## Java Native Interface (JNI)

- Das JNI erlaubt den Aufruf extern implementierter Funktionen. Damit ist ein Zugriff auf alle Ressourcen einer Maschine möglich. Der Aufruf erfolgt über die Java Virtual Machine (JVM).
- Diese Erweiterungen sind selbstverständlich nicht mehr plattformabhängig.
- Wo bringen externe Module Vorteile?
  - Zugriff auf systemspezifische Funktionen möglich.
  - Bereits bestehende Programme in einer anderen Sprache können schrittweise in Java übertragen werden.
  - Firmenspezifisches Know-how kann in externen Modulen geschützt werden.
- Auch umgekehrt ist ein Zugriff aus einer anderen Programmiersprache auf ein Java-Modul möglich. Dies erfolgt über die Java Invocation-API. Java selbst nutzt diese nativen Funktionen. Vor allem Methoden, die systemspezifische Implementierungen benötigen, z.B.:

```
* @(#)FileInputStream.java 1.58 02/02/06
 * Copyright 2002 Sun Microsystems, Inc. All rights reserved.
 */
package java.io;
import java.nio.channels.FileChannel;
import sun.nio.ch.FileChannelImpl;
/**
     * Opens the specified file for reading.
     * @param name the name of the file
    private native void open(String name) throws FileNotFoundException;
     * Reads a byte of data from this input stream. This method blocks
     * if no input is yet available.
     * @return
                 the next byte of data, or <code>-1</code> if the end of the
     * file is reached.
* @exception IOException if an I/O error occurs.
    public native int read() throws IOException;
```

- Für den Aufruf einer extern definierten Funktion muss diese bei Windows in einer DLL vorliegen.
- Das Arbeiten mit JNI erscheint anfangs etwas kompliziert. Wenn man aber das Vorgehen kennt, ist es keine problematische Sache.
- Der Aufruf von Funktionen mit Parameter und Werterückgaben werden nächstes Mal behandelt.

## JNI mit Borland C++ Builder V5.0

Das Erstellen der DLL mit Borland C++ ist an sich problemlos, wenn einige Dinge beachtet werden:

1. Das Tool JavahScript.exe für die Headerfile-Erzeugung im JDK1.4\bin- Verzeichnis einbringen:

🚞 bin					ال_	
Datei Bearbeiten Ansicht Favori	en E	xtras ?				<b>R</b>
🔇 Zurück 🔹 🕥 - 🏂 🔎 S	iuchen	😥 Ordner 🛛 🔝 🗸				
Adresse 🛅 T:\Borland\JBuilder9\jdk1.4	\bin				💌 🔁 Wechsel	ln zu
		Name 🔺	Größe	Тур	Geändert am	
Datei- und Ordneraufgaben	*	appletviewer.exe	29 KB	Anwendung	16.5.2004 18:56	
- B Dataturkananan		extcheck.exe	29 KB	Anwendung	16.5.2004 18:56	
		HtmlConverter.exe	16 KB	Anwendung	16.5.2004 18:56	
🔯 Datei verschieben		🛅 idlj.exe	29 KB	Anwendung	16.5.2004 18:56	
📄 Datei kopieren		🛅 jar.exe	29 KB	Anwendung	16.5.2004 18:56	
🔕 Datei im Web veröffentlichen		🛅 jarsigner.exe	29 KB	Anwendung	16.5.2004 18:56	
Datei in E-Mail versenden		🛅 java.exe	25 KB	Anwendung	16.5.2004 18:56	
> Datei löschen		🛅 javac.exe	29 KB	Anwendung	16.5.2004 18:56	
		🛅 javadoc.exe	29 KB	Anwendung	16.5.2004 18:56	
		🛅 javah.exe	29 KB	Anwendung	16.5.2004 18:56	
Andere Orte	\$	JavahScript.exe	56 KB	Anwendung	23.5.2004 16:59	
		iavap.exe	29 KB	Anwendung	16.5.2004 18:56	
🛅 jdk1.4		🛅 javaw.exe	29 KB	Anwendung	16.5.2004 18:56	
📋 Eigene Dateien		🛅 jdb.exe	29 KB	Anwendung	16.5.2004 18:56	
Arbeitsplatz		📩 keytool.exe	29 KB	Anwendung	16.5.2004 18:56	
		🛅 kinit.exe	29 KB	Anwendung	16.5.2004 18:56	
		📩 klist.exe	29 KB	Anwendung	16.5.2004 18:56	
		📩 ktab.exe	29 KB	Anwendung	16.5.2004 18:56	
Details	¥	mative2ascii.exe	29 KB	Anwendung	16.5.2004 18:56	
		🗂 orbd.exe	29 KB	Anwendung	16.5.2004 18:56	
		policytool.exe	29 KB	Anwendung	16.5.2004 18:56	
		Troic eve	20 //R	Anwendung	16 5 2004 18:56	-

2. Nun wird das Tool im Tools-Menu installiert:



Der Parametereintrag muss genauso lauten: (Mit allen Leerschlägen)
"(\$OutputDir)\(\$FilePackage)" "(\$FileClass)" (\$JDKPath) "(\$FileDir)"

Jetzt wird der Eintrag als Tool im Menü sichtbar.

3. Das Java-Programm codieren und kompilieren. Das .class-File muss erzeugt werden.



In der Klasse muss die zu verwendende externe Funktion als Methoden-Prototyp definiert werden. In einem static-Block wird die DLL mit System.loadLibrary(..) geladen. Hier wird der DLL-Name ohne Extension angegeben. Das Kompilieren erfolgt durch rechten Mausklick auf den Reiter des Sheets:



Nun ist das .class-File erzeugt worden.

4. Nun wird das Headerfile für das C-Modul erzeugt. Dazu im Tools-Menü das Tool anklicken. Im Message-Fenster wird der Erfolg angezeigt.

🕖 JBuilder 9 - S:/EB01-1/Informatik 4	Uebergang/JavaBsp2/src/Ja	avaBeispiele2/JNIHello.java	<u>_</u> _×
Datei Bearbeiten Suchen Ansicht Proj	iekt Start Team Experten	Tools Fenster Hilfe	()
🎦 🗳 👻 🖺 📲 🎒 🗳 🖍	~ 🖻 🛱 🐰 🖊 📃	IDE-Optionen	🖳 🕨 + 🕅 + 🔯 +
🕼 - 4 + 🔗		Selitor-Optionen	
	🗙 🦄 INIHello	🛅 Bibliotheken konfigurieren	
		JDKs konfigurieren	
ButtonBsp.java 🛋	2	Server konfigurieren	
SileReaderToText.java	3 /**	Dateiverknüpfungen konfigurieren	
JavaFile.java	4 * Überschri	Palette konfigurieren	nal
JavaFilter1.java	5 * Beschreit	l ogether-Linstellungen	
JNI1 java	7 * Organisat	Enterprise-betup	
	8 * @author unbe	Enterprise-Deployment	
	9 * @version 1.0	EDB-Clerit-JAK erzeugen	
E morts	10 */	Beaninsignt	
hello()	12 public class JN	JDBC-Monitor	
main(String[] args)	13 public native	Datenbank-Pilot	
<b>⊡</b> > <initializer></initializer>	14 static	Bo JDataStore-Explorer	
	16 ( Jyscem. 107	Bg JDatastore-server	
	17	web Services Explorer	
	18 public static	ee I CP-Monitor	
	19 JNIHello jr 20 jnil bellou	web bervices-konsole	
	21 }	K Tools konfigurieren	
	22 }	Javah JNI Header File Generator	
		RMI-Registrierung	, I I I I I I I I I I I I I I I I I I I
	JNIHello.java	Einfüc	aen 6:13 🔹 CUA 🝷 🔍 👻
I	Quelle Design Bean Doku	Historie	
T:\Borland\JBuilder9\jdkl.4\bin	n\JavahScript.exe "S:\E	801-1\Informatik 4 Uebergang\Java	Bsp2\classes\JavaBeispiele2
Companying INT Norther Dia			
Generating JNI Header File	lder9\idkl.4\ire\lib\rt	.jar:T:\Borland\JBuilder9\jdkl.4\	ire\lib\il8n.iar:T:\Borland
[Loaded S:\EB01-1\Informatik 4	Uebergang\JavaBsp2\cla	sses\JavaBeispiele2\JNIHello.clas:	s]
[Loaded T:\Borland\JBuilder9\j	dkl.4\jre\lib\rt.jar(ja	va/lang/Object.class)]	
[Overwriting file S:\EB01-1\In:	formatik 4 Uebergang\Ja r godo 0	vaBsp2\src\JavaBeispiele2\JNIHell	o.h ]
Javan has cerminated with error	r code u		
Prozess beendet.			
🗙 🔆 Javah JNI Her File Generator			E
Javah JNI Header File Generator			

Die Arbeiten in der Java-Umgebung sind vorläufig abgeschlossen.

5. Borland C++ Builder aufstarten. Die DLL wird als Projekt mit DLL Experten erstellt:

🎯 C++Builder 5 - JNI1 [Dauer: 8.67 Sek]				
Datei Bearbeiten Suchen Ansicht Proje	t Sta <u>r</u> t <u>K</u> omponente	: <u>T</u> ools <u>H</u> ilfe ∭ <mark><kein></kein></mark>	- B	P
1 🗋 Neu				V
🕆 🗊 Neue Anwendung	Objektgalerie			<u>ے</u>
🛄 🗔 Neues Formular	Neu JNI1 F	ormulare Dialoge Projekte	Datenmodule	
Deuer Frame		•	<u></u>	
🗃 Öffnen			<b>N</b>	
🖆 Projekt öffnen Strg+F11	Anwendung	Batch-Datei	Bibliothek	
Neu öffnen 🕨		_		
	<b>1</b>	<b>C</b> <del>T</del>	<b>~</b>	
Strg+5				
CP Durielt and the surface	C-Datei	Upp-Datei	Datenmodul	
Projekt speichern unter				
Schließen			<u>_</u>	
Exp Schließen	DU Evente			
	DLL-Experte	Formular	Frame	
💾 Unit Header einschließen Alt+F11		-0-		<b>_</b>
🖨 Drucken	C Kopieren	C ⊻ererben C Ver <u>w</u> e	nden	
👖 👖 Beenden				
		ОК	Abbrechen	<u>H</u> ilfe

Vorgaben für DLL wählen:



6. Es wird eine Unit erzeugt mit DLLEntryPoint und weiteren Einträgen:



## Der gesamte Inhalt wird gelöscht!

7. Die Include-Directory- Liste muss für <JNI.h> u.a. erweitert werden. Dies erfolgt über die Projektoptionen:

😽 C++Builder 5 - Project1			
Datei Bearbeiten Suchen Ansicht Proj	ojekt Sta <u>r</u> t <u>K</u> omponente <u>T</u> ools <u>H</u> ilfe	(kein) 🔽 🖓 🖓	
Unit.cpp Unit.cpp Unit.cpp #include <jni.h> #include "JNIHel</jni.h>	Dem Projekt hinzufügen Umsch+F11 Aus dem Projekt entfernen Typbibliothek importieren Der Objektablage hinzufügen Quellext anzeigen Optionenquelle bearbeiten Make-Datei exportieren Neues Projekt hinzufügen Existierendes Projekt hinzufügen Unit compilieren Alt+F9 Project1 aktualisieren Strg+F9 Project1 erzeugen Info über [Vichts] Alle Projekte aktualisieren Alle Projekte erzeugen	Projektoptionen         Pascal       Linker         Anwendung       Compiler         Erweiterte Compiler-Optionen       C++         Verzeichnisse/Bedingungen       Versionsinfo         Ordner       Include-Pfadt         Bibliothekspfadt       \wendy\DevTools\Borland\CBuildefS\Project         Pfad für Debugger:       \$(BCB)\source\vcl         Zwischenausgabe:          BPI/LIB-Ausgabe:          Bedingungen       Definition:         DEBUG          Masse          Unit-Aliase          OK       Abbrechen         Bilite	
2: 24 Verändert Einfüger			
j 2. 24 jverandert johninger	21		

Der Include-Pfad muss mit den Einträgen zum Ziel ...jdk1.4\include und ....jdk1.4\include\win32 ergänzt werden. Der Ausgabepfad wird so definiert, dass die DLL in das aktuelle Projektverzeichnis geschrieben wird. (Das Projektverzeichnis ist das Verzeichnis, das vor dem JPX-File des Packages steht)). Damit wird die DLL ohne weitere Vorkehrungen beim Programmlauf gefunden.

Crdner XI	Projektoptionen X
	Pascal Linker Erweiterter Linker
Softierte Liste der Include-Prade:	Anwendung Compiler Erweiterte Compiler-Optionen C++
Wwendy/HTA-BE Dok/EB01-1 Vinformatik 4 Ueberov \$(BCB)\include \$(BCB)\include\vcl T:\Borland\JBuilder9\jdk1.4\include\ T:\Borland\JBuilder9\jdk1.4\include\win32	Verzeichnisse/Bedingungen Versionsinfo Packages Tasm Ordner Include-Pfad: \$(BCB)\include:\$(BCB)\include\vcl:\\wendy\ Bibliothekspfad: \\wendy\DevTools\Borland\CBuilder5\Project Pfad für Debugger: \$(BCB)\source\vcl Zwischenausgabe: Endgültige Ausgabe: \$:\EB01-1\\Informatik 4 Uebergang\JavaBsp2
\\wendy\HTA-BE Dok\EB01-1\Informatik 4 Ue	
Ersetzen Hinzufügen Löschen OK Abbrechen Hilfe	Bedingungen Definition: _DEBUG
	Vorgabe OK Abbrechen Hilfe

8. Das Projekt nun unter einem zweckmässigen Namen abspeichern. Der DLL-Name muss natürlich dem Parameter bei System.loadLibrary(...) entsprechen.

9. Die Projektverwaltung öffnen und das JNI-Headerfile zum Projekt zufügen. Hier: JNIHello.h. (Rechte Maustaste auf den DLL-Namen klicken – Hinzufügen..)



10. Das JNI-Headerfile öffnen und den Funktionsprototypen mit dem Paketnamen ergänzen. Hier JavaBeispiele2 zwischen zwei Unterstreichungszeichen. Diese Ergänzung ist nicht dokumentiert, aber zumindest bei der Benutzung von C++ Builder 5.0 notwendig.



11. Den Code in das .CPP-File eintippen. Der Funktionskopf wird zweckmässigerweise aus dem JNI-Headerfile kopiert:



12. Nun das Projekt kompilieren (Erzeugen) und kontrollieren, dass die DLL wirklich in das Verzeichnis mit dem JPX-File geschrieben worden ist.



13. Nun wieder zur JBuilder-IDE wechseln. Das Java File starten und prüfen, ob der Programmlauf korrekt erfolgt:

🕼 JBuilder 9 - 5:/EB01-1/Informatik 4	Uebergang/JavaBsp2/src/JavaBeispiele2/JNIHello.java	- 🗆 ×
Datei Bearbeiten Suchen Ansicht Proj	ekt Sta <u>r</u> t Tea <u>m</u> Experten <u>T</u> ools <u>F</u> enster <u>H</u> ilfe	
* 6 6 • 🖺 🛯 🖨 🖕	~ 🐂 🛱 X 🗛 🔽 🔽 🐨 🐨 🐨 🐨 🐨 🐨	
<b>₫ • ♦ ♦</b>		
📲 😢 🛋 🕅 🗗 JavaBeispi 🔻	🗙 😫 JNiHelio	
JavaFilter1 java     Java     Jili java     Simple/Java     Simple/Java     StringReaderBsp java     TextToBufferedWriter.is     JiliHello     hello()     main(String[] args)	<pre>1 package JavaBeispiele2; 2 3 /** 4 * Überschrift: Beispiel zu J-Builder 9 Personal 5 * Beschreibung: Einfachbeispiel zu JNI 6 * Copyright: Copyright (c) 2004 7 * Copyright: Copyright (c) 2004 7 * Organisation: Gerhard Krucker 8 * @author unbekamt 9 * @version 1.0 10 */ 11 12 public class JNIHello { 13 public native void hello(); 14 static 15 { System.loadLibrary("JNIHello"); } 16 17 18 public static void main(String[] args) { 19 JNIHello jnil=new JNIHello(); 20 jnil.hello(); 21 jnil.hello(); 22 jnil.hello();</pre>	
	22 }	
	NHello isva	
	Guelle Design Bean Doku Historie	
Hello world!	ı\javaw -ciasspatn ~S:\EBUI-I\informatik 4 Uebergang\JavaBsp2\classes;T:\Borl≀	ma\JBui
Prozess beendet.		
X JNH X JNHello		Ē