Advanced Programming with ImgLib2

ImageJ User and Developer Conference

Tobias Pietzsch, Stephan Preibisch, Stephan Saalfeld

Max Planck Institute of Molecular Cell Biology and Genetics, Dresden

October 25, 2012

http://imglib2.net

http://imglib2.net/

ImgLib2

Library for *n*-dimensional data representation and manipulation.

Write code that is independent of image dimensionality, data type, and data storage strategy.

Design goals:

- Re-usability, avoid code duplication.
- Decouple algorithm development and data management.
- High-level programming interface.
- High performance.
- Extensibility (adding algorithms, pixel types, storage strategies).
- Adaptability (to existing data structures).

ImgLib2

Library for *n*-dimensional data representation and manipulation.

Write code that is independent of image dimensionality, data type, and data storage strategy.

Design goals:

- Re-usability, avoid code duplication.
- Decouple algorithm development and data management.
- High-level programming interface.
- High performance.
- Extensibility (adding algorithms, pixel types, storage strategies).
- Adaptability (to existing data structures).

ImgLib2

Library for *n*-dimensional data representation and manipulation.

Write code that is independent of image dimensionality, data type, and data storage strategy.

Design goals:

- Re-usability, avoid code duplication.
- Decouple algorithm development and data management.
- High-level programming interface.
- High performance.
- Extensibility (adding algorithms, pixel types, storage strategies).
- Adaptability (to existing data structures).

ImgLib2

Library for *n*-dimensional data representation and manipulation.

Write code that is independent of image dimensionality, data type, and data storage strategy.

Design goals:

- Re-usability, avoid code duplication.
- Decouple algorithm development and data management.
- High-level programming interface.
- High performance.
- Extensibility (adding algorithms, pixel types, storage strategies).
- Adaptability (to existing data structures).

About this Workshop

Goals:

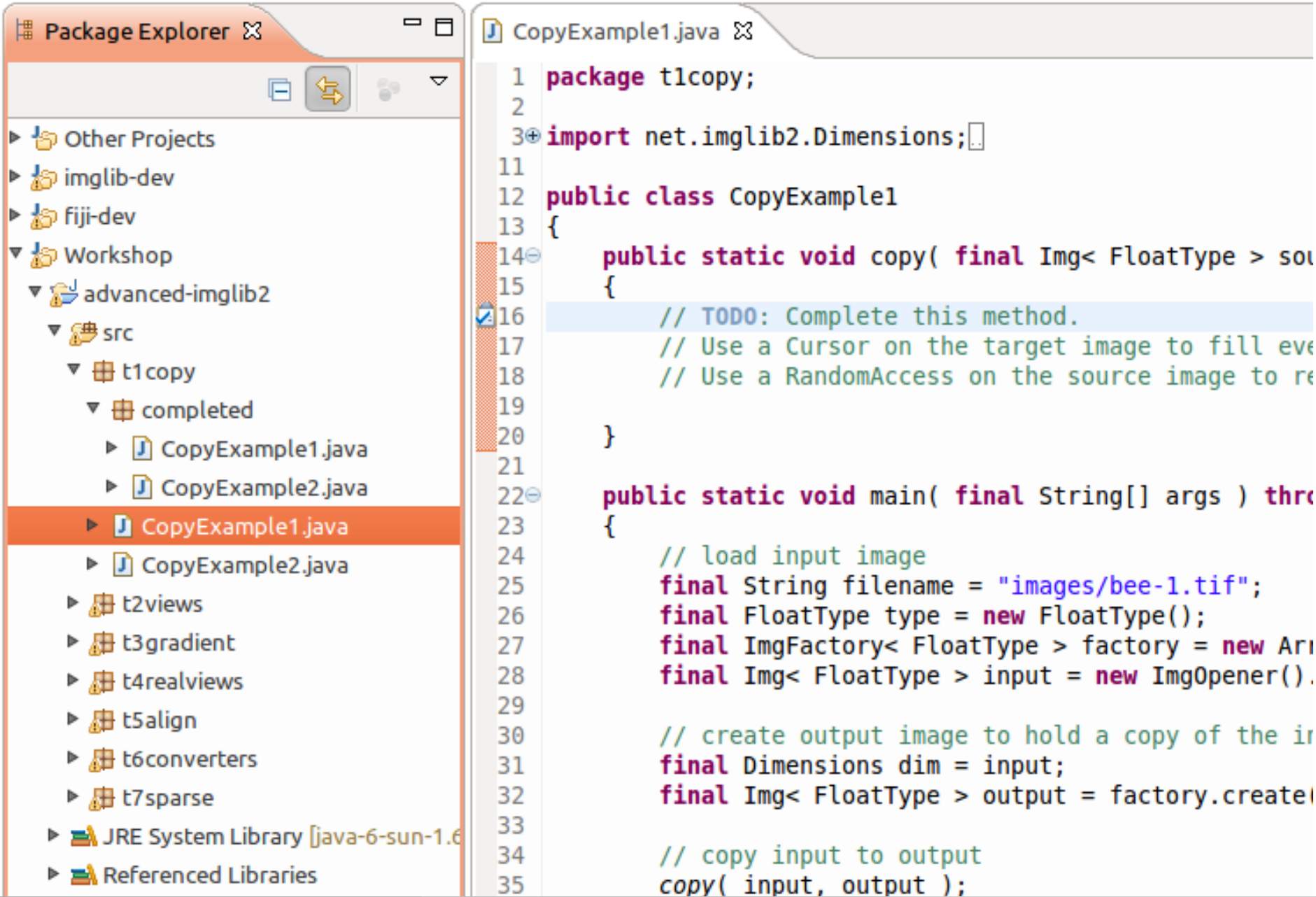
- Write type- and dimensionality-independent code.
- *Type* hierarchy (fragments).
- *Accessibles* hierarchy (fragments).
- Use *Views* for on-the-fly coordinate transforms.
- Use *Converters* for on-the-fly value transforms.
- Interface to ImageJ(1).

About this Workshop

Format:

- Fill in missing parts of example code.
- Completed examples provided, too.

About this Workshop



Tutorial 1 - Writing Generic Code

ImgLib2

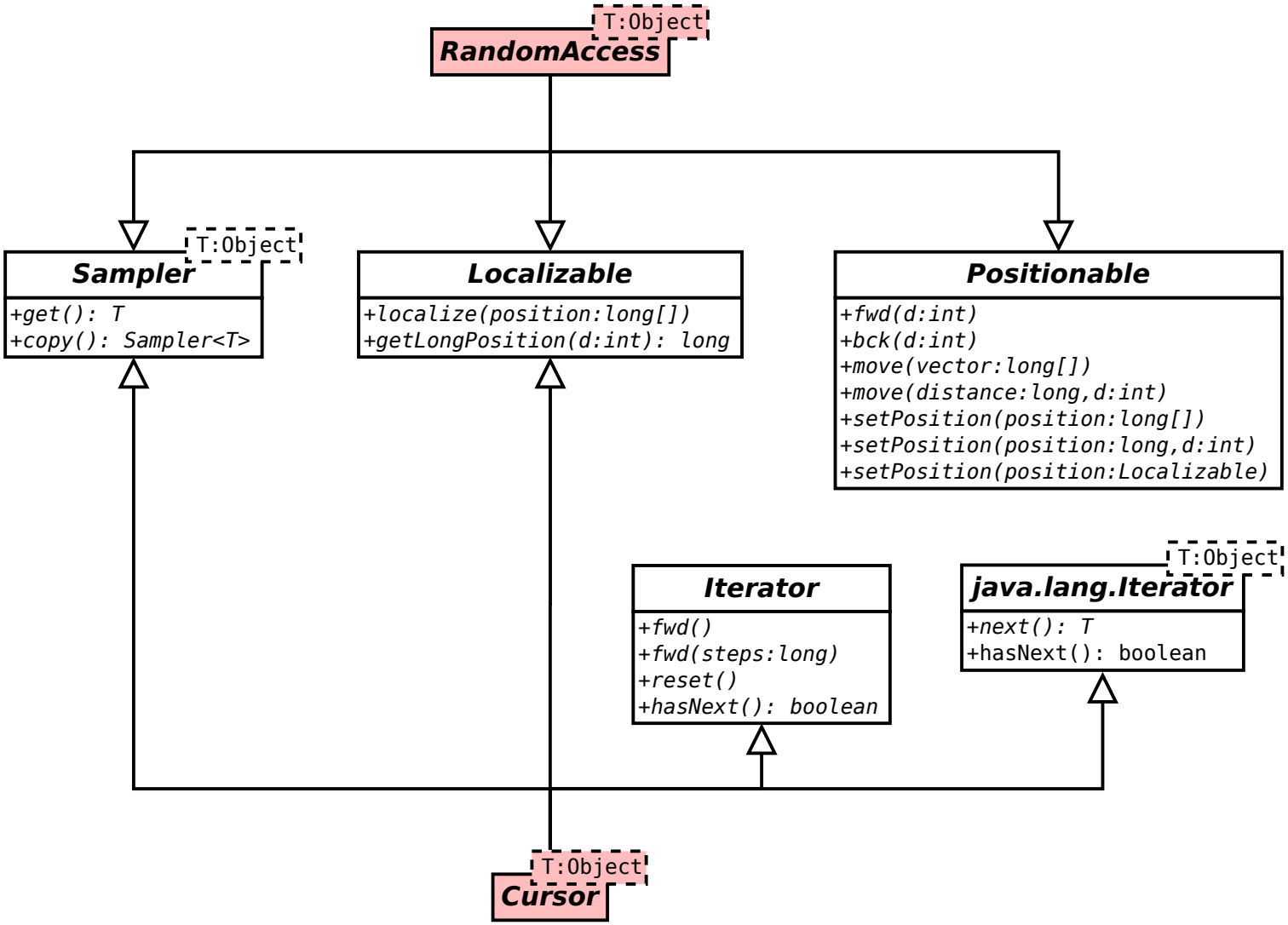
CopyExample1

1. Open and run `CopyExample1`.
2. Complete the `copy()` method
   (You should see a bee).

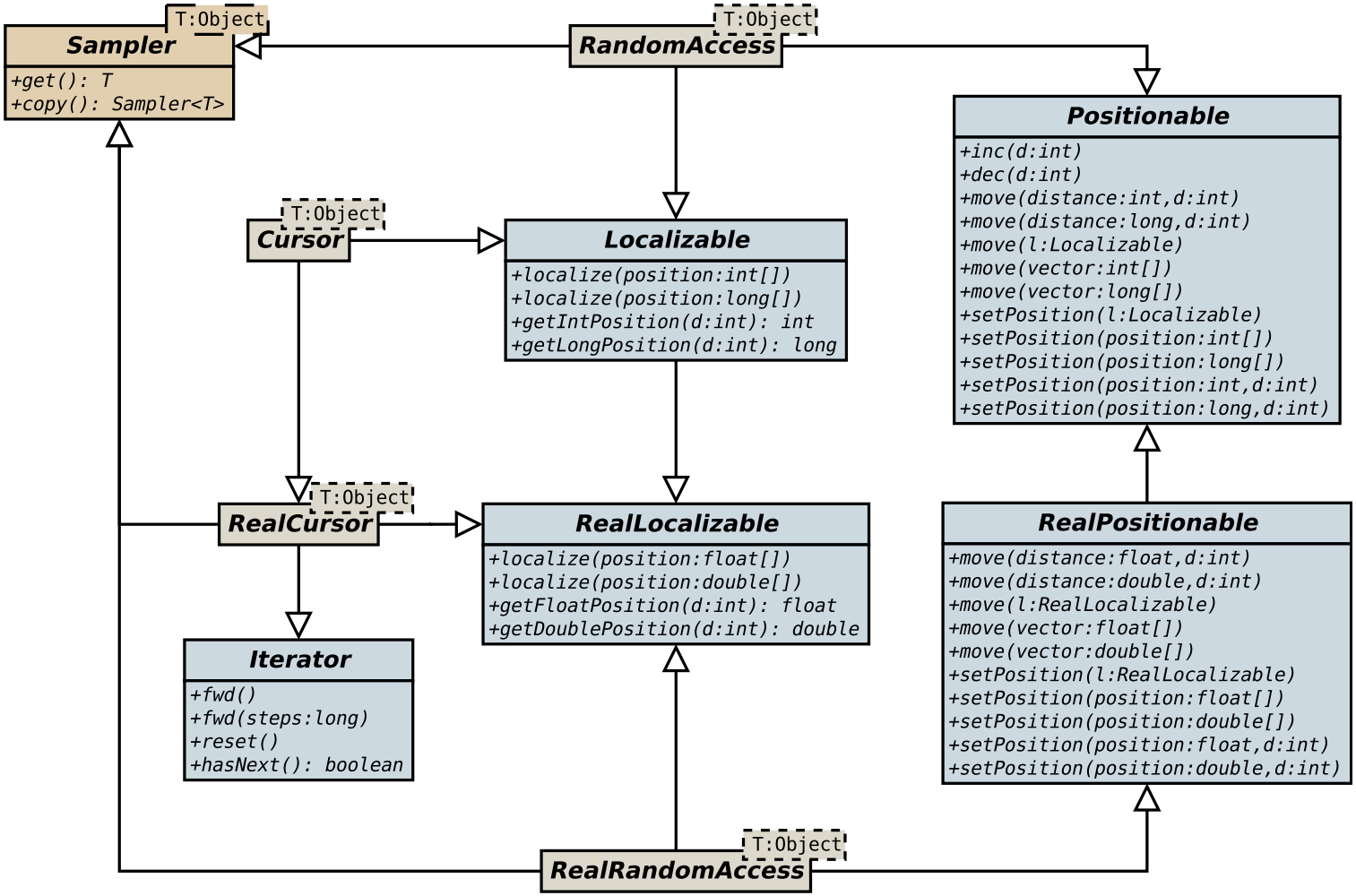
CopyExample2

- Generalize the `copy()` method.

Accessors (simplified):



Accessors:


Sampler
T:Object
+get(): T
+copy(): Sampler<T>
RandomAccess
T:Object
Positionable
+inc(d:int)
+dec(d:int)
+move(distance:int,d:int)
+move(distance:long,d:int)
+move(l:Localizable)
+move(vector:int[])
+move(vector:long[])
+setPosition(l:Localizable)
+setPosition(position:int[])
+setPosition(position:long[])
+setPosition(position:int,d:int)
+setPosition(position:long,d:int)
Cursor
T:Object
Localizable
+localize(position:int[])
+localize(position:long[])
+getIntPosition(d:int): int
+getLongPosition(d:int): long
RealCursor
T:Object
RealLocalizable
+localize(position:float[])
+localize(position:double[])
+getFloatPosition(d:int): float
+getDoublePosition(d:int): double
RealPositionable
+move(distance:float,d:int)
+move(distance:double,d:int)
+move(l:RealLocalizable)
+move(vector:float[])
+move(vector:double[])
+setPosition(l:RealLocalizable)
+setPosition(position:float[])
+setPosition(position:double[])
+setPosition(position:float,d:int)
+setPosition(position:double,d:int)
Iterator
+fwd()
+fwd(steps:long)
+reset()
+hasNext(): boolean
RealRandomAccess
T:Object

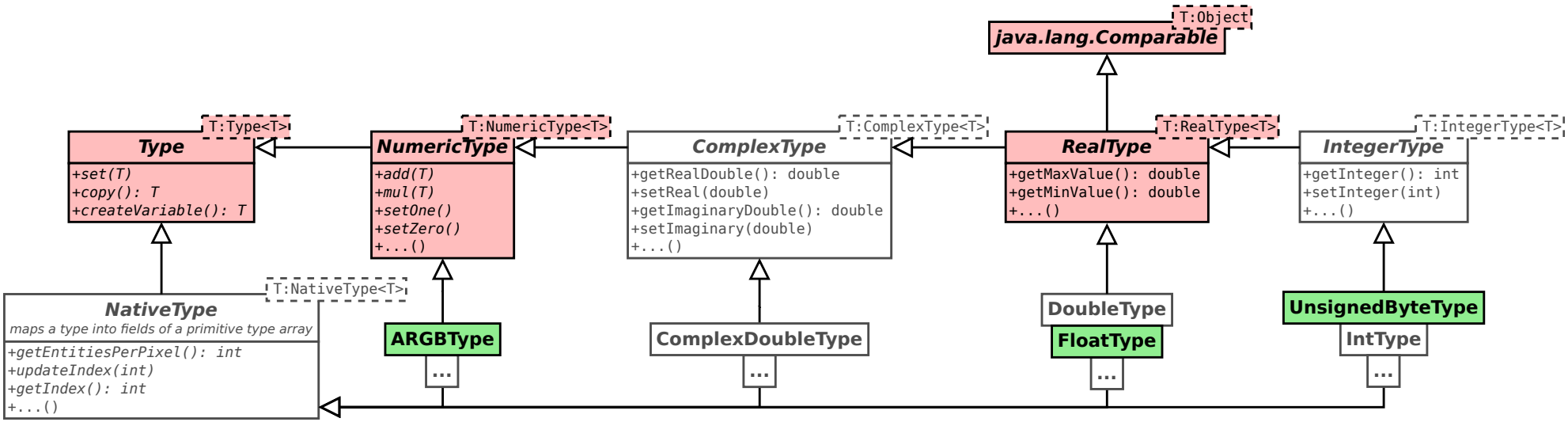
CopyExample1

1. Open and run `CopyExample1`.
2. Complete the `copy()` method
   (You should see a bee).

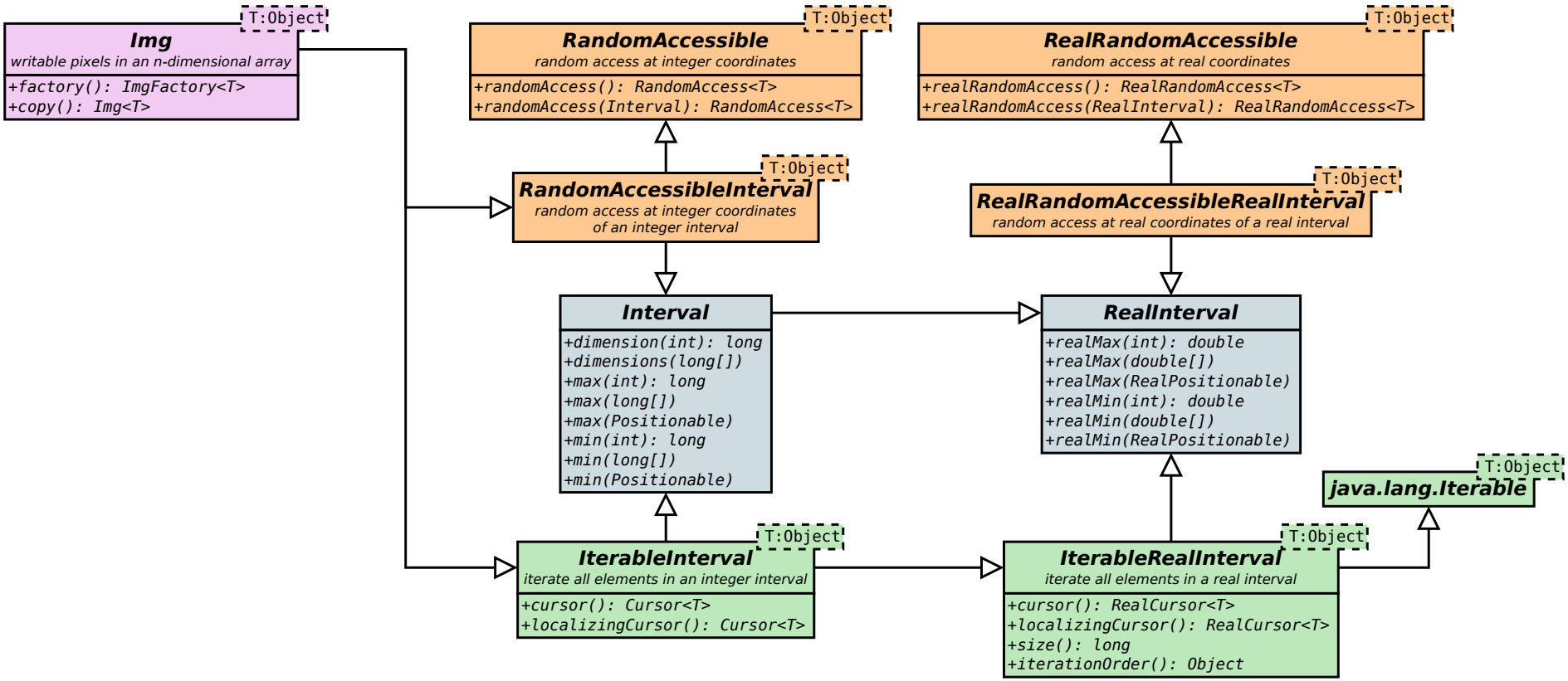
CopyExample2

- Generalize the `copy()` method.

Type Hierarchy:



Accessibles:



ViewsExample1

- Experiment with static convenience methods of the `Views` class.

ViewsExample2

- Experiment with views in 3D.

Tutorial 2 - Views

ViewsExample1

- Experiment with static convenience methods of the `Views` class.

ViewsExample2

- Experiment with views in 3D.

Tutorial 3 - Compute Gradient Using Views

GradientExample1

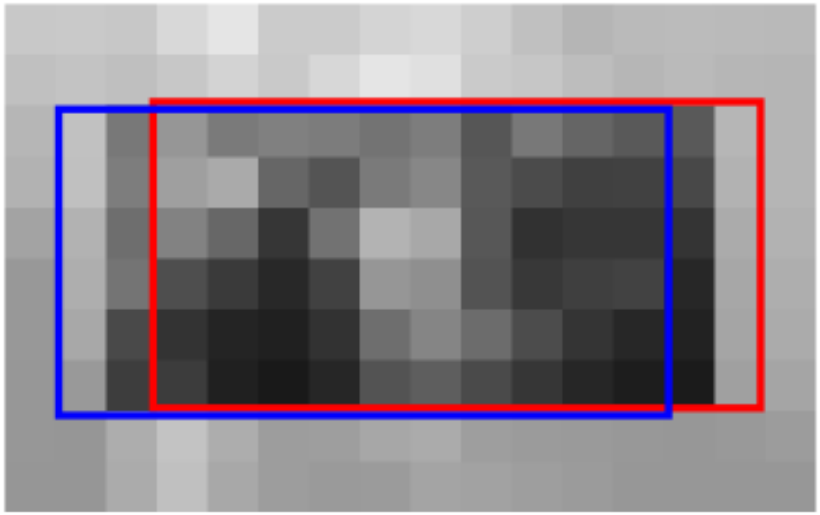
- Compute the partial derivative of a `RandomAccessibleInterval` in a given dimension.

GradientExample2

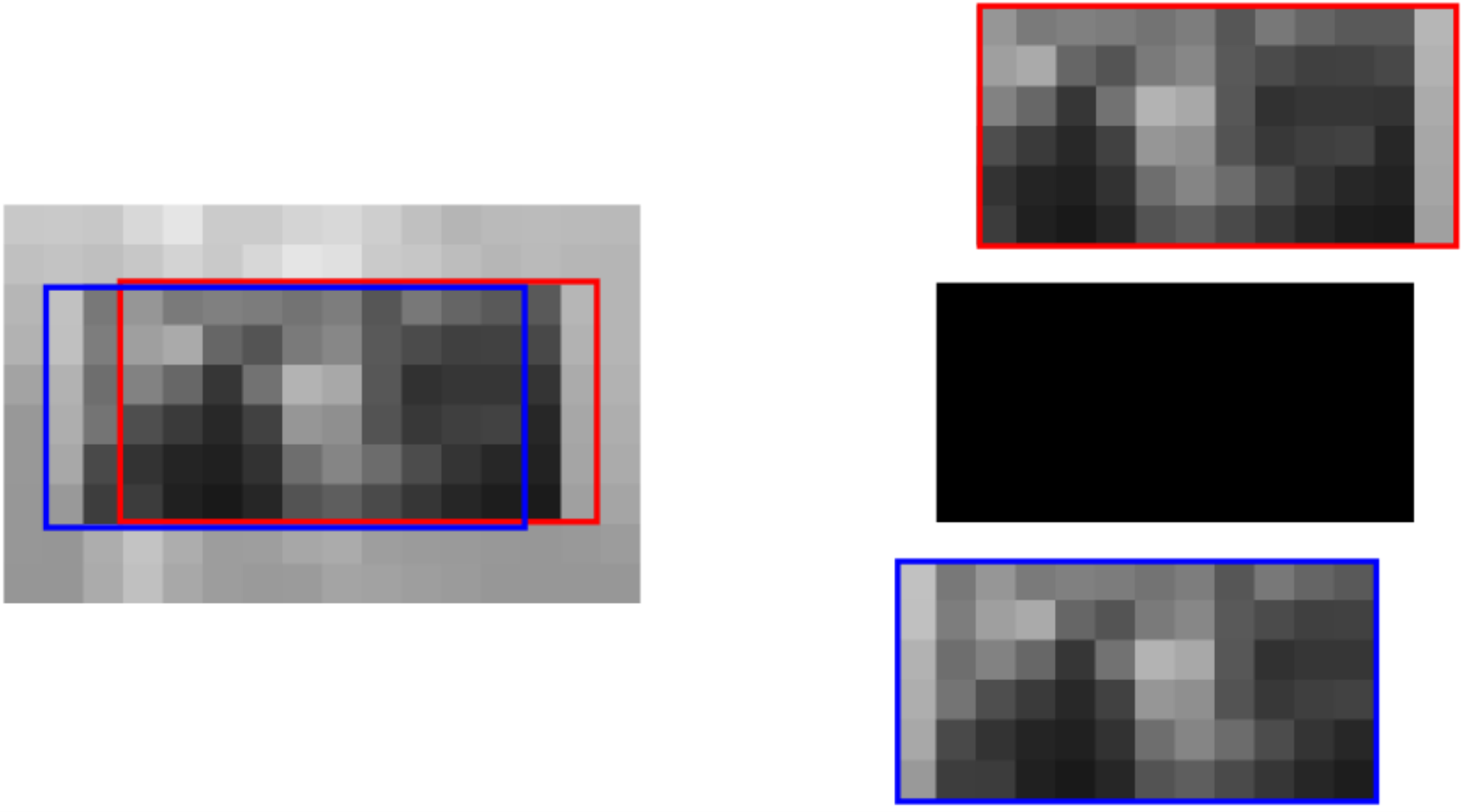
- Compute the partial derivatives of a `RandomAccessibleInterval` in every dimension.

Tutorial 3 - Compute Gradient Using Views

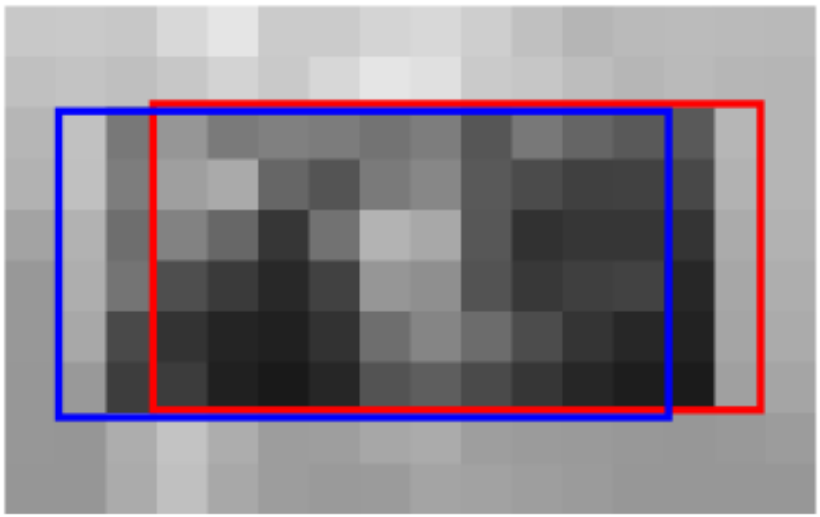
Tutorial 3 - Compute Gradient Using Views

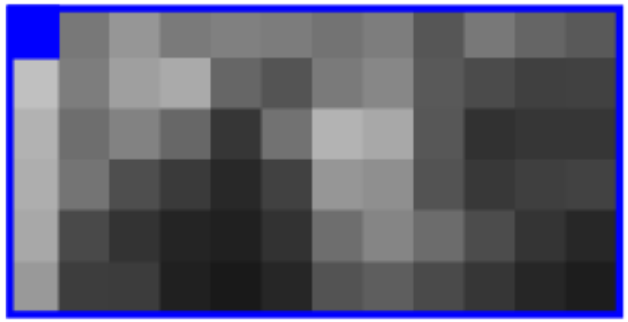


Tutorial 3 - Compute Gradient Using Views

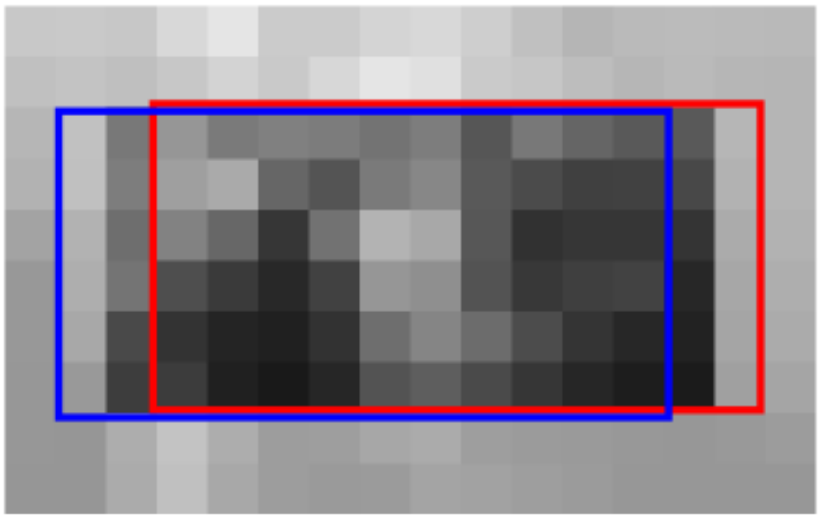


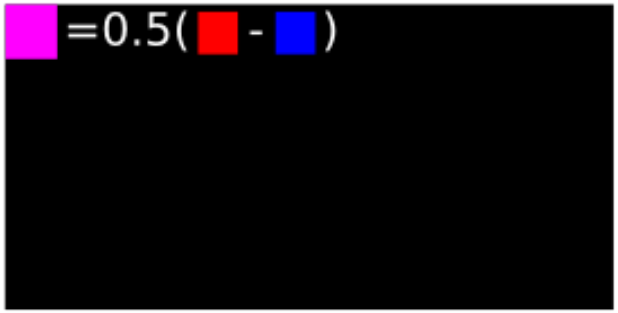
Tutorial 3 - Compute Gradient Using Views

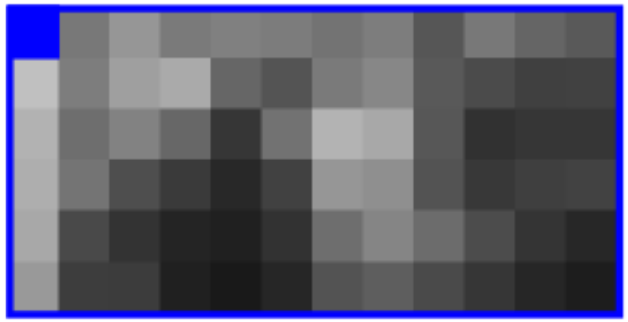




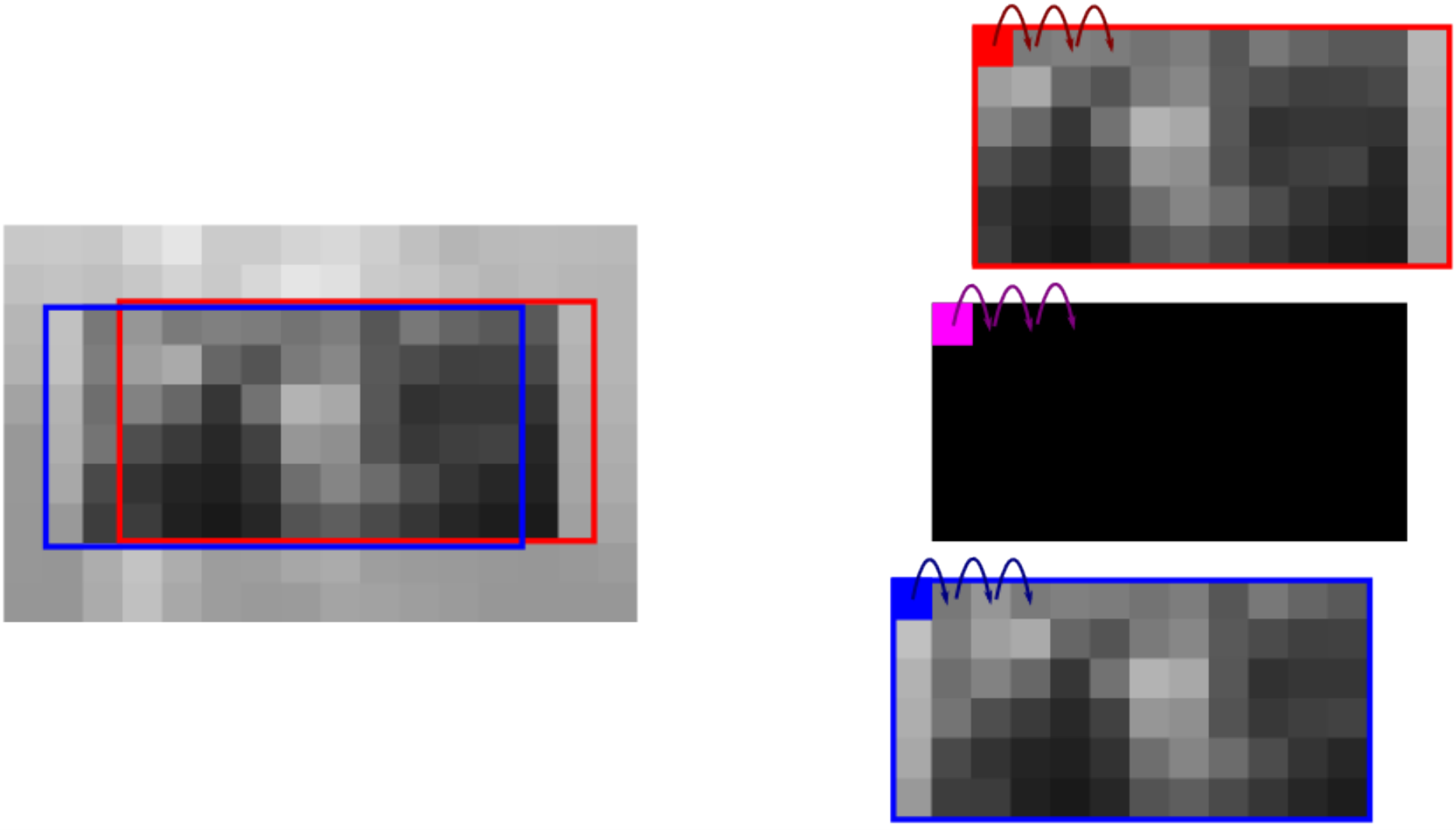
Tutorial 3 - Compute Gradient Using Views







Tutorial 3 - Compute Gradient Using Views



Tutorial 3 - Compute Gradient Using Views

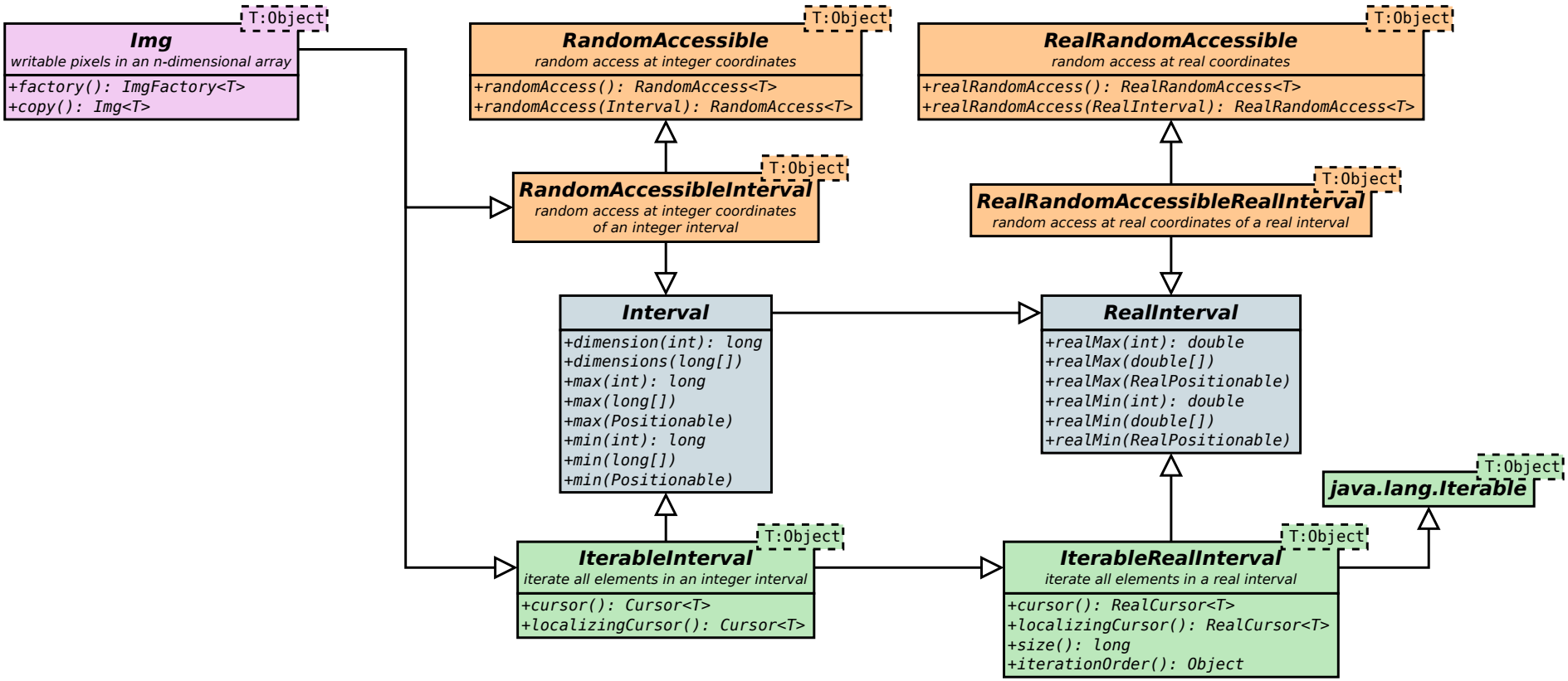
GradientExample1

- Compute the partial derivative of a `RandomAccessibleInterval` in a given dimension.

GradientExample2

- Compute the partial derivatives of a `RandomAccessibleInterval` in every dimension.

Accessibles:



Tutorial 4 - RealViews

RealViewsExample1

- Experiment with interpolation.

RealViewsExample2

1. Affine-transform an interpolated image.
2. Rasterize and crop a `RealRandomAccessible` for display.

RealViewsExample1

- Compute the pixel-wise difference between an affine-transformed source image and a target image.

RealViewsExample1

- Experiment with interpolation.

RealViewsExample2

1. Affine-transform an interpolated image.
2. Rasterize and crop a `RealRandomAccessible` for display.

RealViewsExample1

- Compute the pixel-wise difference between an affine-transformed source image and a target image.

RealViewsExample1

- Experiment with interpolation.

RealViewsExample2

1. Affine-transform an interpolated image.
2. Rasterize and crop a `RealRandomAccessible` for display.

RealViewsExample1

- Compute the pixel-wise difference between an affine-transformed source image and a target image.

Inverse Compositional Image Alignment

Baker, S., and Matthews, I., “Lucas-Kanade 20 Years On: A Unifying Framework”
International Journal of Computer Vision, 2004

- Minimize difference between *template* and *image*.
- Iteratively find best transformation for linearized problem.
- Implemented in `Align` class.
- Uses `gradients()` and `computeDifference()` we implemented earlier.

Inverse Compositional Image Alignment

Baker, S., and Matthews, I., “Lucas-Kanade 20 Years On: A Unifying Framework”
International Journal of Computer Vision, 2004

- Minimize difference between *template* and *image*.
- Iteratively find best transformation for linearized problem.
- Implemented in `Align` class.
- Uses `gradients()` and `computeDifference()` we implemented earlier.

Inverse Compositional Image Alignment

Baker, S., and Matthews, I., “Lucas-Kanade 20 Years On: A Unifying Framework”
International Journal of Computer Vision, 2004

- Minimize difference between *template* and *image*.
- Iteratively find best transformation for linearized problem.
- Implemented in `Align` class.
- Uses `gradients()` and `computeDifference()` we implemented earlier.

Inverse Compositional Image Alignment

Baker, S., and Matthews, I., “Lucas-Kanade 20 Years On: A Unifying Framework”
International Journal of Computer Vision, 2004

- Minimize difference between *template* and *image*.
- Iteratively find best transformation for linearized problem.
- Implemented in `Align` class.
- Uses `gradients()` and `computeDifference()` we implemented earlier.

Tutorial 5 - Image Alignment Plugin

ImgLib2

ShowSteepestDescentExample

ShowIterationErrorExample

AlignExample

Align_Plugin_Example1

- Wrap or convert ImageJ `ImagePlus`.
- Apply `Align` result to input image and show result.

Align_Plugin_Example2

- Choose `T` of `Img<T>` according to type of `ImagePlus`.

Tutorial 5 - Image Alignment Plugin

ShowSteepestDescentExample

ShowIterationErrorExample

AlignExample

Align_Plugin_Example1

- Wrap or convert ImageJ `ImagePlus`.
- Apply `Align` result to input image and show result.

Align_Plugin_Example2

- Choose `T` of `Img<T>` according to type of `ImagePlus`.

Tutorial 5 - Image Alignment Plugin

ShowSteepestDescentExample

ShowIterationErrorExample

AlignExample

Align_Plugin_Example1

- Wrap or convert ImageJ `ImagePlus`.
- Apply `Align` result to input image and show result.

Align_Plugin_Example2

- Choose `T` of `Img<T>` according to type of `ImagePlus`.

Tutorial 5 - Image Alignment Plugin

ShowSteepestDescentExample

ShowIterationErrorExample

AlignExample

Align_Plugin_Example1

- Wrap or convert ImageJ `ImagePlus`.
- Apply `Align` result to input image and show result.

Align_Plugin_Example2

- Choose `T` of `Img<T>` according to type of `ImagePlus`.

Tutorial 5 - Image Alignment Plugin

ShowSteepestDescentExample

ShowIterationErrorExample

AlignExample

Align_Plugin_Example1

- Wrap or convert ImageJ `ImagePlus`.
- Apply `Align` result to input image and show result.

Align_Plugin_Example2

- Choose `T` of `Img<T>` according to type of `ImagePlus`.

Tutorial 6 - Converters

ConverterExample1

- Apply converters to transform pixel values on the fly.

ConvertersExample2

- Implement and apply a `Converter` that inverts pixel values.

Tutorial 6 - Converters

ConverterExample1

- Apply converters to transform pixel values on the fly.

ConvertersExample2

- Implement and apply a `Converter` that inverts pixel values.

Tutorial 7 - Sparse data

SparseExample1

1. Construct a kD-tree on a sparse list of samples (coordinates and values).
2. Use nearest neighbor search to interpolate a continuous `RealRandomAccessible`.
3. Rasterize, crop, and display the result.

http://imglib2.net/

Pietzsch, T., Preibisch, S., Tomančák, P., and Saalfeld, S., “ImgLib2 – generic image processing in Java,” *Bioinformatics* (in press).