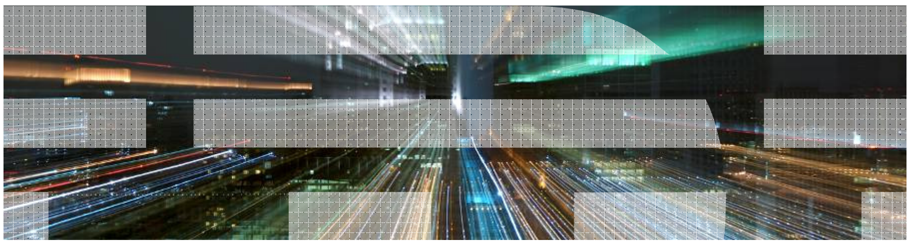
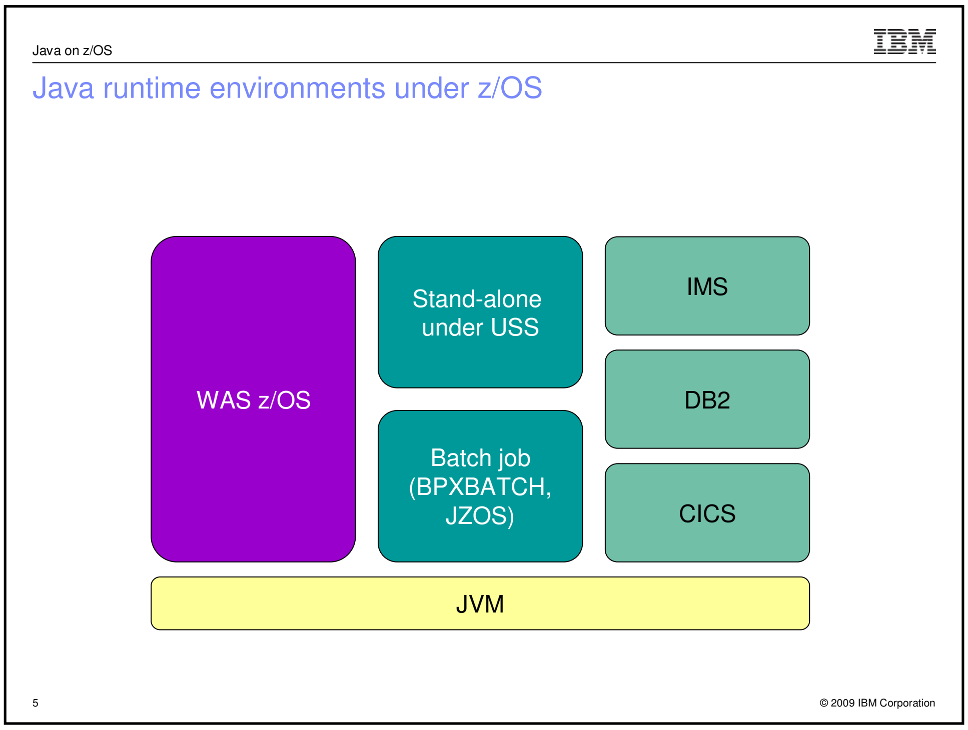
Martina Schmidt
martina.schmidt@de.ibm.com

# Java on z/OS

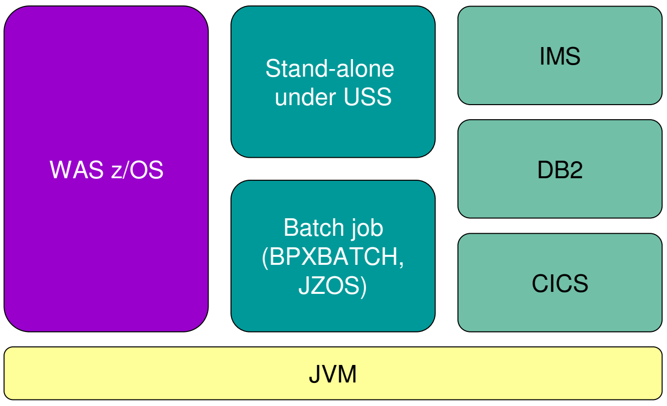


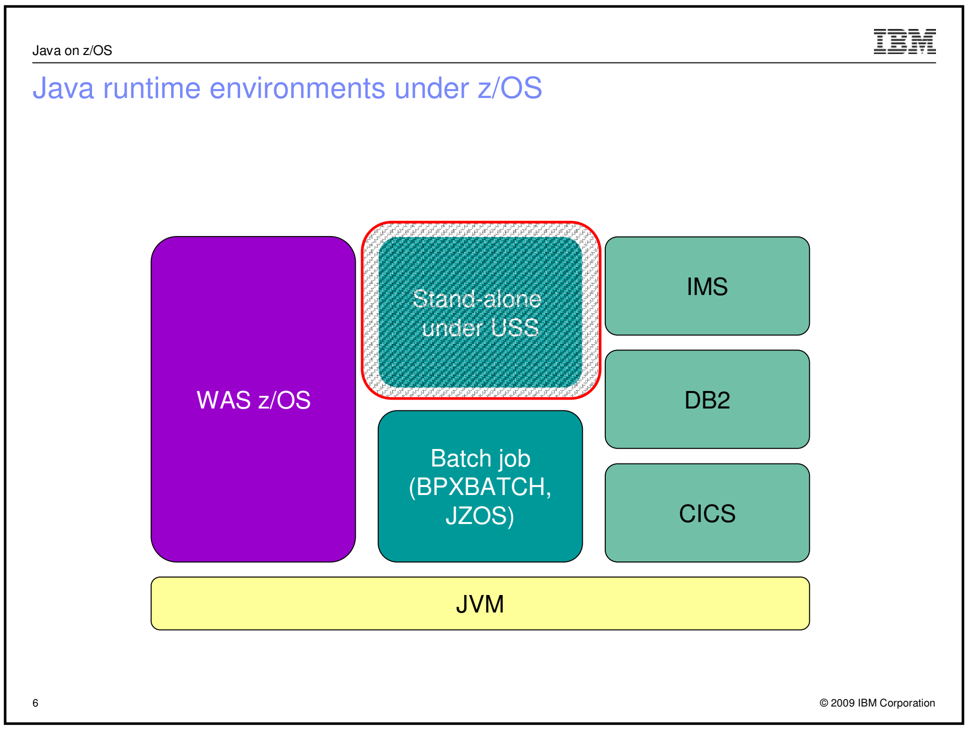
## Agenda

- Java runtime environments on z/OS
- Java SDK 5 and 6
- Java System Resource Integration
- Java Backend Integration
- Java development for z/OS

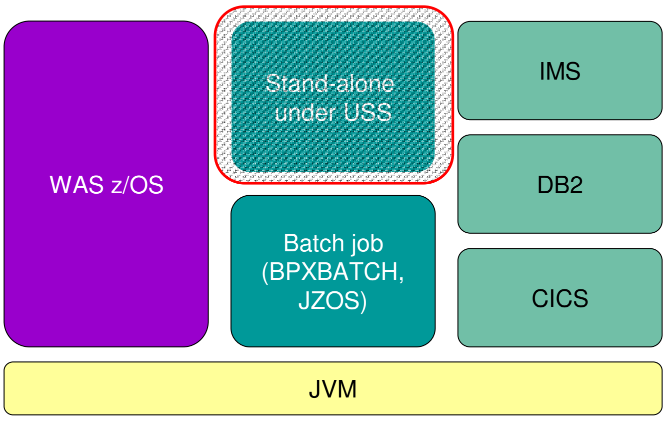


## Java runtime environments under z/OS



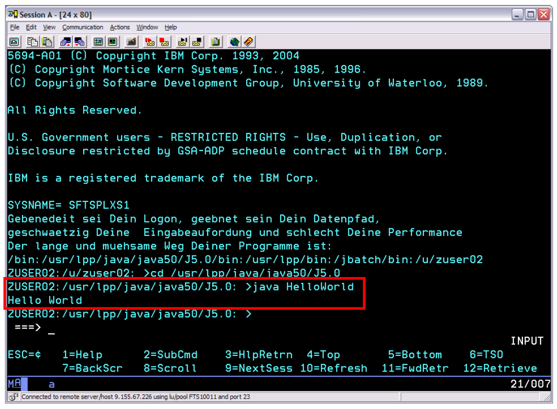


## Java runtime environments under z/OS



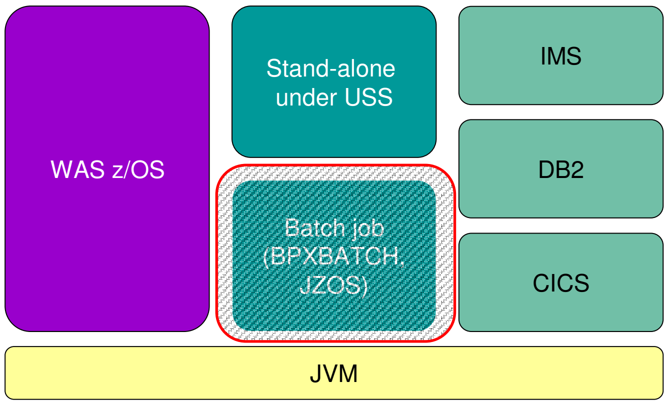
## Stand-alone Java under the USS shell

- Starting Java programs like on the shell of every OS
- Shell scripts supported



```
Session A - [24 x 80]
File  Edit  View  Communication  Actions  Window  Help
5694-A01 (C) Copyright IBM Corp. 1993, 2004
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

SYSNAME= SFTSPLXS1
Gebenedeit sei Dein Logon, geebnet sein Dein Datenpfad,
geschwaetzig Deine  Eingabeaufordung und schlecht Deine Performance
Der lange und muehsame Weg Deiner Programme ist:
/bin:/usr/lpp/java/java50/J5.0/bin:/usr/lpp/bin:/jbatch/bin:/u/zuser02
ZUSER02:/u/zuser02: >cd /usr/lpp/java/java50/J5.0
ZUSER02:/usr/lpp/java/java50/J5.0: >java HelloWorld
Hello World
ZUSER02:/usr/lpp/java/java50/J5.0: >
 ===> _
                                                                  INPUT
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
MA    a                                                              21/007
Connected to remote server/host 9.155.67.226 using lu/pool FTS10011 and port 23
```

## Java runtime environments under z/OS

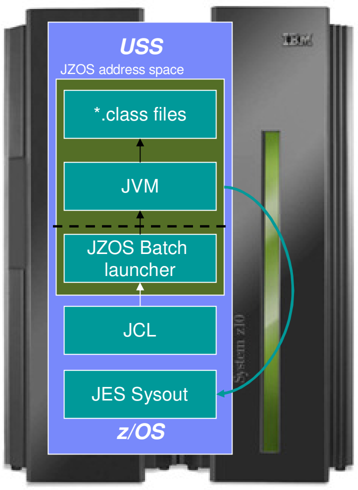


## Java batch option No. 1: BPXBATCH

| | |
|---|---|
| //OPENBATC JOB | Start of Job |
| //S1 EXEC PGM=MVSPROG1 | MVS batch program |
| //S2 EXEC PGM=BPXBATCH,<br>// PARM='pgm parm1 parm2' | z/OS UNIX program, e.g. Java |
| //S3 EXEC PGM=MVSPROG2 | MVS batch program |
| // | End of Job |

## Java batch option No. 2: JZOS

- Full integration of Java into JES
- Easy to integrate new program logic written in Java into classic job nets (e.g. eMail or PDF generation)
- Allows to run Java based servers as started task
- IDE integration
- zAAP eligible

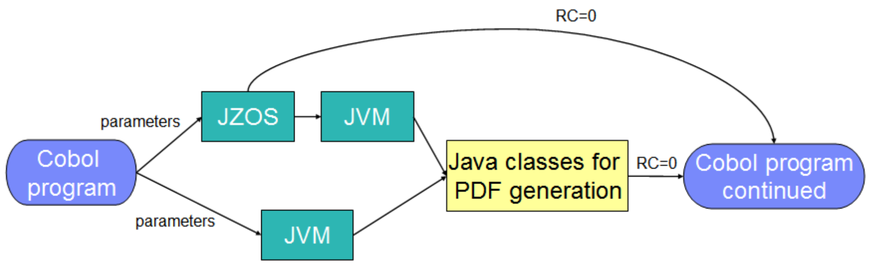


## JZOS – How do I get it

- JZOS is a framework acquired by IBM
- Has become part of IBM z/OS JVM last year in the following version:
  - Java 31-Bit SDK 1.4.2 SR6 for z/OS or higher
  - Java 31-Bit SDK 5.0 SR3
  - Java 64-Bit SDK 5.0 SR3
- More information: http://www-03.ibm.com/servers/eserver/zseries/software/java/

## Examples: 1) Creating PDFs with Java Batch

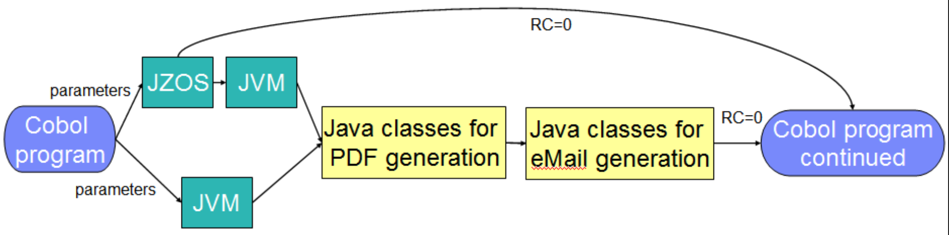
- Traditional batch jobs generate print outs
- Why not generate PDFs? There are lots of Java classes for PDF generation available



… Have you ever tried to generate PDFs with Cobol, PL/I or ASM?

## Examples: 2) Sending Mails with Java Batch

- … Lets take the last example again:

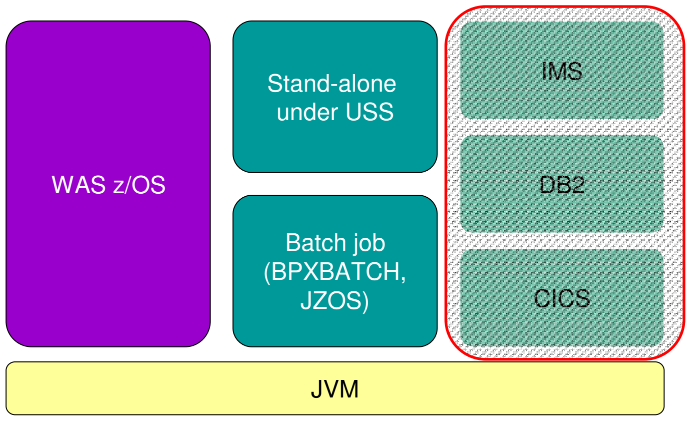


- There are standard APIs available that easily allow to send emails with attachments

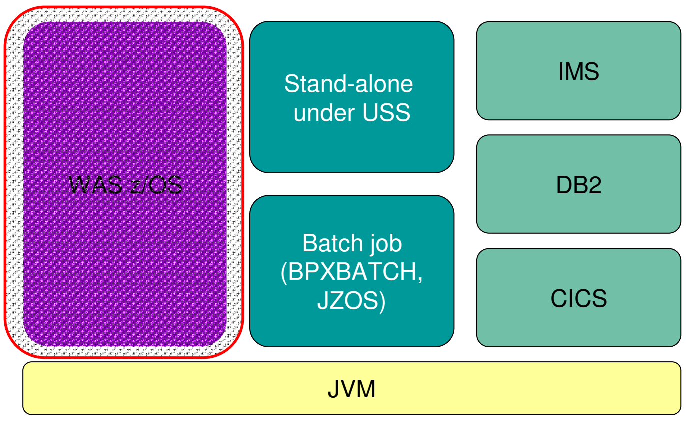
## Comparison BPXBATCH vs. JZOS

| | **BPXBATCH** | **JZOS** |
|---|---|---|
| Run in the same address space | No | Yes |
| DD statements supported | No | Yes |
| Move stdin, stdout and sterr to MVS dataset | No | Yes |
| Console communication | No | Yes |
| Return code | Always 1 | Yes |
| Running programs and shell scripts in USS | yes | No |

## Java runtime environments under z/OS



## Java runtime environments under z/OS



## Agenda

- Java runtime environments on z/OS
- Java SDK 5 and 6
- Java System Resource Integration
- Java Backend Integration
- Java development for z/OS

## Java SDK 5.0 – A complete new JVM for z/OS

- Sun IP-free, but Java 2 (1.3) compliant (J2ME) and J2SE (1.4.2, 5.0)
- **Common code base** across all platforms
  - PowerPC, IA32, x86-64, and 390 (Linux or **z**/**OS**)
- Flexible and sophisticated technology oriented to:
  - Performance (throughput and application startup time)
  - Scalability
  - Reliability, Availability and Serviceability (RAS)

## Java SDK 6.0 – What' new?

Java 6 SDK focuses on platform exploitation, stability, performance and diagnostics

- XML
- XSLT Processing
- Classpath checking
- Data sharing between Java Virtual Machines (JVMs)
- Enhanced diagnostics information

## Just-in-time compiler

- The just-in-time compiler (JIT) is not really part of the JVM, but is essential for a **high performing Java application**

- Java is Write Once Run Anywhere thus it is interpreted by nature and without the JIT could not compete with native code applications

- As your code accesses methods the **JIT determines how frequently specific methods are accessed** and compiles those touched often quickly to optimize performance

- **-Xquickstart** helps to improve JVM startup time for short running Java applications
  – causes the JIT to run with a subset of optimizations

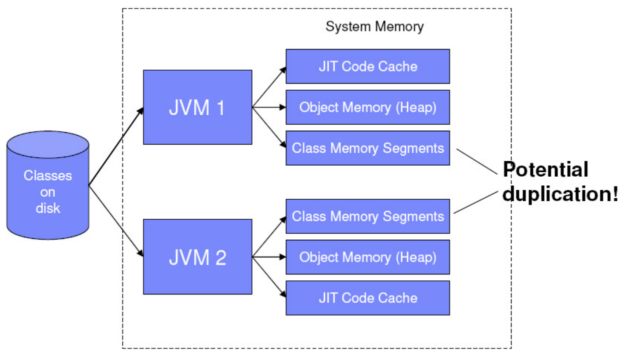
## Garbage collection

Memory management is configurable using different policies

1. **Optimize for Throughput** – flat heap collector focused on maximum throughput
   - -Xgcpolicy:optthruput
2. **Optimize for Pause Time** – flat heap collector with concurrent mark and sweep to minimize GC pause time
   - -Xgcpolicy:optavgpause
3. **Generational Concurrent** – divides heap into “nursery” and “tenured” segments providing fast collection for short lived objects. Can provide maximum throughput with minimal pause times
   - -Xgcpolicy:gencon
4. **Subpool** – a flat heap technique to help increase performance on multiprocessor systems , commonly greater than 8. Available on IBM System p and System z
   - -Xgcpolicy:subpool

## Shared classes

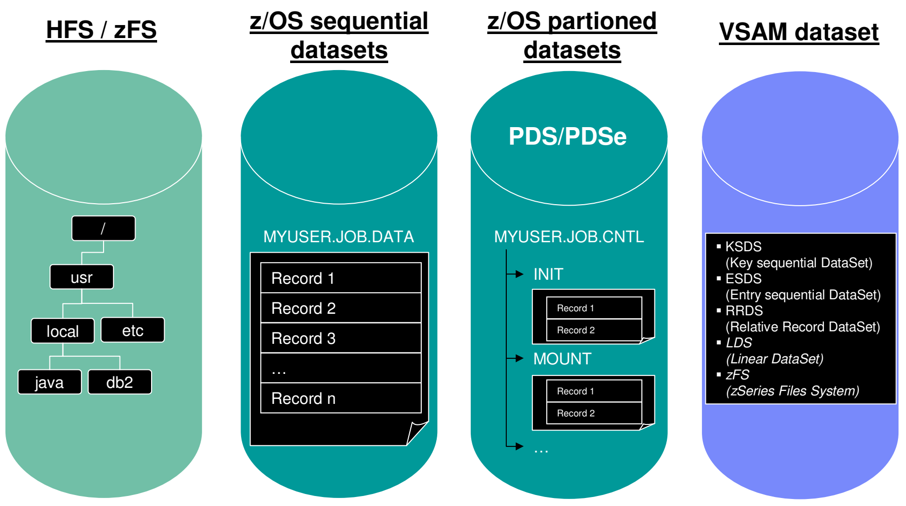
- A shared class area for one or more JVMs
- **Improves startup** time
  - – Lots of classes are already preloaded
- One class cache for many JVMs

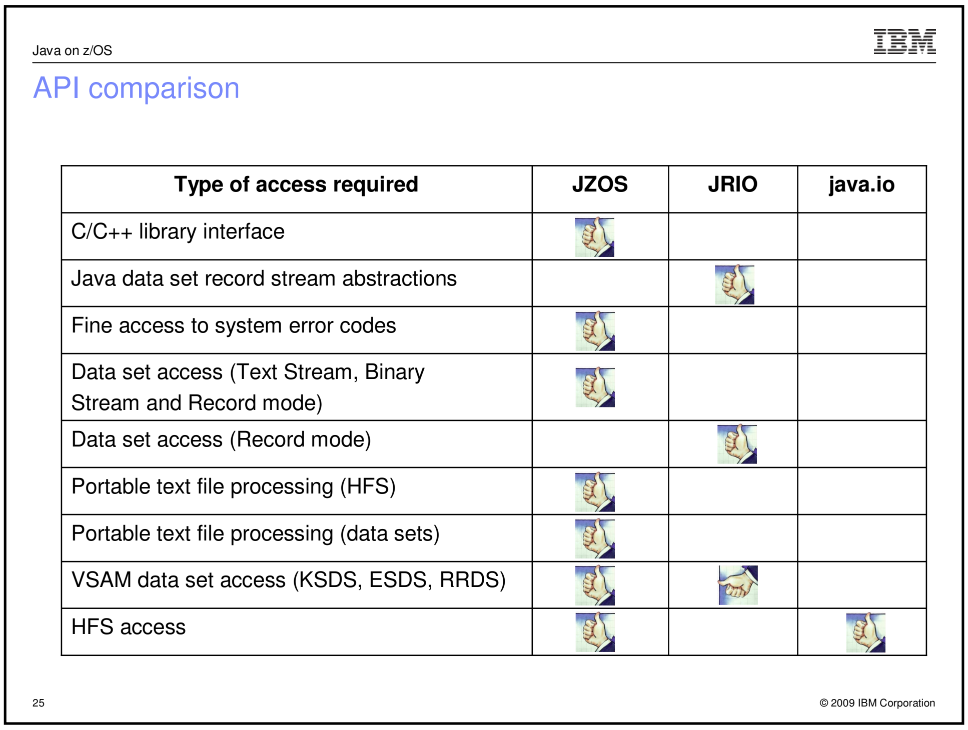


## Agenda

- Java runtime environments on z/OS
- Java SDK 5 and 6
- Java System Resource Integration
- Java Backend Integration
- Java development for z/OS

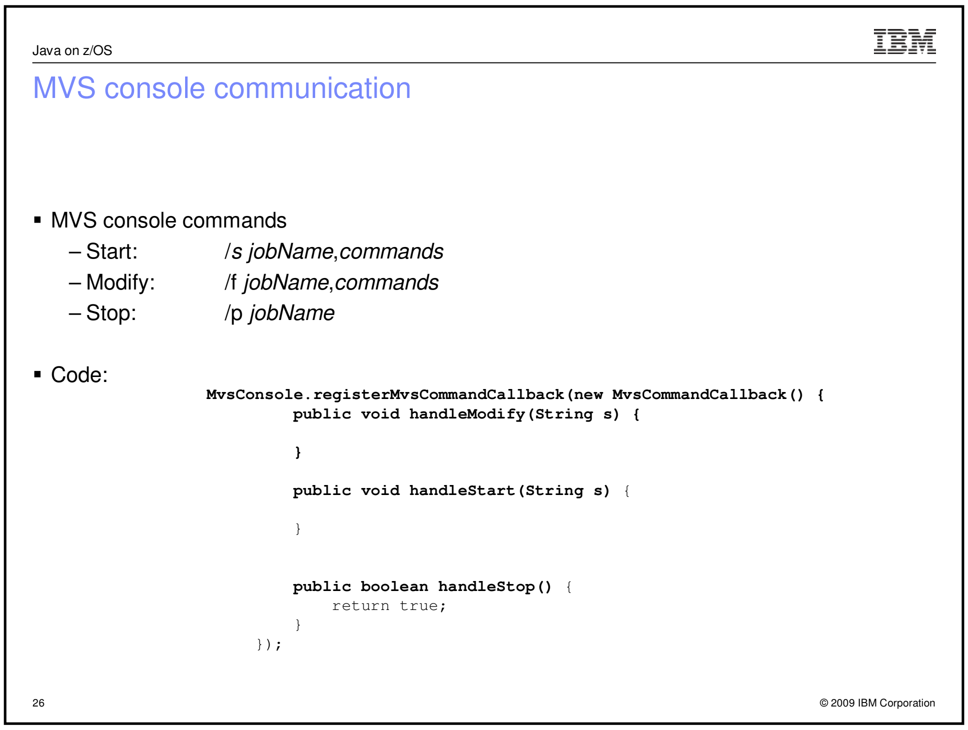
## A short z/OS data overview





## API comparison

| Type of access required | JZOS | JRIO | java.io |
|---|---|---|---|
| C/C++ library interface | 👍 | | |
| Java data set record stream abstractions | | 👍 | |
| Fine access to system error codes | 👍 | | |
| Data set access (Text Stream, Binary Stream and Record mode) | 👍 | | |
| Data set access (Record mode) | | 👍 | |
| Portable text file processing (HFS) | 👍 | | |
| Portable text file processing (data sets) | 👍 | | |
| VSAM data set access (KSDS, ESDS, RRDS) | 👍 | 🤝 | |
| HFS access | 👍 | | 👍 |



## MVS console communication

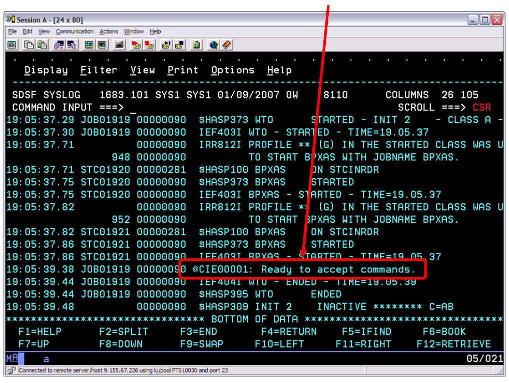
- MVS console commands
  - Start: /s *jobName,commands*
  - Modify: /f *jobName,commands*
  - Stop: /p *jobName*
- Code:

```
MvsConsole.registerMvsCommandCallback(new MvsCommandCallback() {
        public void handleModify(String s) {

        }

        public void handleStart(String s) {

        }


        public boolean handleStop() {
            return true;
        }
    });
```

## MVS console communication

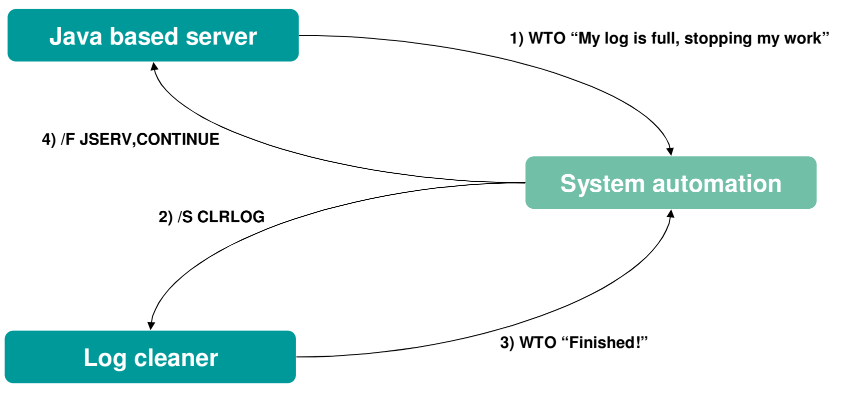
- Write To Operator (WTO) API available

```
{
    MvsConsole.wto("CIE00001: Ready to accept commands.",0x0020, 0x4000);
}
```



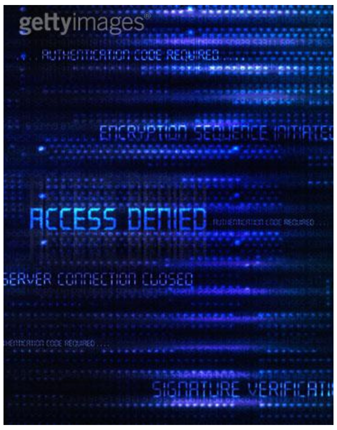
```
Session A - [24 x 80]
File  Edit  View  Communication  Actions  Window  Help

  Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF SYSLOG   1683.101 SYS1 SYS1 01/09/2007 0W    8110      COLUMNS  26 105
 COMMAND INPUT ===> _                                          SCROLL ===> CSR
19:05:37.29 JOB01919 00000090  $HASP373 WTO      STARTED - INIT 2    - CLASS A -
19:05:37.30 JOB01919 00000090  IEF403I WTO - STARTED - TIME=19.05.37
19:05:37.71          00000090  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS U
                 948 00000090          TO START BPXAS WITH JOBNAME BPXAS.
19:05:37.71 STC01920 00000281  $HASP100 BPXAS    ON STCINRDR
19:05:37.75 STC01920 00000090  $HASP373 BPXAS    STARTED
19:05:37.75 STC01920 00000090  IEF403I BPXAS - STARTED - TIME=19.05.37
19:05:37.82          00000090  IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS U
                 952 00000090          TO START BPXAS WITH JOBNAME BPXAS.
19:05:37.82 STC01921 00000281  $HASP100 BPXAS    ON STCINRDR
19:05:37.86 STC01921 00000090  $HASP373 BPXAS    STARTED
19:05:37.86 STC01921 00000090  IEF403I BPXAS - STARTED - TIME=19.05.37
19:05:39.38 JOB01919 00000090 @CIE00001: Ready to accept commands.
19:05:39.44 JOB01919 00000090  IEF404I WTO - ENDED - TIME=19.05.39
19:05:39.44 JOB01919 00000090  $HASP395 WTO      ENDED
19:05:39.48          00000090  $HASP309 INIT 2    INACTIVE ******** C=AB
******************************** BOTTOM OF DATA ********************************
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
MA     a                                                                  05/021
Connected to remote server/host 9.155.67.226 using lu/pool FTS10030 and port 23
```

## Java and z/OS console communication



## Security interfaces in the IBM z/OS Java SDK

- Java Cryptography Extension (IBMJCE)
  - Java Cryptography Extension in Java 2 Platform Standard Edition, Hardware Cryptography (IBMJCECCA)
- Java Secure Sockets Extension (IBMJSSE)
- Java Certification Path (CertPath)
- Java Authentication and Authorization Service (JAAS)
- Java Generic Security Services (JGSS)
- SAF interfaces

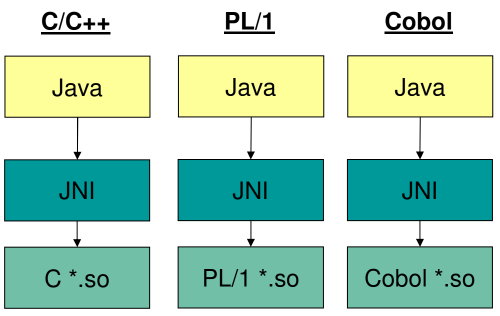


## SMF API

- There are existing C API for SMF
- This API can be wrapped via JNI
  - You can use this API to write SMF records that write CPU time on a user level instead on application level for accounting reasons
  - You can also use it for writing data concerning access violations
  - Now also available as new Java API on Alphaworks for evaluation (together with new Features like DFSORT, Job management, z/OS Logstreams) http://www.alphaworks.ibm.com/tech/zosjavabatchtk

## JNI – Accessing C, C++, PL/1, Cobol

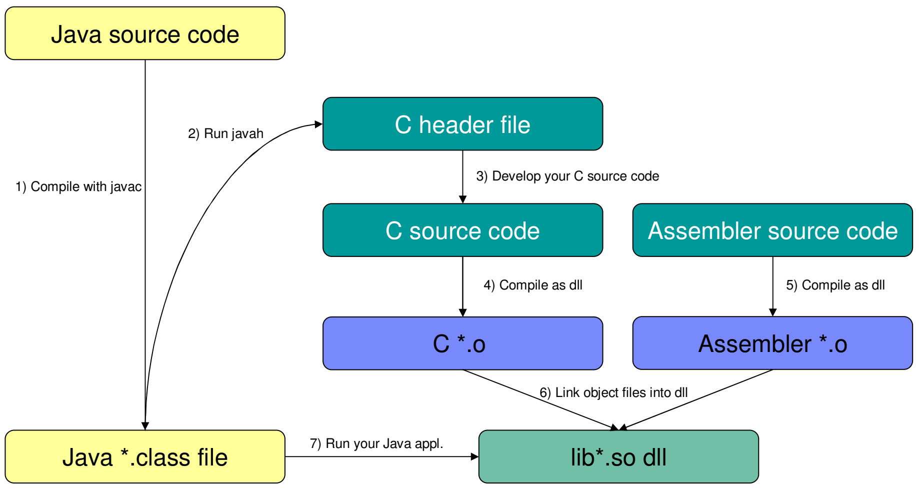
- Write Java Code
- Compile Java code with javac.exe
- C/C++:
  Run javah.exe → *.h is generated
- Write native code, compile it and link it into a shared library



- You can also call explicitly a JVM From C/C++, Cobol and PL/1!

**Note: JNI is a wonderful thing, but treat it with care!**
**It might be dangerous to your JVM…**

## Assembler access from Java

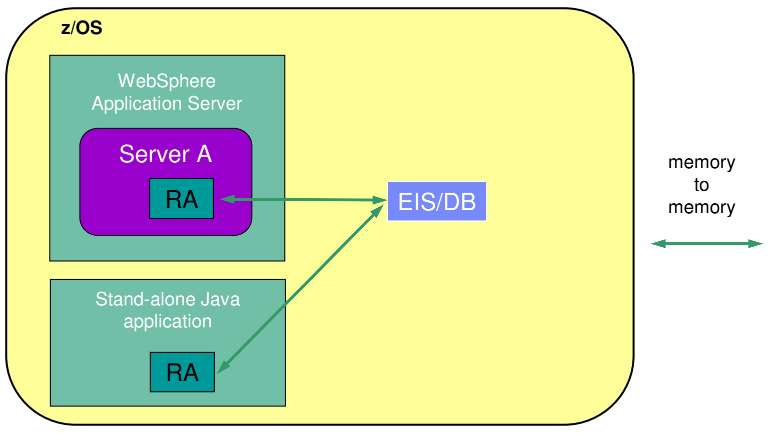


**Reference**: IBM Redbook SG24-7177-00 (Java Stand-alone Applications on z/OS, Volume I
*http://www.redbooks.ibm.com/abstracts/sg247177.html*

## Agenda

- Java runtime environments on z/OS
- Java SDK 5 and 6
- Java System Resource Integration
- Java Backend Integration
- Java development for z/OS

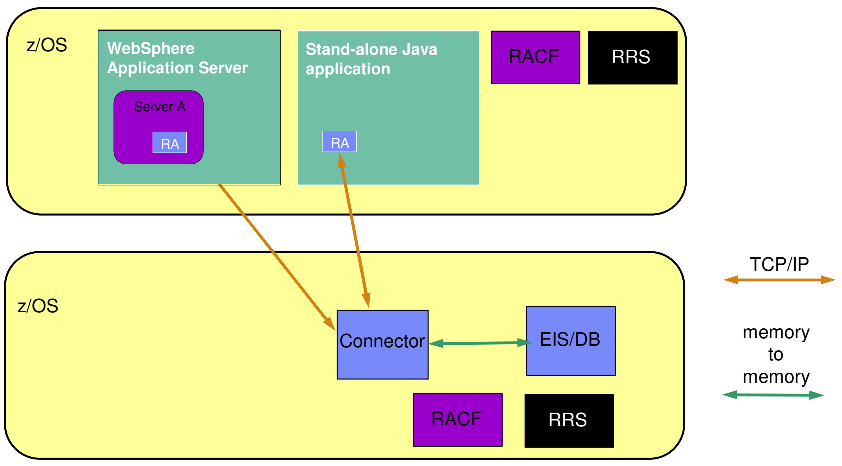
## Sample of a local connection



## Attributes of a local connection

- Performance
  – Is good since there is no network delay
- Availability
  – Is good as long as Java app and the target EIS/DB are both available
- Scalability
  – Is available only by adding additional servers
- Security
  – Is able to use the thread identity or RUNAS value as the connection identity
- Transactionality
  – 2PC is supported when all resource managers are RRS enabled

## Sample of a remote connection



## Attributes of a remote connection

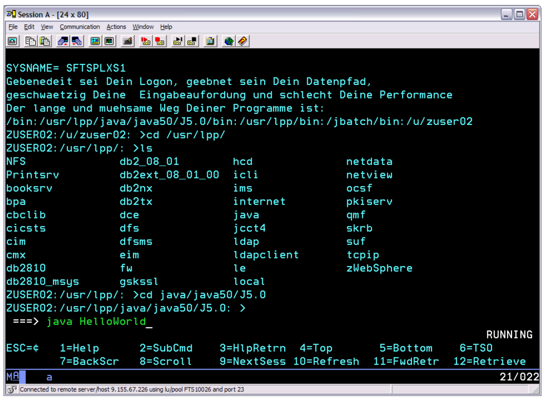
- Performance
  - Can be adversely affected by any network delay
- Availability
  - The restarts are faster, less (failed) components to restart
  - If multiple LPARs are used, better availability
- Scalability
  - Is easier since options like using Sysplex Distributor are available
- Security
  - Does not offer as many options for connection identities
- Transactionality
  - Can be limited in circumstances when transactions span multiple LPARs
  - Recovery of in-doubt LUWs can become complex

## Agenda

- Java runtime environments on z/OS
- Java SDK 5 and 6
- Java System Resource Integration
- Java Backend Integration
- Java development for z/OS

## How to develop Java applications for the mainframe

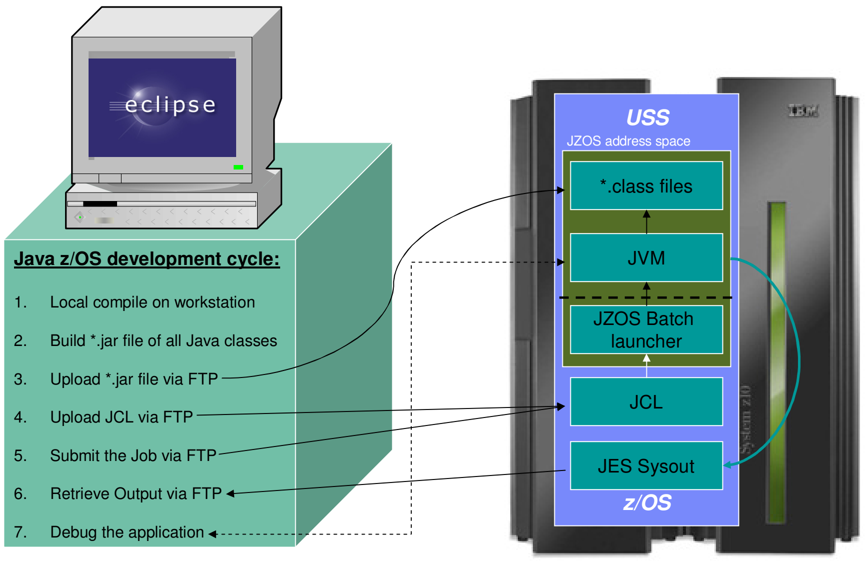
- This is a picture people often associate with the mainframe:



```
SYSNAME= SFTSPLXS1
Gebenedeit sei Dein Logon, geebnet sein Dein Datenpfad,
geschwaetzig Deine  Eingabeaufordung und schlecht Deine Performance
Der lange und muehsame Weg Deiner Programme ist:
/bin:/usr/lpp/java/java50/J5.0/bin:/usr/lpp/bin:/jbatch/bin:/u/zuser02
ZUSER02:/u/zuser02: >cd /usr/lpp/
ZUSER02:/usr/lpp/: >ls
NFS                 db2_08_01          hcd                 netdata
Printsrv            db2ext_08_01_00    icli                netview
booksrv             db2nx              ims                 ocsf
bpa                 db2tx              internet            pkiserv
cbclib              dce                java                qmf
cicsts              dfs                jcct4               skrb
cim                 dfsms              ldap                suf
cmx                 eim                ldapclient          tcpip
db2810              fw                 le                  zWebSphere
db2810_msys         gskssl             local
ZUSER02:/usr/lpp/: >cd java/java50/J5.0
ZUSER02:/usr/lpp/java/java50/J5.0: >
 ===> java HelloWorld_
                                                                      RUNNING
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
MA     a                                                               21/022
Connected to remote server/host 9.155.67.226 using lu/pool FTS10026 and port 23
```

- ... But it is much **easier**!
  - Eclipse as an IDE can be easily used for Mainframe Java development

## Development tools for Java Batch: 1) Eclipse



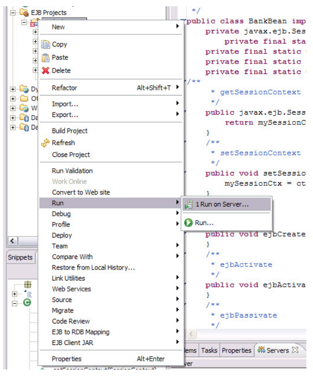
**Java z/OS development cycle:**

1. Local compile on workstation
2. Build *.jar file of all Java classes
3. Upload *.jar file via FTP
4. Upload JCL via FTP
5. Submit the Job via FTP
6. Retrieve Output via FTP
7. Debug the application

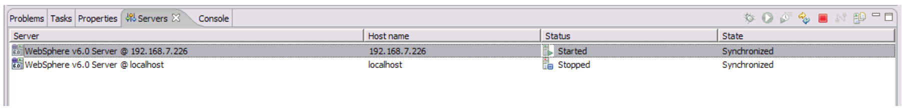
## Development tools for Java Batch: 2) RDz

RDz and J2EE development:

- RDz is put on top of Rational Application Developer
- **Integrated** WebSphere Application **Server test environment**
- **Remote deployment** of applications
- Wizards for EJB creation
- EJB Test client



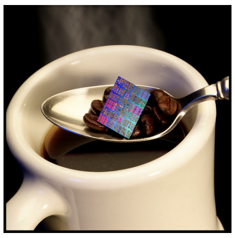
```
 */
public class BankBean imp
    private javax.ejb.Ses
        private final sta
    private final static
    private final static
    private final static
/**
  * getSessionContext
  */
    public javax.ejb.Sess
        return mySessionC
    }
    /**
  * setSessionContext
  */
    public void setSessio
        mySessionCtx = ct
    }
    public void ejbCreate
    }
    /**
  * ejbActivate
  */
    public void ejbActiva
    }
    /**
  * ejbPassivate
  */
```



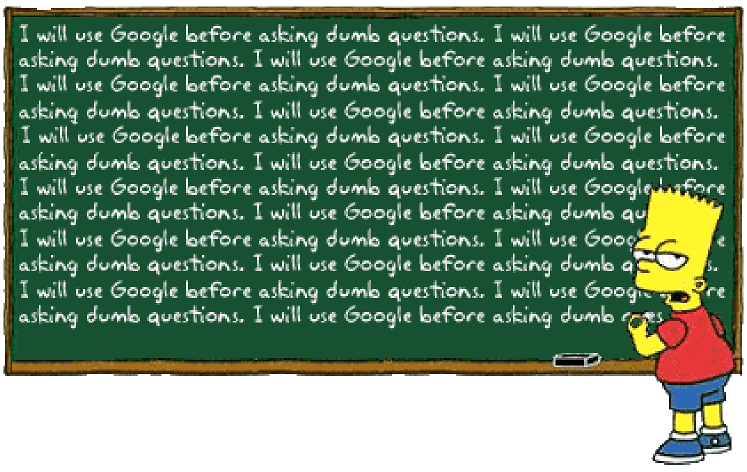
| Server | Host name | Status | State |
|---|---|---|---|
| WebSphere v6.0 Server @ 192.168.7.226 | 192.168.7.226 | Started | Synchronized |
| WebSphere v6.0 Server @ localhost | localhost | Stopped | Synchronized |

## Summary: For which Java applications does a mainframe fit?

- Batch is still one of the mainframes biggest strengths
  - The mainframe was designed for batch (punch cards)
  - The mainframe has the longest experience in the batch environment
  - Special facilities in z/OS allow a huge complex job management for batch jobs (JES, SDSF,...)
  - Java inherits these functionalities
- Business critical Java based servers that need:
  - High availability (99,999%)
  - Best security
- Java applications which use lots of transactions
  - Data proximity



## Questions?


I will use Google before asking dumb questions. I will use Google before asking dumb questions. I will use Google before asking dumb questions. I will use Google before asking dumb questions. I will use Google before asking dumb questions. I will use Google before asking dumb questions. I will use Google before asking dumb questions. I will use Google before asking dumb questions.