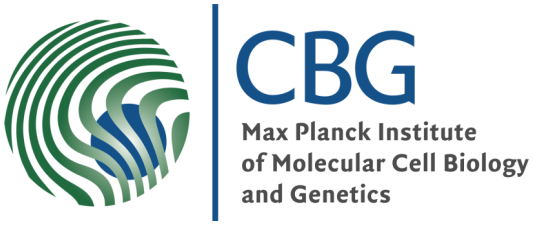
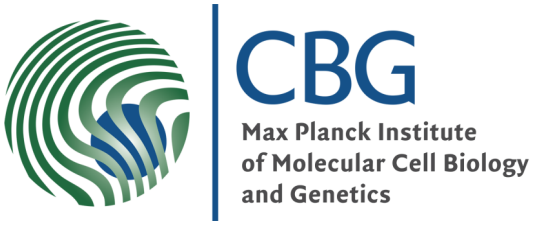


Introduction to ImgLib2

Stephan Preibisch, Tobias Pietzsch, Stephan Saalfeld

Albert Einstein College of Medicine, New York
MPI-CBG, Dresden, Germany
HHMI Janelia Farm Research Campus, Virginia



Special thanks to …

- Tobias Pietzsch (MPI-CBG)
- Stephan Saalfeld (MPI-CBG)

- Pavel Tomancak (MPI-CBG)
- Gene Myers (MPI-CBG)
- Rob Singer
  (Einstein College & Janelia Farm)

- ImageJ2 crew
  - Johannes Schindelin
  - Curtis Rueden
  - Barry DeZonia
  - Kevin Eliceiri
- Albert Cardona
- KNIME guys
  - Christian Dietz
  - Martin Horn

OXFORD JOURNALS CONTACT US MY BASKET MY ACCOUNT

Bioinformatics

ABOUT THIS JOURNAL CONTACT THIS JOURNAL SUBSCRIPTIONS CURRENT ISSUE ARCHIVE SEARCH

Oxford Journals › Life Sciences & Mathematics & Physical Sciences › Bioinformatics › Advance Access › 10.1093/bioinformatics/bts543

ImgLib2 – Generic Image Processing in Java

Tobias Pietzsch,[1,*] Stephan Preibisch,[1,2,*] Pavel Tomančák [1] and Stephan Saalfeld [1,*,†]

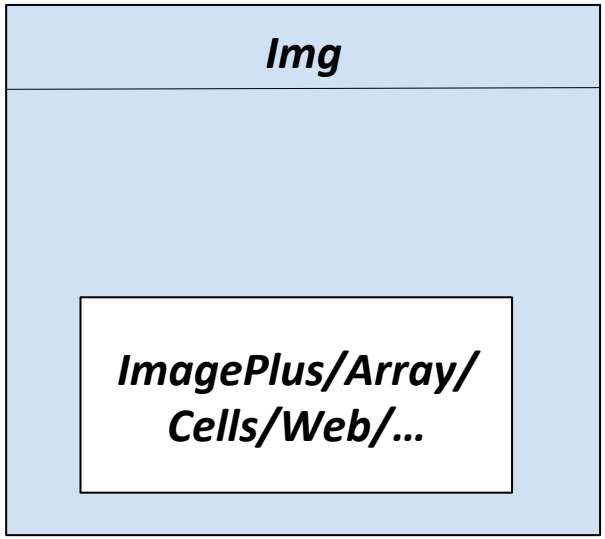
[1]Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany,
[2]Janelia Farm Research Campus, Howard Hughes Medical Institute, Ashburn, Virginia, USA

Associate Editor: Dr. Jonathan Wren

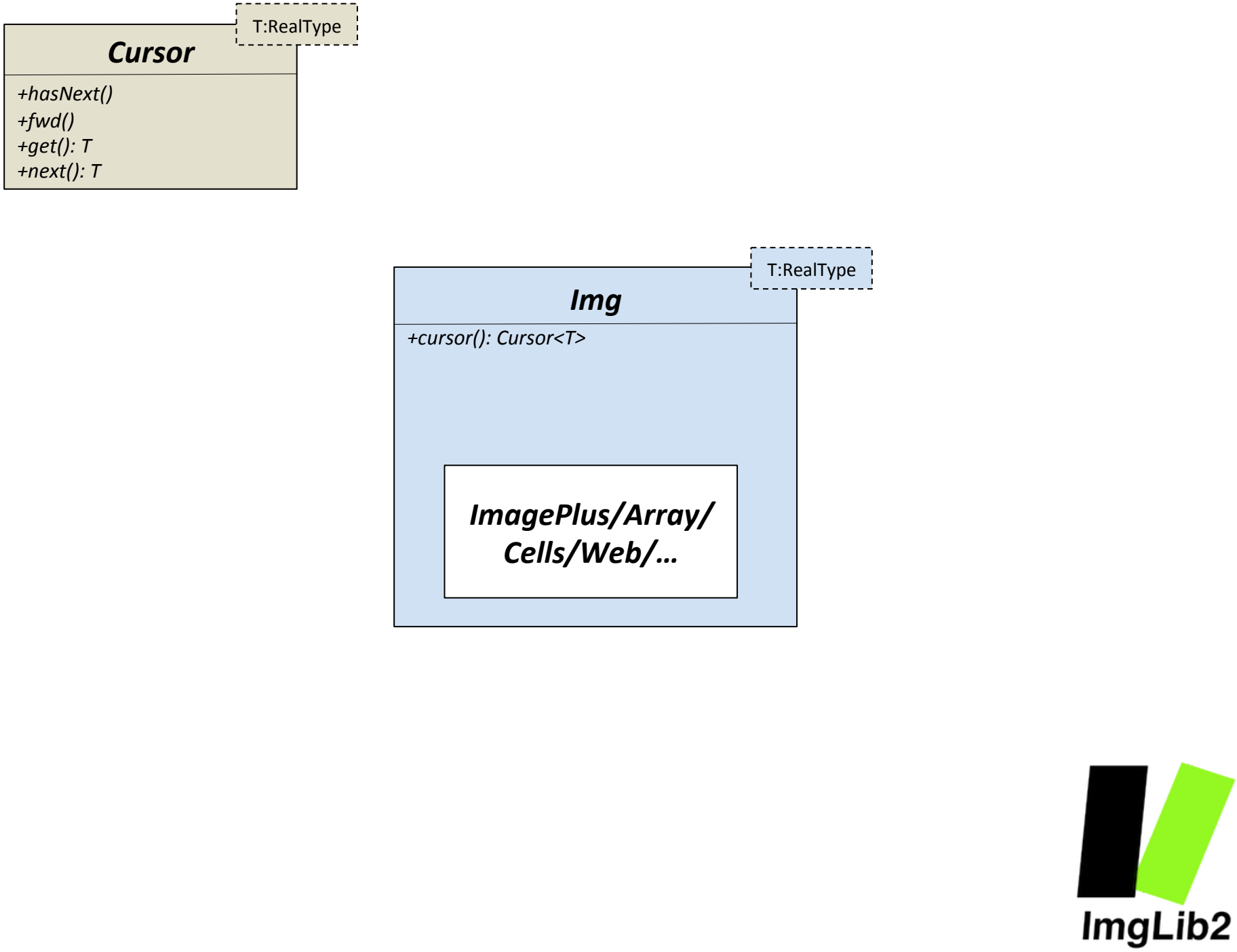
Why using ImgLib2?

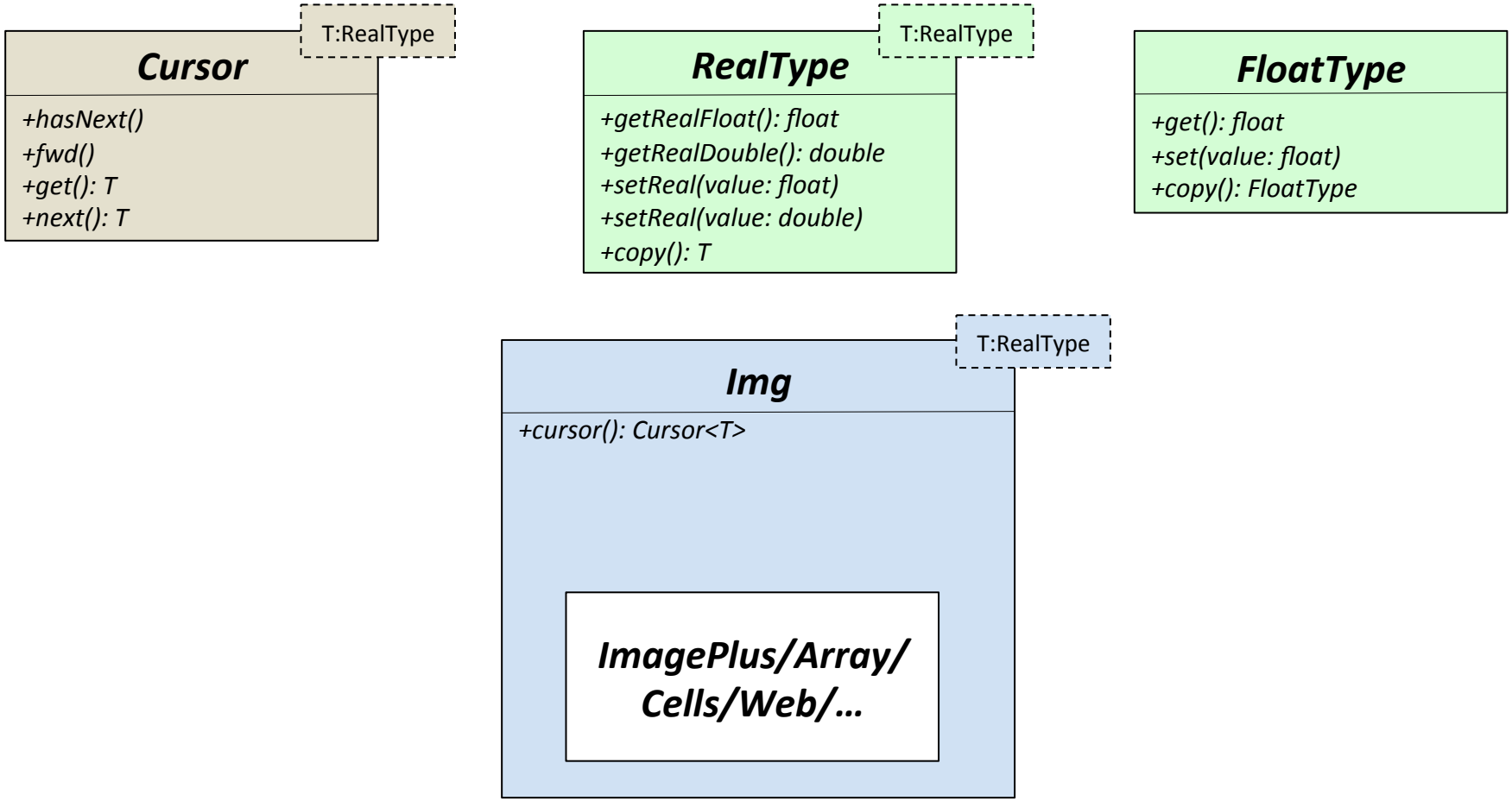
- Generic programming – Write it once!
- Directly applicable to very large datasets, different dimensionality or pixel type
- ImgLib2 does not force you to implement type independent or n-dimensional – it is still applicable to very large datasets
- More algorithm-like programming
- Fewer simple programming mistakes
- Easier exchange of code
- Smaller source code

ImagePlus/Array/ Cells/Web/…

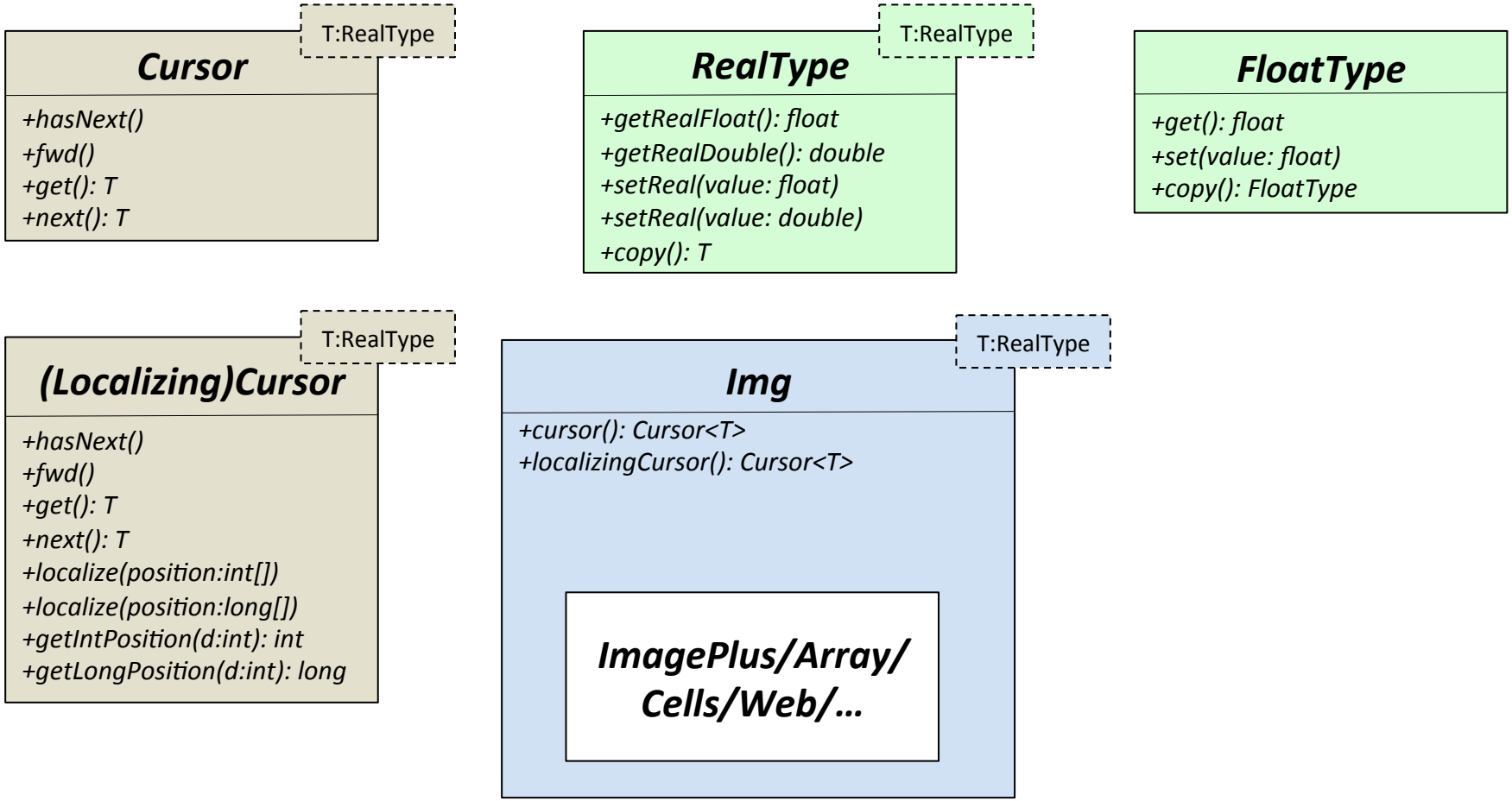


ImgLib2

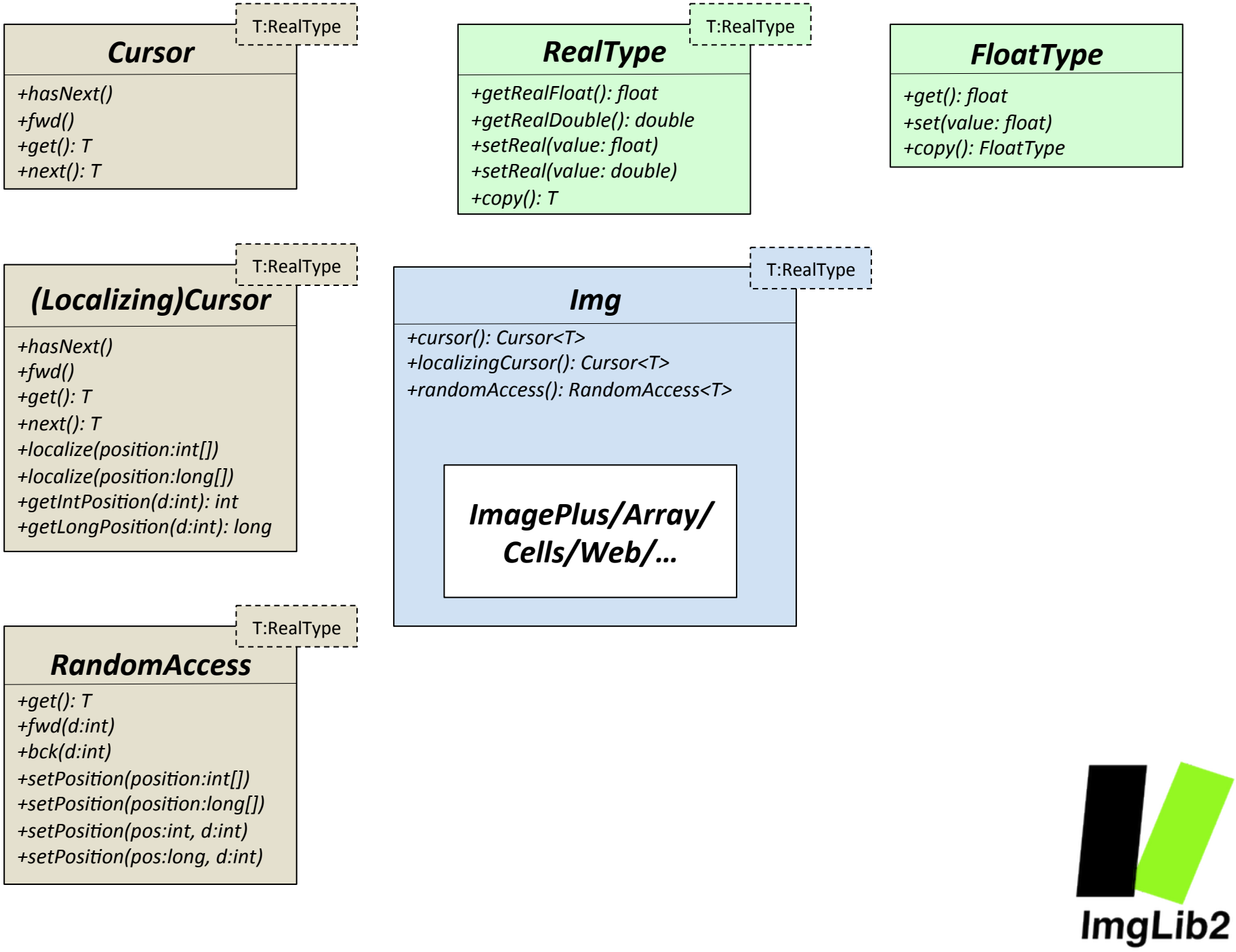

Cursor
T:RealType
+hasNext()
+fwd()
+get(): T
+next(): T
Img
T:RealType
+cursor(): Cursor<T>
ImagePlus/Array/
Cells/Web/…
ImgLib2

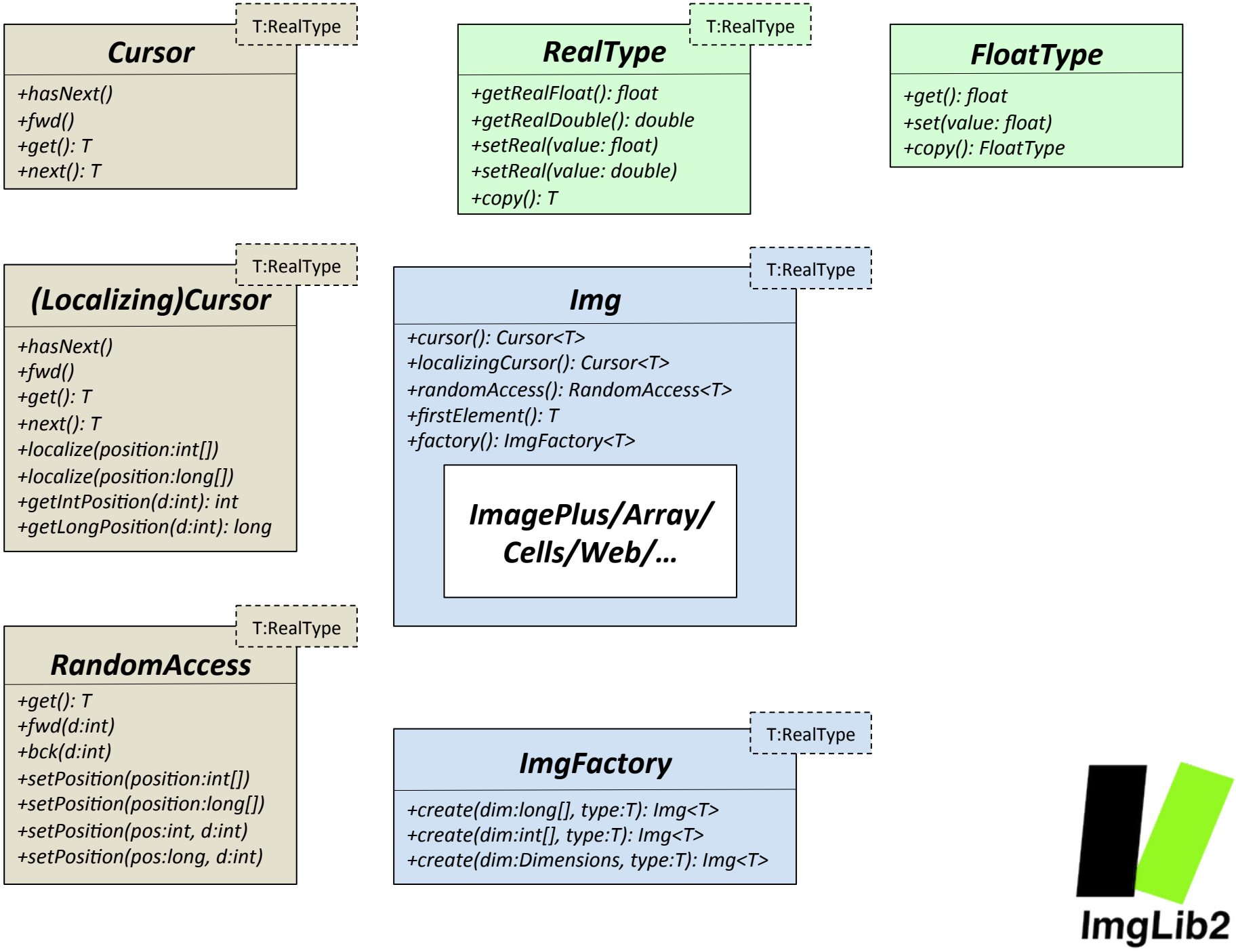

Cursor
T:RealType
+hasNext()
+fwd()
+get(): T
+next(): T
RealType
T:RealType
+getRealFloat(): float
+getRealDouble(): double
+setReal(value: float)
+setReal(value: double)
+copy(): T
FloatType
+get(): float
+set(value: float)
+copy(): FloatType
Img
T:RealType
+cursor(): Cursor<T>
ImagePlus/Array/
Cells/Web/...

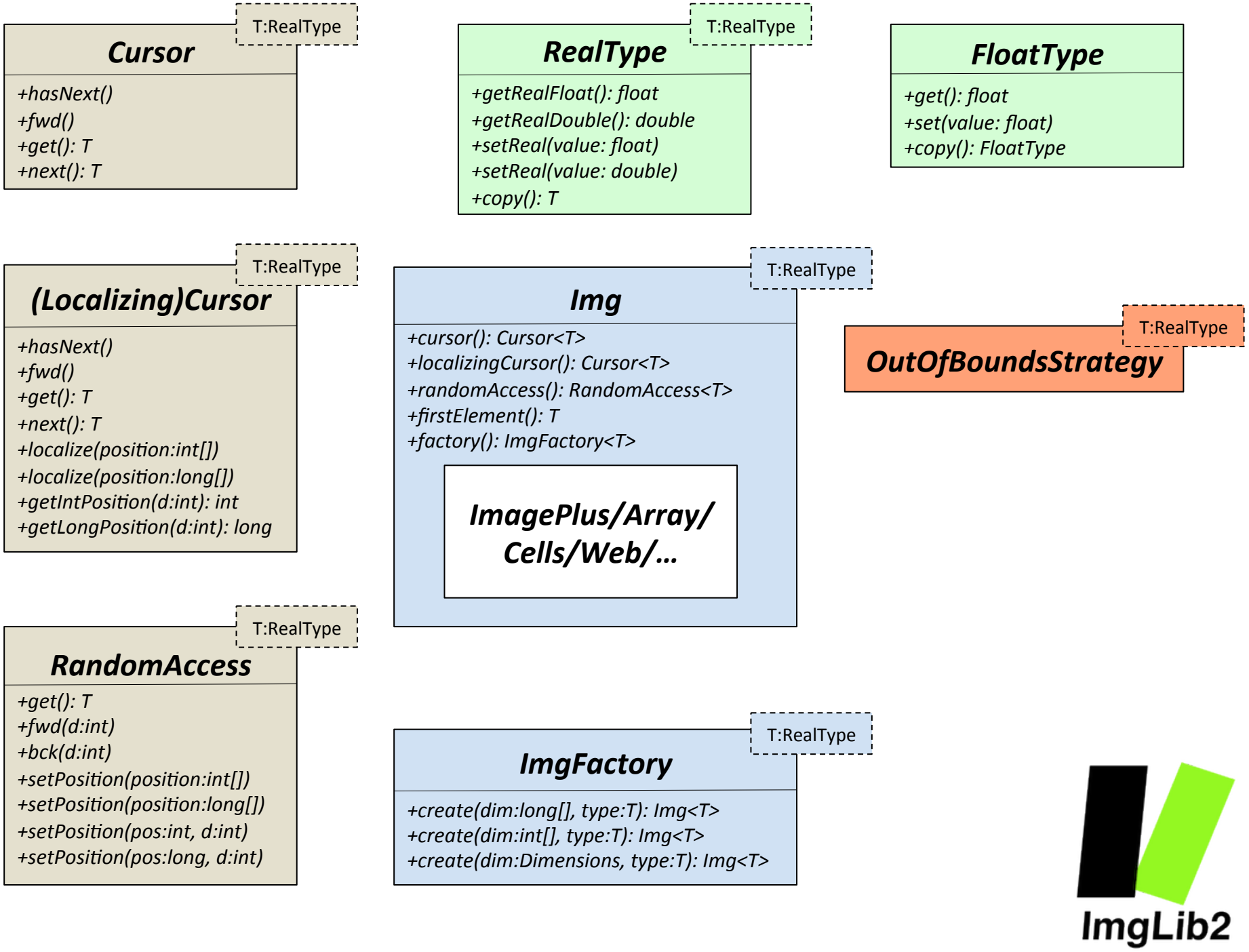
ImgLib2

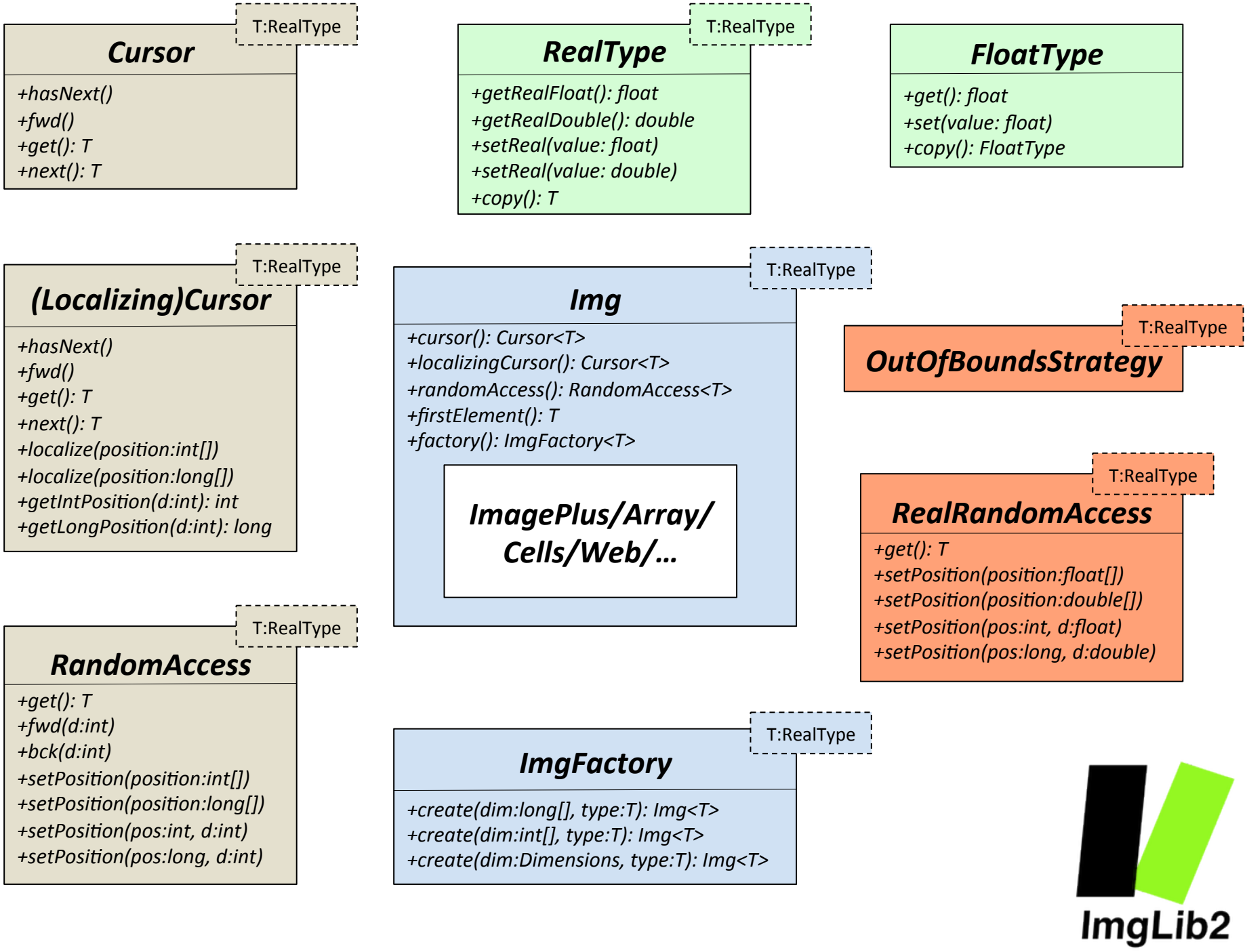

T:RealType
Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
T:RealType
RealType
+getRealFloat(): float
+getRealDouble(): double
+setReal(value: float)
+setReal(value: double)
+copy(): T
FloatType
+get(): float
+set(value: float)
+copy(): FloatType
T:RealType
(Localizing)Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
+localize(position:int[])
+localize(position:long[])
+getIntPosition(d:int): int
+getLongPosition(d:int): long
T:RealType
Img
+cursor(): Cursor<T>
+localizingCursor(): Cursor<T>
ImagePlus/Array/
Cells/Web/...

ImgLib2


T:RealType
Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
T:RealType
RealType
+getRealFloat(): float
+getRealDouble(): double
+setReal(value: float)
+setReal(value: double)
+copy(): T
FloatType
+get(): float
+set(value: float)
+copy(): FloatType
T:RealType
(Localizing)Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
+localize(position:int[])
+localize(position:long[])
+getIntPosition(d:int): int
+getLongPosition(d:int): long
T:RealType
Img
+cursor(): Cursor<T>
+localizingCursor(): Cursor<T>
+randomAccess(): RandomAccess<T>
ImagePlus/Array/
Cells/Web/...
T:RealType
RandomAccess
+get(): T
+fwd(d:int)
+bck(d:int)
+setPosition(position:int[])
+setPosition(position:long[])
+setPosition(pos:int, d:int)
+setPosition(pos:long, d:int)
ImgLib2


T:RealType
Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
T:RealType
RealType
+getRealFloat(): float
+getRealDouble(): double
+setReal(value: float)
+setReal(value: double)
+copy(): T
FloatType
+get(): float
+set(value: float)
+copy(): FloatType
T:RealType
(Localizing)Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
+localize(position:int[])
+localize(position:long[])
+getIntPosition(d:int): int
+getLongPosition(d:int): long
T:RealType
Img
+cursor(): Cursor<T>
+localizingCursor(): Cursor<T>
+randomAccess(): RandomAccess<T>
+firstElement(): T
+factory(): ImgFactory<T>
ImagePlus/Array/
Cells/Web/...
T:RealType
RandomAccess
+get(): T
+fwd(d:int)
+bck(d:int)
+setPosition(position:int[])
+setPosition(position:long[])
+setPosition(pos:int, d:int)
+setPosition(pos:long, d:int)
T:RealType
ImgFactory
+create(dim:long[], type:T): Img<T>
+create(dim:int[], type:T): Img<T>
+create(dim:Dimensions, type:T): Img<T>
ImgLib2


T:RealType
Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
T:RealType
RealType
+getRealFloat(): float
+getRealDouble(): double
+setReal(value: float)
+setReal(value: double)
+copy(): T
FloatType
+get(): float
+set(value: float)
+copy(): FloatType
T:RealType
(Localizing)Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
+localize(position:int[])
+localize(position:long[])
+getIntPosition(d:int): int
+getLongPosition(d:int): long
T:RealType
Img
+cursor(): Cursor<T>
+localizingCursor(): Cursor<T>
+randomAccess(): RandomAccess<T>
+firstElement(): T
+factory(): ImgFactory<T>
ImagePlus/Array/
Cells/Web/...
T:RealType
OutOfBoundsStrategy
T:RealType
RandomAccess
+get(): T
+fwd(d:int)
+bck(d:int)
+setPosition(position:int[])
+setPosition(position:long[])
+setPosition(pos:int, d:int)
+setPosition(pos:long, d:int)
T:RealType
ImgFactory
+create(dim:long[], type:T): Img<T>
+create(dim:int[], type:T): Img<T>
+create(dim:Dimensions, type:T): Img<T>
ImgLib2

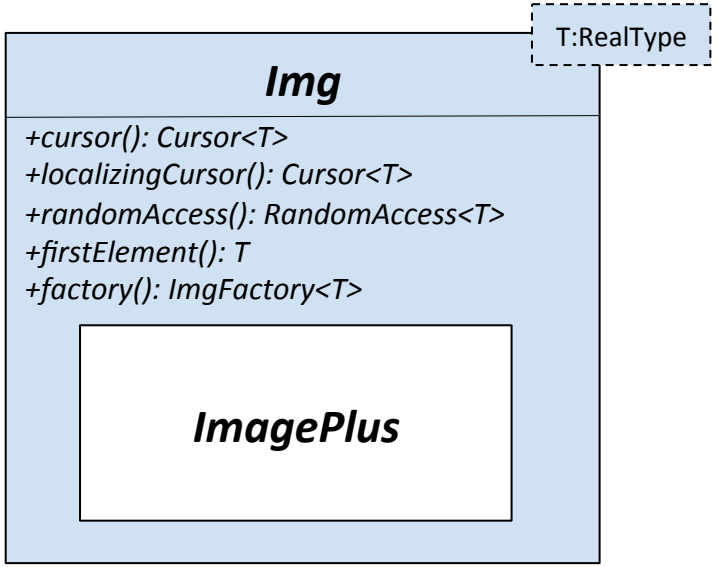

T:RealType
Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
T:RealType
RealType
+getRealFloat(): float
+getRealDouble(): double
+setReal(value: float)
+setReal(value: double)
+copy(): T
FloatType
+get(): float
+set(value: float)
+copy(): FloatType
T:RealType
(Localizing)Cursor
+hasNext()
+fwd()
+get(): T
+next(): T
+localize(position:int[])
+localize(position:long[])
+getIntPosition(d:int): int
+getLongPosition(d:int): long
T:RealType
Img
+cursor(): Cursor<T>
+localizingCursor(): Cursor<T>
+randomAccess(): RandomAccess<T>
+firstElement(): T
+factory(): ImgFactory<T>
ImagePlus/Array/
Cells/Web/...
T:RealType
OutOfBoundsStrategy
T:RealType
RealRandomAccess
+get(): T
+setPosition(position:float[])
+setPosition(position:double[])
+setPosition(pos:int, d:float)
+setPosition(pos:long, d:double)
T:RealType
RandomAccess
+get(): T
+fwd(d:int)
+bck(d:int)
+setPosition(position:int[])
+setPosition(position:long[])
+setPosition(pos:int, d:int)
+setPosition(pos:long, d:int)
T:RealType
ImgFactory
+create(dim:long[], type:T): Img<T>
+create(dim:int[], type:T): Img<T>
+create(dim:Dimensions, type:T): Img<T>
ImgLib2

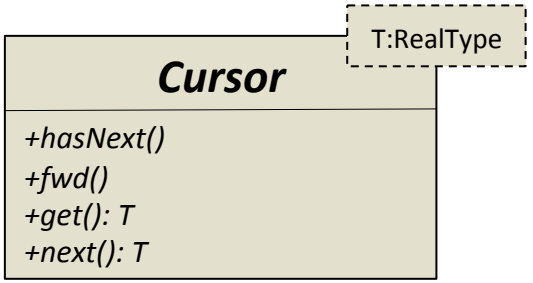
Hands on Programming

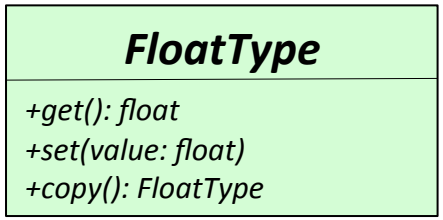
① Threshold on an image
② Center of mass of an image
③ Gradient of an image
④ Rigid transformation of an image

Thresholding 1

- Wrap a float image
- Compute threshold for each pixel
- Overwrite the original data
- ImgLib2_Threshold1.java



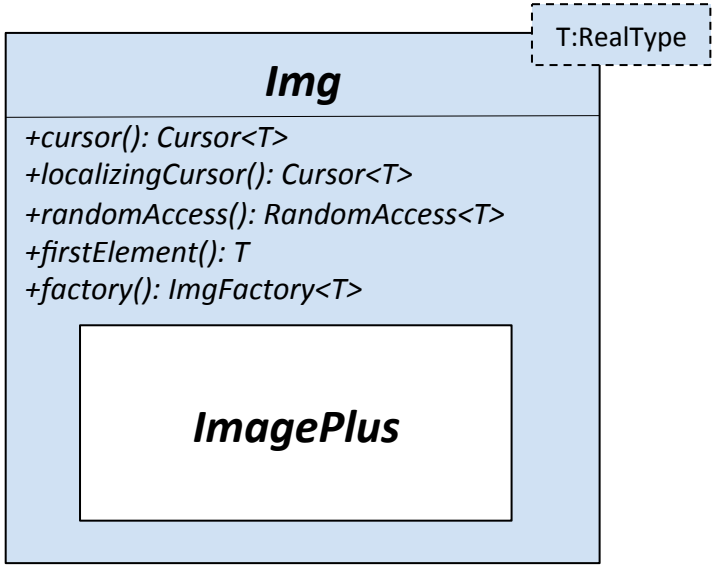


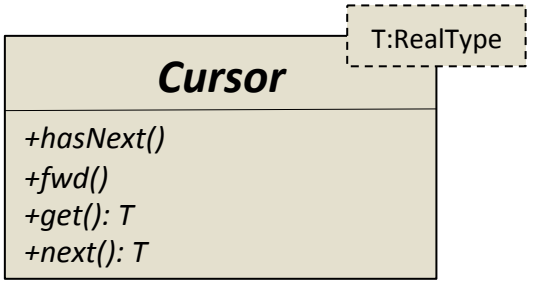


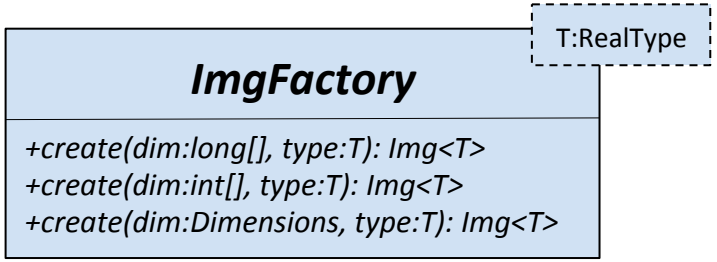
ImgLib2

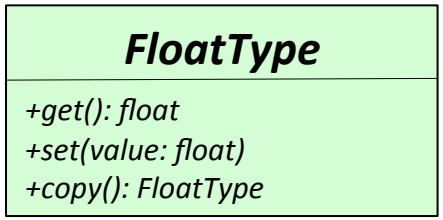
Thresholding 2

- Wrap a float image
- Create a new Img
- Compute threshold for each pixel
- Write threshold into the new Img
- ImgLib2_Threshold2.java





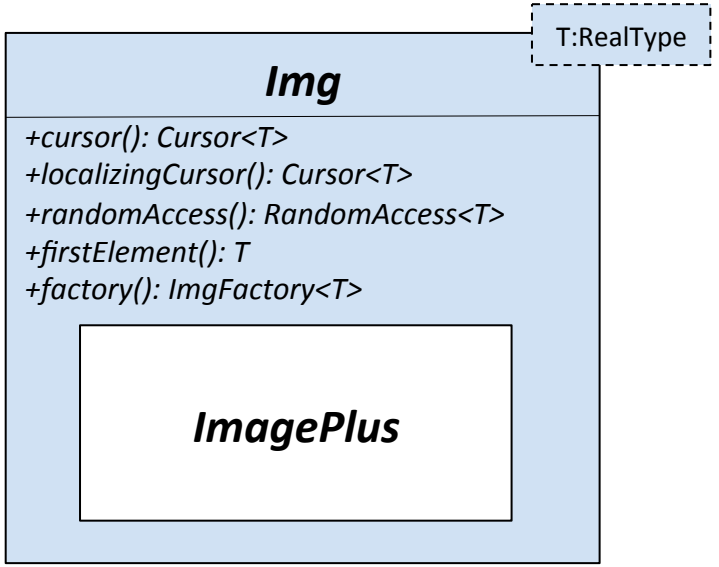


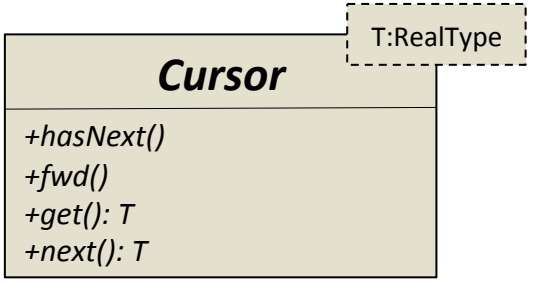


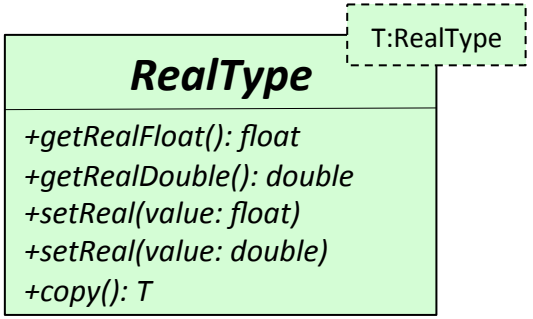
ImgLib2

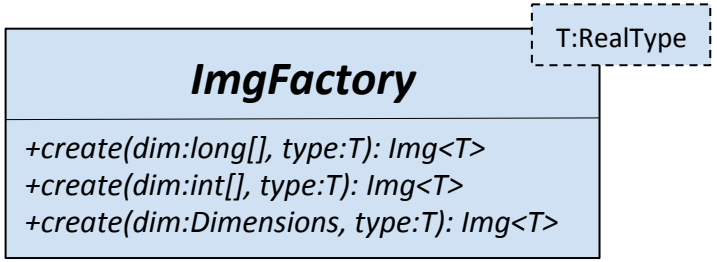
Thresholding 3

- Work on <u>any real valued</u> image
- Create a new Img
- Compute threshold for each pixel
- Write threshold into the new Img
- ImgLib2_Threshold3.java





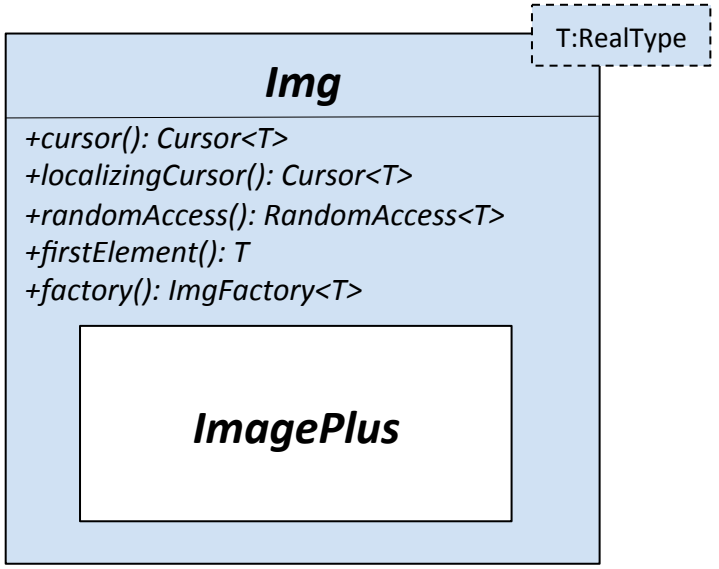


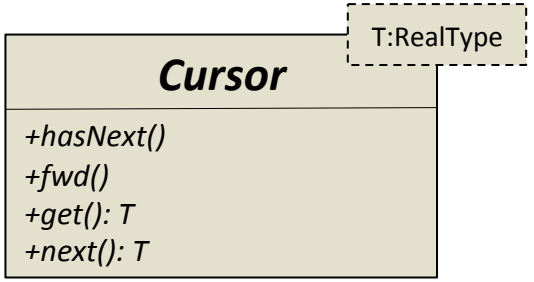


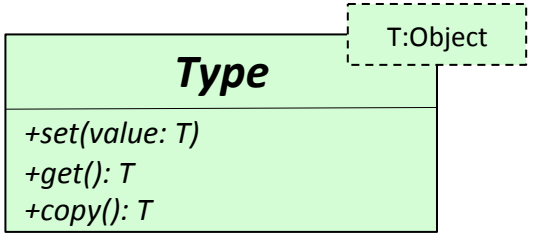
ImgLib2

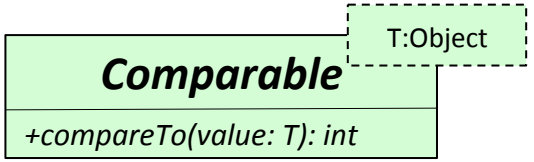
Thresholding 4

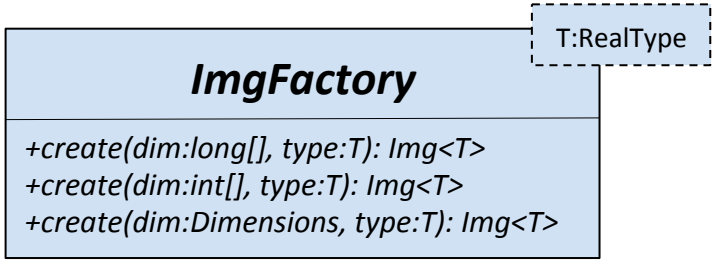
- Work on any real valued image
- Create a new Img
- Compute threshold for each pixel for all Comparable
- Write threshold into the new Img of BitType (1 bit per pixel)
- ImgLib2_Threshold4.java







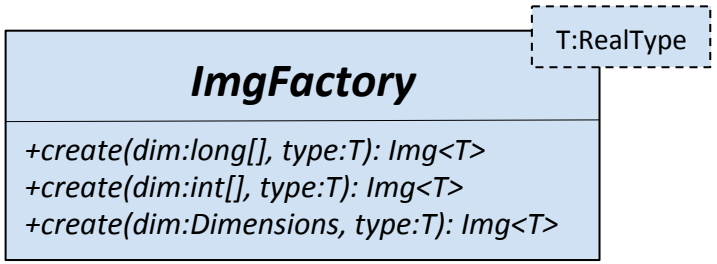


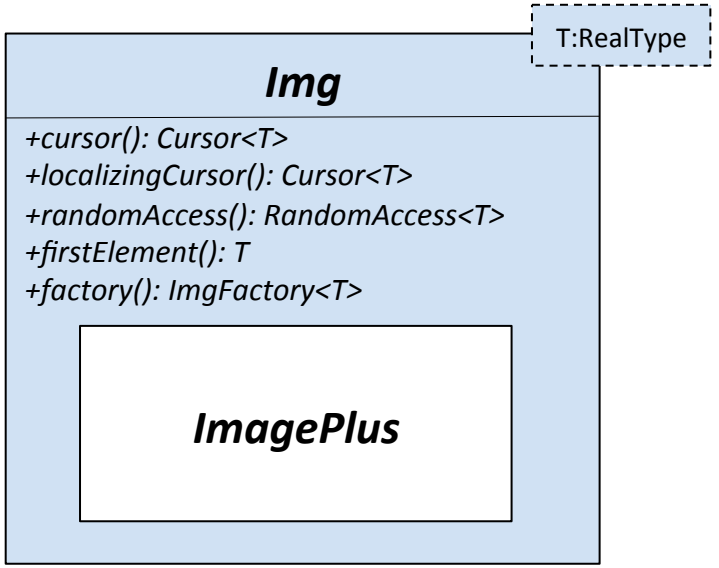


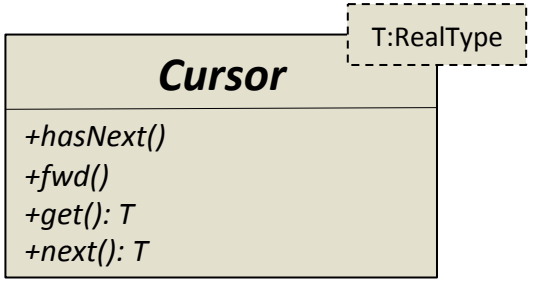
ImgLib2

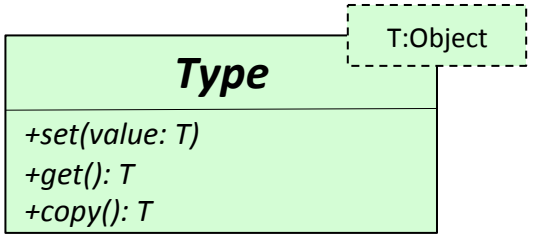
Thresholding 5

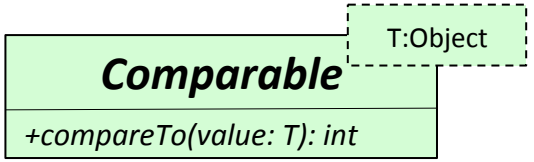
- Work on any real valued image
- Create a new Img
- Compute threshold for a subset using Views
- Write threshold into the new Img of BitType (1 bit per pixel)
- ImgLib2_Threshold5.java







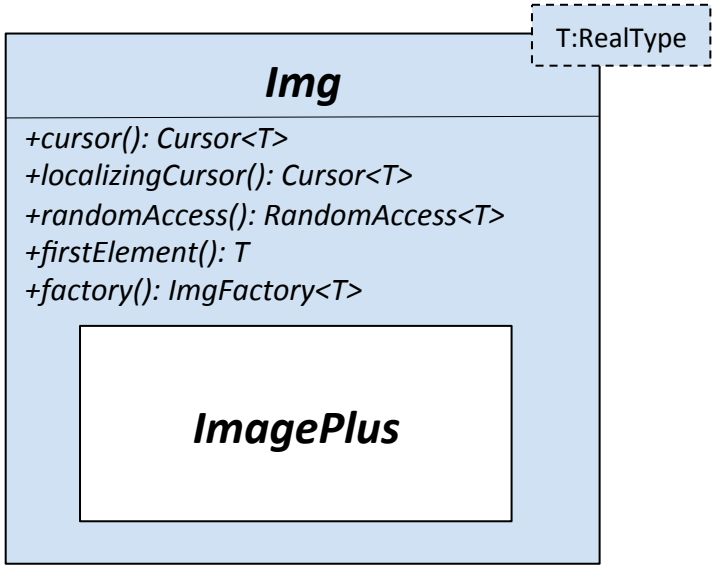


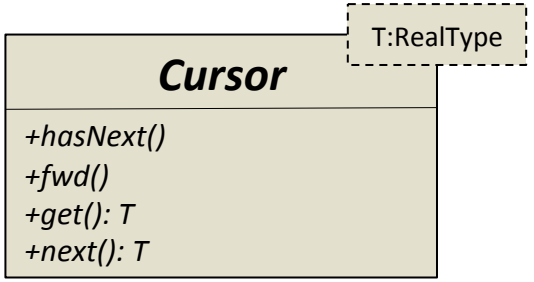


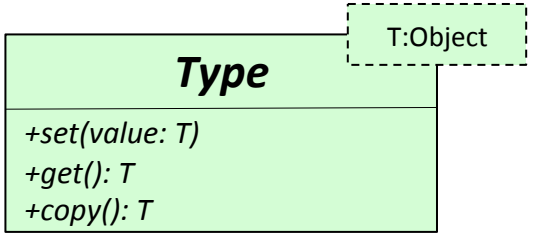
ImgLib2

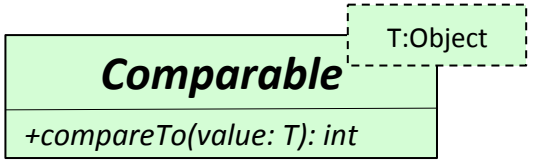
Thresholding 6

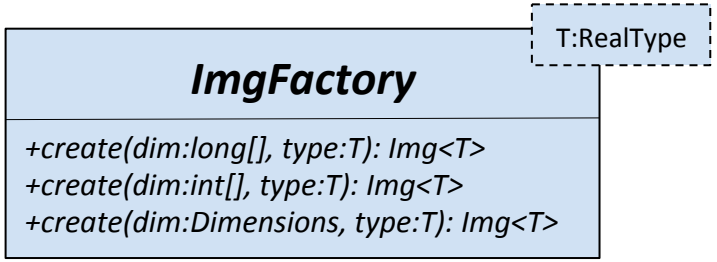
- Work on any real valued image
- Create a new Img
- Compute threshold for a subset using IterableInterval
- Write threshold into the new Img of BitType (1 bit per pixel)
- ImgLib2_Threshold6.java





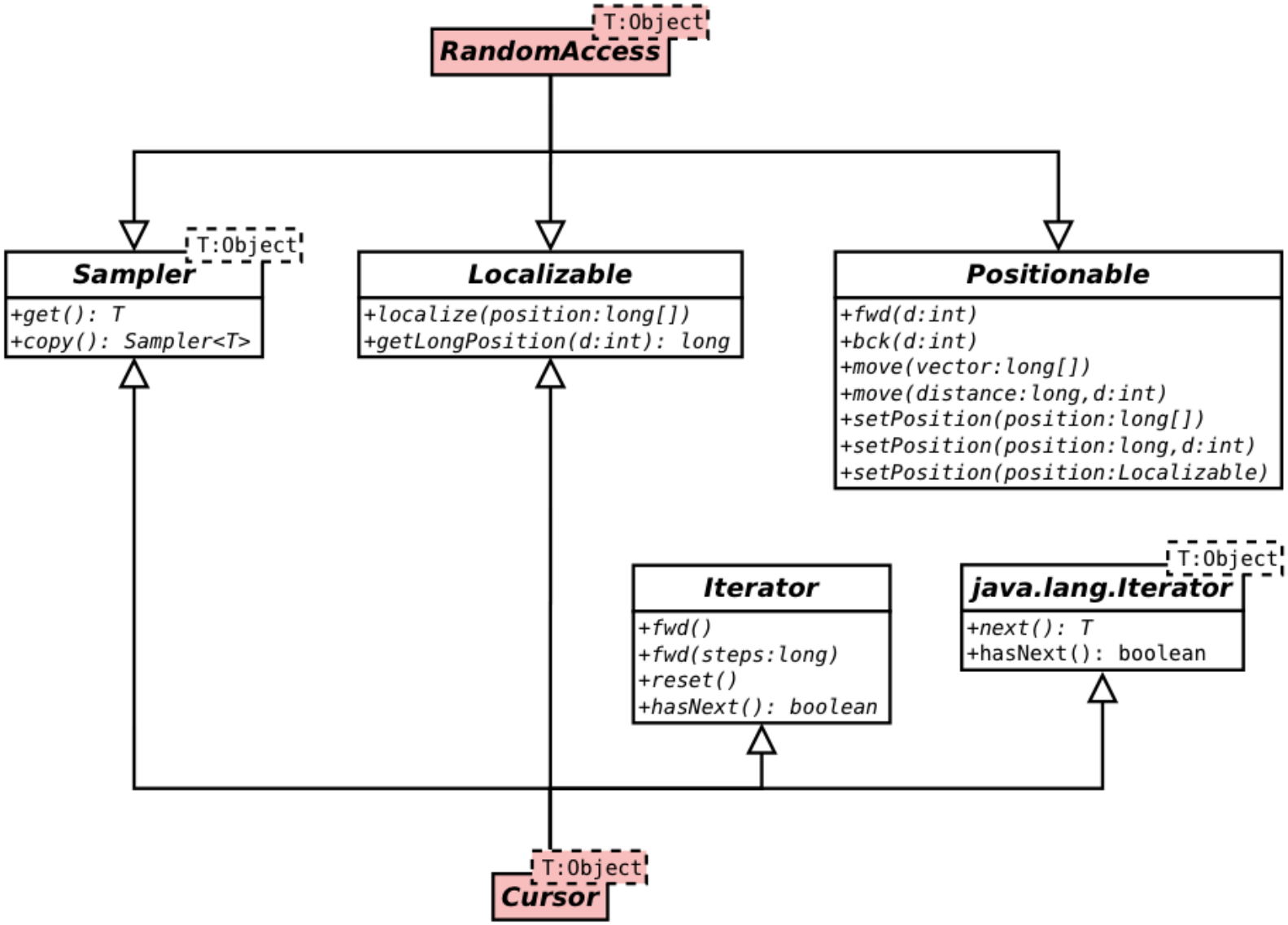




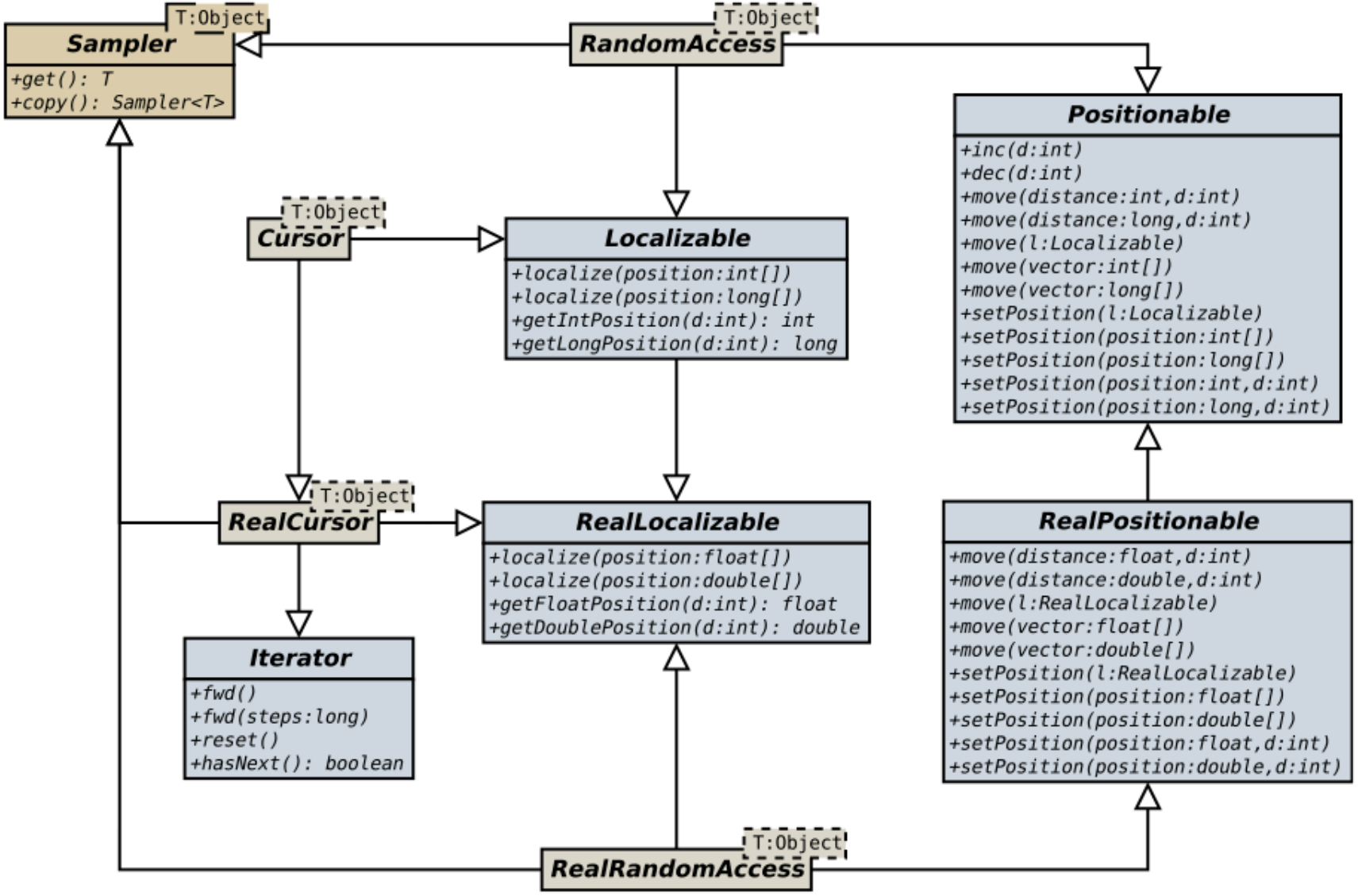


ImgLib2

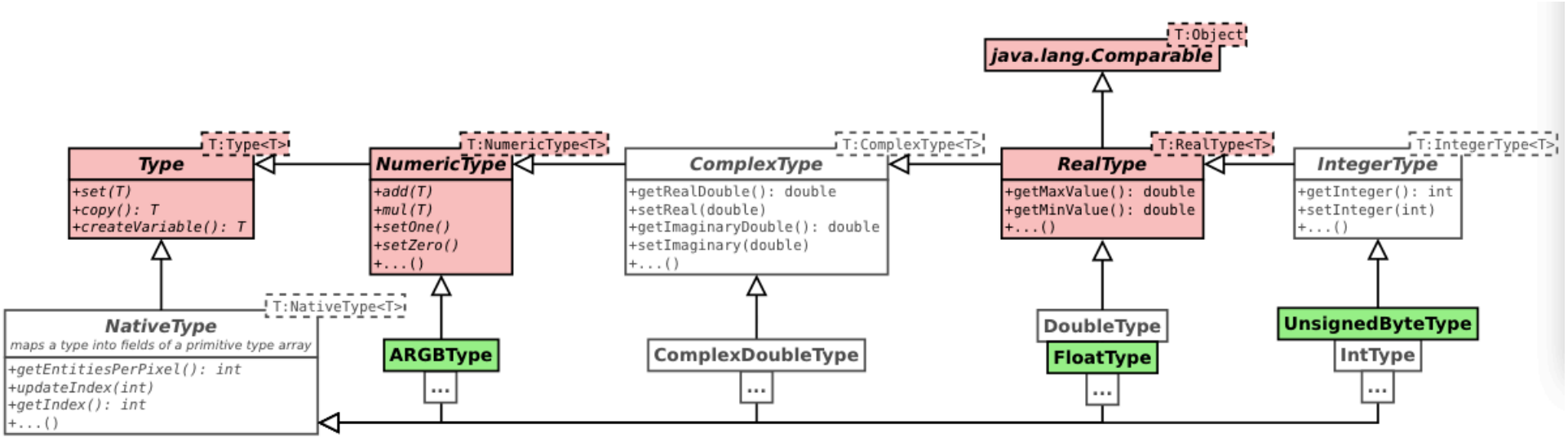
Accessors (simplified)



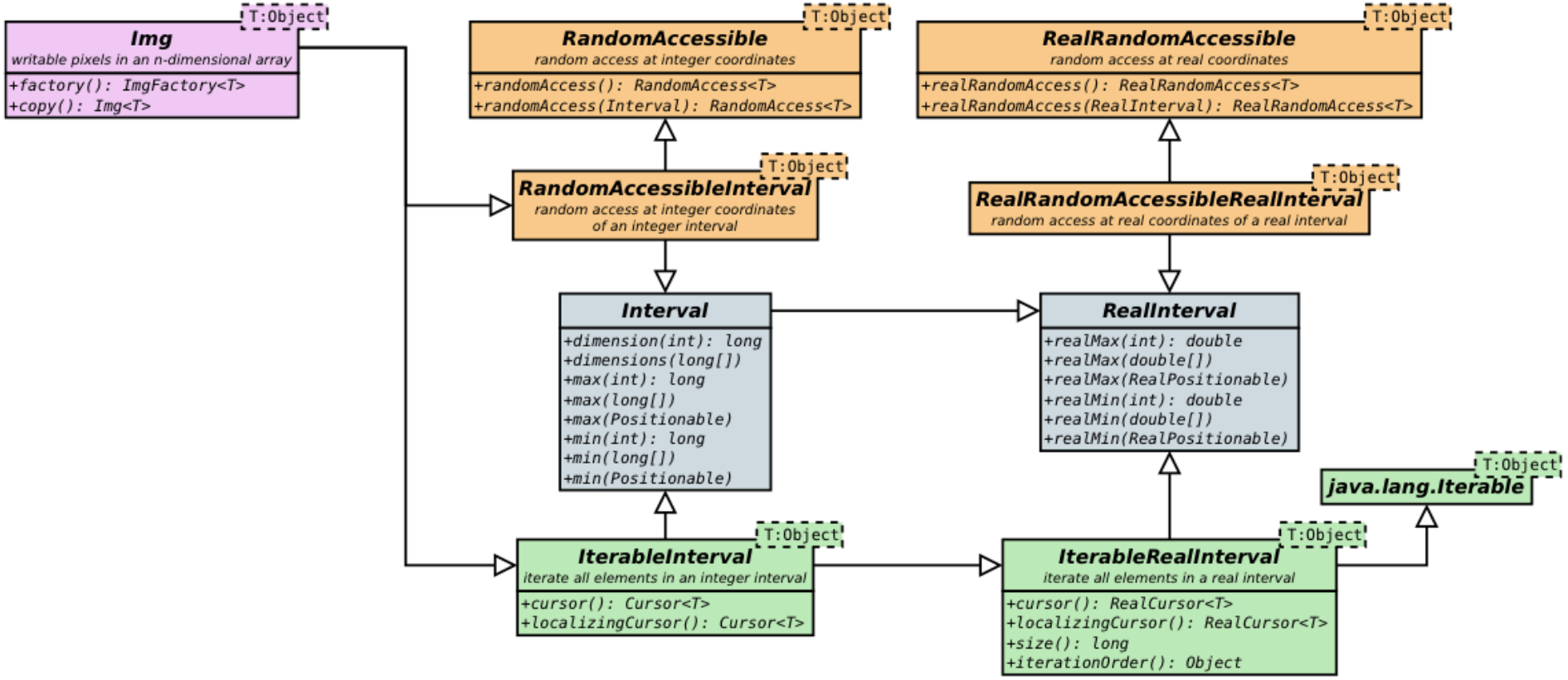
Accessors


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

Type Hierarchy

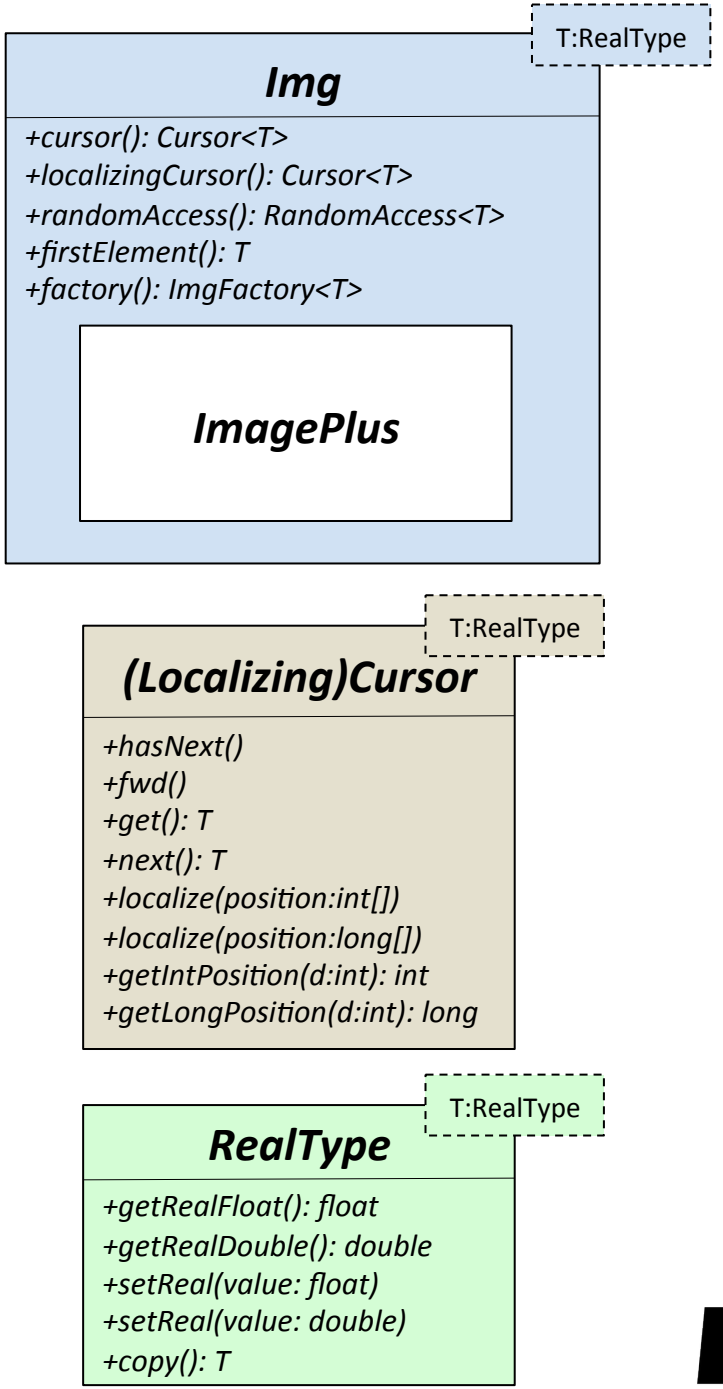


Accessibles



Center of Mass 1

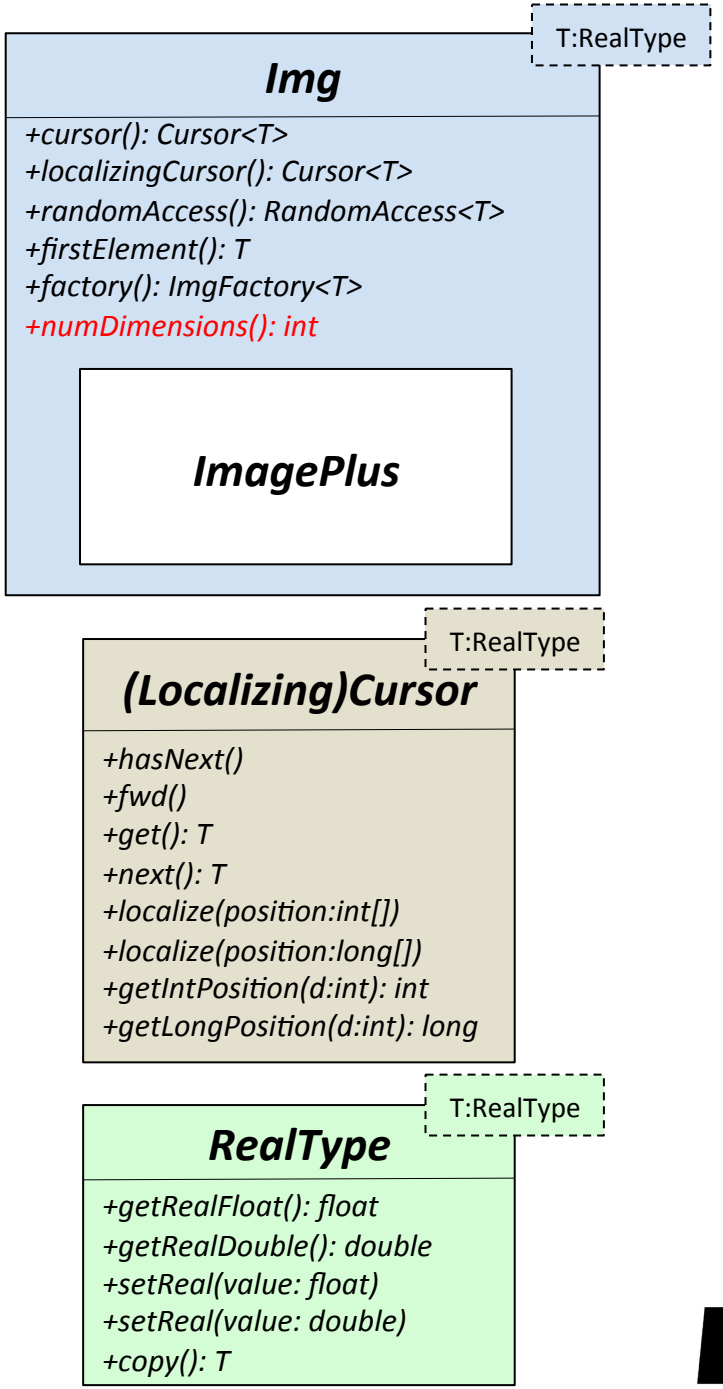
- Work on any real valued image
- Use a localizing Cursor to Compute the center of mass in two dimensions (x,y)
- Write the result to the log window
- ImgLib2_CenterOfMass1.java



ImgLib2

Center of Mass 2

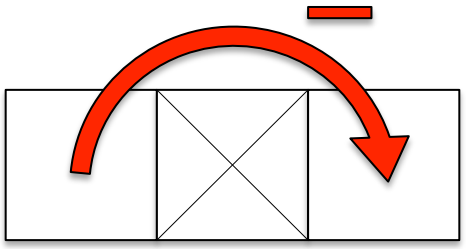
- Work on any real valued image
- Use a localizing Cursor to Compute the center of mass in n dimensions
- Write the result to the log window
- ImgLib2_CenterOfMass2.java



ImgLib2

Gradient 1

- Work on any real valued image
- Approximate the magnitude of the gradient for each pixel using a localizing Cursor on the output and a RandomAccess on the input

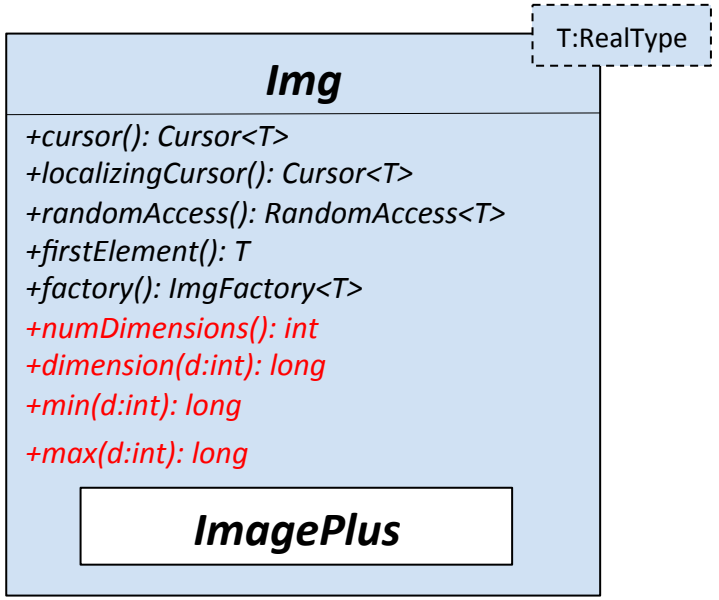


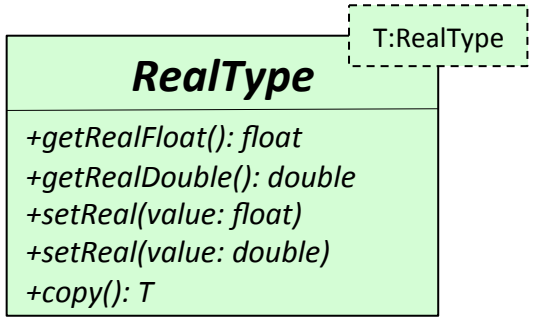
$$\nabla f_x = \frac{I(x+1,y,\ldots) - I(x-1,y,\ldots)}{2}$$

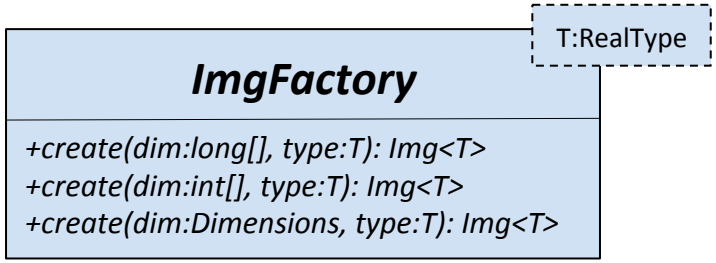
$$\nabla f_y = \frac{I(x,y+1,\ldots) - I(x,y-1,\ldots)}{2}$$

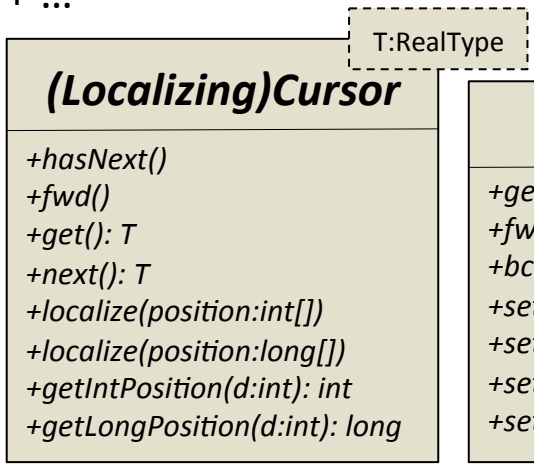
$$|\nabla f| = \sqrt{\nabla f_x^2 + \nabla f_y^2 + \ldots}$$

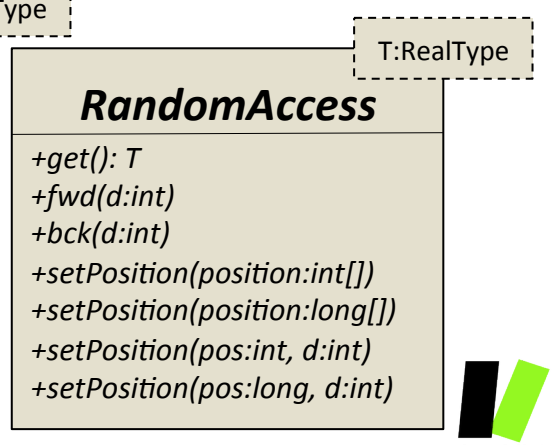
- ImgLib2_Gradient1.java





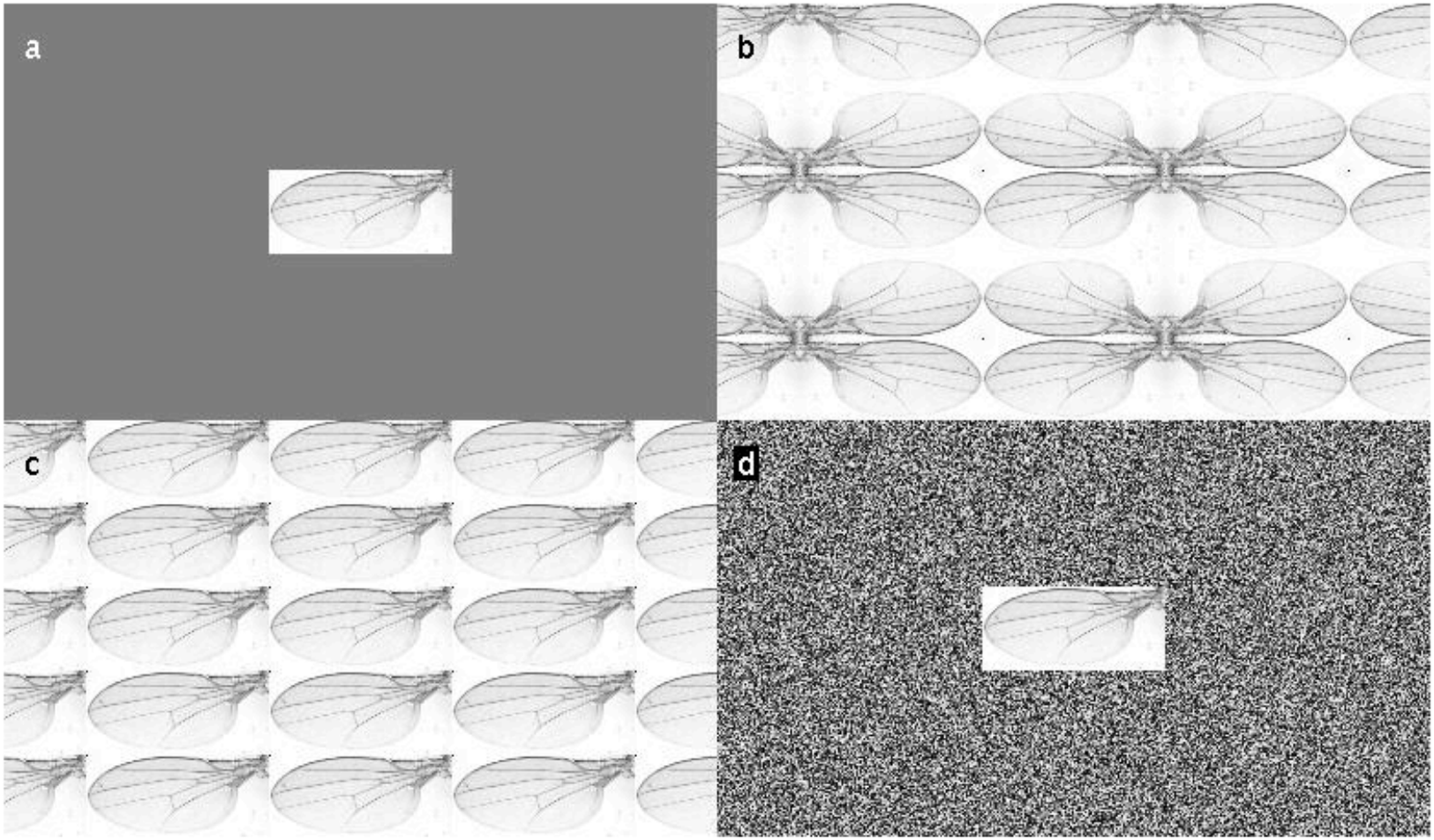






ImgLib2

OutOfBoundsStrategies

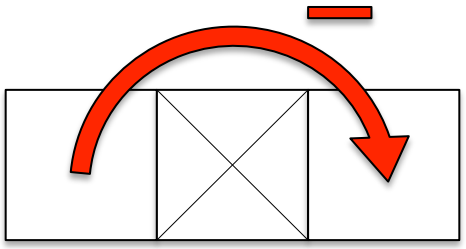


Illustrates the effect of various OutOfBoundsStrategies. (a) shows out of bounds with a constant value, (b) shows a mirroring strategy, (c) shows the periodic strategy, and (d) shows a strategy that uses random values.

ImgLib2

Gradient 2

- Use OutOfBoundsStrategy to compute gradient for all pixels
- Approximate the magnitude of the gradient for each pixel using a localizing Cursor on the output and a RandomAccess on the input

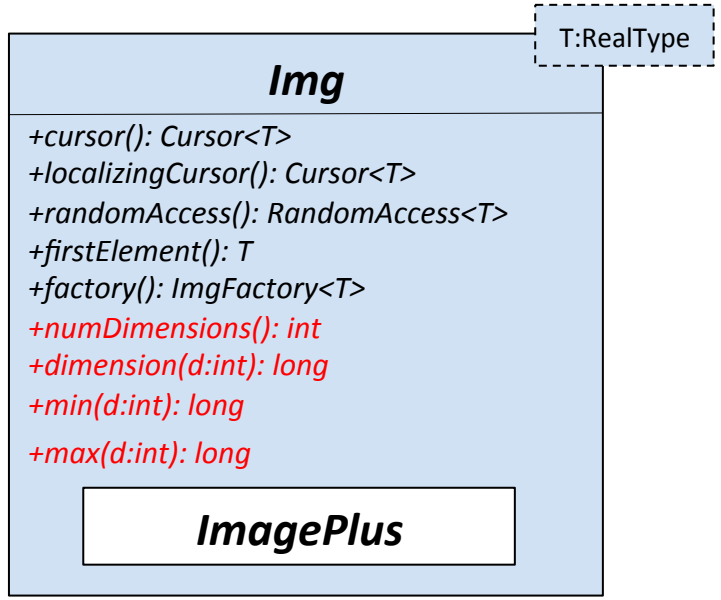


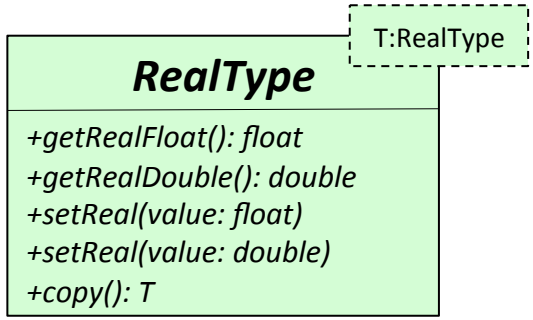
$$\nabla f_x = \frac{I(x+1,y,\ldots) - I(x-1,y,\ldots)}{2}$$

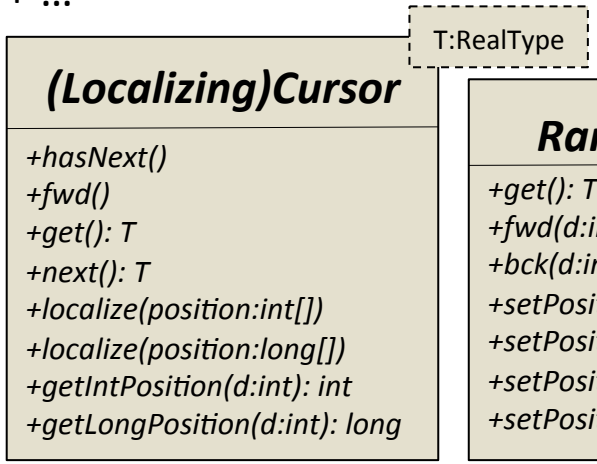
$$\nabla f_y = \frac{I(x,y+1,\ldots) - I(x,y-1,\ldots)}{2}$$

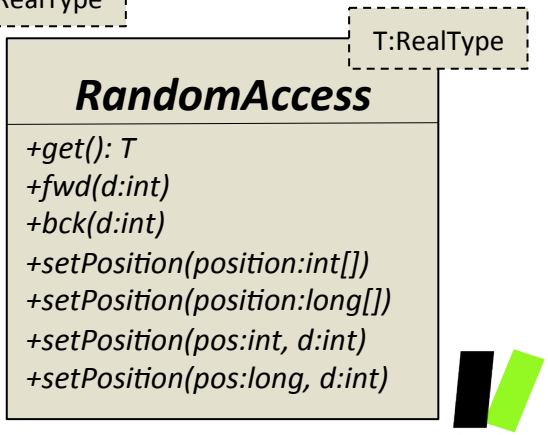
$$|\nabla f| = \sqrt{\nabla f_x^2 + \nabla f_y^2 + \ldots}$$

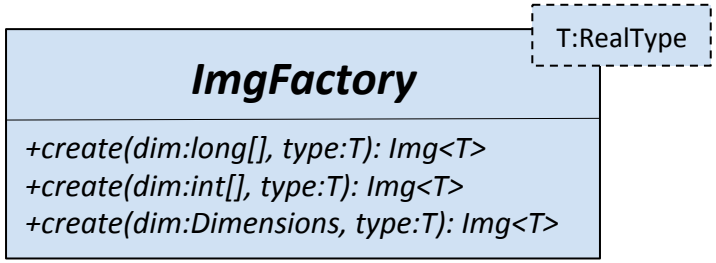
- ImgLib2_Gradient2.java







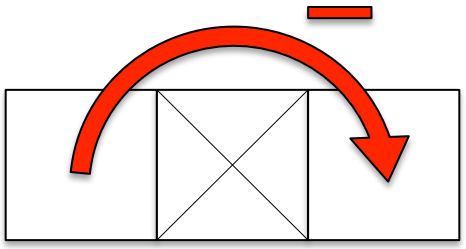




ImgLib2

Gradient 3

- Always return an Img<FloatType> to prevent overflows
- Approximate the magnitude of the gradient for each pixel using a localizing Cursor on the output and a RandomAccess on the input

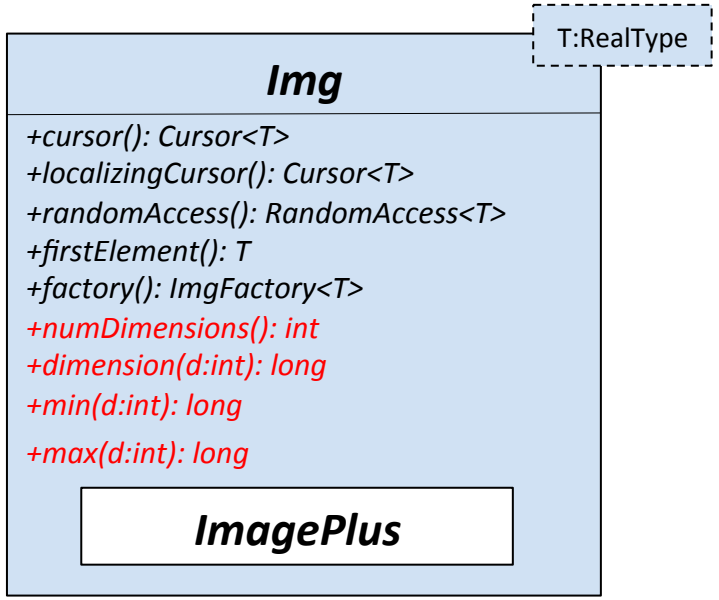


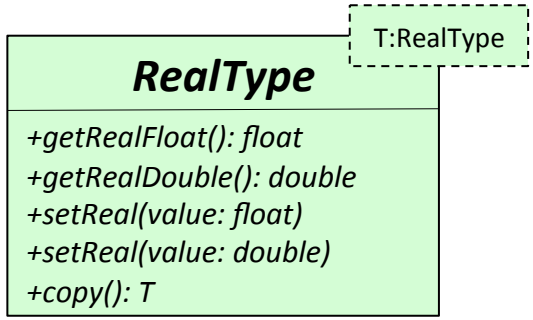
$$\nabla f_x = \frac{I(x+1,y,\ldots) - I(x-1,y,\ldots)}{2}$$

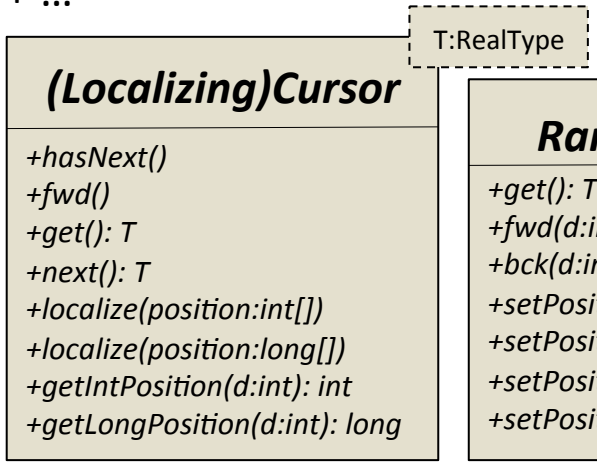
$$\nabla f_y = \frac{I(x,y+1,\ldots) - I(x,y-1,\ldots)}{2}$$

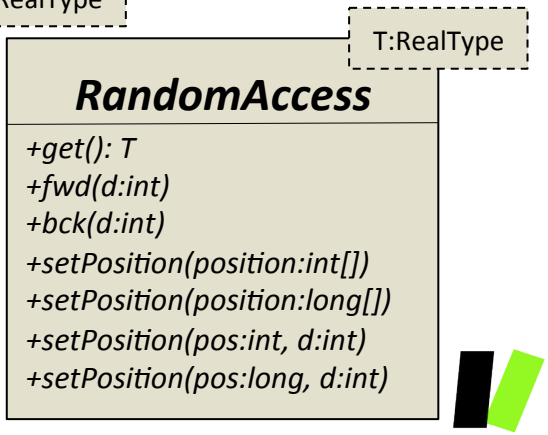
$$|\nabla f| = \sqrt{\nabla f_x^2 + \nabla f_y^2 + \ldots}$$

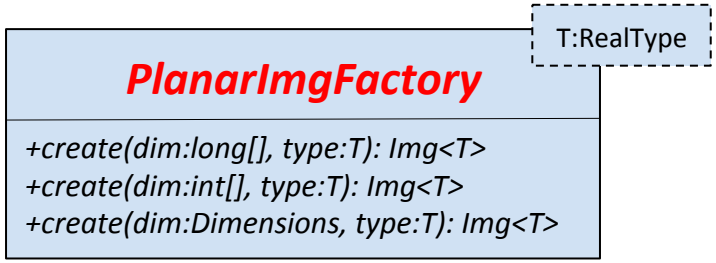
- ImgLib2_Gradient3.java







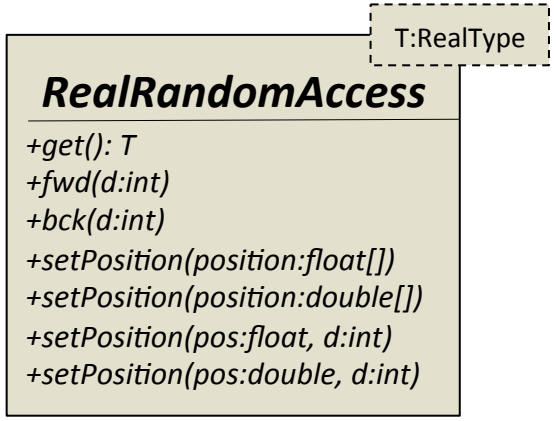


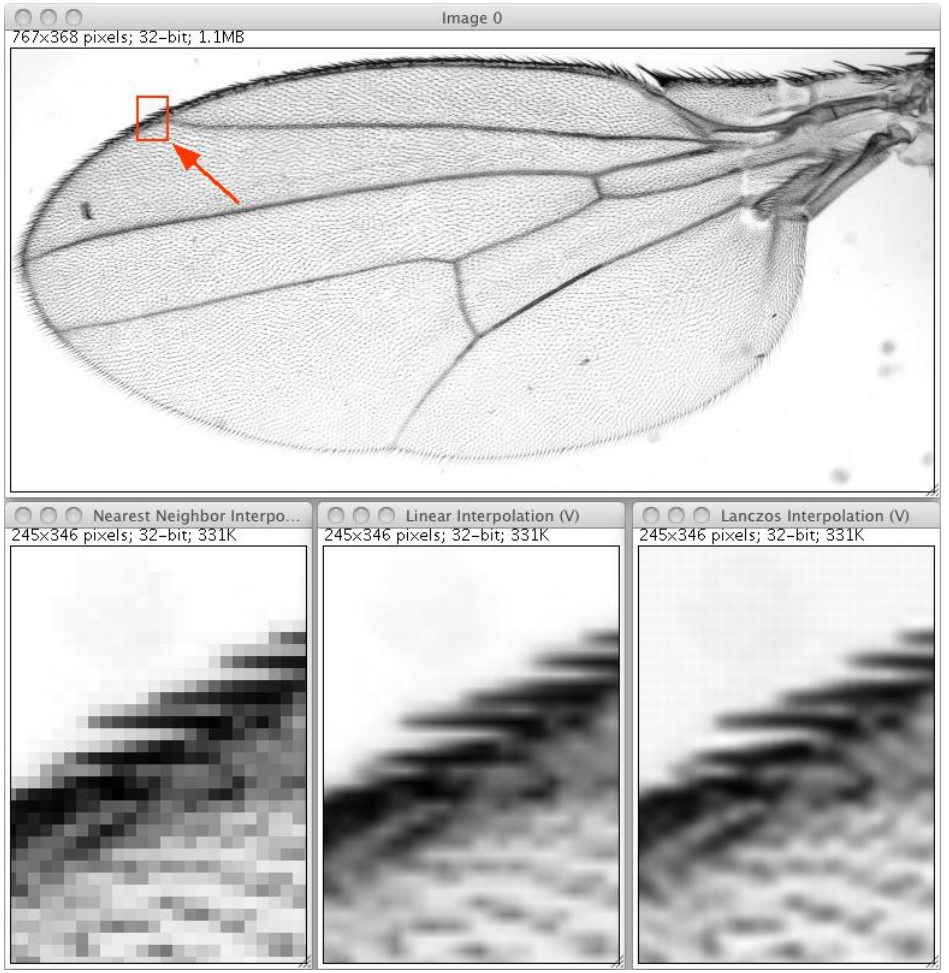


ImgLib2

Interpolation

- Transform an Img (RandomAccessible) into a RealRandomAccessible that can return values at any real-valued location in space
- First extend by OutOfBounds

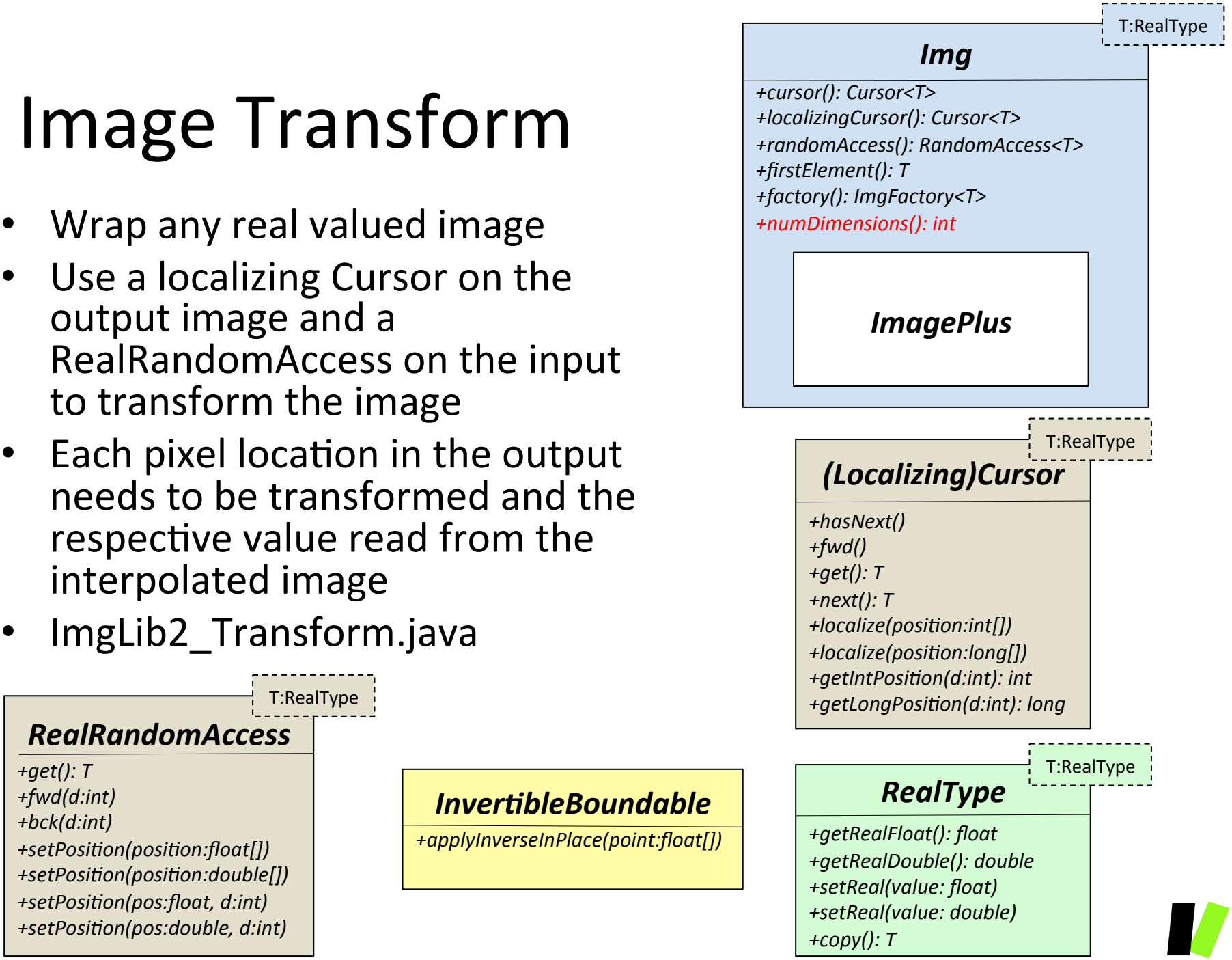


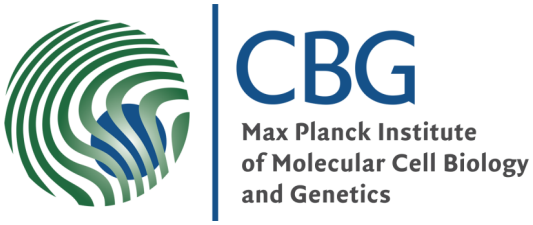


ImgLib2

Image Transform

- Wrap any real valued image
- Use a localizing Cursor on the output image and a RealRandomAccess on the input to transform the image
- Each pixel location in the output needs to be transformed and the respective value read from the interpolated image
- ImgLib2_Transform.java





Special thanks to …

- Tobias Pietzsch (MPI-CBG)
- Stephan Saalfeld (MPI-CBG)

- Pavel Tomancak (MPI-CBG)
- Gene Myers (MPI-CBG)
- Rob Singer
  (Einstein College & Janelia Farm)

- ImageJ2 crew
  - Johannes Schindelin
  - Curtis Rueden
  - Barry DeZonia
  - Kevin Eliceiri
- Albert Cardona
- KNIME guys
  - Christian Dietz
  - Martin Horn