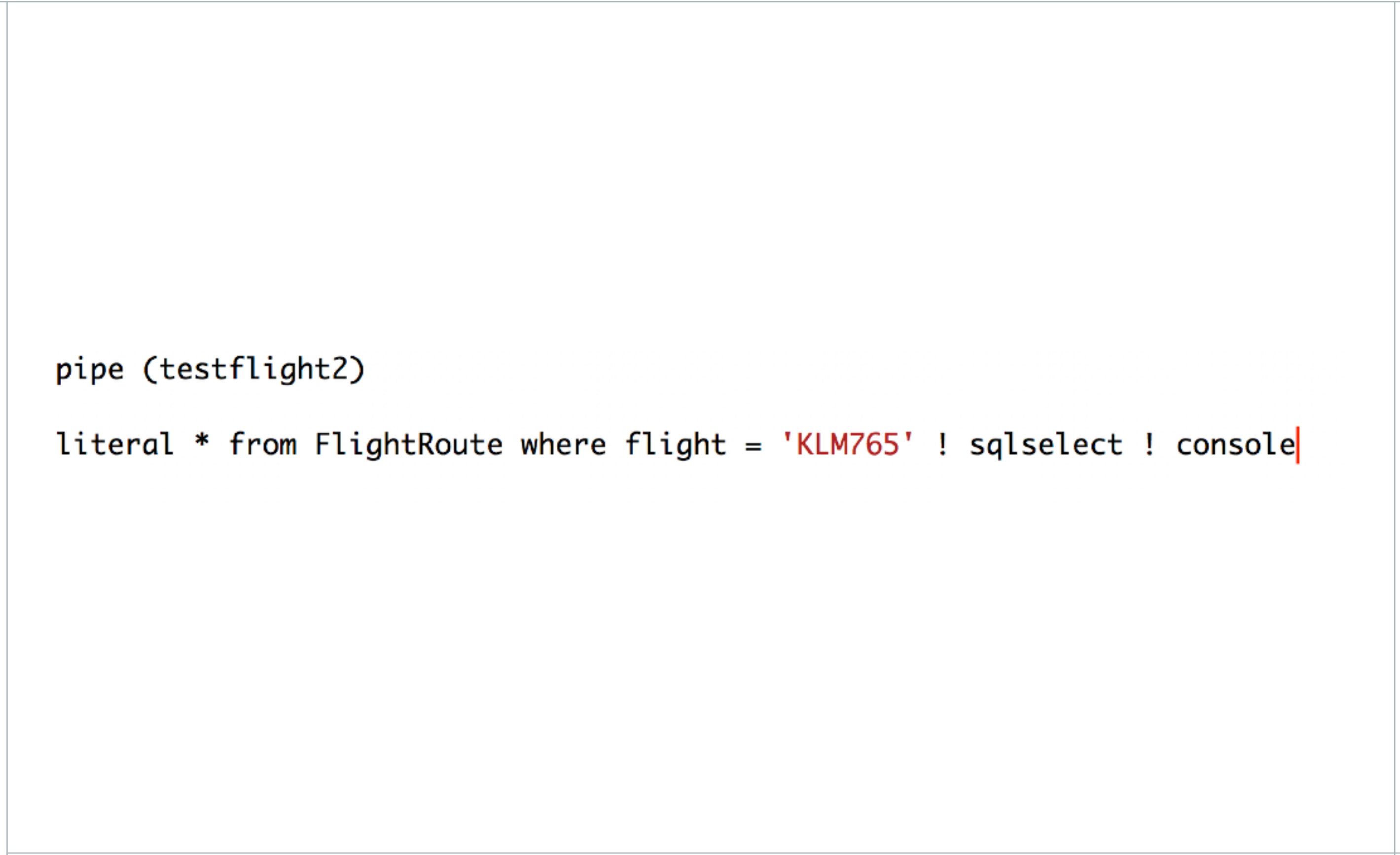
NetRexx 3.07 New Features

René Vincent Jansen
29th International Rexx Language Symposium 2018
Aruba, Dutch West Indies

Agenda

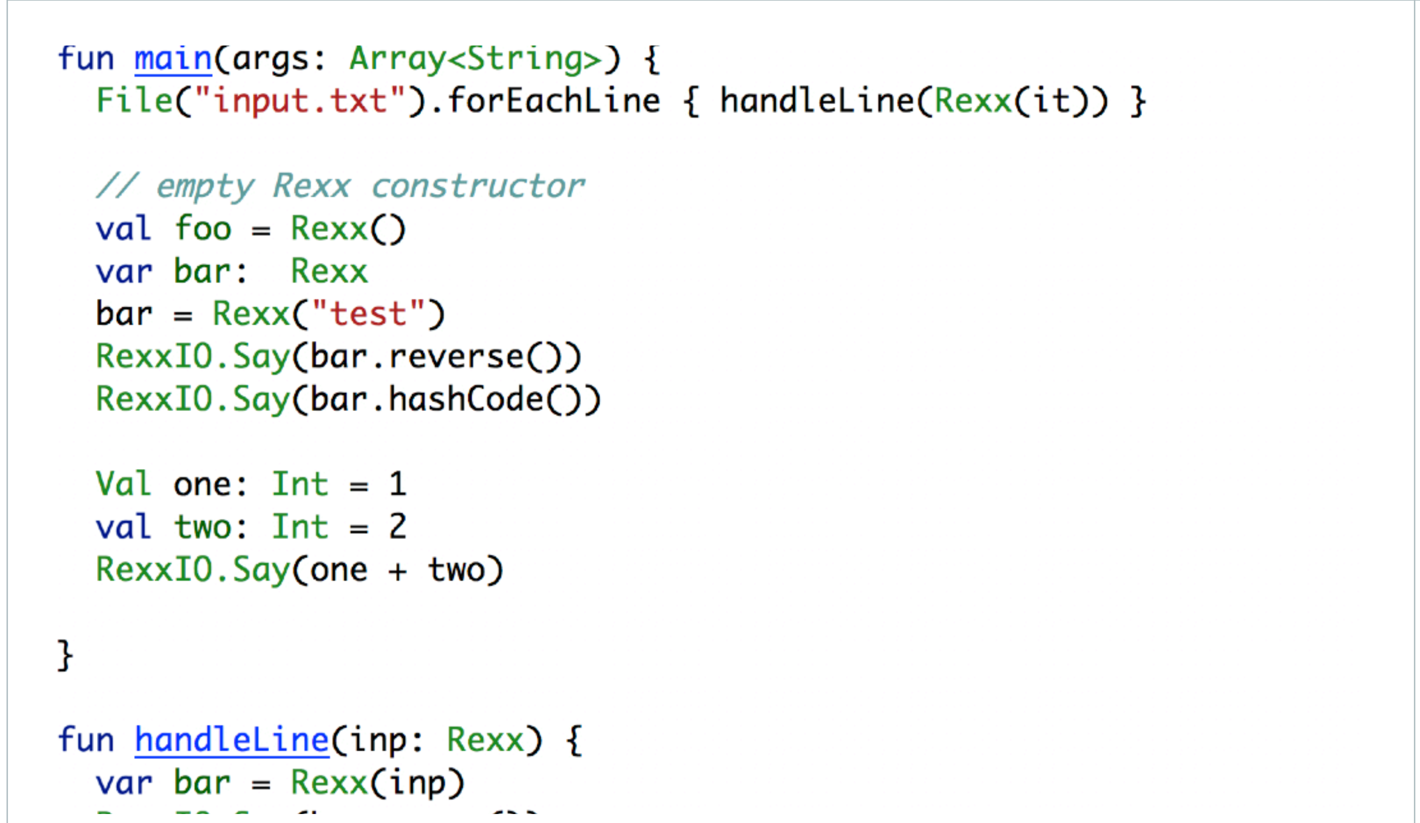


```
pipe (testflight2)

literal * from FlightRoute where flight = 'KLM765' ! sqlselect ! console
```

Pipelines: SQLSelect Stage

Pipe the sql statement into it

From 3.07

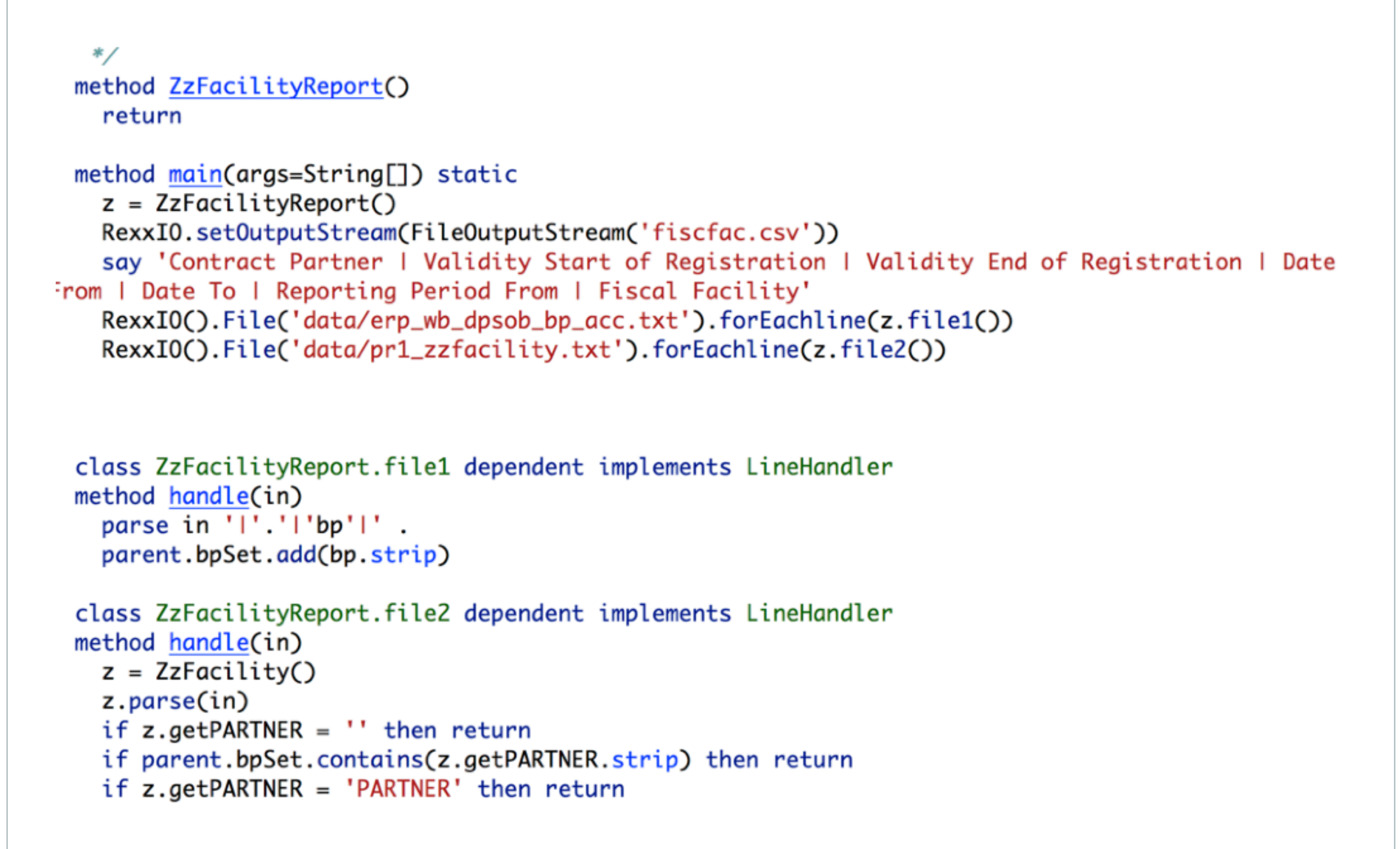


```
fun main(args: Array<String>) {
  File("input.txt").forEachLine { handleLine(Rexx(it)) }

  // empty Rexx constructor
  val foo = Rexx()
  var bar:  Rexx
  bar = Rexx("test")
  RexxIO.Say(bar.reverse())
  RexxIO.Say(bar.hashCode())

  Val one: Int = 1
  val two: Int = 2
  RexxIO.Say(one + two)

}

fun handleLine(inp: Rexx) {
  var bar = Rexx(inp)
```

Rexx() Constructor Unshared

Make it usable from Kotlin

From 3.07



```
   */
 method ZzFacilityReport()
   return

 method main(args=String[]) static
   z = ZzFacilityReport()
   RexxIO.setOutputStream(FileOutputStream('fiscfac.csv'))
   say 'Contract Partner | Validity Start of Registration | Validity End of Registration | Date
rom | Date To | Reporting Period From | Fiscal Facility'
   RexxIO().File('data/erp_wb_dpsob_bp_acc.txt').forEachline(z.file1())
   RexxIO().File('data/pr1_zzfacility.txt').forEachline(z.file2())



 class ZzFacilityReport.file1 dependent implements LineHandler
 method handle(in)
   parse in '|'.'|'bp'|' .
   parent.bpSet.add(bp.strip)

 class ZzFacilityReport.file2 dependent implements LineHandler
 method handle(in)
   z = ZzFacility()
   z.parse(in)
   if z.getPARTNER = '' then return
   if parent.bpSet.contains(z.getPARTNER.strip) then return
   if z.getPARTNER = 'PARTNER' then return
```

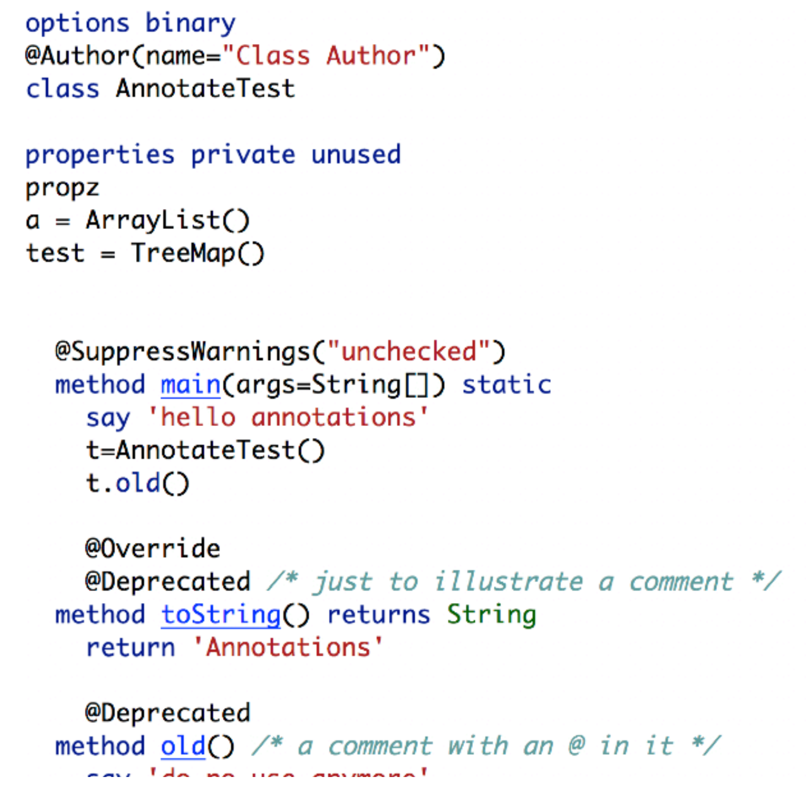
RexxIO Runtime Improvements

Set/Push/PopOutputStream

From 3.07

Annotations

From 3.06

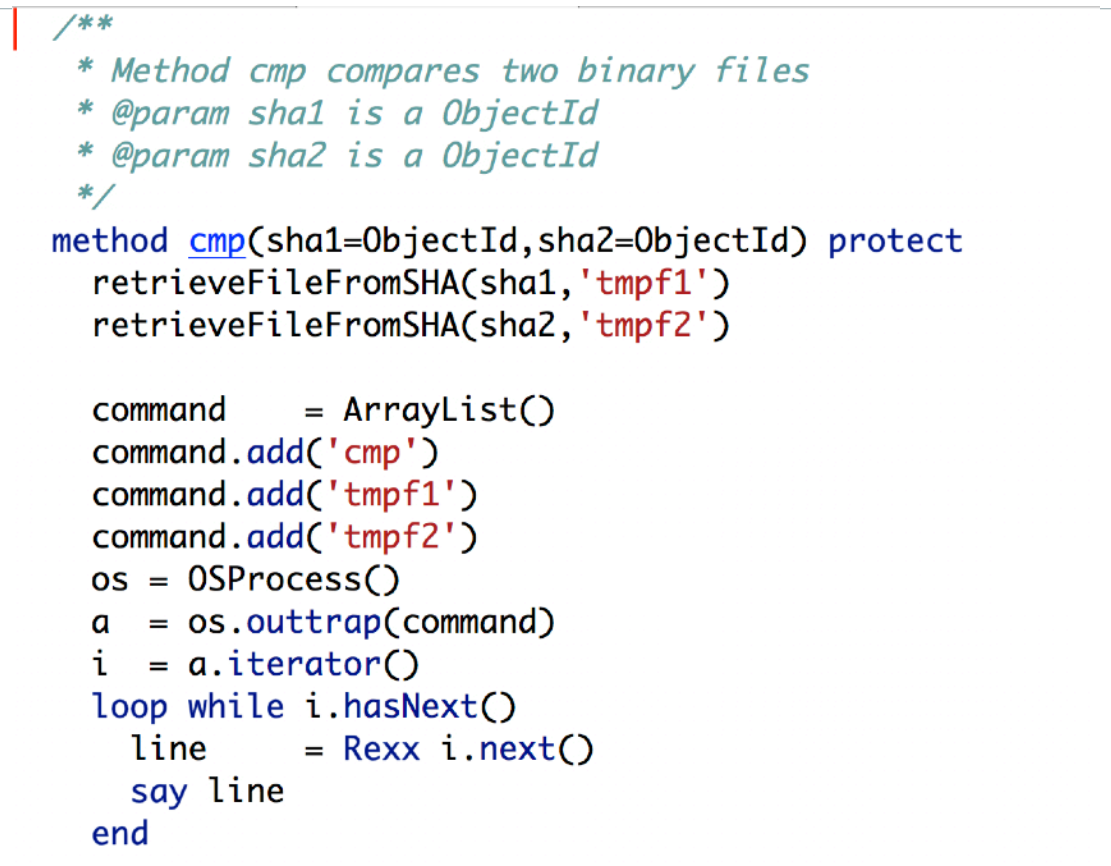


```
options binary
@Author(name="Class Author")
class AnnotateTest

properties private unused
propz
a = ArrayList()
test = TreeMap()


  @SuppressWarnings("unchecked")
  method main(args=String[]) static
    say 'hello annotations'
    t=AnnotateTest()
    t.old()

    @Override
    @Deprecated /* just to illustrate a comment */
  method toString() returns String
    return 'Annotations'

    @Deprecated
  method old() /* a comment with an @ in it */
```

OSProcess()

From 3.06

Runtime support for ADDRESS()

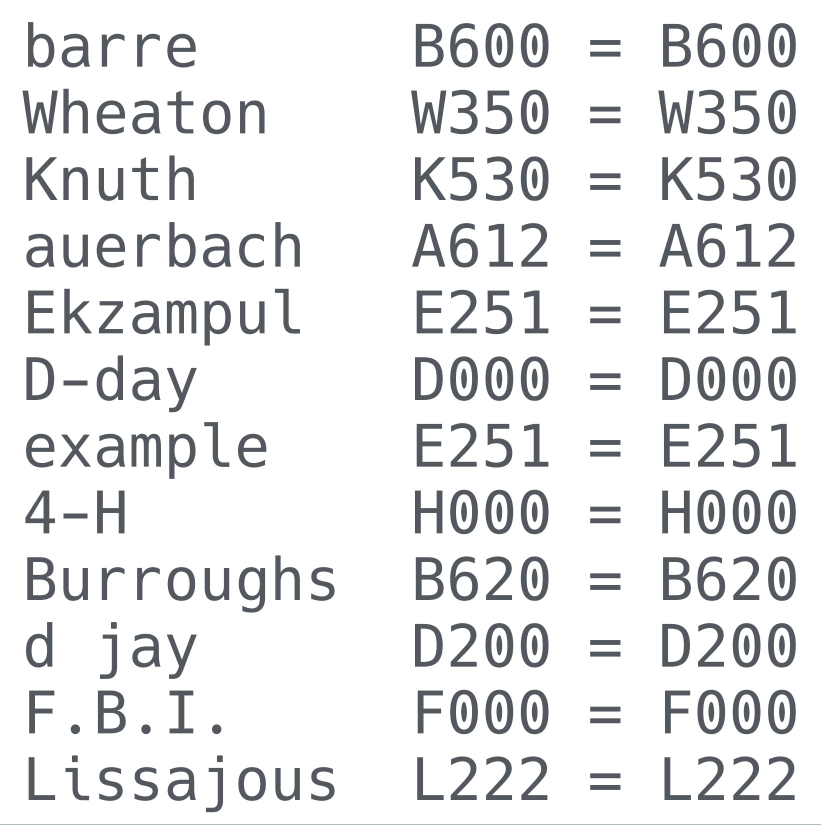


```
/**
 * Method cmp compares two binary files
 * @param sha1 is a ObjectId
 * @param sha2 is a ObjectId
 */
method cmp(sha1=ObjectId,sha2=ObjectId) protect
  retrieveFileFromSHA(sha1,'tmpf1')
  retrieveFileFromSHA(sha2,'tmpf2')

  command     = ArrayList()
  command.add('cmp')
  command.add('tmpf1')
  command.add('tmpf2')
  os = OSProcess()
  a  = os.outtrap(command)
  i  = a.iterator()
  loop while i.hasNext()
    line     = Rexx i.next()
    say line
  end
```

Rexx.soundex()

From 3.07

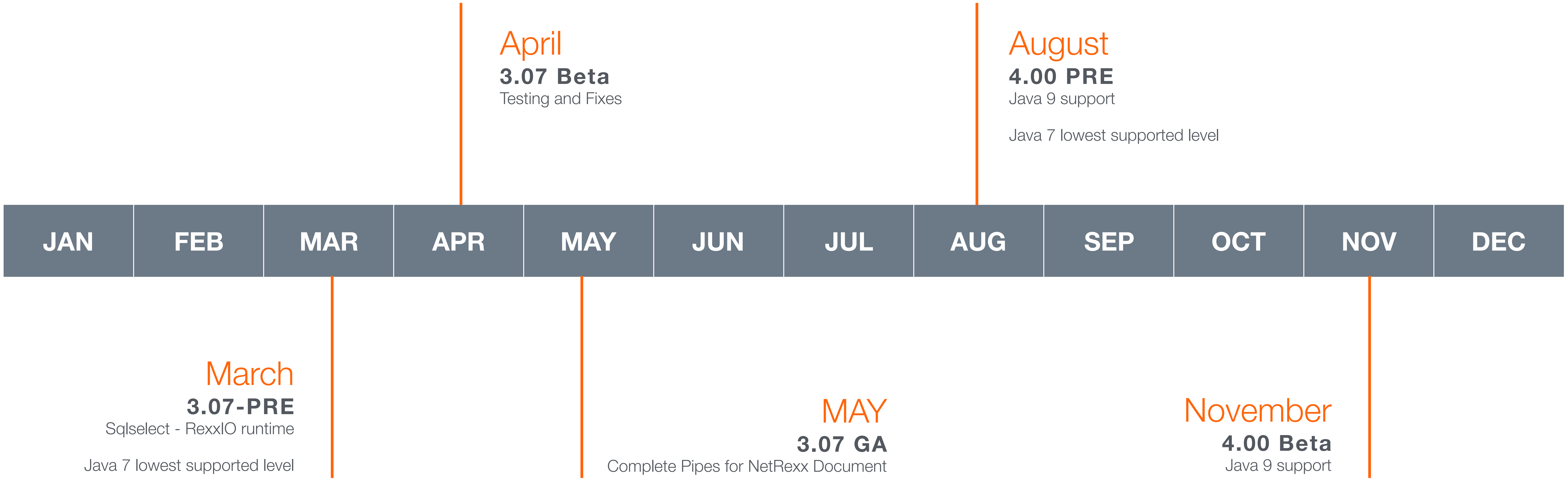
Method soundex() for Rexx strings



```
barre       B600 = B600
Wheaton     W350 = W350
Knuth       K530 = K530
auerbach    A612 = A612
Ekzampul    E251 = E251
D-day       D000 = D000
example     E251 = E251
4-H         H000 = H000
Burroughs   B620 = B620
d jay       D200 = D200
F.B.I.      F000 = F000
Lissajous   L222 = L222
```

Agenda

Release Timeline for NetRexx 3.0x - 4.00



March
3.07-PRE
Sqlselect - RexxIO runtime

Java 7 lowest supported level

April
3.07 Beta
Testing and Fixes

MAY
3.07 GA
Complete Pipes for NetRexx Document

August
4.00 PRE
Java 9 support

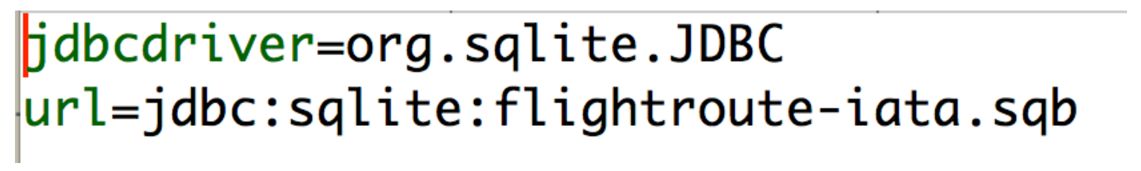
Java 7 lowest supported level

November
4.00 Beta
Java 9 support

SQLSelect stage of pipelines

- One of the first NetRexx programs I wrote
- It only accepted input from its commandline input
- It needed to accept input from a previous stage in the pipeline
- It nows does, after some 20 years

- This also prompted some experimentation with SQLite
- Which works wonderful with NetRexx

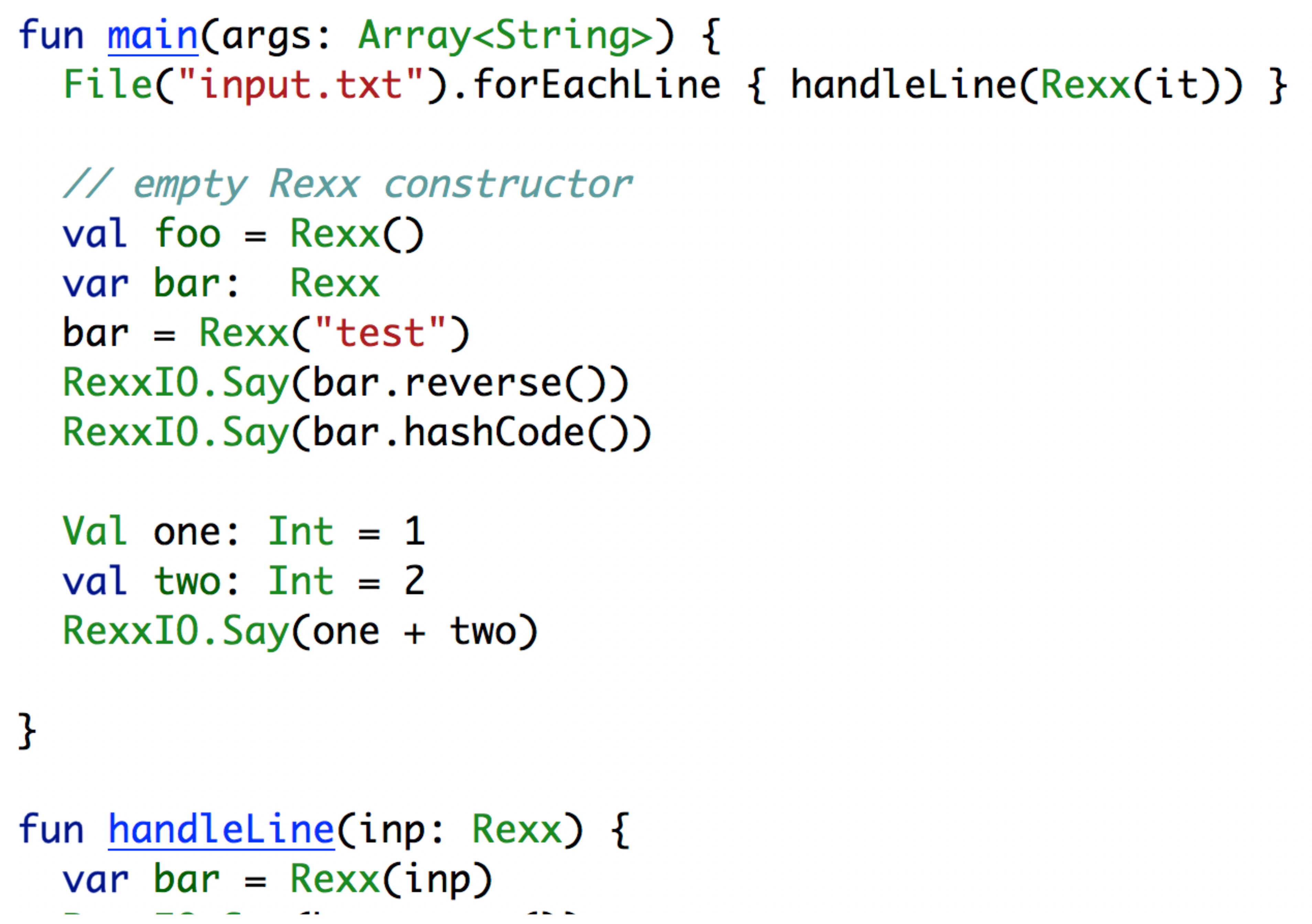


```
jdbcdriver=org.sqlite.JDBC
url=jdbc:sqlite:flightroute-iata.sqb
```

```
pipe (testflight2)

literal * from FlightRoute where flight = 'KLM765' ! sqlselect ! console
```

Rexx() constructor unshared

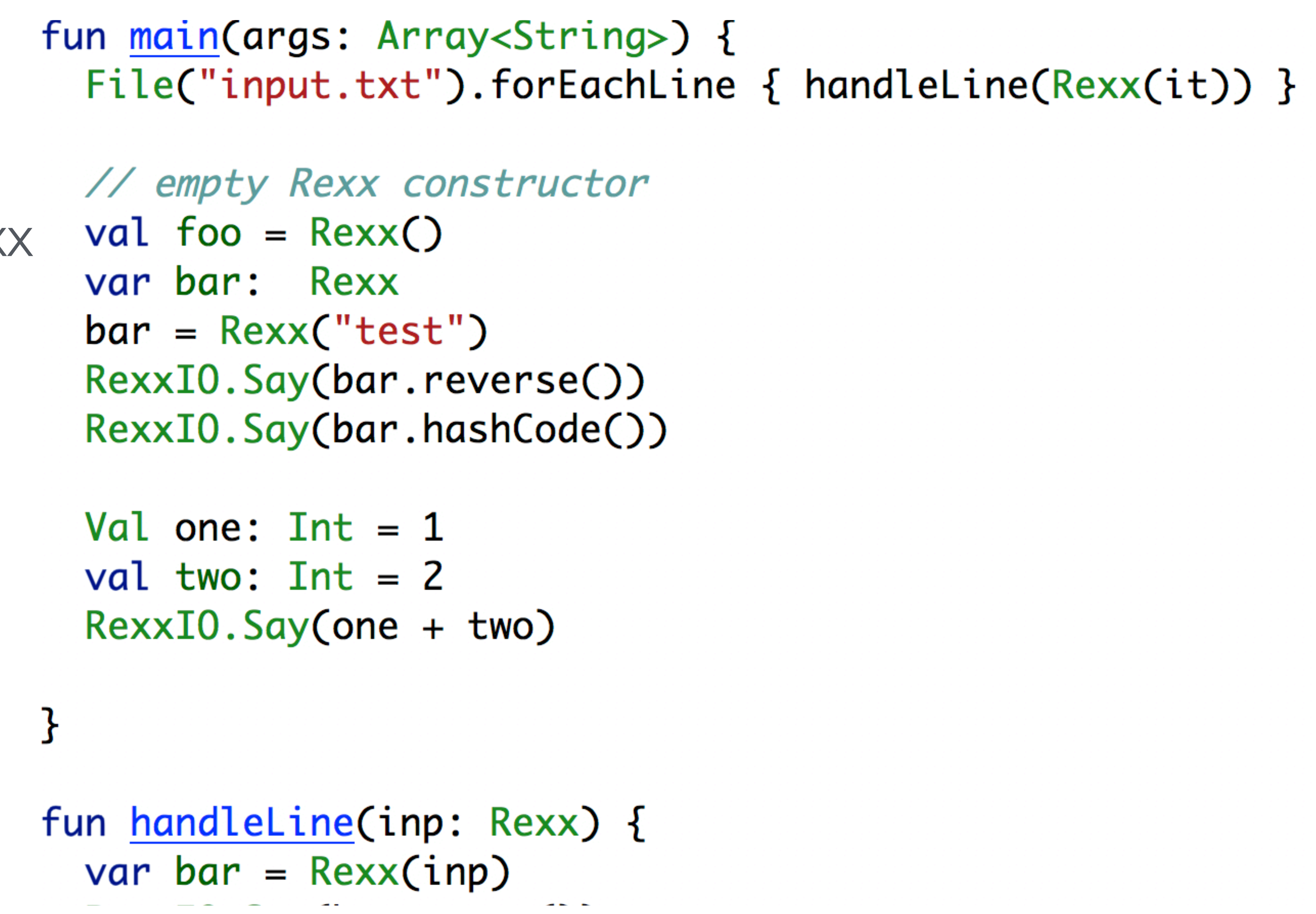
- Admittedly, this is not really useful for NetRexx but makes for a much better first impression when using the Rexx class in **Kotlin**
- Kotlin: upcoming, en-vogue language
  - It has a lot of the good things we know in NetRexx
  - Needs more investigation,
    - at least the first thing you try does not fail
  - If you are hired for a Kotlin project: yes, you can use Rexx
    - All the string functions we know from the 1980's there



```
fun main(args: Array<String>) {
  File("input.txt").forEachLine { handleLine(Rexx(it)) }

  // empty Rexx constructor
  val foo = Rexx()
  var bar:  Rexx
  bar = Rexx("test")
  RexxIO.Say(bar.reverse())
  RexxIO.Say(bar.hashCode())

  Val one: Int = 1
  val two: Int = 2
  RexxIO.Say(one + two)

}

fun handleLine(inp: Rexx) {
  var bar = Rexx(inp)
```

This one I liked in Kotlin

- Open a file with its name and specify in on line how and where to handle each record
  - It tempted me to do some work (at work) in Kotlin
  - Until I realised we can to this in about the same manner in NetRexx



```
fun main(args: Array<String>) {
  File("input.txt").forEachLine { handleLine(Rexx(it)) }

  // empty Rexx constructor
  val foo = Rexx()
  var bar:  Rexx
  bar = Rexx("test")
  RexxIO.Say(bar.reverse())
  RexxIO.Say(bar.hashCode())

  Val one: Int = 1
  val two: Int = 2
  RexxIO.Say(one + two)

}

fun handleLine(inp: Rexx) {
  var bar = Rexx(inp)
```

```
class ZzFacilityReport
  properties inheritable
  bpSet = TreeSet()

  /**
   * Default constructor
   */
  method ZzFacilityReport()
    return

  method main(args=String[]) static
    z = ZzFacilityReport()

    RexxIO.setOutputStream(FileOutputStream('fiscfac.csv'))

    RexxIO().File('data/erp_wb_dpsob_bp_acc.txt').forEachline(z.file1())
    RexxIO().File('data/pr1_zzfacility.txt').forEachline(z.file2())



  class ZzFacilityReport.file1 dependent implements LineHandler
  method handle(in)
    parse in '|'.'|'bp'|' .
    parent.bpSet.add(bp.strip)

  class ZzFacilityReport.file2 dependent implements LineHandler
  method handle(in)
    z = ZzFacility()
    z.parse(in)
    if z.getPARTNER = '' then return
    if parent.bpSet.contains(z.getPARTNER.strip) then return
    if z.getPARTNER = 'PARTNER' then return
    if z.getZZACTVTSTART = '' then z.setZZACTVTSTART('01.01.1900')
```

Oneliner file handler

Using a minor class and inheritable properties

Support for this in RexxIO runtime class

- Previously not documented, contains Say(), Ask(), AskOne()
- Method file()
  - Accepts a filename and constructs a BufferedReader
  - Returns RexxIO (static) to be able to chain methods
- Method forEachLine()
  - accepts any implementation of the LineHandler interface

```
package netrexx.lang

class LineHandler interface
  method handle(in=Rexx)
```

```
method File(nm) returns RexxIO
  do
    fileIn = BufferedReader(FileReader(nm))
  catch IOException
    return null
  end
  return this

method forEachline(c=LineHandler)
  do
    loop forever
      line = Rexx fileIn.readLine()
      if line = null then leave
      c.handle(line)
    end
  catch IOException
  end -- do
```

Other RexxIO changes: OutputStream

- I noticed how everything that is prototyped with **say** always ends up needing to be written to a file
  - We can redirect, but that means all System.out and System.err ends up in between the output
  - We can open a PrintWriter and change all say statements to println()
  - Opening a file in a number of lines and changing all say statements is drudge work
  - How about if we could just **say** something (in)to a file
  - Thats is what the experiment is about

setOutputStream

- You can set an OutputStream on the RexxIO class (which is static)
  - For the first time, you can switch between stdout and stderr
  - You may also specify a FileOutputStream
  - All **say** output from that moment on will go to that file
  - Reset it by setting it back to System.out

  - Every **say** always flushes the output stream (and always did)
  - Even when this is taken into account:
    - On systems with slow consoles (read: windows):
      - The speedup is stunning when writing to a file

```
class ZzFacilityReport
  properties inheritable
  bpSet = TreeSet()

  /**
   * Default constructor
   */
  method ZzFacilityReport()
    return

  method main(args=String[]) static
    z = ZzFacilityReport()

    RexxIO.setOutputStream(FileOutputStream('fiscfac.csv'))

    RexxIO().File('data/erp_wb_dpsob_bp_acc.txt').forEachline(z.file1())
    RexxIO().File('data/pr1_zzfacility.txt').forEachline(z.file2())


  class ZzFacilityReport.file1 dependent implements LineHandler
  method handle(in)
    parse in '|'.'|'bp'|' .
    parent.bpSet.add(bp.strip)

  class ZzFacilityReport.file2 dependent implements LineHandler
  method handle(in)
    z = ZzFacility()
    z.parse(in)
    if z.getPARTNER = '' then return
    if parent.bpSet.contains(z.getPARTNER.strip) then return
    if z.getPARTNER = 'PARTNER' then return
    if z.getZZACTVTSTART = '' then z.setZZACTVTSTART('01.01.1900')
```

What if we want some **say** output going to more outputstreams?

- To make **say** output go to more streams (stdout, a file, stderr) we can:
  - pushOutputStream
    - Add one outputstream
  - popOutputStream
    - Remove the latest added outputstream
  - StdOut in RexxIO is now a ConcurrentLinkedDeque
    - Which should make it reasonable thread safe

```
method setOutputStream(out=OutputStream) static
    StdOut.clear()
    StdOut.push(PrintWriter(out))

method pushOutputStream(out=OutputStream) static
    StdOut.push(PrintWriter(out))

method popOutputStream() static
  do
    StdOut.pop()
  catch java.util.NoSuchElementException
    StdOut.push(PrintWriter(System.out))
  end
```

Annotations (in 3.06)

- Adding annotations was not avoidable due to the large amount of Java classes using mandatory annotations - jUnit, vaadin, Jakarta Spring
- Unlike generics, the way to handle these in NetRexx without language support would be much more complex (though not impossible, everything becomes a method call in the end)
- For this reason, the parser was adapted to recognise and pass through @annotations
- This was not easy and there still are some snags
- Most of the things you need do work, though

```
options binary
@Author(name="Class Author")
class AnnotateTest

properties private unused
propz
a = ArrayList()
test = TreeMap()


 @SuppressWarnings("unchecked")
 method main(args=String[]) static
  say 'hello annotations'
  t=AnnotateTest()
  t.old()

  @Override
  @Deprecated /* just to illustrate a comment */
 method toString() returns String
  return 'Annotations'

  @Deprecated
 method old() /* a comment with an @ in it */
  say 'do no use anymore'
```

OSProcess - Runtime support for ADDRESS and OUTTRAP (since 3.06)

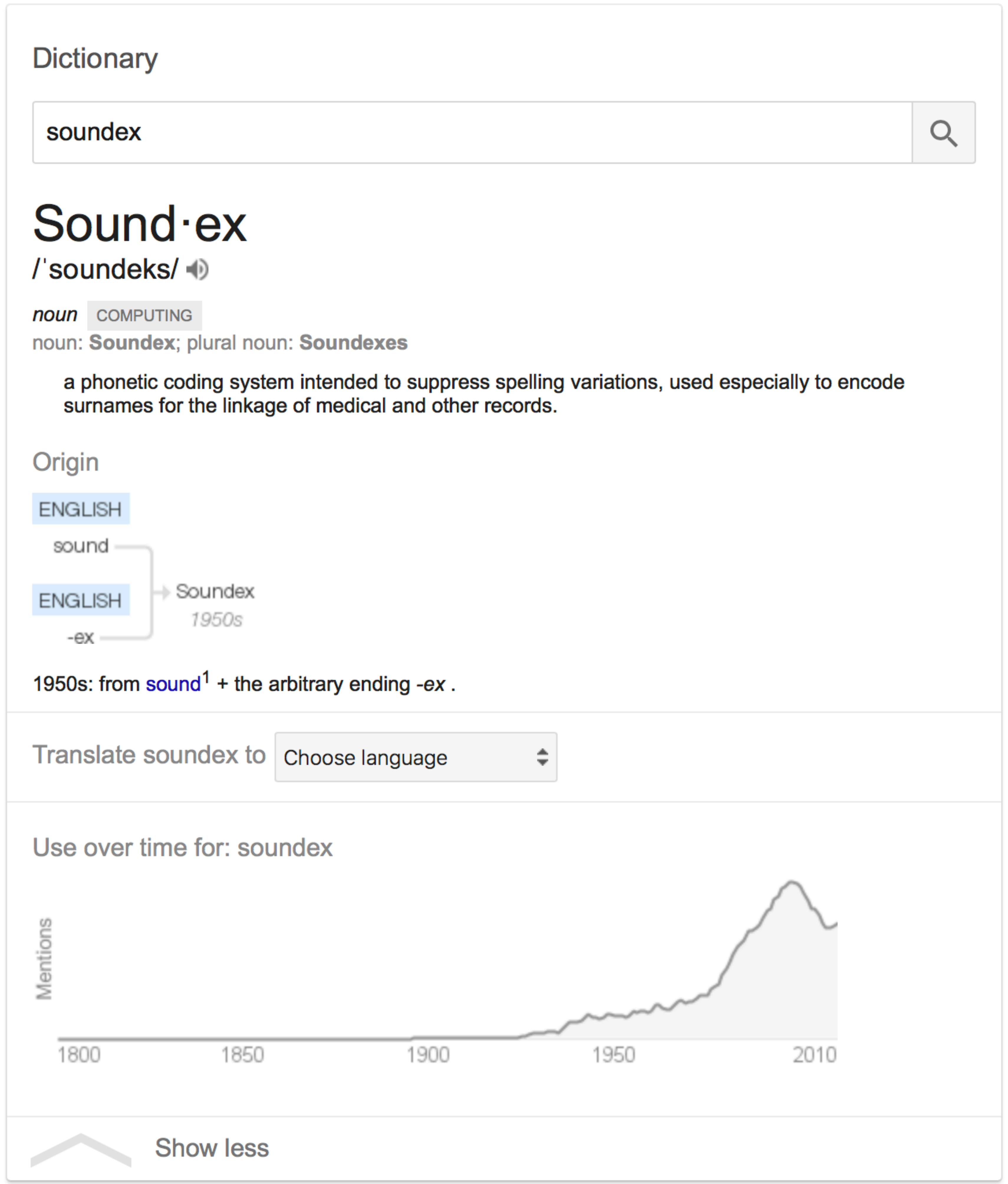
- NetRexx was designed with the following assumptions
  - Java is going to be used for I/O
  - Java interfaces are going to be used for native functionality
  - Java handles pretty much everything and native is not needed
  - Here NetRexx diverges from other dialects
  - Scripting is closely related to the (OS/Platform) environment
  - These can be building blocks for an ADDRESS command
  - Let’s see what ooRexx is doing with ADDRESS WITH

```
/**
  * Method cmp compares two binary files
  * @param sha1 is a ObjectId
  * @param sha2 is a ObjectId
  */
method cmp(sha1=ObjectId,sha2=ObjectId) protect
  retrieveFileFromSHA(sha1,'tmpf1')
  retrieveFileFromSHA(sha2,'tmpf2')

  command     = ArrayList()
  command.add('cmp')
  command.add('tmpf1')
  command.add('tmpf2')
  os = OSProcess()
  a  = os.outtrap(command)
  i  = a.iterator()
  loop while i.hasNext()
    line      = Rexx i.next()
    say line
  end
```

Soundex (3.07)

- Rexx variables have to ways for comparison
  - A strict (==) comparator
  - A less strict (more what a human would do) comparator (=)

  - But it misses a loose comparator
  - For this, the Soundex algorithm is the standard
  - For data cleansing operations this was needed so often, it was put as a method on the Rexx string
- Why put it in the runtime
  - the algorithm is just not trivial enough to assume that language users will easily roll their own
  - It is a good addition to the other two comparators


Dictionary

soundex

Sound·ex
/ˈsoundeks/

noun COMPUTING
noun: **Soundex**; plural noun: **Soundexes**

a phonetic coding system intended to suppress spelling variations, used especially to encode surnames for the linkage of medical and other records.

Origin

1950s: from sound[1] + the arbitrary ending *-ex* .

Translate soundex to Choose language

Use over time for: soundex

Soundex example & testset

- We need to normalize a database that has a free field for street name
- We know people have put in various forms of 'unknown'
- We know that **'unknown'.soundex()** is U525
- We now find:

Unkown/ Onbekend
Unknown\ Onbekend
Unknown/Onbekend
Unknown/Onbeken
Unknown/ Onbekend
Unknown Onbekend
UNKNOWN /ONBEKND
Unknown /Onbekend
Unknown / Onbekend
unknown /onbekend
Unknown
Unknowm/ Onbekend
Unknnown/Onbekend

```
barre      B600 = B600
Wheaton    W350 = W350
Knuth      K530 = K530
auerbach   A612 = A612
Ekzampul   E251 = E251
D-day      D000 = D000
example    E251 = E251
4-H        H000 = H000
Burroughs  B620 = B620
d jay      D200 = D200
F.B.I.     F000 = F000
Lissajous  L222 = L222
Burrows    B620 = B620
coöp       C100 = C100
de la Rosa D462 = D462
Gauss      G200 = G200
Donnell    D540 = D540
Ghosh      G200 = G200
Dracula    D624 = D624
Ellery     E460 = E460
he         H000 = H000
Gutierrez  G362 = G362
Drakula    D624 = D624
Williams   W452 = W452
Heilbronn  H416 = H416
Du Pont    D153 = D153
Robert     R163 = R163
Pfister    P236 = P236
Moskowitz  M232 = M232
Euler      E460 = E460
Hilbert    H416 = H416
Rupert     R163 = R163
Uhrbach    U612 = U612
Moskovitz  M213 = M213
Lukasiewic L222 = L222
Woolcock   W422 = W422
Tymczak    T522 = T522
Rubin      R150 = R150
Swhgler    S460 = S460
```

Soundex implementation

- Somewhat dependent on language
- The canonical form is for English
- The numbers are dependent on pronunciation
- In case of popular demand:
    - We need to make these strings swappable

```
/** soundex returns the normalized soundex value of the string */
method soundex() returns Rexx
  in = this.upper()
  old_alphabet= 'AEIOUYHWBFPVCGJKQSXZDTLMNR'
  new_alphabet= '@@@@@@**111122222222334556'
  word=Rexx('')
  loop i=1 for intlength()
    tmp_=in.substr(i, 1)
    if tmp_.datatype('M')  then word=word||tmp_
  end
  value=word.strip().left(1)
  word=word.translate(new_alphabet, old_alphabet)
  prev=value.translate(new_alphabet, old_alphabet)
  loop j=2  to word.length()
    q=word.substr(j, 1)
    if q\==prev & q.datatype('W')  then do
      value=value || q;  prev=q
    end
    else if q=='@'  then prev=q
  end
  return value.left(4,0)
```

NetRexx 4.00

- NetRexx 3.X does not run on Java 9
- This is due to an incompatible change by Java - the Oracle team
- Reason for the change is the module system
- NetRexx reads all jars and zip, and classes on the classpath for every compilation
  - This has become impossible now
- Later this week we will have a workshop on reflection and method handles
- Results of this workshop will be highly important to the future of NetRexx

Thank you for your attention

- Q? rvjansen@xs4all.nl or president@rexxla.org

29th International Rexx Language Symposium

Aruba, Dutch West Indies