**HÜ 09 Gr. 03** (2 in 1 Bsp):

Input:

command="curl --silent [https://bach.wu.ac.at/z/start"](https://bach.wu.ac.at/z/start%22)
outArr=.array~new -- array for stdout
ADDRESS SYSTEM command WITH OUTPUT USING (outArr)
bach=outArr~makeString -- turn array into string

parse arg bach
/\* create an instance of the JAXP DocumentBuilderFactory \*/
factory=bsf.loadClass("javax.xml.parsers.DocumentBuilderFactory")~newInstance
factory~setNamespaceAware(.true) -- set desired parser to namespace aware
parser=factory~newDocumentBuilder -- create the parser from the factory
eh=.errorHandler~new -- create an error handler Rexx object
-- wrap up the Rexx error handler as a Java object
javaEH=BsfCreateRexxProxy(eh, , "org.xml.sax.ErrorHandler")
parser~setErrorHandler(javaEH) -- set the error handler for this parser
rootNode=parser~parse(bach) -- parse the file, returns root node
/\* make important constants available via .local \*/
clzDomNode=bsf.loadClass("org.w3c.dom.Node") -- load the Java interface class
.local~CDATA\_SECTION\_NODE=clzDomNode~CDATA\_SECTION\_NODE -- save field value
.local~TEXT\_NODE =clzDomNode~TEXT\_NODE -- save field value
/\* now collect all text and CDATA nodes and display them \*/
call followNode rootNode
::requires BSF.CLS /\* get the Java support \*/
::routine followNode /\* walks the document tree recursively \*/
use arg node
call processNode node -- process received node
if node~hasChildNodes then
do
children=node~getChildNodes -- get NodeList
loop i=0 to children~length-1 -- 0-based indexes!
call followNode children~item(i) -- recurse
end
end
::routine processNode /\* processes each node \*/
use arg node
nodeType=node~getNodeType -- get type of node
if nodeType=.text\_node | nodeType=.cdata\_section\_node then
say pp(node~nodeValue)
::class ErrorHandler -- a Rexx error handler ("org.xml.sax.ErrorHandler")
::method unknown /\* handles "warning", "error" and "fatalError" events \*/
use arg methName, argArray -- arguments from the Java SAX parser
exception=argArray[1] -- retrieve SAXException argument
.error~say(methName":" -
"line="exception~getLineNumber",col="exception~getColumnNumber":" -
pp(exception~getMessage))

Output:
Fehlermeldung:

