Talking Java from Delphi the real world

Jeroen Pluimers
better office benelux

Agenda… ☺

- Sometimes you need to access Java code from your Delphi applications.
- This is done through the Java Native Interfaces, and this session shows you how to use that interface to call Java methods, instantiate objects, perform Garbage Collection, measure memory usage, and provide access to the JNI instrumentation interface.
- Also you will learn how to marshal data to and from your Java layer.

Part 1

The app that lead
me to writing this session

Source of all this

- Insurance company
  - sells products
    through insurance brokers
    - WebSphere Extranet site
    - Delphi win32 app
- Both share:
    - Java core
      - has all generic product knowledge
    - Delphi calculation boxes
      - know about actuarial aspects

Delphi from Java

- Not much of an issue here
  - Calculations done once per session
  - Similar to “WinExecAndWait”

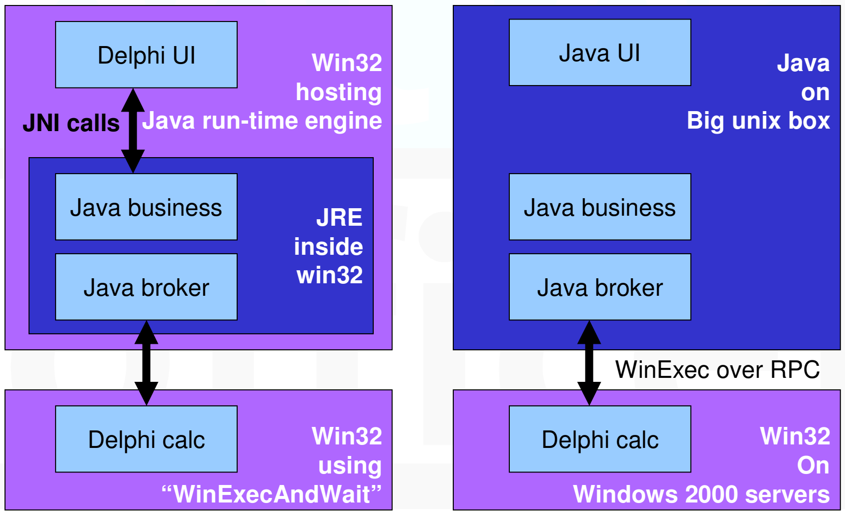
  - We don’t do it here

Java from Delphi

- Tricky, as it needs to be in-process
  - All user-interaction steered by Java core
  - A lot of user-interaction
    - Demo
  - Out-of-process is too slow

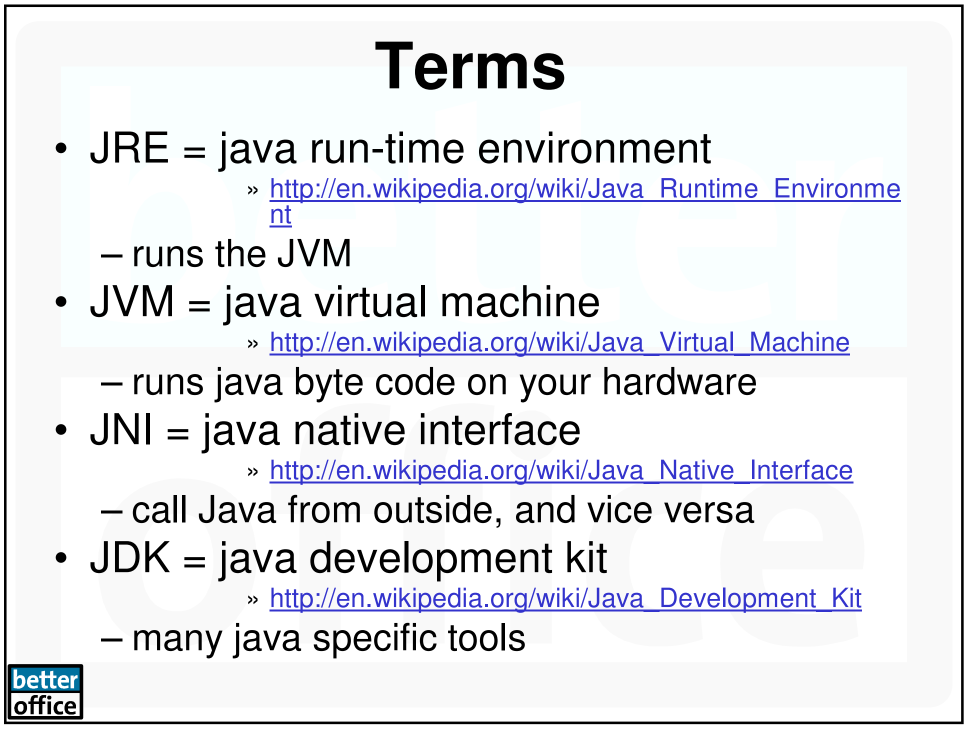
Shared structure

- Win32
- WebSpere



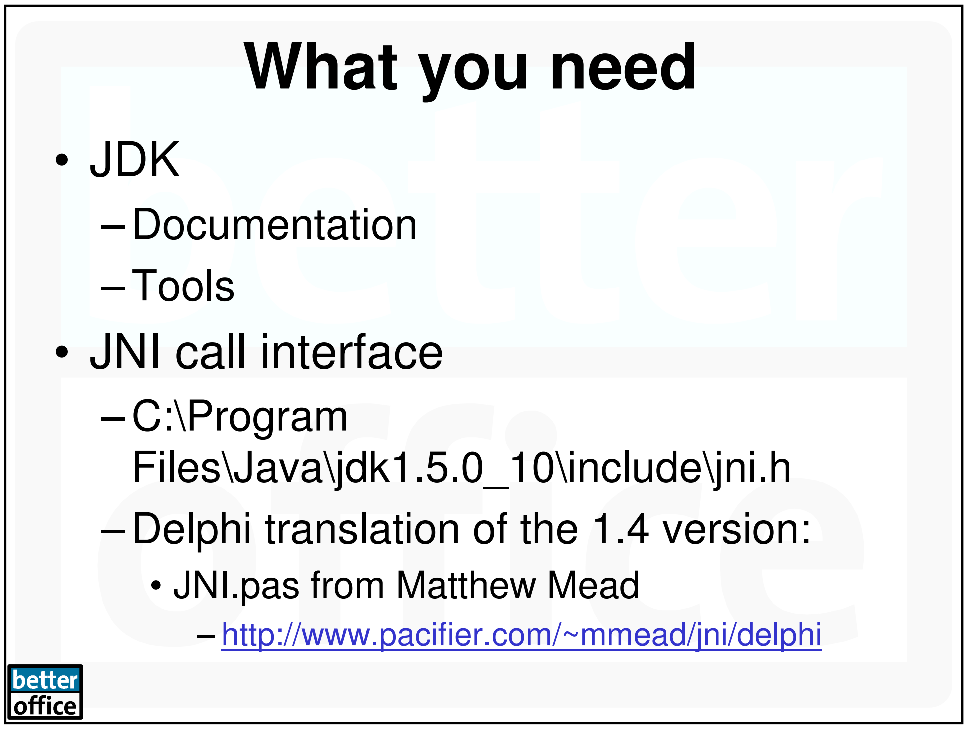
Part 2

Info and tools needed



Terms

- JRE = java run-time environment
  - » http://en.wikipedia.org/wiki/Java_Runtime_Environment
  - – runs the JVM
- JVM = java virtual machine
  - » http://en.wikipedia.org/wiki/Java_Virtual_Machine
  - – runs java byte code on your hardware
- JNI = java native interface
  - » http://en.wikipedia.org/wiki/Java_Native_Interface
  - – call Java from outside, and vice versa
- JDK = java development kit
  - » http://en.wikipedia.org/wiki/Java_Development_Kit
  - – many java specific tools



What you need

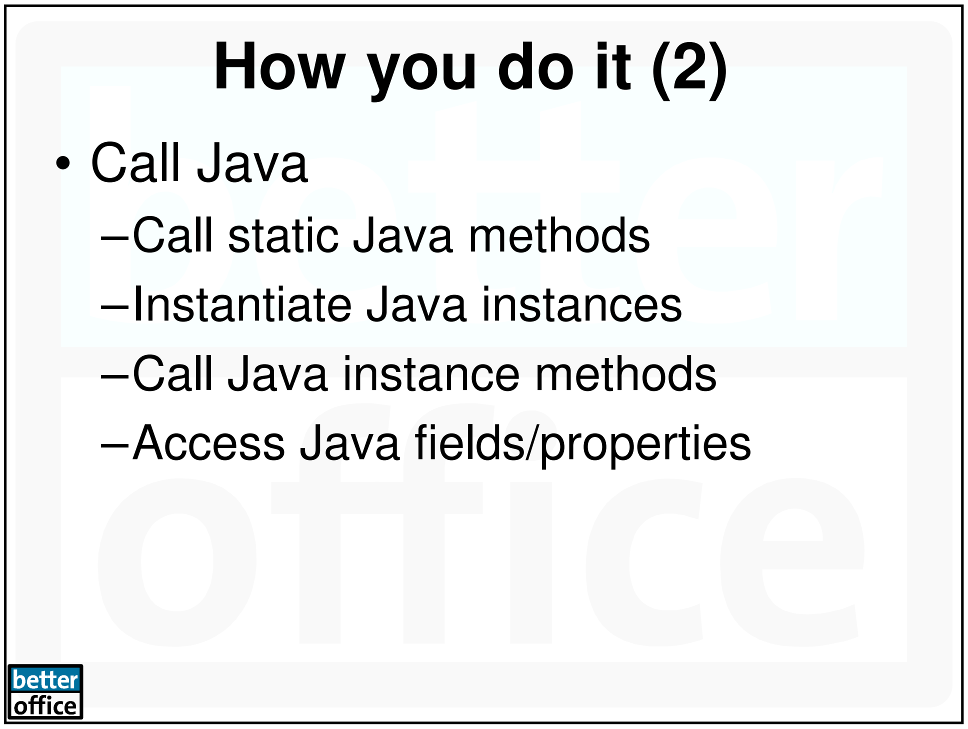
- JDK
  - – Documentation
  - – Tools
- JNI call interface
  - – C:\Program Files\Java\jdk1.5.0_10\include\jni.h
  - – Delphi translation of the 1.4 version:
    - JNI.pas from Matthew Mead
      - – http://www.pacifier.com/~mmead/jni/delphi

Part 3

The “how-to”

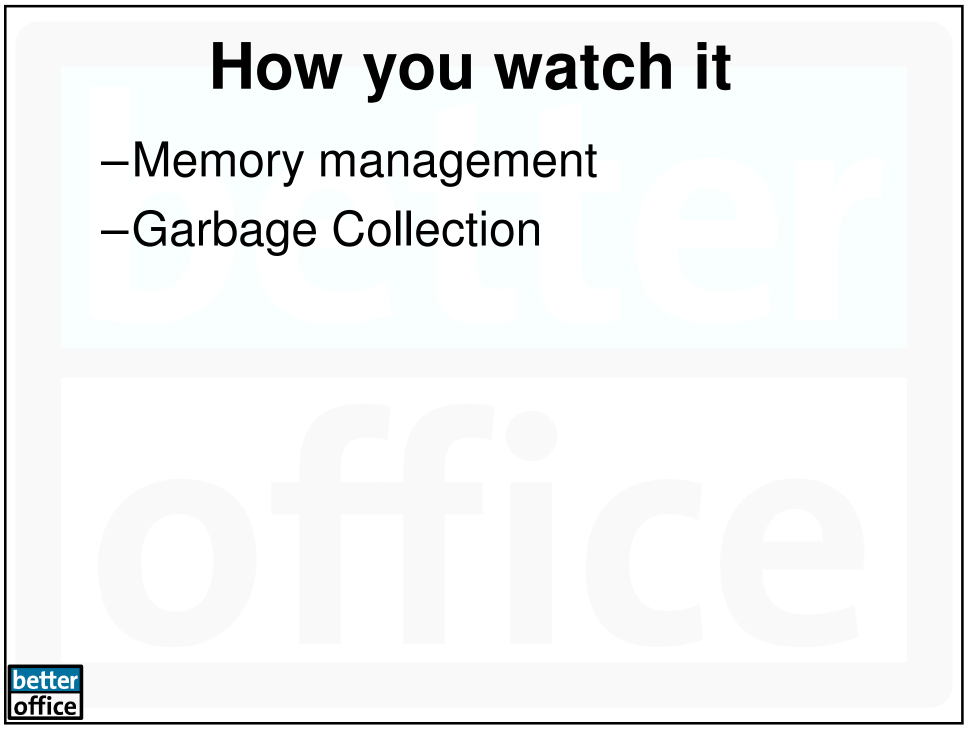
How you do it

- Setup your environment
  - –Detect available JRE’s
  - –Host a JRE’s JVM inside a win32 process
    - “un”hosting a JVM is not possible!



How you do it (2)

- Call Java
  - –Call static Java methods
  - –Instantiate Java instances
  - –Call Java instance methods
  - –Access Java fields/properties



How you watch it

- –Memory management
- –Garbage Collection

Part 4

Easy demo’s

Demo: detecting JRE’s

- HKEY_LOCAL_MACHINE \SOFTWARE\JavaSoft \Java Runtime Environment
  - Value of “CurrentVersion” allows switching
  - Sub-keys determine available JVM’s
    - Value of “RuntimeLib” has full path to jvm.dll
- You can have private JRE’s
  - Be aware of security vulnerabilities!

Demo: loading JVM

- JNI.pas has a TJavaVM
  - Extension with readable error messages:

```
type
  TJavaVMEx = class(TJavaVM)
  public
    destructor Destroy; override;
    class procedure CheckJNIResult(ErrorCode: Integer);
    class function ErrorMessageFromCode(ErrorCode: Integer): string; overload;
  end;

destructor TJavaVMEx.Destroy;
begin
  //##jwp: The JVM cannot be unloaded; see also
  // http://forum.java.sun.com/thread.jspa?threadID=454590&messageID=2078392
  // http://java.sun.com/docs/books/jni/html/invoke.html
//  if Assigned(JavaVM) then
//    CheckJNIResult(JavaVM^.DestroyJavaVM(JavaVM));
  inherited;
end;

class procedure TJavaVMEx.CheckJNIResult(ErrorCode: Integer);
begin
  if ErrorCode < 0 then
    raise EJNIErrorFromCode.Create(ErrorCode);
end;
```

Demo: loading JVM (2)

```
class function TJavaVMEx.ErrorMessageFromCode(ErrorCode: Integer): string;
var
  ErrorMessage: string;
begin
  if ErrorCode > 0 then
    ErrorMessage := 'No Error'
  else
    case ErrorCode of
      JNI_OK:        ErrorMessage := 'JNI_OK: success';
      JNI_ERR:       ErrorMessage := 'JNI_ERR: unknown error';
      JNI_EDETACHED:
        ErrorMessage := 'JNI_EDETACHED: thread detached from the VM';
      JNI_EVERSION:  ErrorMessage := 'JNI_EVERSION: JNI version error';
      JNI_ENOMEM:    ErrorMessage := 'JNI_ENOMEM: not enough memory';
      JNI_EEXIST:    ErrorMessage := 'JNI_EEXIST: VM already created';
      JNI_EINVAL:    ErrorMessage := 'JNI_EINVAL: invalid arguments';
    else
      ErrorMessage := 'Unknown JNI error';
    end;
  Result := Format('%s, code = %d(0x%x).', [ErrorMessage, ErrorCode,
   ErrorCode]);
end;
```

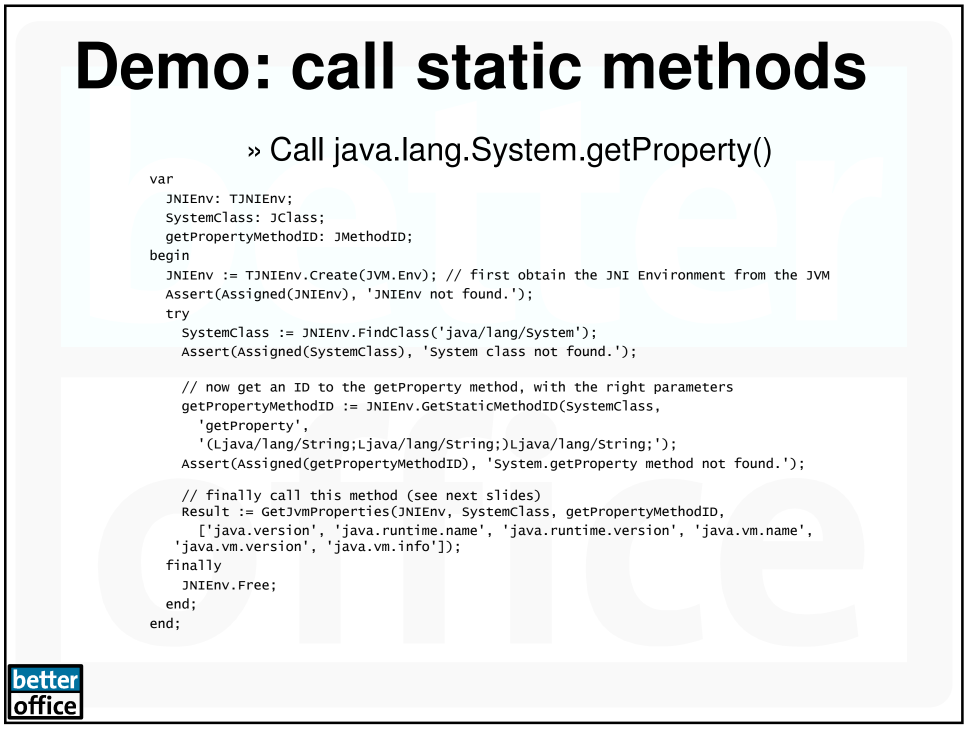
Demo: loading JVM (3)

```
type
  EJNIErrorFromCode = class(EJNIError)
    constructor Create(AErrorCode: Integer);
  private
    FErrorCode: Integer;
  public
    function ErrorMessageFromCode: string;
    property ErrorCode: Integer read FErrorCode;
  end;

constructor EJNIErrorFromCode.Create(AErrorCode: Integer);
begin
  FErrorCode := AErrorCode;
  inherited Create(ErrorMessageFromCode);
end;

function EJNIErrorFromCode.ErrorMessageFromCode: string;
begin
  Result := TJavaVMEx.ErrorMessageFromCode(ErrorCode);
end;
```

Demo: loading JVM (4)

» Core loading logic

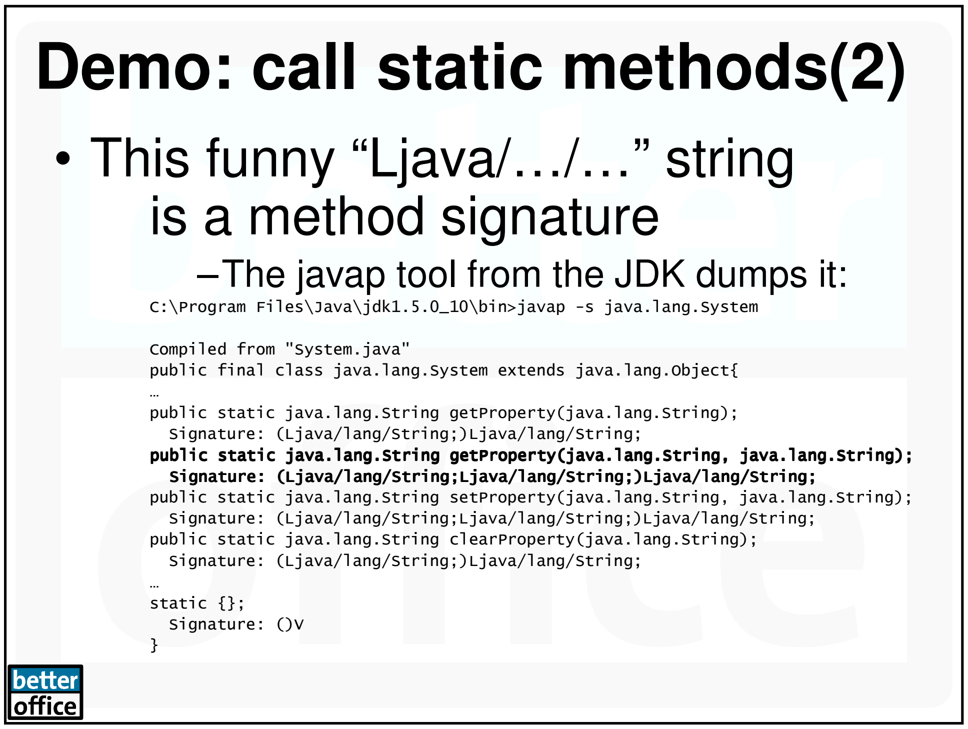
```
uses
  …, JNI, …;
procedure TestLoadingJVM(JvmDllPath: string): string;
var
  J: TJavaVMEx;
  VM_args: JavaVMInitArgs;
begin
  try
    J := TJavaVMEx.Create(JNI_VERSION_1_2, JvmDllPath);
    try
      FillChar(VM_args, SizeOf(VM_args), 0);
      VM_args.version := J.Version;
      J.CheckJNIResult(J.LoadVM(VM_args));
    finally
      J.Free;
    end;
  except
    on E: Exception do
    begin
      OutputDebugString(E.Message);
      raise;
    end;
  end;
end;
```



Demo: call static methods

» Call java.lang.System.getProperty()

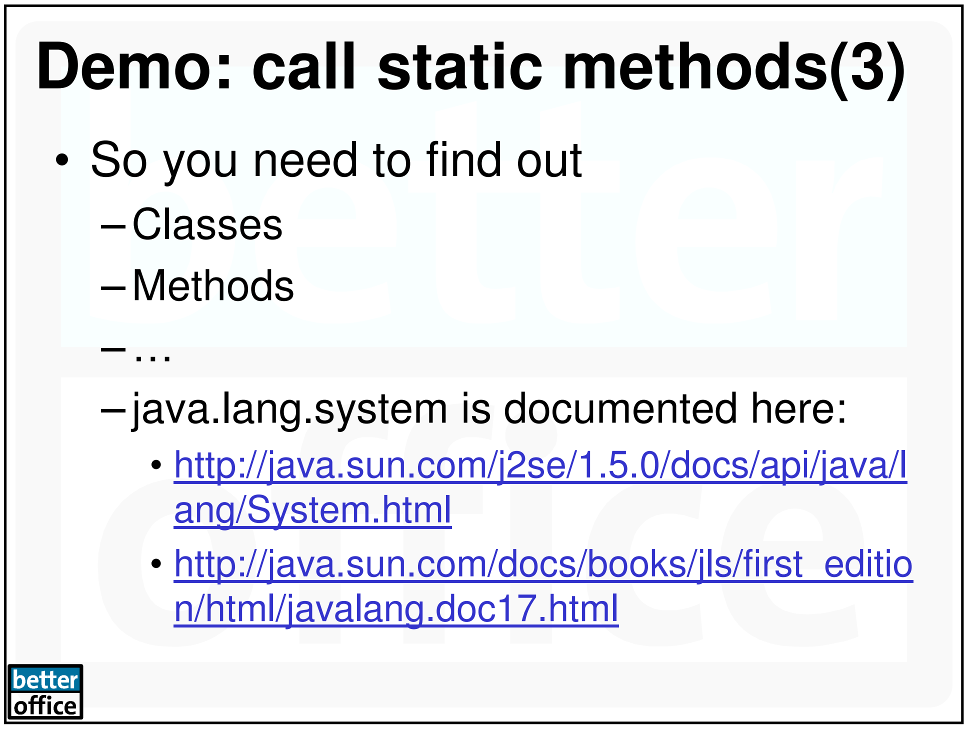
```
var
  JNIEnv: TJNIEnv;
  SystemClass: JClass;
  getPropertyMethodID: JMethodID;
begin
  JNIEnv := TJNIEnv.Create(JVM.Env); // first obtain the JNI Environment from the JVM
  Assert(Assigned(JNIEnv), 'JNIEnv not found.');
  try
    SystemClass := JNIEnv.FindClass('java/lang/System');
    Assert(Assigned(SystemClass), 'System class not found.');

    // now get an ID to the getProperty method, with the right parameters
    getPropertyMethodID := JNIEnv.GetStaticMethodID(SystemClass,
      'getProperty',
      '(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;');
    Assert(Assigned(getPropertyMethodID), 'System.getProperty method not found.');

    // finally call this method (see next slides)
    Result := GetJvmProperties(JNIEnv, SystemClass, getPropertyMethodID,
      ['java.version', 'java.runtime.name', 'java.runtime.version', 'java.vm.name',
   'java.vm.version', 'java.vm.info']);
  finally
    JNIEnv.Free;
  end;
end;
```



Demo: call static methods(2)

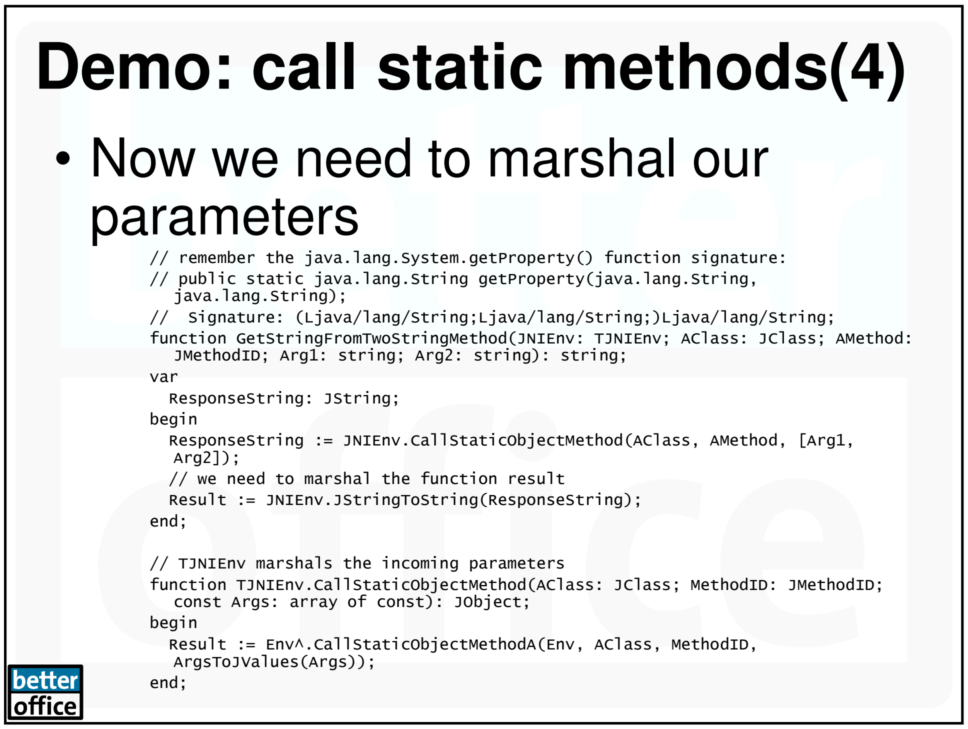
- This funny “Ljava/…/…” string is a method signature
  - –The javap tool from the JDK dumps it:

```
C:\Program Files\Java\jdk1.5.0_10\bin>javap -s java.lang.System

Compiled from "System.java"
public final class java.lang.System extends java.lang.Object{
…
public static java.lang.String getProperty(java.lang.String);
  Signature: (Ljava/lang/String;)Ljava/lang/String;
public static java.lang.String getProperty(java.lang.String, java.lang.String);
  Signature: (Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
public static java.lang.String setProperty(java.lang.String, java.lang.String);
  Signature: (Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
public static java.lang.String clearProperty(java.lang.String);
  Signature: (Ljava/lang/String;)Ljava/lang/String;
…
static {};
  Signature: ()V
}
```



Demo: call static methods(3)

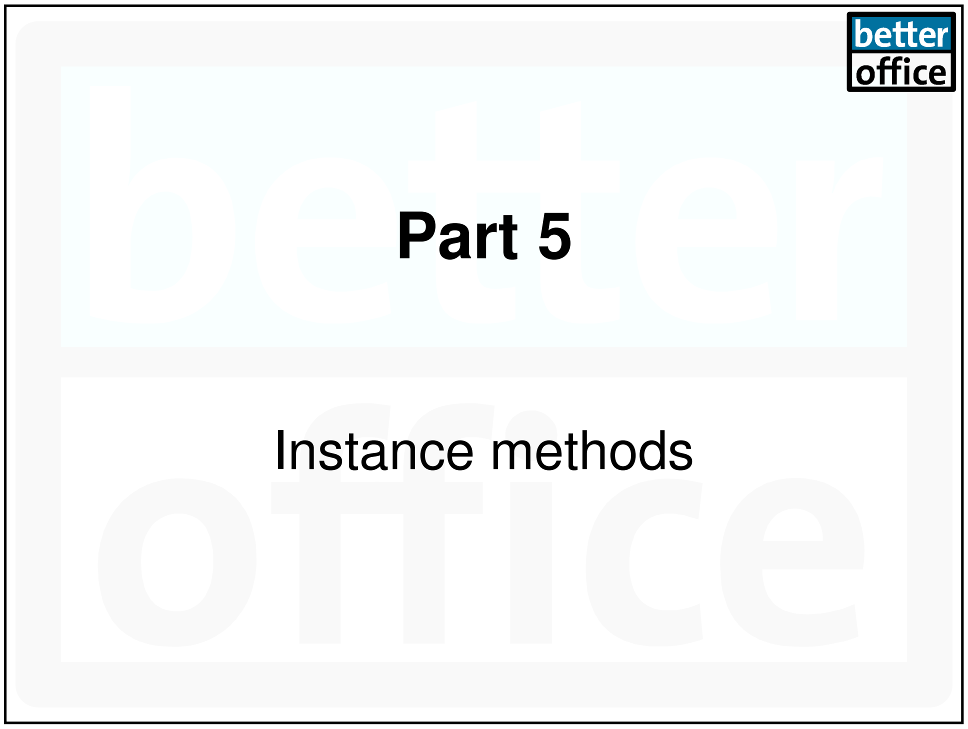
- So you need to find out
  - Classes
  - Methods
  - …
  - java.lang.system is documented here:
    - http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html
    - http://java.sun.com/docs/books/jls/first_edition/html/javalang.doc17.html



Demo: call static methods(4)

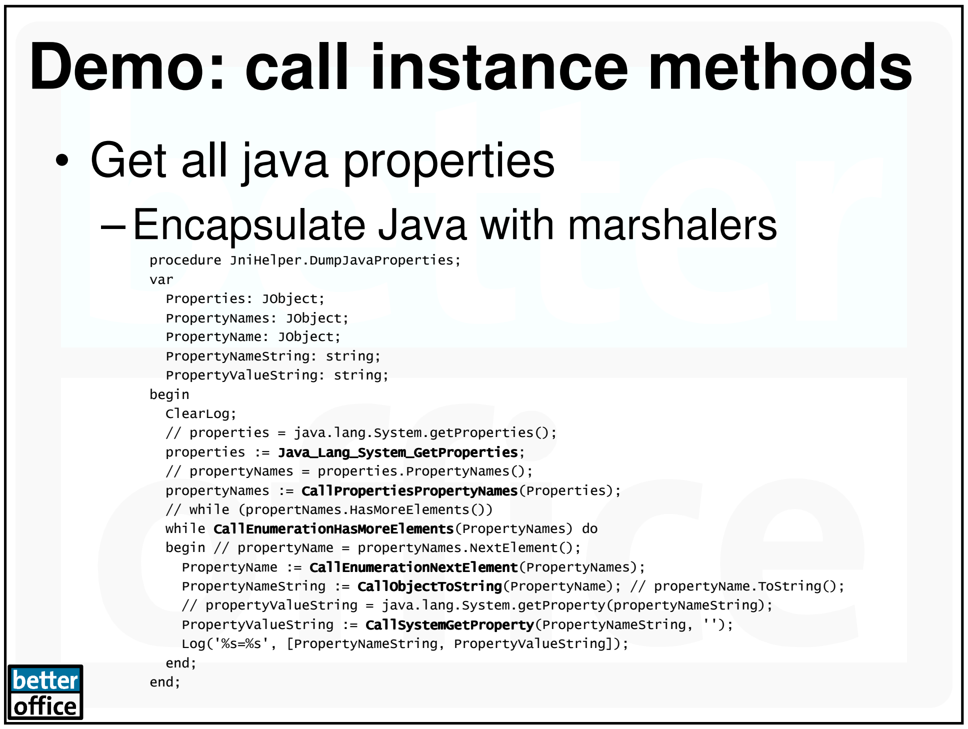
- Now we need to marshal our parameters

```
// remember the java.lang.System.getProperty() function signature:
// public static java.lang.String getProperty(java.lang.String,
   java.lang.String);
//  Signature: (Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
function GetStringFromTwoStringMethod(JNIEnv: TJNIEnv; AClass: JClass; AMethod:
   JMethodID; Arg1: string; Arg2: string): string;
var
  ResponseString: JString;
begin
  ResponseString := JNIEnv.CallStaticObjectMethod(AClass, AMethod, [Arg1,
   Arg2]);
  // we need to marshal the function result
  Result := JNIEnv.JStringToString(ResponseString);
end;

// TJNIEnv marshals the incoming parameters
function TJNIEnv.CallStaticObjectMethod(AClass: JClass; MethodID: JMethodID;
   const Args: array of const): JObject;
begin
  Result := Env^.CallStaticObjectMethodA(Env, AClass, MethodID,
   ArgsToJValues(Args));
end;
```



Part 5

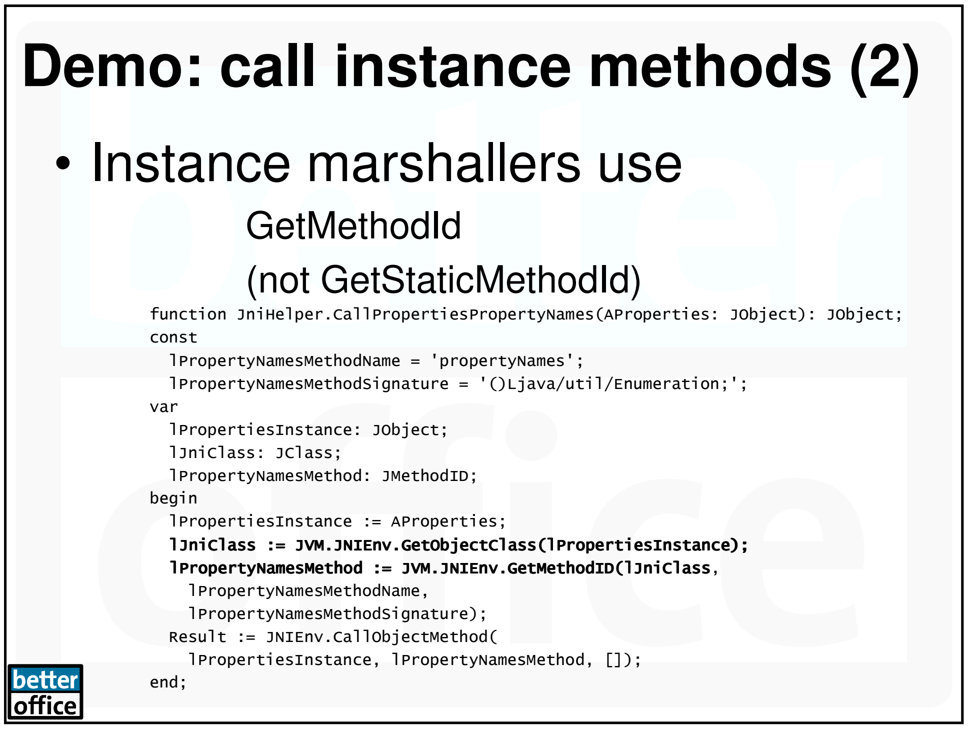
Instance methods



Demo: call instance methods

- Get all java properties
  - Encapsulate Java with marshalers

```
procedure JniHelper.DumpJavaProperties;
var
  Properties: JObject;
  PropertyNames: JObject;
  PropertyName: JObject;
  PropertyNameString: string;
  PropertyValueString: string;
begin
  ClearLog;
  // properties = java.lang.System.getProperties();
  properties := Java_Lang_System_GetProperties;
  // propertyNames = properties.PropertyNames();
  propertyNames := CallPropertiesPropertyNames(Properties);
  // while (propertNames.HasMoreElements())
  while CallEnumerationHasMoreElements(PropertyNames) do
  begin // propertyName = propertyNames.NextElement();
    PropertyName := CallEnumerationNextElement(PropertyNames);
    PropertyNameString := CallObjectToString(PropertyName); // propertyName.ToString();
    // propertyValueString = java.lang.System.getProperty(propertyNameString);
    PropertyValueString := CallSystemGetProperty(PropertyNameString, '');
    Log('%s=%s', [PropertyNameString, PropertyValueString]);
  end;
end;
```



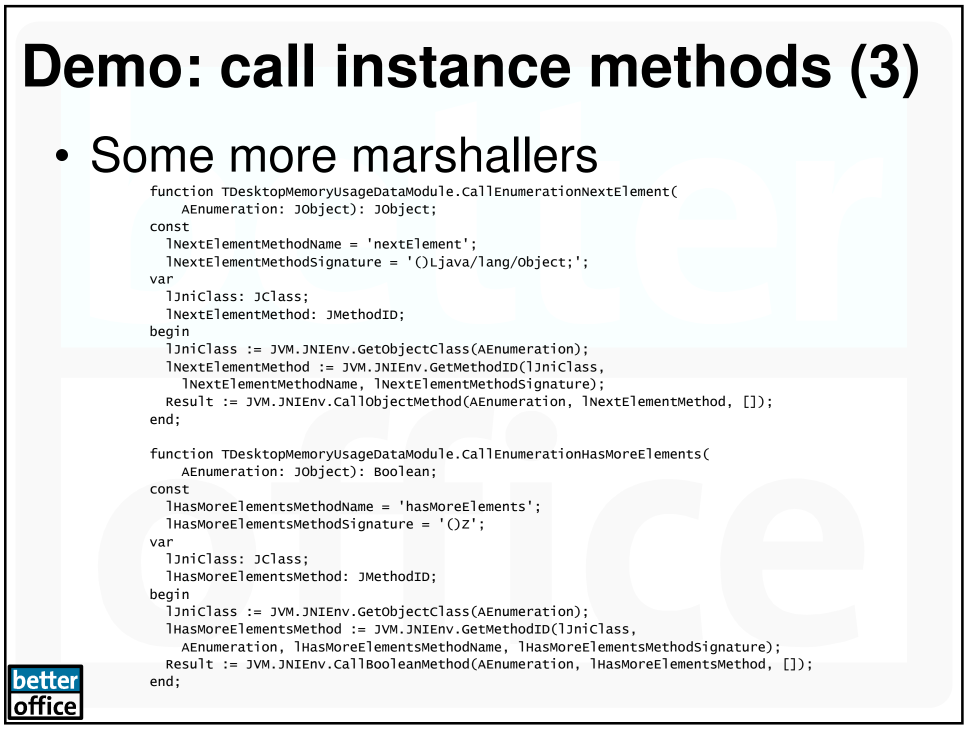
Demo: call instance methods (2)

- Instance marshallers use

  GetMethodId

  (not GetStaticMethodId)

```
function JniHelper.CallPropertiesPropertyNames(AProperties: JObject): JObject;
const
  lPropertyNamesMethodName = 'propertyNames';
  lPropertyNamesMethodSignature = '()Ljava/util/Enumeration;';
var
  lPropertiesInstance: JObject;
  lJniClass: JClass;
  lPropertyNamesMethod: JMethodID;
begin
  lPropertiesInstance := AProperties;
  lJniClass := JVM.JNIEnv.GetObjectClass(lPropertiesInstance);
  lPropertyNamesMethod := JVM.JNIEnv.GetMethodID(lJniClass,
    lPropertyNamesMethodName,
    lPropertyNamesMethodSignature);
  Result := JNIEnv.CallObjectMethod(
    lPropertiesInstance, lPropertyNamesMethod, []);
end;
```



Demo: call instance methods (3)

- Some more marshallers

```
function TDesktopMemoryUsageDataModule.CallEnumerationNextElement(
    AEnumeration: JObject): JObject;
const
  lNextElementMethodName = 'nextElement';
  lNextElementMethodSignature = '()Ljava/lang/Object;';
var
  lJniClass: JClass;
  lNextElementMethod: JMethodID;
begin
  lJniClass := JVM.JNIEnv.GetObjectClass(AEnumeration);
  lNextElementMethod := JVM.JNIEnv.GetMethodID(lJniClass,
    lNextElementMethodName, lNextElementMethodSignature);
  Result := JVM.JNIEnv.CallObjectMethod(AEnumeration, lNextElementMethod, []);
end;

function TDesktopMemoryUsageDataModule.CallEnumerationHasMoreElements(
    AEnumeration: JObject): Boolean;
const
  lHasMoreElementsMethodName = 'hasMoreElements';
  lHasMoreElementsMethodSignature = '()Z';
var
  lJniClass: JClass;
  lHasMoreElementsMethod: JMethodID;
begin
  lJniClass := JVM.JNIEnv.GetObjectClass(AEnumeration);
  lHasMoreElementsMethod := JVM.JNIEnv.GetMethodID(lJniClass,
    AEnumeration, lHasMoreElementsMethodName, lHasMoreElementsMethodSignature);
  Result := JVM.JNIEnv.CallBooleanMethod(AEnumeration, lHasMoreElementsMethod, []);
end;
```

Conclusion so far

- Most important things to learn
  - Method signatures
  - Data marshalling
  - Readable encapsulation

- JNI docs help a lot:
  - http://java.sun.com/j2se/1.4.2/docs/guide/jni/spec/jniTOC.html
  - http://java.sun.com/j2se/1.5.0/docs/guide/jni/spec/jniTOC.html
  - http://java.sun.com/j2se/1.5.0/docs/guide/jni/spec/types.html
  - Note Sun tends to move around things, and their stuff is not always well indexed in search engines
    - this url does NOT exist:
      - http://java.sun.com/j2se/1.6.0/docs/guide/jni/spec/jniTOC.html
    - use this one in stead:
      - http://java.sun.com/javase/6/docs/technotes/guides/jni/spec/jniTOC.html
  - Sun tends to move around urls
    - Sometimes external docs are more static:
      - http://www.science.uva.nl/ict/ossdocs/java/tutorial/native1.1/implementing/method.html

Part 6

Memory Management

Global and local references

- Local references
  - Allocated automatically
  - Live guarantee is only 1 JNI call
  - Are automatically released
  - Can be manually allocated by calling
    function NewLocalRef(Ref: JObject): JObject;
  - Can be manually released by calling
    procedure DeleteLocalRef(Obj: JObject);
- Global references
  - Allocate by calling
    function NewGlobalRef(LObj: JObject): JObject;
  - Live any number of calls
  - Need to to be manually released by calling
    procedure DeleteLocalRef(Obj: JObject);
    (use a *try…finally…end* block for this)
- http://java.sun.com/javase/6/docs/technotes/guides/jni/spec/functions.html#global_local

better office

Talking to the memory manager

```
function TDesktopMemoryUsageDataModule.Java_Lang_RunTime_GetRunTime: JObject;
const
  lRunTimeClassName = 'java.lang.Runtime';
  lGetRunTimeMethodName = 'getRuntime';
  lGetRunTimeMethodSignature = '()Ljava/lang/Runtime;'; // let op ;
var
  lRunTimeClass: JClass;
  lGetRunTimeMethod: JMethodID;
begin
 lRunTimeClass := JVM.JNIEnv.FindClass(lRunTimeClassName);
 if lRunTimeClass = nil then
    raise EJNICommunicator.CreateFmt(
      'cannot find static Java class "%s"', [lRunTimeClassName]);
 lGetRunTimeMethod := JVM.JNIEnv.GetStaticMethodID(lRunTimeClass,
   lGetRunTimeMethodName, lGetRunTimeMethodSignature);
 Result := JVM.JNIEnv.CallStaticObjectMethod(lRunTimeClass, lGetRunTimeMethod,
   []);
end;
```

better office

Talking to the memory manager

```
// you also have 'maxMemory' and 'totalMemory'
function CallRunTimeFreeMemory: Integer;
const
  lFreeMemoryMethodName = 'freeMemory';
  lFreeMemoryMethodSignature = '()J';
var
  lRunTimeInstance: JObject;
  lRunTimeClass: JClass;
  lFreeMemoryMethod: JMethodID;
begin
  lRunTimeInstance := Java_Lang_RunTime_GetRunTime;
  // watch it: instance method, so call GetJniMethodID in stead of GetJniStaticMethodID
  lRunTimeClass := JVM.JNIEnv.GetObjectClass(lRunTimeInstance);
  lFreeMemoryMethod := JVM.JNIEnv.GetJniMethodID(lRunTimeClass,
    lFreeMemoryMethodName, lFreeMemoryMethodSignature);
    Result := JVM.JNIEnv.CallIntMethod(lRunTimeInstance, lFreeMemoryMethod, []);
end;

procedure CallRunTimeGC;
const
  lGCMethodName = 'gc';
  lGCMethodSignature = '()V';
var
  lRunTimeInstance: JObject;
  lRunTimeClass: JClass;
  lGCMethod: JMethodID;
begin
  lRunTimeInstance := Java_Lang_RunTime_GetRunTime;
  lRunTimeClass := JVM.JNIEnv.GetObjectClass(lRunTimeInstance);
  lGCMethod := JVM.JNIEnv.GetJniMethodID(lRunTimeClass, lGCMethodName, lGCMethodSignature);
  Regie.JVM.JNIEnv.CallVoidMethod(lRunTimeInstance, lGCMethod, []);
end;
```

Part 7

in the actual insurance application

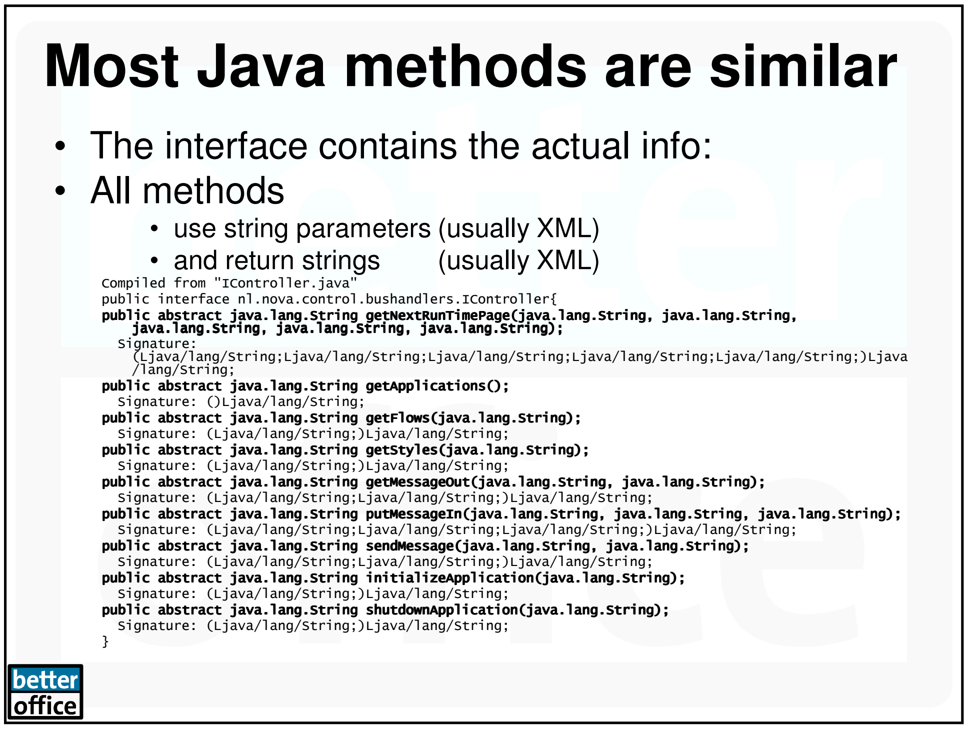
Most Java methods are similar

- You can find that using javap
- Dumping with javap requires your classpath to be complete:

```
C:\005 Source.d5\ZZ Oplevering\bin>"C:\Program
   Files\Java\jdk1.5.0_10\bin\javap.exe" -s -classpath
   ;../Broker/Config;../broker/lib;../broker/lib/Broker.jar
   ;../broker/lib/nnutils.jar;../Broker/Lib/BcsKernel.jar
   ;../product/config;../product/lib/Regie.jar
   ;../product/lib/BusinessLogic.jar
   ;../product/lib/TransferComponents.jar;../product/lib/KFUtil.jar
   ;../product/lib/KFLog4j.jar;../product/lib/XMLUtil.jar
   ;../product/lib/EvalExpr.jar;../product/lib/HtmlConverter.jar
   ;../product/lib/WAFBus.jar;../product/lib/log4j.jar
   ;../product/lib/sa;../product/lib/sa/castor-0.9.5.4.jar
   ;../product/lib/sa/msbase.jar
   ;../product/lib/sa/mssqlserver.jar;../product/lib/sa/msutil.jar
   ;../product/lib/sa/j2ee1_3_1.jar
   ;../product/lib/sa/tyrex-0.9.8.5.jar
   ;../product/lib/DocomStandalone.jar
   ;../product/lib/validation.jar;../product/lib/sa/jcommon-1.0.0.jar
   ;../product/lib/sa/jfreechart-1.0.0.jar;../product/lib/soap.jar
   ;../product/lib/normen.jar;../product/lib/riskMapping.jar
   ;../product/lib/IbisServiceDispatcher.jar
   ;../product/lib/externalXml.jar; nl.nova.control.bushandlers.Regie
```

Most Java methods are similar

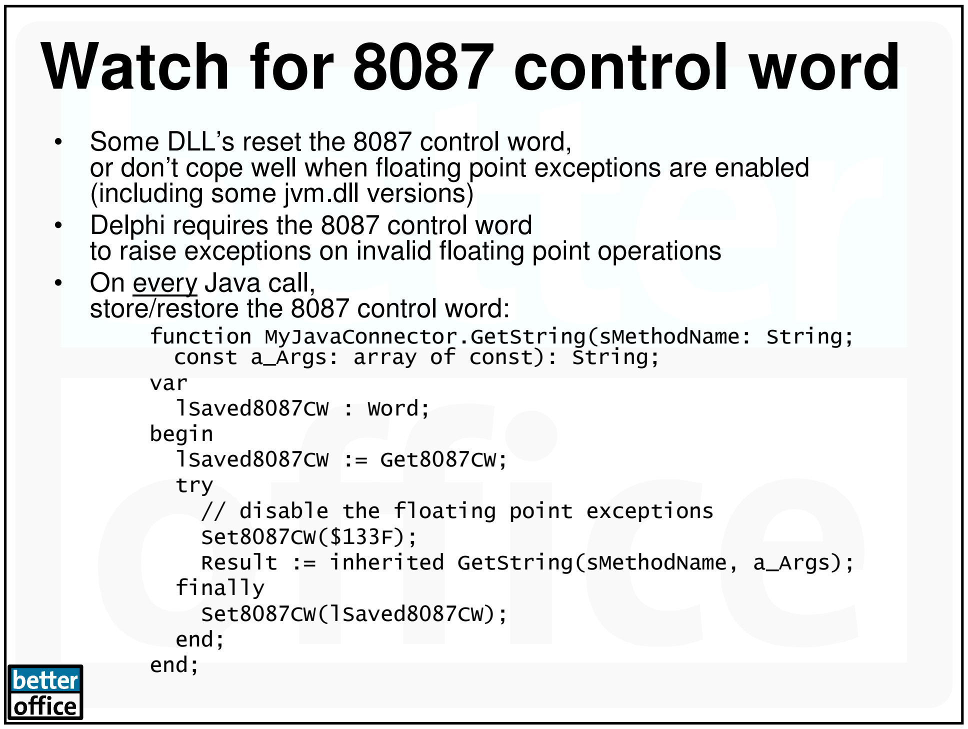
- javap prints only that exact class
- you need the ancestors too,
  this is the complete class list:
    - nl.nova.control.bushandlers.Regie
    - nl.nova.control.bushandlers.Controller
    - java.lang.Object
  - and this interface
    - nl.nova.control.bushandlers.IController



Most Java methods are similar

- The interface contains the actual info:
- All methods
  - use string parameters (usually XML)
  - and return strings   (usually XML)

```
Compiled from "IController.java"
public interface nl.nova.control.bushandlers.IController{
public abstract java.lang.String getNextRunTimePage(java.lang.String, java.lang.String,
    java.lang.String, java.lang.String, java.lang.String);
  Signature:
    (Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)Ljava
    /lang/String;
public abstract java.lang.String getApplications();
  Signature: ()Ljava/lang/String;
public abstract java.lang.String getFlows(java.lang.String);
  Signature: (Ljava/lang/String;)Ljava/lang/String;
public abstract java.lang.String getStyles(java.lang.String);
  Signature: (Ljava/lang/String;)Ljava/lang/String;
public abstract java.lang.String getMessageOut(java.lang.String, java.lang.String);
  Signature: (Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
public abstract java.lang.String putMessageIn(java.lang.String, java.lang.String, java.lang.String);
  Signature: (Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
public abstract java.lang.String sendMessage(java.lang.String, java.lang.String);
  Signature: (Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
public abstract java.lang.String initializeApplication(java.lang.String);
  Signature: (Ljava/lang/String;)Ljava/lang/String;
public abstract java.lang.String shutdownApplication(java.lang.String);
  Signature: (Ljava/lang/String;)Ljava/lang/String;
}
```



Watch for 8087 control word

- Some DLL's reset the 8087 control word, or don't cope well when floating point exceptions are enabled (including some jvm.dll versions)
- Delphi requires the 8087 control word to raise exceptions on invalid floating point operations
- On <u>every</u> Java call, store/restore the 8087 control word:

```
function MyJavaConnector.GetString(sMethodName: String;
  const a_Args: array of const): String;
var
  lSaved8087CW : Word;
begin
  lSaved8087CW := Get8087CW;
  try
    // disable the floating point exceptions
    Set8087CW($133F);
    Result := inherited GetString(sMethodName, a_Args);
  finally
    Set8087CW(lSaved8087CW);
  end;
end;
```

Demo time

- constructor TJNICommunicator.Create(aJvm: TJVM; aClassName: String);
- function TJNICommunicator.GetClass : JClass;
- function TJNICommunicator.GetJavaObject : JObject;
- function TJNICommunicator.GetString(sMethodName: String; const a_Args: array of const): String;

Part 8

Advanced: instrumentation (when time permits)

jconsole connects to JVM

- jconsole shows you the inside of a JVM
    - http://java.sun.com/javase/6/docs/technotes/guides/management/jconsole.html
  - It is part of the JDK
  - It needs the JVM to be started with an extra option:
    - **-Dcom.sun.management.jmxremote**
  - JDK 6 should be easier
- A regular JVM always needs this option:
  - **-Djava.class.path=**
- Your JVM loader needs to be option aware

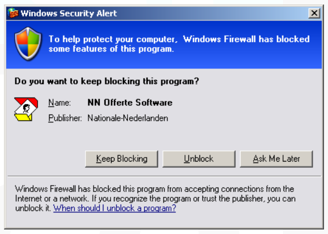
-->

Options during JVM start

```
procedure TJVMLoader.Start(aJvmDllPath: string; aClassPath: string; aJvmOptions: TStrings);
var
  Errcode: Integer;
  VM_args: JavaVMInitArgs;
  JavaClassPathOption: string;
  OptionIndex: Integer;
  Options: array of JavaVMOption;
begin
  FJavaVM := TJavaVM.Create(JNI_VERSION_1_4, aJvmDllPath);
  JavaClassPathOption := '-Djava.class.path=' + aClassPath;
  SetLength(Options, 1 + aJvmOptions.Count);
  Options[0].optionString := PChar(JavaClassPathOption);
  Options[0].extraInfo := nil;
  VM_args.nOptions := Length(Options);
  for OptionIndex := 0 to aJvmOptions.Count-1 do
  begin
    Options[OptionIndex+1].optionString := PChar(aJvmOptions[OptionIndex]);
    Options[OptionIndex+1].extraInfo := nil;
  end;
  VM_args.version := JNI_VERSION_1_4;
  VM_args.options := @Options[0];
  Errcode := FJavaVM.LoadVM(VM_args);
  if Errcode < 0 then
    // Loading the VM more than once will cause this error
    if Errcode = JNI_EEXIST then
      raise EJVMException.Create('Java VM has already been loaded. Only one VM can be loaded.')
    else
      raise EJVMException.CreateFmt('Error creating JavaVM, code = %d', [Errcode]);
  FJNIEnv := TJNIEnv.Create(FJavaVM.Env);
end;
```

Demo jconsole

- nnos.ini
  - INSTRUMENTATION=**1**
- Conditional define
  - **FastMM**
- Run KlantMap
  - Unblock the JVM instrumentation port
  - Start a financial product
- Run jconsole
  - Connect to JVM in KlantMap


Windows Security Alert

To help protect your computer, Windows Firewall has blocked some features of this program.

Do you want to keep blocking this program?

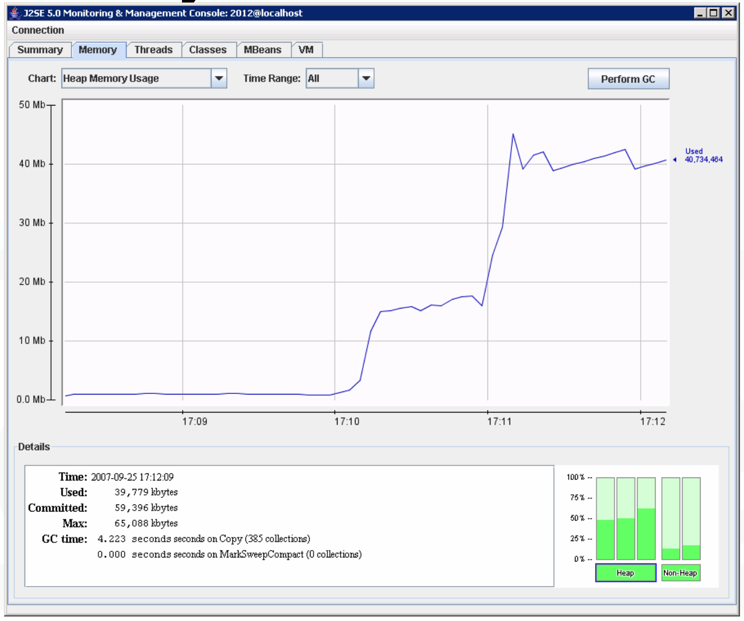
Name: NN Offerte Software
Publisher: Nationale-Nederlanden

Windows Firewall has blocked this program from accepting connections from the Internet or a network. If you recognize the program or trust the publisher, you can unblock it. When should I unblock a program?

Test jconsole

- Open two consoles:
  - C:\Program Files\Java\jdk1.5.0_10> java -Dcom.sun.management.jmxremote -jar demo\plugin\jfc\Java2D\Java2Demo.jar
  - C:\Program Files\Java\jdk1.5.0_10> bin\jconsole.exe
- Now jconsole should see the Java 2D demo
- If not, see the next picture

In case you had VM trouble:



```
     Time: 2007-09-25 17:12:09
     Used:      39,779 kbytes
Committed:      59,396 kbytes
      Max:      65,088 kbytes
  GC time:    4.223  seconds seconds on Copy (385 collections)
              0.000  seconds seconds on MarkSweepCompact (0 collections)
```

better office

better office

Part X

Q&A time...

That's all folks!

Please fill in the evaluations…

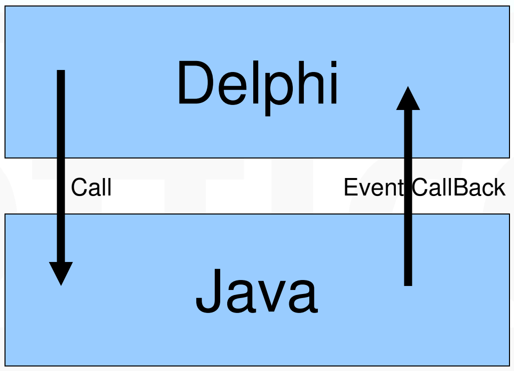
If you have questions after the session, please mail:
jpluimers@better-office.com

Talking Java from Delphi the real world

Jeroen Pluimers
better office benelux
jpluimers@better-office.com

Callbacks can be done

- It can be done, but
  - –This decouples control and
  - –Introduces synchronization issues


Delphi
Call
Event CallBack
Java