId>net.imagej</groupId>
   <artifactId>imagej-legacy</artifactId>
</dependency>
```

Define plugins

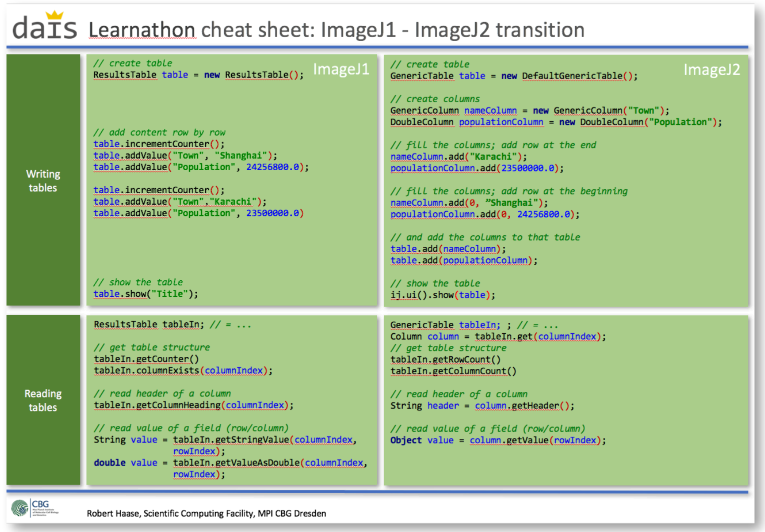
```java
public class ImageNormalizerPlugin implements
             PlugInFilter {
```

resources/plugins.config:

```
Plugins>Filtering, "Normalisation",
                   ImageNormalizerPlugin
```

```java
@Plugin(type = Command.class, menuPath = "Plugins>Normalisation
(IJ2)")
public class ImageNormalizerIJ2Plugin implements Command {
```

Robert Haase, Scientific Computing Facility, MPI CBG Dresden



daIs Learnathon cheat sheet: ImageJ1 - ImageJ2 transition

Writing tables

ImageJ1

```java
// create table
ResultsTable table = new ResultsTable();


// add content row by row
table.incrementCounter();
table.addValue("Town", "Shanghai");
table.addValue("Population", 24256800.0);

table.incrementCounter();
table.addValue("Town","Karachi");
table.addValue("Population", 23500000.0)


// show the table
table.show("Title");
```

ImageJ2

```java
// create table
GenericTable table = new DefaultGenericTable();

// create columns
GenericColumn nameColumn = new GenericColumn("Town");
DoubleColumn populationColumn = new DoubleColumn("Population");

// fill the columns; add row at the end
nameColumn.add("Karachi");
populationColumn.add(23500000.0);

// fill the columns; add row at the beginning
nameColumn.add(0, ”Shanghai");
populationColumn.add(0, 24256800.0);

// and add the columns to that table
table.add(nameColumn);
table.add(populationColumn);

// show the table
ij.ui().show(table);
```

Reading tables

```java
ResultsTable tableIn; // = ...

// get table structure
tableIn.getCounter()
tableIn.columnExists(columnIndex);

// read header of a column
tableIn.getColumnHeading(columnIndex);

// read value of a field (row/column)
String value = tableIn.getStringValue(columnIndex,
               rowIndex);
double value = tableIn.getValueAsDouble(columnIndex,
               rowIndex);
```

```java
GenericTable tableIn; ; // = ...
Column column = tableIn.get(columnIndex);
// get table structure
tableIn.getRowCount()
tableIn.getColumnCount()

// read header of a column
String header = column.getHeader();

// read value of a field (row/column)
Object value = column.getValue(rowIndex);
```

Robert Haase, Scientific Computing Facility, MPI CBG Dresden