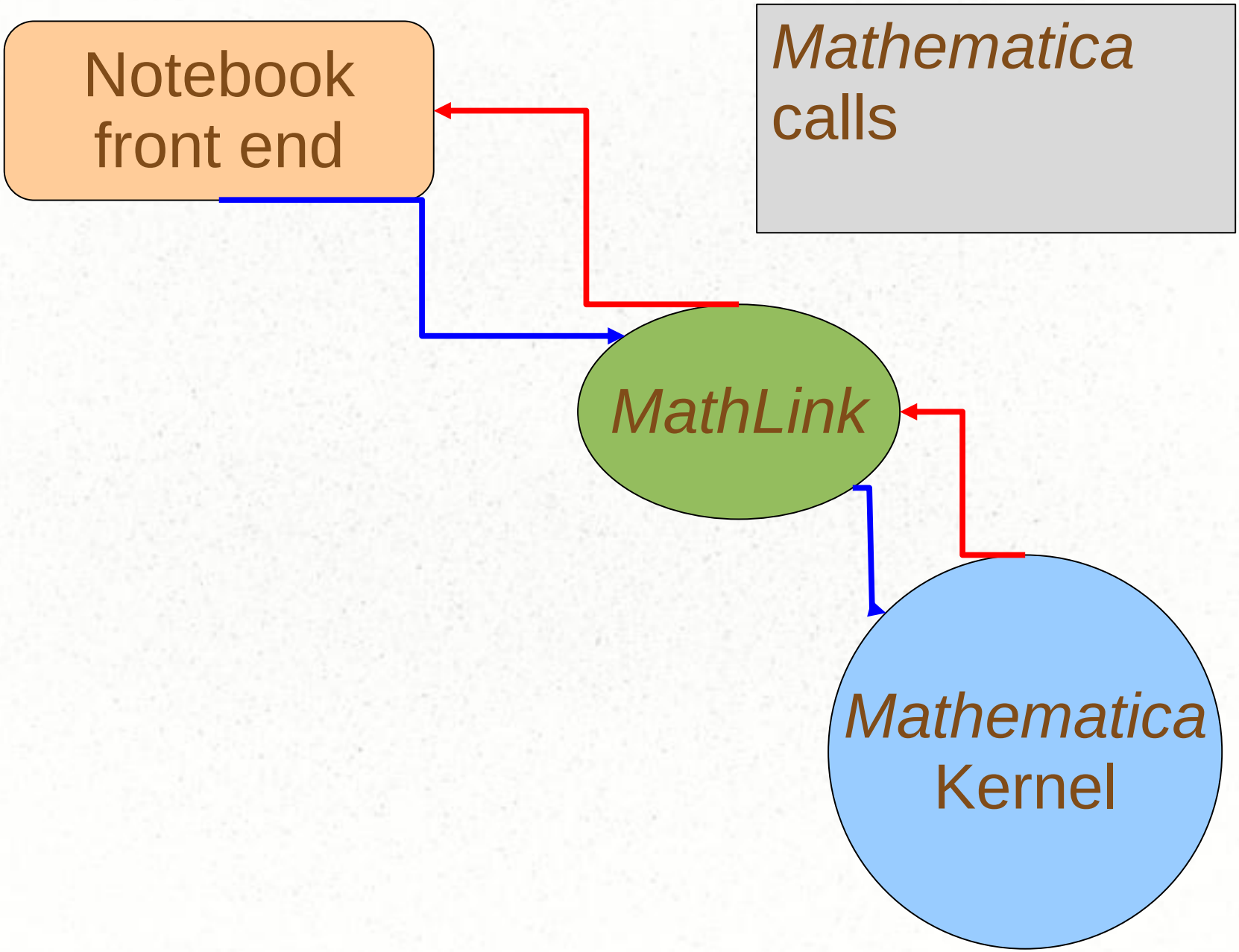
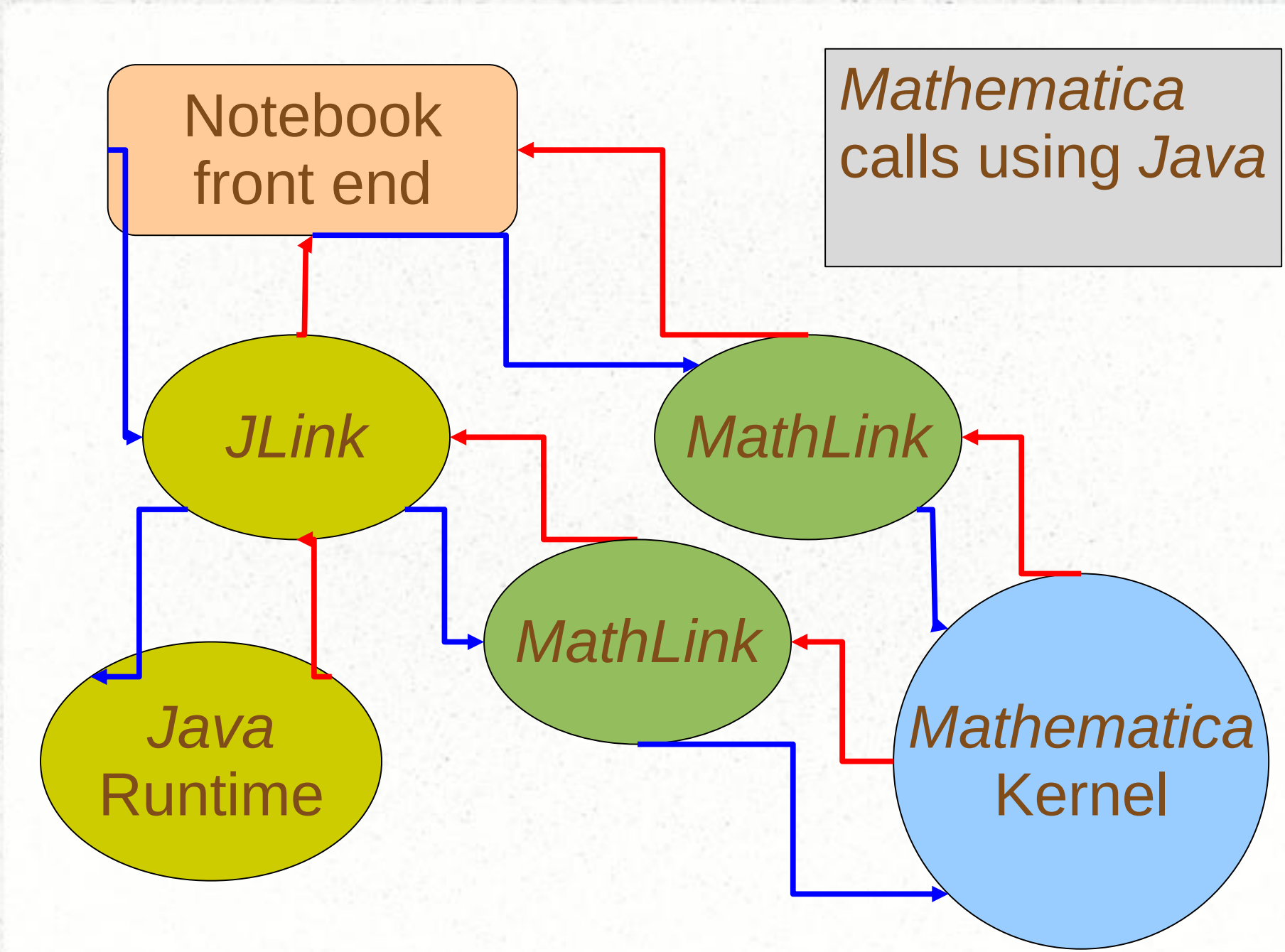
JLink

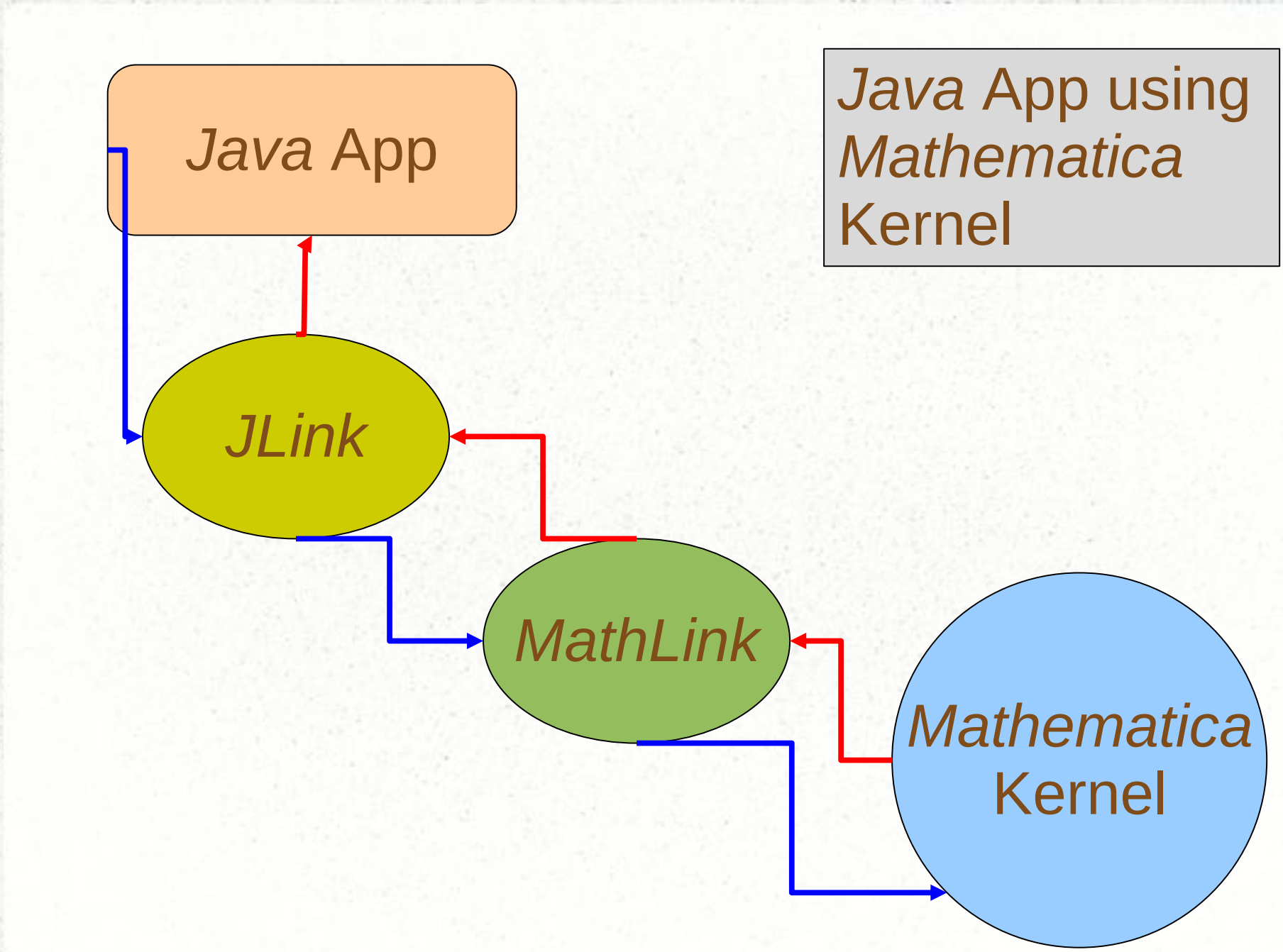
Linking *Mathematica* with *Java*
and the other way round…

Outline

- Introduction
- Calling *Java* from *Mathematica*
- Using *Mathematica Kernel* in *Java* applications


Notebook
front end
Mathematica
calls
MathLink
Mathematica
Kernel


Notebook
front end
Mathematica
calls using Java
JLink
MathLink
MathLink
Java
Runtime
Mathematica
Kernel


Java App
Java App using Mathematica Kernel
JLink
MathLink
Mathematica Kernel

Introduction

- *JLink* uses *MathLink* in its core and tries to hide it from user
- Allow users to (include *Java* in *Mathematica*):
  - Call *Java* methods from *Mathematica*
  - Create alternative front-ends for *Mathematica*
  - Create dialog boxes and other popup user interface elements for *Mathematica* programms

Introduction

- Allow users to (include *Mathematica* in *Java*):
  - Write *Java* programs that uses *Mathematica* services
  - Write applets that use *Mathematica* Kernels on the client server
  - Write servlets that make *Mathematica services available to HTTP clients*

Calling Java from Mathematica

- Load the *JLink* package:

```
In[2]:= Needs["JLink`"]
```

- Launching the *Java* Runtime:

```
In[3]:= InstallJava[]

Out[3]= LinkObject["C:\Program Files\Wolfram
        Research\Mathematica\8.0\SystemFiles\Java\Windows\bin\javaw"
        -classpath "C:\Program Files\Wolfram
        Research\Mathematica\8.0\SystemFiles\Links\JLink\JLink.jar" -Xmx256m
        -Djava.system.class.loader=com.wolfram.jlink.JLinkSystemClassLoader
        com.wolfram.jlink.Install -init "C:\Documents and Settings\atopalou\Local
        Settings\Temp\c\m-56e7bedf-52d0-4631-aee5-ffd7bbe90807", 4, 4]
```

Calling Java from Mathematica

- Loading a class:

```
In[11]:= urlClass = LoadJavaClass["java.net.URL"]

Out[11]= JavaClass[java.net.URL, <>]
```

- Creating objects:

```
In[15]:= frameClass = LoadJavaClass["java.awt.Frame"];
         frm = JavaNew[frameClass, "My Example"];
```

or

```
In[12]:= frame = JavaNew["java.awt.Frame", "My Example"]

Out[12]= «JavaObject[java.awt.Frame]»
```

Calling Java from Mathematica

- Creating objects with JavaNew[]
  - JavaNew[] returns a reference to the object
  - All the data (fields) stay to the *Java* side
  - That makes its call, *fast*
  - Except: the times that is more convenient to be returned “by value”, that means the times that there is a corresponding type between the two languages

Calling Java from Mathematica

- Conversion of types between *Java* and *Mathematica*:

Java type	*Mathematica type*
byte, char, short, int, long	Integer
Byte, Character, Short, Integer, Long, BigInteger	Integer
float, double	Real
Float, Double, BigDecimal	Real
boolean	True or False
String	String
array	List
controlled by user (see "Complex Numbers")	Complex
Object	JavaObject
Expr	any expression
null	Null

Calling Java from Mathematica

- Calling methods and accessing fields:

	constructors
Java:	`MyClass obj=new MyClass(args);`
Mathematica:	`obj=JavaNew["MyClass",args];`
	methods
Java:	`obj.methodName(args);`
Mathematica:	`obj@methodName[args]`
	fields
Java:	`obj.fieldName=1; value=obj.fieldName;`
Mathematica:	`obj@fieldName=1; value=obj@fieldName;`
	static methods
Java:	`MyClass.staticMethod(args);`
Mathematica:	`MyClass`staticMethod[args];`
	static fields
Java:	`MyClass.staticField=1; value=MyClass.staticField;`
Mathematica:	`MyClass`staticField=1; value=MyClass`staticField;`

Calling Java from Mathematica

- Releasing the reference to the objects
  - Must tell *Mathematica* (if needed) that the JavaObject is no longer used

`ReleaseJavaObject[`*obj*`]`	let Java know that you are done using *obj* in *Mathematica*
`ReleaseObject[`*obj*`]`	deprecated; replaced by `ReleaseJavaObject` in *J/Link* 2.0
`JavaBlock[`*expr*`]`	all novel Java objects returned to *Mathematica* during the evaluation of *expr* will be released when *expr* finishes
`BeginJavaBlock[]`	all novel Java objects returned to *Mathematica* between now and the matching `EndJavaBlock[]` will be released
`EndJavaBlock[]`	release all novel objects seen since the matching `BeginJavaBlock[]`
`LoadedJavaObjects[]`	return a list of all objects that are in use in *Mathematica*
`LoadedJavaClasses[]`	return a list of all classes loaded into *Mathematica*

Calling Java from Mathematica

- JavaBlocks
  - Pretty much as C functions, do the job, return an object, release any temporary data used

```
MyOtherFunc[args__] :=
    JavaBlock[
        Module[{obj},
            ...
            obj = JavaNew["java.awt.Frame"];
            ...
            Return[obj]
    (* OK: obj will not be released when JavaBlock finishes. *)
        ]
    ]
```

Calling Java from Mathematica

- Exceptions are handled by *JLink* automatically
  - If an uncaught exception is thrown in the *Java* side, a message will be printed in *Mathematica*

```
In[18]:= frameClass = LoadJavaClass["java.awt.frame"];
         frm = JavaNew[frameClass, "My Example"];

         Java::excptn : A Java exception occurred: java.lang.ClassNotFoundException: java.awt.frame
                at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
                at java.security.AccessController.doPrivileged(Native Method)
                at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
                at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
                at java.lang.ClassLoader.loadClass(ClassLoader.java:252)
                at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:320)
                at java.lang.Class.forName0(Native Method)
                at java.lang.Class.forName(Class.java:247).

         LoadJavaClass::fail : Java failed to load class java.awt.frame. >>
```

Calling Java from Mathematica

- Creating windows
  - "Modal" window: the *Mathematica* kernel waits for the window to be dismissed (input dialog)
    - DoModal[] / EndModal[]
    - SetModal[] (for MathFrames)
  - "Modeless" window: the *Mathematica* kernel is shared between the frond end notebook and the window (a window that allows user to load packages)
    - ShareKernel[] / UnshareKernel[] (default for *Mathematica* releases after 5.1)

Using Mathematica kernel in Java applications

- Import the *JLink.jar*
- *MathLink Interface*
  - The root of all link objects in *JLink*
- *KernelLink Interface*
  - Extends *MathLink*
  - Makes the assumption that the other side of the link is a *Mathematica* Kernel

Calling Java from Mathematica

- Creating Links with *MathLinkFactory*
  - *createMathLink()* (connect with other than *Mathematica* Kernel)
  - *createKernelLink(String cmdLine)*
  - *createKernelLink(String[] argv)*
  - all return the link object or throw *MathLinkException*

Using Mathematica kernel in Java applications

- Asking for evaluation
  - *MathLink:*
    - void *put(arg)* throws MathLinkException
    - different types of arg: int, long, double, String, boolean, Object
    - void *putFunction(String f, int argCount)* throws MathLinkException
    - int *getInteger()*, long *getLongInteger()*, etc...
    - All throw MathLinkExcetion
    - All public

Using Mathematica kernel in Java applications

- Asking for evaluation
  - *KernelLink:*
    - void *evaluate(String or expr)* throws MathLinkException
    - void *waitForAnswer()* throws MathKernelException
    - void *disgardAnswer()* throws MathLinkException
    - *"evaluateTo"* methods
      - No need to call *waitForAnswer()* or *disgardAnswer()*
      - Doesn't throw any Exception
      - Instead, returns null if there is an error

Using Mathematica kernel in Java applications

- Asking for evaluation
  - *KernelLink:*

```
String evaluateToInputForm(String s, int pageWidth);
String evaluateToInputForm(Expr e, int pageWidth);

String evaluateToOutputForm(String s, int pageWidth);
String evaluateToOutputForm(Expr e, int pageWidth);

byte[] evaluateToImage(String s, int width, int height);
byte[] evaluateToImage(Expr e, int width, int height);
byte[] evaluateToImage(String s, int width, int height, int dpi, boolean
useFrontEnd);
byte[] evaluateToImage(Expr e, int width, int height, int dpi, boolean useFrontEnd);

byte[] evaluateToTypeset(String s, int pageWidth, boolean useStdForm);
byte[] evaluateToTypeset(Expr e, int pageWidth, boolean useStdForm);

// Returns the exception that caused the most recent "evaluateTo" method to return
null
Throwable getLastError();
```

Conclusion

- *JLink* is a high-level protocol (based on *MathLink*) to make easier the communication between *Mathematica* and *Java*
- *Works both ways*

- *for more informations visit:*

  http://reference.wolfram.com/mathematica/JLink/tutorial/Overview.html