

UML Class Diagrams

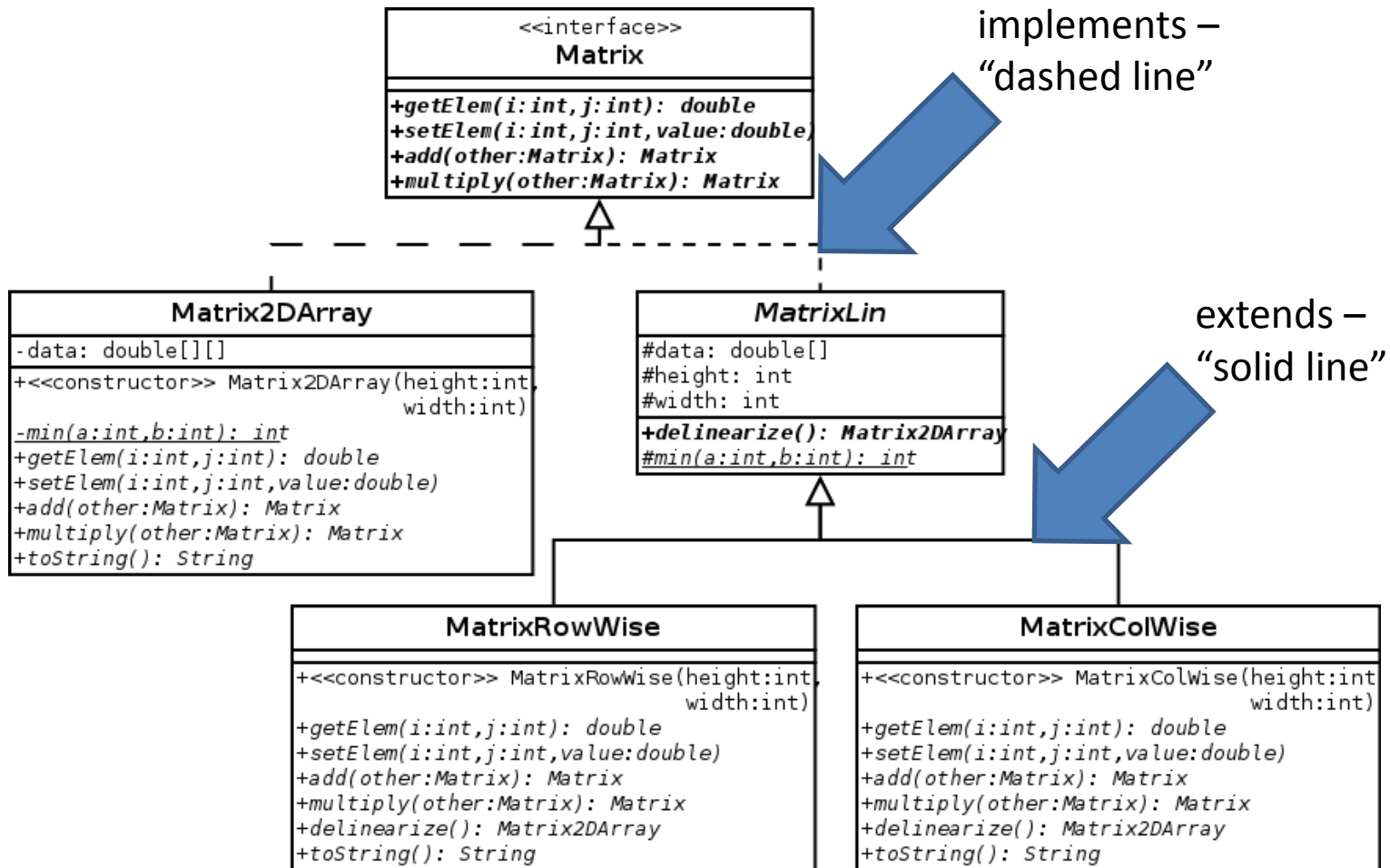
Mihail Georgiev

Creative Commons Attribution-Noncommercial 3.0 License

Unified Modeling Language

- standard way of drawing diagrams representing programs' structure and behaviour
- includes a way to draw diagrams of classes and class hierarchies
- code (e.g. Java code) is translated into diagram

Implements and Extends




Interface, Abstract Class, Class

Interface	Abstract Class	Class
write <<interface>>	written in <i>italics</i>	written upright (normally)
<pre> <<interface>> Matrix +getElem(i:int, j:int): double +setElem(i:int, j:int, value:double) +add(other:Matrix): Matrix +multiply(other:Matrix): Matrix </pre>	<pre> MatrixLin #data: double[] #height: int #width: int +<u>delinearize(): Matrix2DArray</u> #<u>min(a:int, b:int): int</u> </pre>	<pre> MatrixRowWise +<<constructor>> MatrixRowWise(height:int width:int) +getElem(i:int, j:int): double +setElem(i:int, j:int, value:double) +add(other:Matrix): Matrix +multiply(other:Matrix): Matrix +delinearize(): Matrix2DArray +toString(): String </pre>


Access Modifiers

- + means public
- - means private
- # means protected

Matrix2DArray
-data: double[][]
+<<constructor>> Matrix2DArray(height:int width:int)
<u>-min(a:int,b:int): int</u>
+getElem(i:int,j:int): double
+setElem(i:int,j:int,value:double)
+add(other:Matrix): Matrix
+multiply(other:Matrix): Matrix
+toString(): String

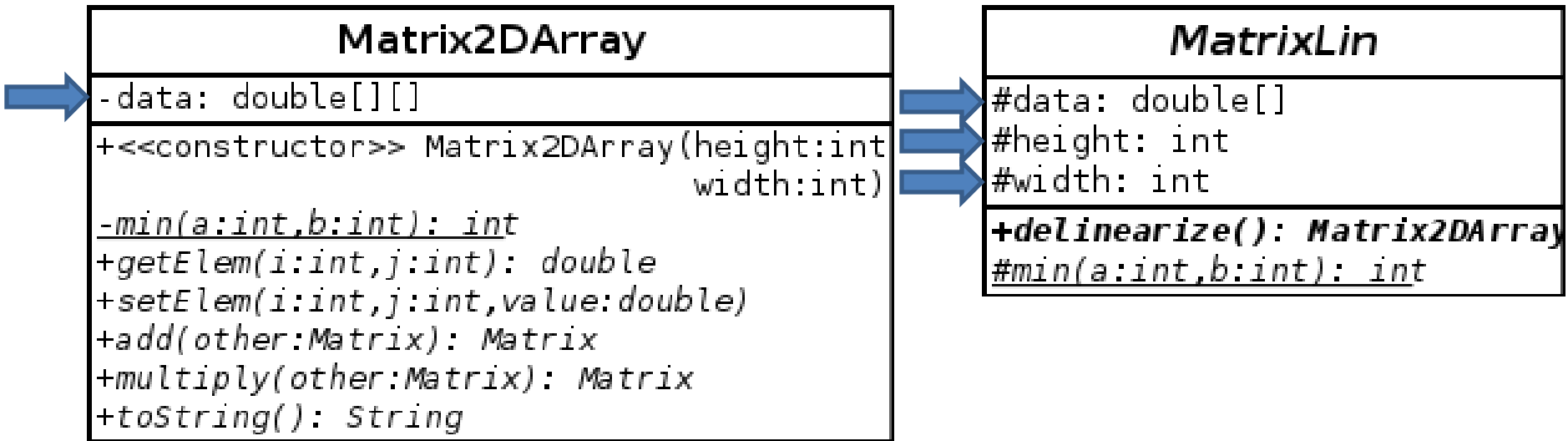


MatrixLin
#data: double[]
#height: int
#width: int
+delinearize(): Matrix2DArray
<u>#min(a:int,b:int): int</u>



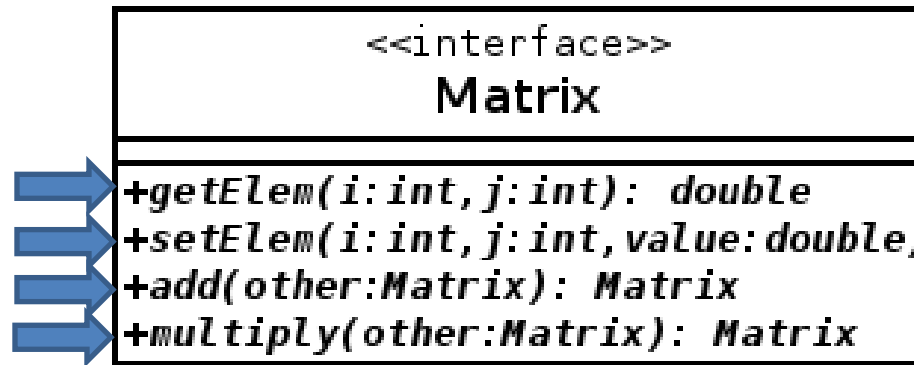
Fields

- written in regular font
- syntax is `name : type`
 - e.g. “`private double[][] data;`” becomes “`-data: double[][]`”



Abstract Methods

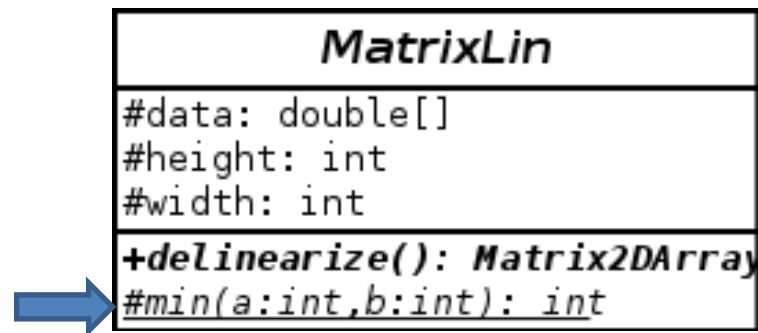
- written in ***bold italic***
- return types come after (nothing for `void`)



```
interface Matrix {  
    public double getElem(int i, int j);  
    public void setElem(int i, int j, double value);  
    public Matrix add(Matrix other);  
    public Matrix multiply(Matrix other);  
}
```

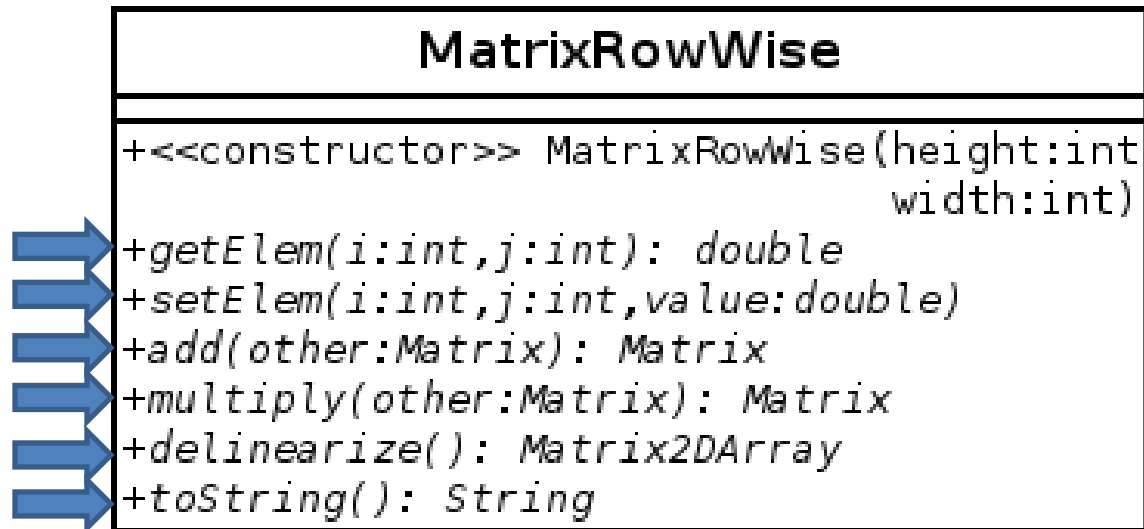
Implemented Methods

- written in *italic*
- implementations not shown in class diagrams
- static fields/methods are underlined



```
abstract class MatrixLin implements Matrix {  
    protected double[] data;  
    protected int height, width;  
    public abstract Matrix2DArray delinearize();  
    protected static int min(int a, int b) { ... }  
}
```

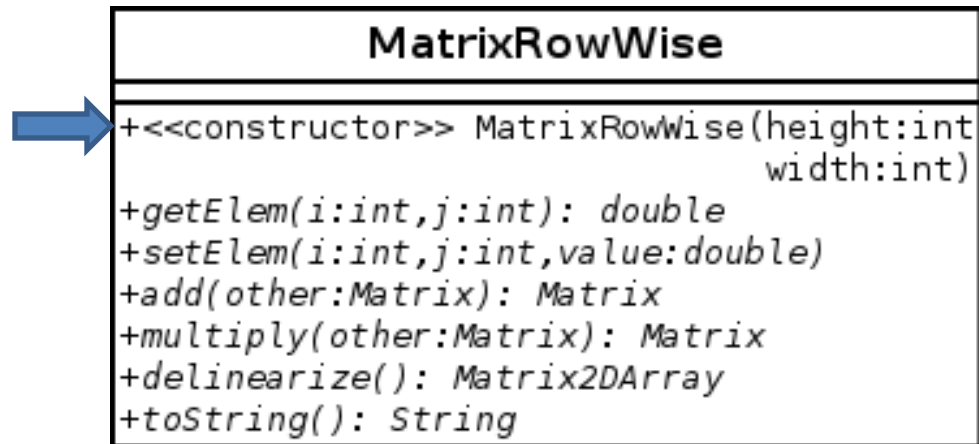

More Implemented Methods



```
class MatrixRowWise extends MatrixLin {  
    public MatrixRowWise(int height, int width) { ... }  
    public double getElem(int i, int j) { ... }  
    public void setElem(int i, int j, double value) { ... }  
    public Matrix add(Matrix other) { ... }  
    public Matrix multiply(Matrix other) { ... }  
    public Matrix2DArray delinearize() { ... }  
    public String toString() { ... }  
}
```

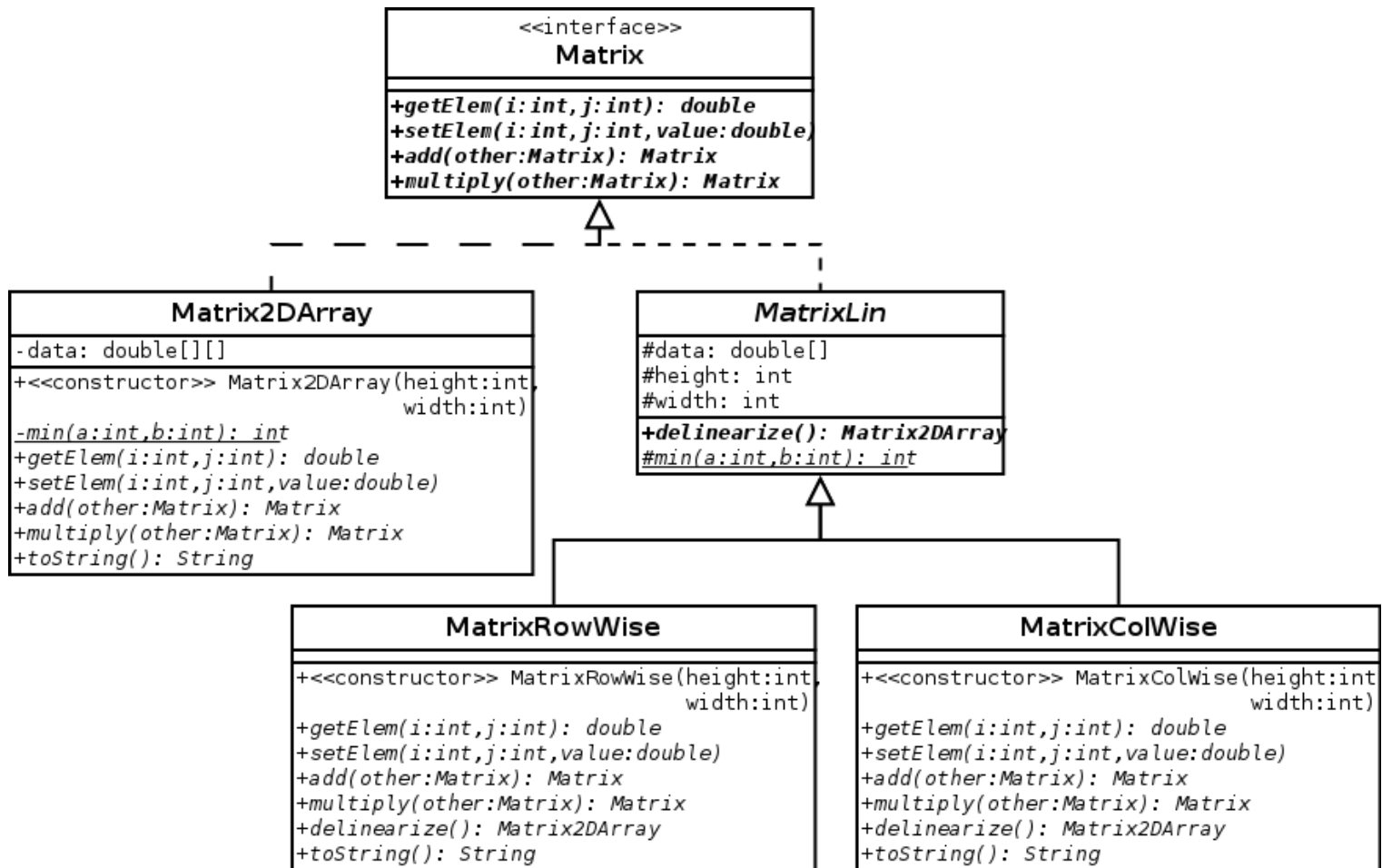
Constructors

- prepend <<constructor>>



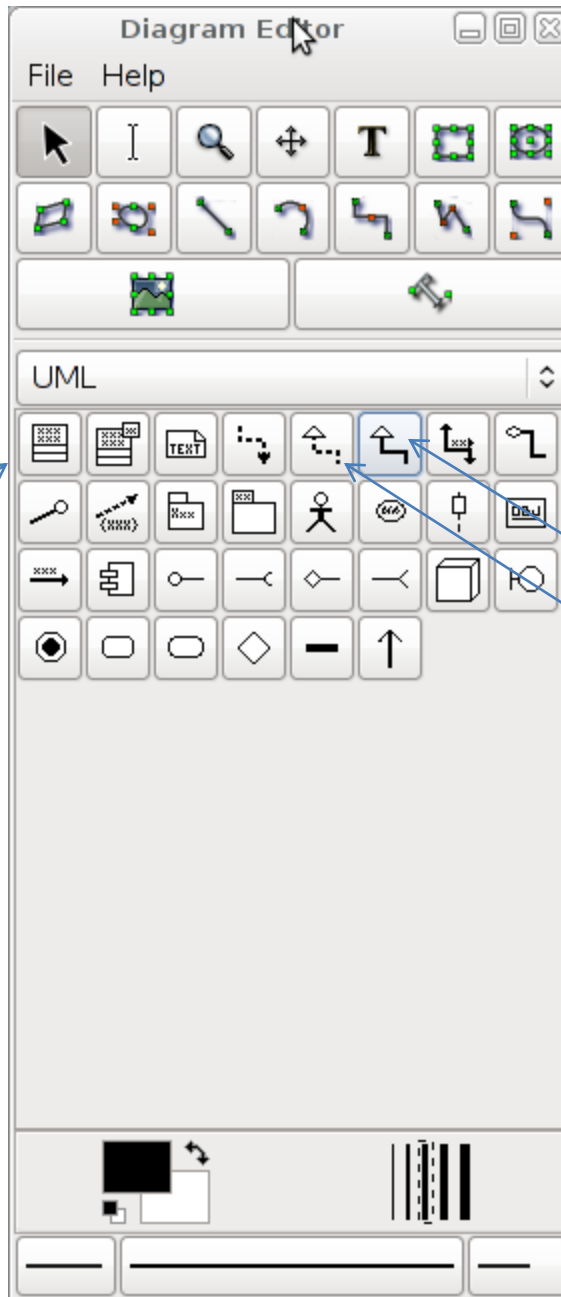
```
class MatrixRowWise extends MatrixLin {
    public MatrixRowWise(int height, int width) { ... }
    public double getElem(int i, int j) { ... }
    public void setElem(int i, int j, double value) { ... }
    public Matrix add(Matrix other) { ... }
    public Matrix multiply(Matrix other) { ... }
    public Matrix2DArray delinearize() { ... }
    public String toString() { ... }
}
```

Class Hierarchy (again)



dia (Diagram Editor)

- easy way of creating class diagrams
- create classes and relationships
- double-click on class to get a self-explanatory dialog that lets you modify it
- Google for anything that's unclear
- can export to PNG or PDF (PNG is a bit more straightforward)
- you can use any software you wish



create class or interface
HINT: modify
"stereotype" for
interfaces

extends
implements