From zero to zHero: Java Batch development for IBM System z

Martina Schmidt
STG Technical Sales Mainframe Systems
Mail: Martina.Schmidt@de.ibm.com

Table of contents

Disclaimer

This document was created as an instruction guide for the Java Batch workshop in Germany and is no official IBM reference guide.

1 Introduction and general hints

This lab guide shell is intended as an introduction for people to become familiar with Java on z/OS.
You should not need any Java skill to run through these labs, but basic knowledge of z/OS and Mainframe technologies is required to understand them.

Here are some general hints that you should read before you begin with this lab:

- Java is case sensitive. So always be careful when you type Java source code!
- This lab guide is also available as PDF-document on the target machine under /u/fhbmstr/jbatch/docs/howto/lab_guide.pdf.
- When you copy content from the lab guide and paste it into development tools, always be aware that some line breaks in this document might cause failures. In that case, remove the line break.
- Here are some general hints for ISPF:
    - When you are requested to press <enter>, please press the <right CTRL> key!
    - If you see stars (three stars!) *** please press <enter>
    - You can only enter data in special screen areas. Use the <TAB> key to go to the next typo field
    - If you try to enter data in a non-typo area, your keyboard will be locked (see red sign, last line, left side ← ☺ →)

2 Getting started

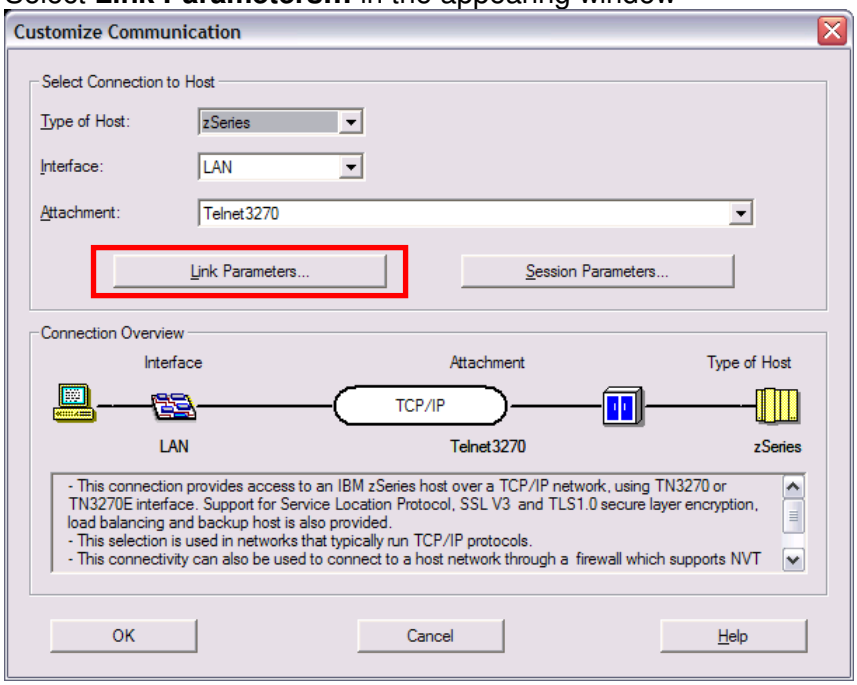
This lab explains how to connect to the workshop host system and lists general information on the system structure.

2.1 Setup PCOMM and start TSO

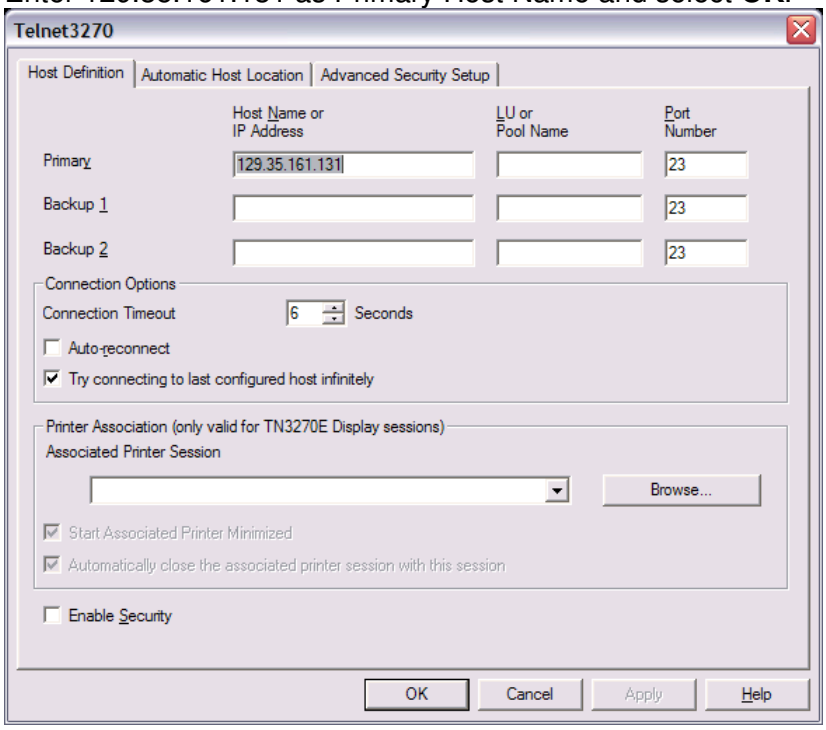
In this lab, we will establish a new TSO connection via PCOMM to the workshop host system.

Note: if you are using Host on Demand, please go directly to chapter 2.2 on page 8.

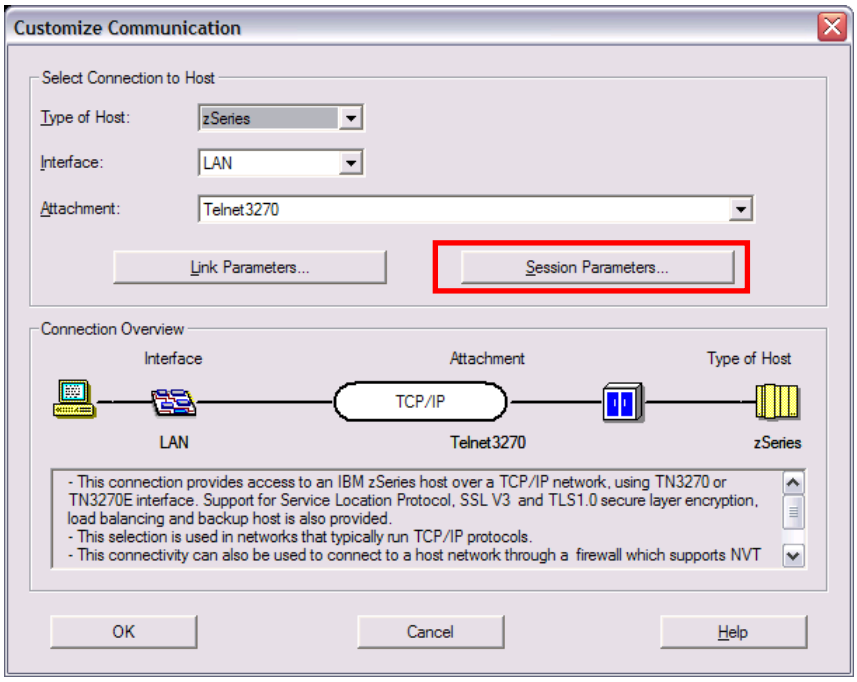
1) To establish a 3270 via TSO, Select **Programs** from the Windows Start menu → **IBM Personal Communications** → **Start or Configure Session** ☐
2) Select **New Session…** ☐
3) Select **Link Parameters…** in the appearing window ☐



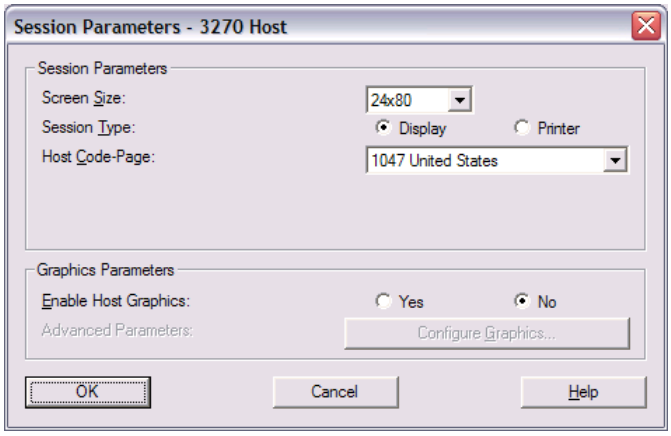
4) Enter 129.35.161.131 as Primary Host Name and select **OK**. ☐



5) Select **Session Parameters…** ☐

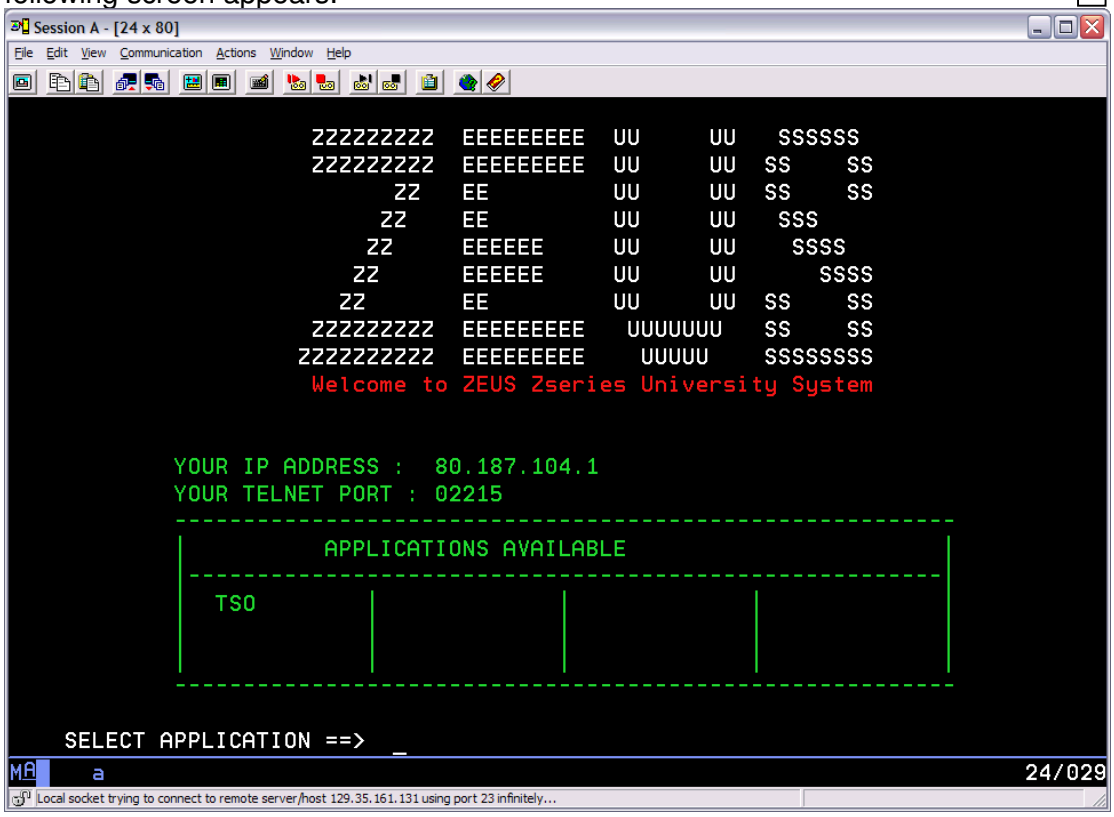


6) Select **24x80** as screen size and **1047** as Host Code-Page. Click **OK**. ☐



7) Select **OK**. ☐

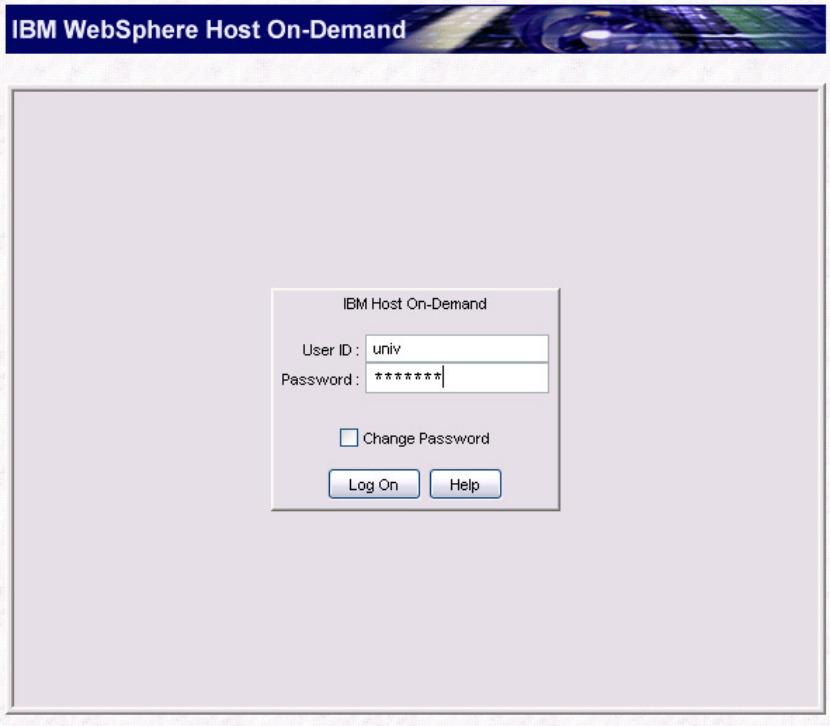
8) You should now be able to login to TSO. Enter **TSO** and press Enter when the following screen appears: ☐



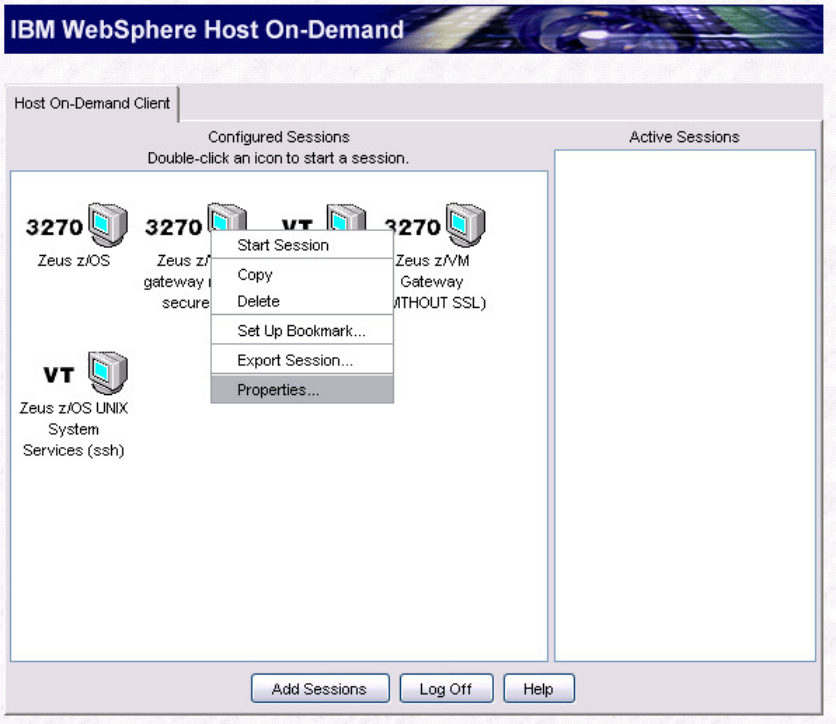
9) Login with your credentials. ☐

2.2 Setup Host On-Demand

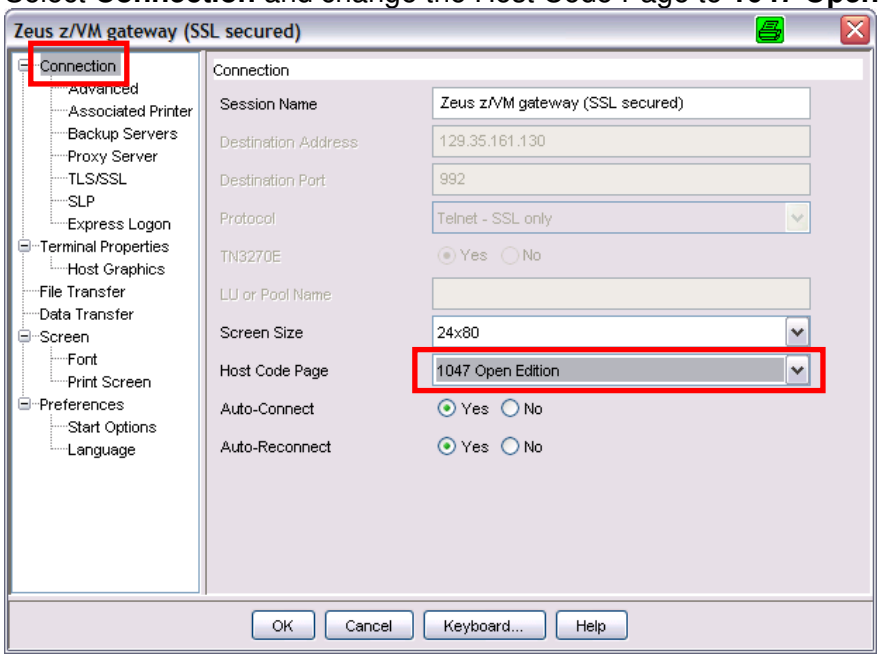
1) Go to http://zeus.moppssc.com/hod/HOD_en.html and login with your credentials. ☐



2) Right-click on **Zeus z/VM gateway (SSL secured)** and select **Properties**. ☐

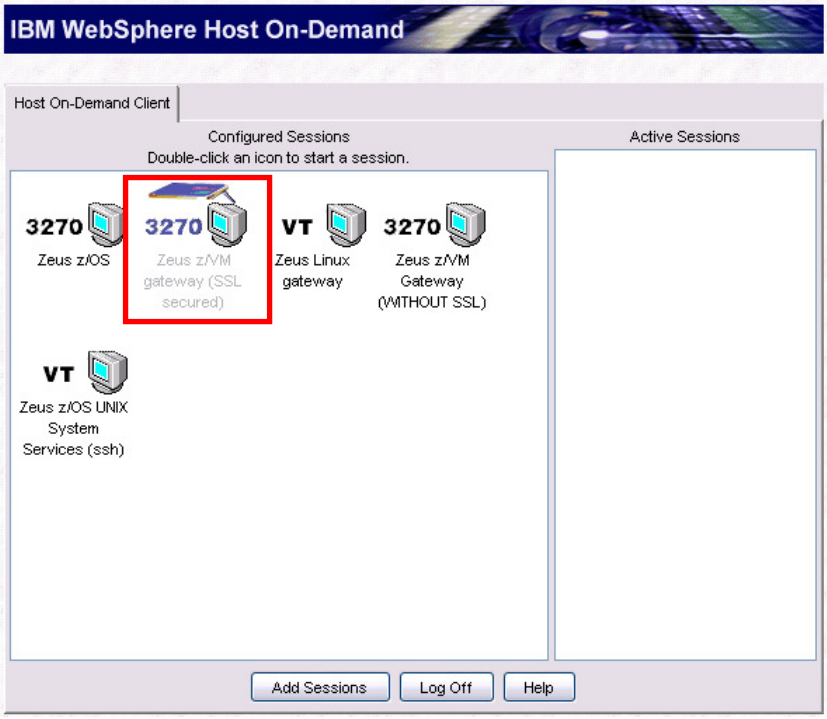


3) Select **Connection** and change the Host Code Page to **1047 Open Edition**. ☐

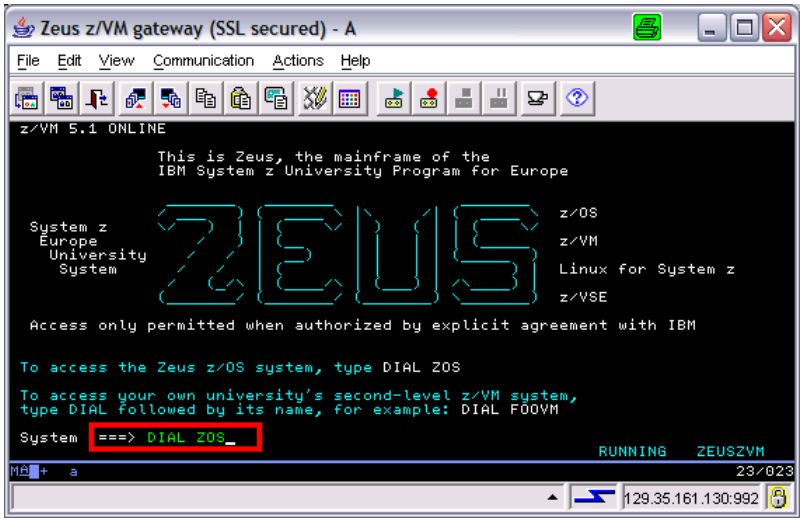


4) Select **OK**. ☐

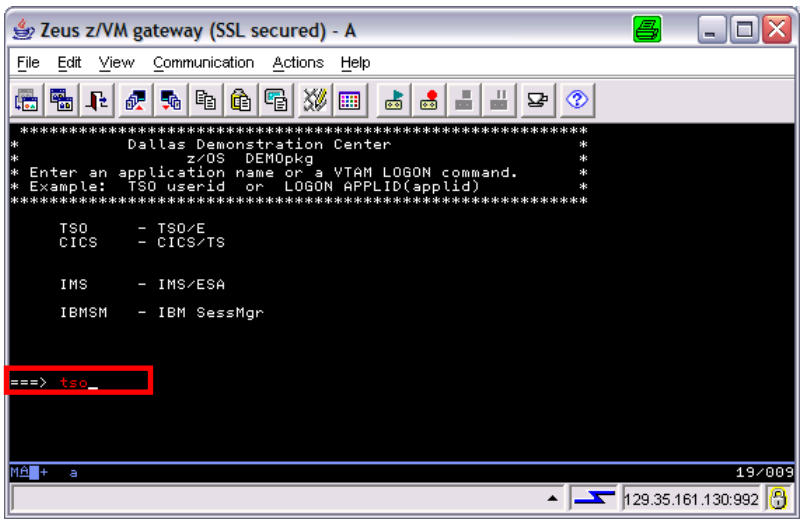
5) Double-click on **Zeus z/VM gateway (SSL secured).** ☐



6) Enter **DIAL ZOS**. ☐



7) Enter **TSO.** ☐



8) Login to TSO with your credentials. ☐

End of lab ☺

2.3 Basic information

The following table shows the most important properties of the workshop host system.

	Value	**Comment**
Userid	UDE00##	
Password	UDEPW##	
IP	129.35.161.131	
Ports	129.35.161.131:23 129.35.161.131:21	Telnet 3270 FTP
Java home directories	/usr/lpp/java/IBM/J1.3 /u/fhbmstr/jbatch/local/java/J5.0 /u/fhbmstr/jbatch/local/java/J6.0	Java 1.3 Java 5.0 Java 6.0
User's home directory	/u/UDE00##	
Workshop files	/u/fhbmstr/jbatch	
User's HLQ	UDE00##	
Master home directory	/u/fhbmstr	
Master HLQ	UDEMSTR	

3 HelloWorld (terminal based)

In this lab, you will create your first Java HelloWorld application on z/OS with a terminal session.

3.1 Verify Java installation

Before you start with HelloWorld, you have to verify that Java is installed correctly on z/OS.

1) Login to TSO as described at the end of chapter 2.1. ☐
2) Open an OMVS shell: ☐

```
TSO OMVS
```

3) In the Unix shell type: ☐

```
java -fullversion
```

4) If Java reports its version the JVM seems to be OK ☐

End of lab ☺

3.2 Your first Java HelloWorld

This lab will show you how to develop a Java Hello World on the terminal.

1) Logon to TSO if you have not done so far. ☐
2) Open an OMVS shell if you have not done so far: ☐

```
TSO OMVS
```

3) Change to your home directory /u/UDE00## and create a new directory myjava. After each step, select enter: ☐

```
cd /u/UDE00##
mkdir myjava
```

4) Create a new Java file in this directory and edit it. After each step, select enter: ☐

```
cd myjava
oedit HelloWorld.java
```

5) Insert the following text into the new file: ☐

```
class HelloWorld
{
       public static void main(String[] args)
       {
              System.out.println("Hello World!");
       }
}
```

6) Save and exit with: ☐

```
F3
```

7) Compile the Java file by entering the following command: ☐

```
javac HelloWorld.java
```

8) This will create a new file HelloWorld.class. Enter the following command to run the HelloWorld example in the Java Virtual Machine (JVM): ☐

```
java  HelloWorld
```

9) You should see a *HelloWorld* on the command line. ☐

10) Exit OMVS by entering ☐

```
Exit
```

End of lab ☺

4 BPXBATCH Labs

4.1 HelloWorld mit BPXBATCH

1) Logon to TSO if you have not done so far. ☐

2) Copy from UDEMSTR.JBATCH.SAMPLES(BPXBATCH) the following job in a new PDSe UDE00##.SAMPLES.JCL(BPXHELLO) using ISPF 3.3: ☐

```
//BPXBATCH JOB ,REGION=0M
//*********************************************************************
//* Run Java under a UNIX System Service shell
//*********************************************************************
//STEP2 EXEC PGM=BPXBATCH,
// PARM='SH java HelloWorld'
//STDIN DD DUMMY
//STDOUT DD PATH='/u/UDE00##/myjava/bpxbatch.out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/u/UDE00##/myjava/bpxbatch.err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDENV DD *
CLASSPATH=/u/UDE00##/myjava
//* Copy HFS output files to SYSOUT, since BPXBATCH can only write
//* STDOUT and STDERR to HFS files.
//*********************************************************************
//STEP3 EXEC PGM=IKJEFT01,DYNAMNBR=300,COND=EVEN
//SYSTSPRT DD SYSOUT=*
//HFSOUT DD PATH='/u/UDE00##/myjava/bpxbatch.out'
//HFSERR DD PATH='/u/UDE00##/myjava/bpxbatch.err'
//STDOUTL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
OCOPY INDD(HFSOUT) OUTDD(STDOUTL)
OCOPY INDD(HFSERR) OUTDD(STDERRL)
//
```

3) Change the Classpath to the directory where you have created the .class file and point STDOUT, STDERR, HSFOUT and HSFERR to /u/UDE00##/myjava... ☐

4) Submit the job ☐

```
sub
```

5) Check results with SDSF. You should see something like this ☐

```
Hello World!
```

End of lab ☺

4.2 Java BPXBATCH with parameters

This lab will show how to use BPXBATCH with Java and parameters. The parameters you specify in the JCL for the Java program will be printed out by the Java program to SDSF.

1) Call OMVS: Enter ☐

```
TSO OMVS
```

2) Change to your myjava sub directory of your home directory ☐

```
cd myjava
```

3) In this directory, create (or copy) a new Java file: ☐

```
cp /u/fhbmstr/jbatch/source/Parameter.java /u/UDE00##/myjava
```

Or

```
oedit Parameter.java
```

accordingly:

```
class Parameter
{
       public static void main(String[] args)
       {
              for (int i=0; i<args.length; i++)
              {
                     System.out.println(args[i]);
              }
       }
}
```

4) Save and exit with ☐

```
F3
```

5) Compile the Java file by entering the following command: ☐

```
javac Parameter.java
```

This will create a new file Parameter.class.

6) Exit OMVS by entering ☐

```
Exit
```

7) Copy from UDEMSTR.JBATCH.SAMPLES.JCL(BPXPARM) the following job in your UDE00##.SAMPLES.JCL(BPXPARM) using ISPF 3.3: ☐

```
//BPXBATCH JOB ,REGION=0M
//*********************************************************************
//* Run Java under a UNIX System Service shell
//*********************************************************************
//STEP2 EXEC PGM=BPXBATCH,
// PARM='SH java Parameter Test1 Test2'
//STDIN DD DUMMY
//STDOUT DD PATH='/u/UDE00##/myjava/bpxbatch.out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDERR DD PATH='/u/UDE00##/myjava/bpxbatch.err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=SIRWXU
//STDENV DD *
CLASSPATH=/u/UDE00##/myjava
//* Copy HFS output files to SYSOUT, since BPXBATCH can only write
```

```
//* STDOUT and STDERR to HFS files.
//*********************************************************************
//STEP3 EXEC PGM=IKJEFT01,DYNAMNBR=300,COND=EVEN
//SYSTSPRT DD SYSOUT=*
//HFSOUT DD PATH='/u/UDE00##/myjava/bpxbatch.out'
//HFSERR DD PATH='/u/UDE00##/myjava/bpxbatch.err'
//STDOUTL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//STDERRL DD SYSOUT=*,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
OCOPY INDD(HFSOUT) OUTDD(STDOUTL)
OCOPY INDD(HFSERR) OUTDD(STDERRL)
//
```

8) Change the Classpath to the directory where you have created the .class file and point STDOUT, STDERR, HSFOUT and HSFERR to /u/UDE00##/myjava ☐

9) Submit the job ☐

```
sub
```

10) Check results with SDSF. You should see something like this: ☐

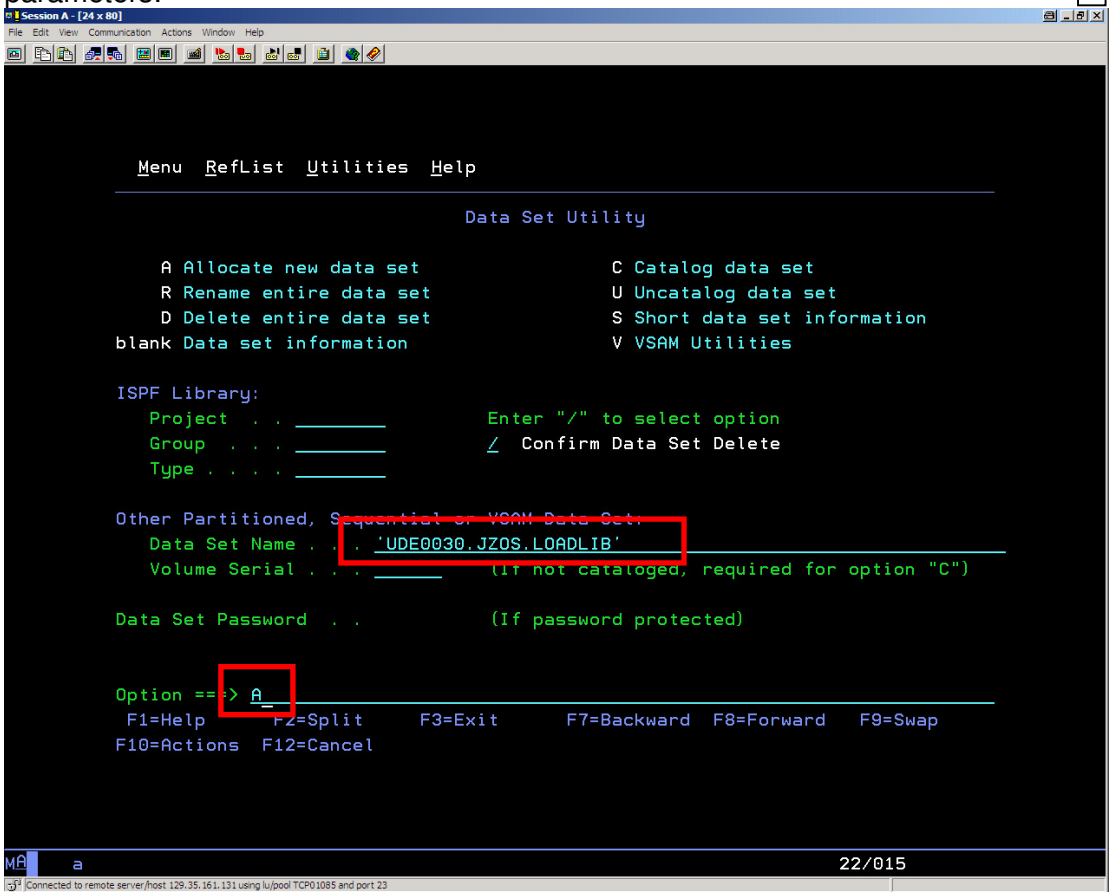
```
Test1
Test2
```

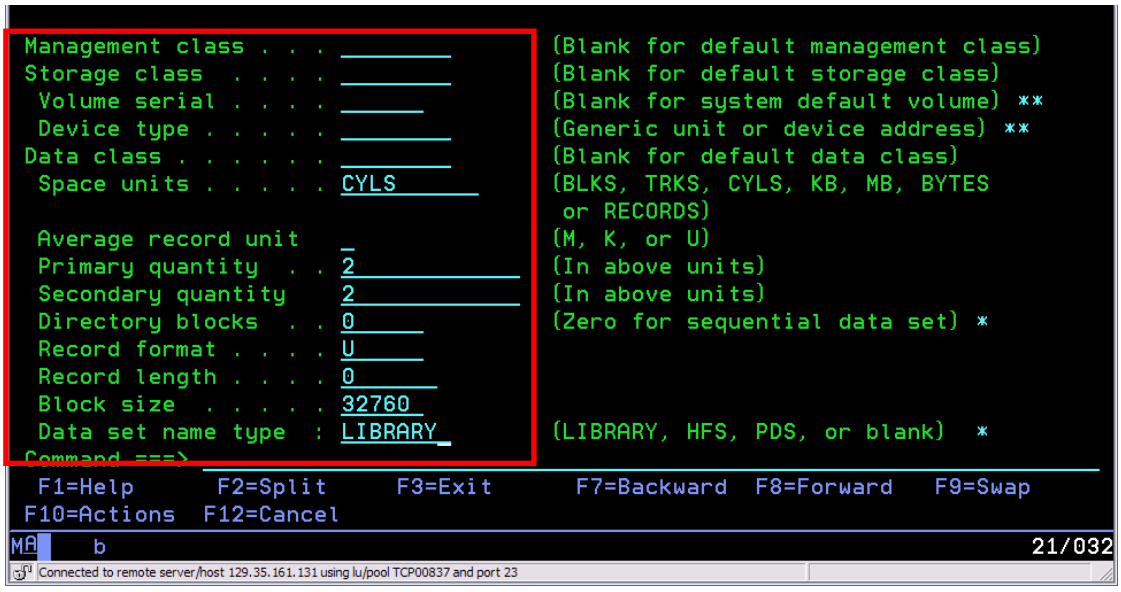
End of lab ☺

4.3 JZOS Labs

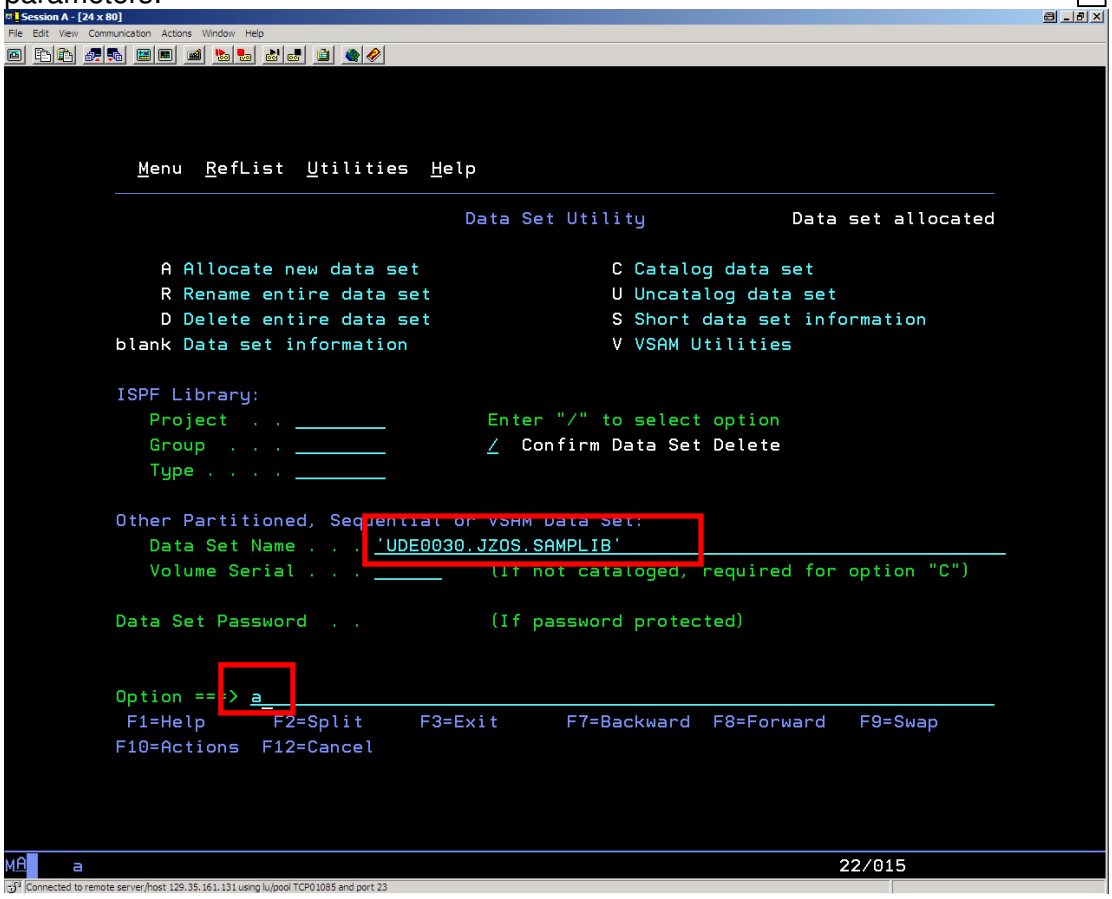
4.4 Installation of JZOS for JVM 6.0

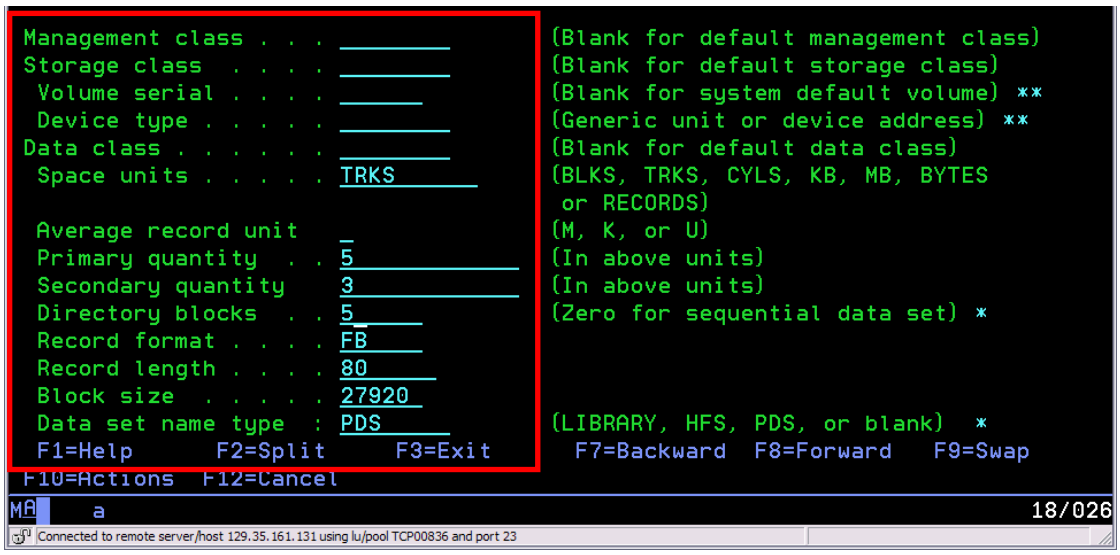
1) Allocate a new dataset UDE00##.JZOS.LOADLIB under 3.2 with the specified parameters:



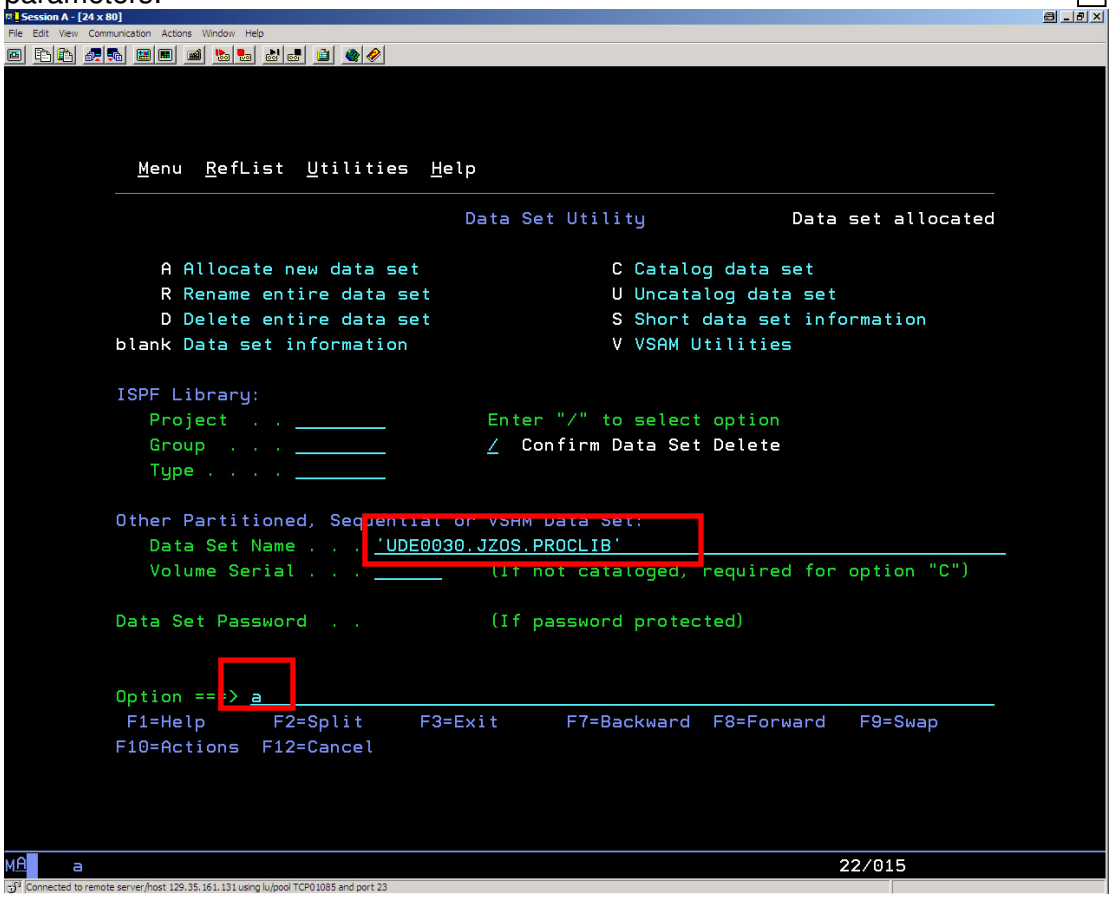


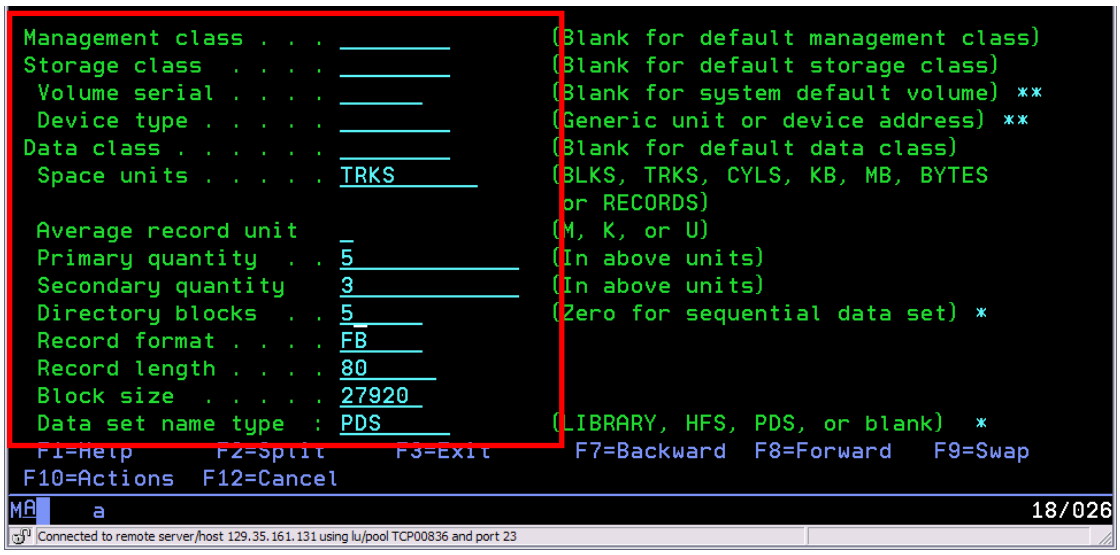
2) Allocate another dataset UDE00##.JZOS.SAMPLIB under 3.2 with the specified parameters:





3) Allocate another dataset UDE00##.JZOS.PROCLIB under 3.2 with the specified parameters: ☐





4) Call OMVS: Enter ☐

```
TSO OMVS
```

5) Change to the Java 6.0 Home directory: ☐

```
cd /u/fhbmstr/jbatch/local/java/J6.0
```

6) Change to the mvstools directory: ☐

```
cd mvstools
```

7) Enter the following command to extract the JVM 6.0 JZOS load module to your just created load library: ☐

```
cp -X JVMLDM60 "//'UDE00##.JZOS.LOADLIB(JVMLDM60)'"
```

8) Change to the samples/jcl directory: ☐

```
cd samples/jcl
```

9) Enter the following command to extract the JVM 6.0 PROC to your just created proclib: ☐

```
cp JVMPRC60 "//'UDE00##.JZOS.PROCLIB(JVMPRC60)'"
```

10) Enter the following command to extract the JVM 6.0 sample JCL to your just created samplib: ☐

```
cp JVMJCL60 "//'UDE00##.JZOS.SAMPLIB(JVMJCL60)'"
```

OPTIONAL:

11) To also install JZOS for the JVM 5.0, perform the following steps:
Change to the Java 5.0 Home directory: ☐

```
cd /u/fhbmstr/jbatch/local/java/J5.0
```

12) Change to the mvstools directory: ☐

```
cd mvstools
```

13) Enter the following command to extract the JVM 5.0 JZOS load module to your just created load library: ☐

```
cp -X JVMLDM50 "//'UDE00##.JZOS.LOADLIB(JVMLDM50)'"
```

14) Change to the samples/jcl directory: ☐

```
cd samples/jcl
```

15) Enter the following command to extract the JVM 5.0 PROC to your just created proclib: ☐

```
cp JVMPRC50 "//'UDE00##.JZOS.PROCLIB(JVMPRC50)'"
```

16) Enter the following command to extract the JVM 5.0 sample JCL to your just created samplib: ☐

```
cp JVMJCL50 "//'UDE00##.JZOS.SAMPLIB(JVMJCL50)'"
```

End of lab ☺

4.5 HelloWorld with JZOS

To run your first HelloWorld with JZOS and Java 6.0, perform the following steps.

1) Edit the JZOS batch launcher proc contained in 'UDE00##.JZOS.PROCLIB(JVMPRC60)', updating it to point to the JZOS load module library and uncomment this statement. ☐

```
//JVMPRC50 PROC JAVACLS=,                  < Fully Qfied Java class..RQD
//   ARGS=,                                < Args to Java class
//   LIBRARY='UDE00##.JZOS.LOADLIB',       < STEPLIB FOR JVMLDM module
//   VERSION='60',                         < JVMLDM version: 60
//   LOGLVL='',                            < Debug LVL: +I(info) +T(trc)
//   REGSIZE='0M',                         < EXECUTION REGION SIZE
//   LEPARM=''
```

2) Uncomment STEPLIB and save your changes: ☐

```
//JAVAJVM  EXEC PGM=JVMLDM&VERSION,REGION=&REGSIZE,
//   PARM='&LEPARM/&LOGLVL &JAVACLS &ARGS'
//STEPLIB  DD DSN=&LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*          < System stdout
//SYSOUT   DD SYSOUT=*          < System stderr
//STDOUT   DD SYSOUT=*          < Java System.out
//STDERR   DD SYSOUT=*          < Java System.err
//CEEDUMP  DD SYSOUT=*
//ABNLIGNR DD DUMMY
//*
```

3) Following the instructions contained in 'UDE00##.JZOS.SAMPLIB(JVMJCL60)', tailor the member: ☐

```
//UDE00##A JOB                                          ← JOB-Name
//PROCLIB JCLLIB ORDER=UDE00##.JZOS.PROCLIB
//*********************************************************************
...
//*********************************************************************

//JAVA EXEC PROC=JVMPRC60,
...
export JAVA_HOME=/u/fhbmstr/jbatch/local/java/J6.0
...
APP_HOME=/u/UDE00##/myjava
...
```

4) SUBMIT the modified JCL and check the job log. ☐

If everything was set up properly, the SYSOUT DD should contain output like this:

```
JVMJZBL1001N JZOS batch Launcher Version: 2.3.0 2008-05-12
JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
java version "1.6.0"
Java(TM) SE Runtime Environment (build jvmmz3160-20081107_25433)
IBM J9 VM (build 2.4, J2RE 1.6.0 IBM J9 2.4 z/OS s390-31 jvmmz3160-
20081107_2543
J9VM - 20081105_025433_bHdSMr
JIT  - r9_20081031_1330
GC   - 20081027_AB)
JVMJZBL1023N Invoking HelloWorld.main()...
JVMJZBL1024N HelloWorld.main() completed.
JVMJZBL1021N JZOS batch launcher completed, return code=0
```

And the JOB STDOUT DD should contain:

```
Hello World!
```

OPTIONAL:
If you want to do the same for Java 5.0 repeat the following steps:

5) Edit the JZOS batch launcher proc contained in 'UDE00##.JZOS.PROCLIB(JVMPRC50)', updating it to point to the JZOS load module library and uncomment this statement. ☐

```
//JVMPRC50 PROC JAVACLS=,                  < Fully Qfied Java class..RQD
//   ARGS=,                                < Args to Java class
//   LIBRARY='UDE00##.JZOS.LOADLIB',       < STEPLIB FOR JVMLDM module
//   VERSION='50',                         < JVMLDM version: 50
//   LOGLVL='',                            < Debug LVL: +I(info) +T(trc)
//   REGSIZE='0M',                         < EXECUTION REGION SIZE
//   LEPARM=''
```

6) Uncomment STEPLIB and save your changes: ☐

```
//JAVAJVM  EXEC PGM=JVMLDM&VERSION,REGION=&REGSIZE,
//   PARM='&LEPARM/&LOGLVL &JAVACLS &ARGS'
//STEPLIB  DD DSN=&LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*          < System stdout
//SYSOUT   DD SYSOUT=*          < System stderr
//STDOUT   DD SYSOUT=*          < Java System.out
//STDERR   DD SYSOUT=*          < Java System.err
//CEEDUMP  DD SYSOUT=*
//ABNLIGNR DD DUMMY
//*
```

7) Following the instructions contained in 'UDE00##.JZOS.SAMPLIB(JVMJCL50)', tailor the member: ☐

```
//UDE00##A JOB                                          ← JOB-Name
//PROCLIB JCLLIB ORDER=UDE00##.JZOS.PROCLIB
//*********************************************************************
...
//*********************************************************************

//JAVA EXEC PROC=JVMPRC50,
...
export JAVA_HOME=/u/fhbmstr/jbatch/local/java/J6.0
...
APP_HOME=/u/UDE00##/myjava
...
```

8) SUBMIT the modified JCL and check the job log. ☐

If everything was set up properly, the SYSOUT DD should contain output like this:

```
JVMJZBL1001N JZOS batch Launcher Version: 2.3.0 2008-05-12
JVMJZBL1002N Copyright (C) IBM Corp. 2005. All rights reserved.
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pmz31dev-20081210
(SR9-0
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390-31 j9vmmz3123-20081130
(JI
J9VM - 20081126_26240_bHdSMr
JIT  - 20081112_1511ifx1_r8
GC   - 200811_07)
JVMJZBL1023N Invoking HelloWorld.main()...
JVMJZBL1024N HelloWorld.main() completed.
JVMJZBL1021N JZOS batch launcher completed, return code=0
```

And the JOB STDOUT DD should contain:

```
Hello World!
```

End of lab ☺

4.5.1 Optional JZOS Lab 1.1 - diagnose problems

1) To diagnose problems with the JZOS batch launcher, change the LOGLVL parameter to '+I' : ☐

```
// EXEC EXJZOSVM,LOGLVL='+I',
```

NOTE: Setting this logging level (+I) will dump the environment that is passed to the JVM. The trace level setting "+T" will produce many messages, some of which may be helpful in tracking down installation problems.

End of lab ☺

4.6 More MVS Java Programs

This chapter covers different Java programs that access native MVS resources. The Java source files for this chapter can be found here: /u/fhbmstr/jbatch/source.

4.6.1 Write to operator console with Java

This chapter shows to write messages to the operator console from Java

2) Copy /u/fhbmstr/jbatch/source/Wto.java to /u/UDE00##/Wto.java: ☐

```
cp /u/fhbmstr/jbatch/source/Wto.java /u/UDE00##/myjava/Wto.java
```

3) Have a look at the source code: ☐

```
cd /u/UDE00##/myjava
oedit Wto.java
```

4) Close the file with F3. ☐

5) Compile that file with javac: ☐

```
javac Wto.java -classpath
/u/fhbmstr/jbatch/local/java/J6.0/lib/ext/ibmjzos.jar
```

6) Exit the OMVS shell by entering ☐

```
Exit
```

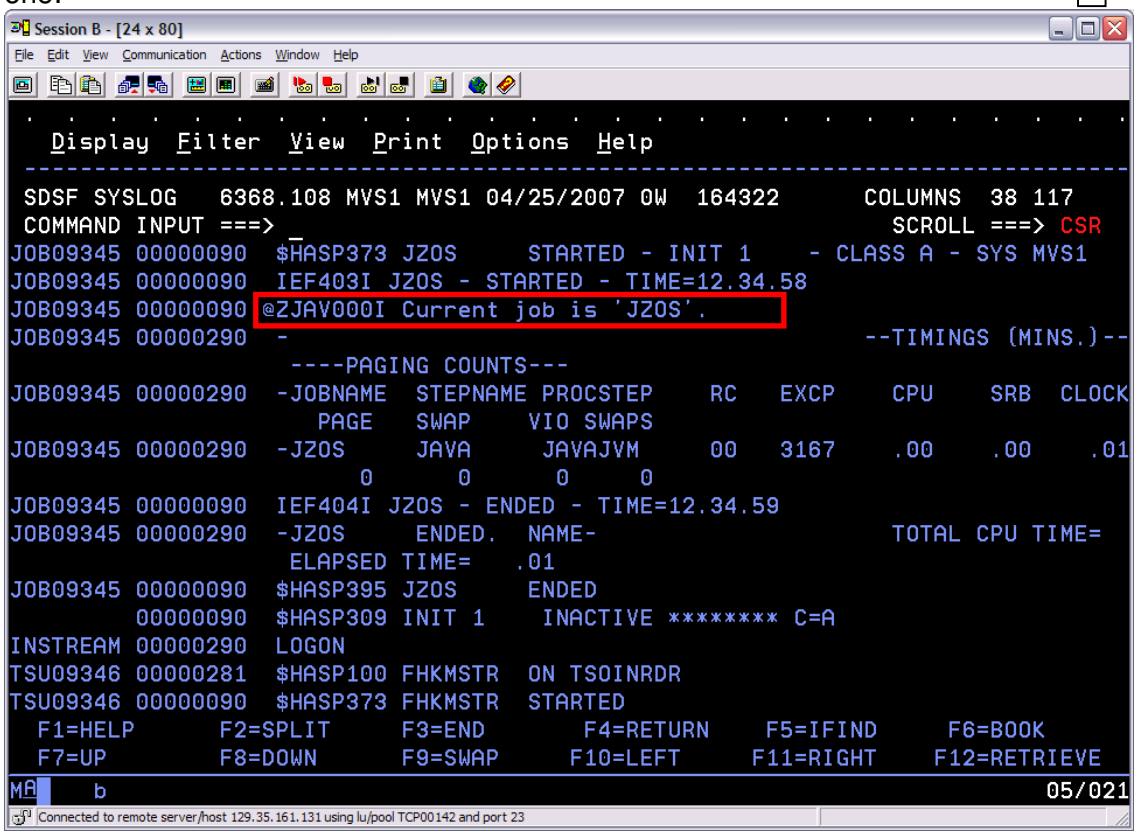
7) Copy the JCL that you have used in chapter 4.5, step 3) into a new JCL 'UDE00##.JZOS.SAMPLIB (WTO)' ☐

8) Modify JAVACLS in the JCL: ☐

```
//UDE00##A JOB                                          ← JOB-Name
//PROCLIB JCLLIB ORDER=UDE00##.JZOS.PROCLIB
...
//JAVA EXEC PROC=JVMPRC60,
// JAVACLS='Wto'
//STDENV DD *
...
export JAVA_HOME=/u/fhbmstr/jbatch/local/java/J6.0
...
```

9) Submit the JCL and check the output with SDSF. ☐

10) Go to the operator console and check for the WTO. It should look similar to this one:



4.6.2 Handling condition codes with Java

This chapter shows to write create a little Job net based on Java return codes.

1) Copy /u/fhbmstr/jbatch/source/ConditionCode.java to /u/UDE00##/ConditionCode.java: ☐

```
cp /u/fhbmstr/jbatch/source/ConditionCode.java
/u/UDE00##/myjava/ConditionCode.java
```

2) Have a look at the source code: ☐

```
cd /u/UDE00##/myjava
oedit ConditionCode.java
```

3) Close the file with F3. ☐

4) Compile that file with javac: ☐

```
javac ConditionCode.java
```

5) Exit the OMVS shell by entering ☐

```
Exit
```

6) Copy the JCL that you have used in chapter 4.5, step 3) into a new JCL 'UDE00##.JZOS.SAMPLIB(CC)' ☐

7) Modify JAVACLS and ARGS: ☐

```
//UDE00##A JOB                                          ← JOB-Name
//PROCLIB JCLLIB ORDER=UDE00##.JZOS.PROCLIB
...
//JAVA EXEC PROC=JVMPRC60,
// JAVACLS='ConditionCode',
// ARGS='1'
//STDENV DD *
...
export JAVA_HOME=/u/fhbmstr/jbatch/local/java/J6.0
…
```

8) In the JCL, add the following script which starts a HelloWorld if the return code of the Java ConditionCode program is '1' ☐

```
//*********************************************
// IF (RC = 1) THEN
//*********************************************
//HELLO EXEC PROC=JVMPRC60,
// JAVACLS='HelloWorld'
//STDENV DD *
# This is a shell script which configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.
. /etc/profile
export APP_HOME=/u/UDE00##/myjava
export JAVA_HOME=/u/fhbmstr/jbatch/local/java/J6.0
export PATH="$PATH":"${JAVA_HOME}"/bin:
LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin/classic
LIBPATH="$LIBPATH":"${JZOS_HOME}"
export LIBPATH="$LIBPATH":
# Customize your CLASSPATH here
CLASSPATH=$APP_HOME
export CLASSPATH="$CLASSPATH":
# Set JZOS specific options
# Use this variable to specify encoding for DD STDOUT and STDERR
#export JZOS_OUTPUT_ENCODING=Cp1047
# Use this variable to prevent JZOS from handling MVS operator commands
#export JZOS_ENABLE_MVS_COMMANDS=false
# Use this variable to supply additional arguments to main
```

```
#export JZOS_MAIN_ARGS=""
# Configure JVM options
IJO="-Xms16m -Xmx128m"
IJO="$IJO -Djzos.home=${JZOS_HOME}"
# Uncomment the following if you want to run without JIT
#IJO="$IJO -Djava.compiler=NONE"
# Uncomment the following if you want to run with Ascii file encoding..
IJO="$IJO -Dfile.encoding=ISO8859-1"
export IBM_JAVA_OPTIONS="$IJO "
export JAVA_DUMP_HEAP=false
export JAVA_PROPAGATE=NO
//*********************************************
// ENDIF
//*********************************************
//
```

9) Submit the JCL and check the output with SDSF. ☐

10) Modify ARGS='0' in the JCL ☐

11) Again, submit the JCL and check the output with SDSF. The HelloWorld should **not** be invoked. ☐

End of lab ☺

4.7 Tomcat with JZOS

The Apache Tomcat servlet container can be installed quickly and easily using the JZOS toolkit.

1) Create a new directory /u/fhbmstr/jbatch/users/UDE00## ☐

```
mkdir /u/fhbmstr/jbatch/users/UDE00##
```

2) Download the .zip version of the binary distribution of Tomcat and upload it (in binary mode) to /u/fhbmstr/jbatch/users. ☐

   Apache Jakarta download page: http://jakarta.apache.org/site/binindex.cgi
   binary .zip distribution: http://tomcat.apache.org/download-55.cgi

   Note: You can also find the binary zip installation files in the /u/fhbmstr/jbatch/setup directory and copy it directly to your home directory /u/UDE00##!

```
cp /u/fhbmstr/jbatch/setup/apache-tomcat-5.5.15.zip
/u/fhbmstr/jbatch/users/UDE00##
```

3) From a z/OS Unix shell, change to your home directory extract the Tomcat zip file. ☐

   Note: this will create a directory "apache-tomcat-5.5.15" under the current directory!

```
cd /u/fhbmstr/jbatch/users/UDE00##
jar -xvf apache-tomcat-5.5.15.zip
```

4) Delete the zip file: ☐

```
rm apache-tomcat-5.5.15.zip
```

5) For convenience, create a symbolic link to the Tomcat distribution: ☐

```
ln -s apache-tomcat-5.5.15 tomcat
```

   Note: You can now use “tomcat” instead of “apache-tomcat-5.5.15”.

6) Modify ports in server.xml: ☐
   - Download /u/fhbmstr/jbatch/users/UDE00##/apache-tomcat-5.5.15/conf/server.xml via FTP in binary mode ☐
   - Edit server.xml on your local workstation and change the http port depending on your userid, i.e. change 8080 to 80## ☐
   - Upload the modified server.xml to /u/fhbmstr/jbatch/users/UDE00##/users/apache-tomcat-5.5.15/conf/server.xml via FTP in binary mode. ☐

7) Create a new member TOMCAT in the dataset UDE00##.JZOS.SAMPLIB' and insert the following JCL content: ☐

```
//TOMCAT JOB
//PROCLIB JCLLIB ORDER=UDE00##.JZOS.SAMPJCL
//...
//JAVA EXEC PROC=JVMPRC50,
// JAVACLS='org.apache.catalina.startup.Bootstrap',
// ARGS='start'
//STDENV DD *
# This is a shell script which configures
# any environment variables for the Java JVM.
```

```
# Variables must be exported to be seen by the launcher.
. /etc/profile

export TOMCAT_HOME=/u/fhbmstr/jbatch/users/UDE00##/apache-tomcat-5.5.15
export JAVA_HOME=/u/fhbmstr/jbatch/local/java/J5.0

export PATH=/bin:"${JAVA_HOME}"/bin:

LIBPATH=/lib:/usr/lib:"${JAVA_HOME}"/bin:"$JAVA_HOME"/bin/classic

LIBPATH="$LIBPATH":"${JZOS_HOME}"
export LIBPATH="$LIBPATH":

CLASSPATH="${JAVA_HOME}/lib/tools.jar"
CLASSPATH="$CLASSPATH":"${TOMCAT_HOME}/bin/bootstrap.jar"
CLASSPATH="$CLASSPATH":"${JZOS_HOME}/jzos.jar"
CLASSPATH="$CLASSPATH":"${TOMCAT_HOME}/bin/commons-logging-api.jar"
export CLASSPATH="$CLASSPATH":
# Set JZOS specific options
# Use this variable to specify encoding for DD STDOUT and STDERR
#export JZOS_OUTPUT_ENCODING=Cp1047
# Use this variable to prevent JZOS from handling MVS operator commands
#export JZOS_ENABLE_MVS_COMMANDS=false
# Use this variable to supply additional arguments to main
#export JZOS_MAIN_ARGS=""
# Configure JVM options
# Note that Tomcat requires default ASCII file.encoding
IJO="-Xms64m -Xmx128m"
IJO="$IJO -Dfile.encoding=ISO8859-1"
IJO="$IJO -Djzos.home=${JZOS_HOME}"
IJO="$IJO -Dcatalina.base=${TOMCAT_HOME}"
IJO="$IJO -Dcatalina.home=${TOMCAT_HOME}"
IJO="$IJO -Djava.io.tmpdir=${TOMCAT_HOME}/temp"
IJO="$IJO -Djava.endorsed.dirs=${TOMCAT_HOME}/common/endorsed"
# Uncomment the following if you want to run without JIT
#IJO="$IJO -Djava.compiler=NONE"
export IBM_JAVA_OPTIONS="$IJO "
export JAVA_DUMP_HEAP=false
export JAVA_PROPAGATE=NO
export IBM_JAVA_ZOS_TDUMP=NO
//
```

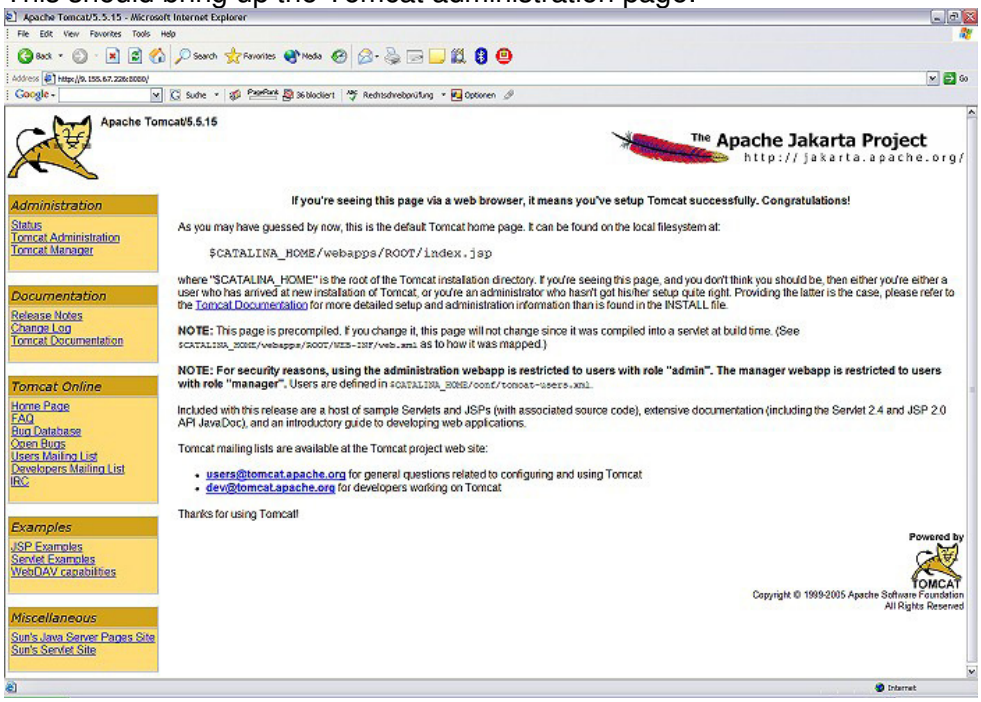
8) Submit the JCL ☐

9) Check results with SDSF: ☐
   STDOUT DD should contain something like this (but will vary depending on the JDK version):

```
[INFO] Http11Protocol - Initializing Coyote HTTP/1.1 on http-8080
[INFO] Catalina - Initialization processed in 129605 ms
[INFO] StandardService - Starting service Catalina
[INFO] StandardEngine - Starting Servlet Engine: Apache Tomcat/
[INFO] StandardHost - XML validation disabled
[INFO] StandardHost - Create Host deployer for direct deployment ( non-jmx )
[INFO] StandardHostDeployer - Processing Context configuration file URL
...
[INFO] Catalina - Server startup in 454422 ms
```

10) Test Tomcat: ☐
    Go to http://129.35.161.131:80##/

This should bring up the Tomcat administration page:



11) Purge the TOMCAT Job

Note: Please ask your instructor to purge tomcat job for you, you won't have enough authority to do that on the system.

4.7.1 Install JZOS Samples

1) To deploy applications, you first of all have to install the Tomcat Admin Page: unzip the /u/fhbmstr/jbatch/setup/apache-tomcat-5.5.15-admin.zip to your Tomcat home: ☐

```
Cd /u/fhbmstr/jbatch/users/UDE00##
jar -xvf /u/fhbmstr/jbatch/setup/apache-tomcat-5.5.15-admin.zip
```

2) Download /u/fhbmstr/jbatch/users/UDE00##/apache-tomcat-5.5.15/conf/tomcat-users.xml in binary mode via FTP to your workstation. ☐

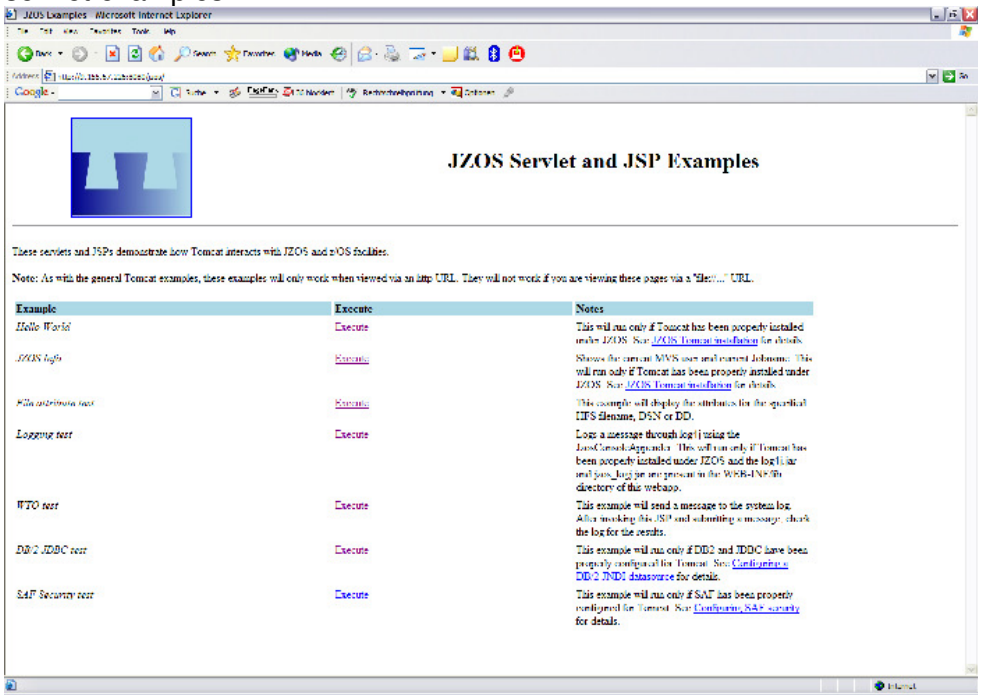
3) Add the following lines in the xml file: ☐
<role rolename="admin"/>
<role rolename="manager"/>
…
<user username="admin" password="manager" roles="admin,manager"/>

4) Save the tomcat-users.xml file and upload again to the host in binary mode to /u/fhbmstr/jbatch/users/UDE00##/apache-tomcat-5.5.15/conf ☐

5) Restart Tomcat by submitting the TOMCAT job again. ☐

6) Install and test JZOS servlet examples: ☐
Download the jzos.war from /u/fhbmstr/jbatch/setup to your local workstation and deploy it with the Tomcat Manager on the admin page (Login with username "admin" and password "manager").

7) After deploying the .war file, go to http://129.35.161.131:80xx/jzos and test the servlet examples. ☐



8) Purge the TOMCAT Job ☐

Note: Please ask your instructor to purge tomcat job for you, you won't have enough authority to do that on the system.

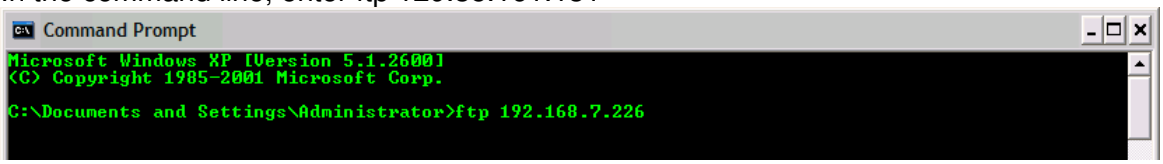
End of lab ☺

Appendix

A1 Basic FTP Tutorial

This tutorial explains how to use FTP for downloading files from the host.

1) In Windows, select **Start** → **Run** → Enter **cmd** and press Enter.
2) In the command line, enter ftp 129.35.161.131



3) Enter UDE00## as username and <your password> as password.
4) Enter **lcd <local_dir>** to change to your local workstation directory where you want to place your downloaded files.
5) Enter **bin** to use binary mode.
6) Enter **cd <host_dir>** to change to the host directory where you want to download the files from.
7) Enter **get <destination_file>** to get the desired file.
8) If you have finished downloading all files, enter **quit** to leave the FTP session

A2 Java Syntax

Further information on Java can be found here:

- *Handbuch der Java-Programmierung*: http://www.javabuch.de/ (German)
- *Java ist auch eine Insel* and *Java 2 und Praxisbuch Objektorientierung*: http://www.galileocomputing.de/openbook (German)

A3 z/OS Basics

A good introduction to z/OS basics can be found in the redbook *Introduction to the New Mainframe: z/OS Basics* under
http://www.redbooks.ibm.com/abstracts/sg246366.html?Open

Table of contents:

- Part 1. Introduction to z/OS and the mainframe environment
- Chapter 1. Introduction to the new mainframe
- Chapter 2. Mainframe hardware systems and high availability
- Chapter 3. z/OS overview
- Chapter 4. TSO/E, ISPF, and UNIX: Interactive facilities of z/OS
- Chapter 5. Working with data sets
- Chapter 6. Using JCL and SDSF
- Chapter 7. Batch processing and JES
- Part 2. Application programming on z/OS
- Chapter 8. Designing and developing applications for z/OS