

Syntax: Übersetzungshilfe		
	ObjectPascal	Java
<b>Groß-/Kleinschreibung</b>	egal	kontextsensitiv
1-zeiliger Kommentar	//	//
n-zeiliger Kommentar	{...}	/*...*/
JavaDoc-Kommentar		/**...*/
JavaDoc-Elemente		@author @version @param @return

© T. Hempel

6

Syntax: Übersetzungshilfe			
Pascal	Java	Pascal	Java
begin	{	bedingungen	(bedingungen)
end;	}	integer	int
:=	=	real	double
=	==	string	String
<>	!=	case	switch
and	&&	repeat	do
or		until	while
not	!	do	
div	/	then	
mod	%		

© T. Hempel

7

Syntax: Grundgerüst	
ObjectPascal	Java
program name;	public class name{
var	
...	
begin	public static void main(String[] args){
...	...
end.	}
	}

© T. Hempel

8

Syntax: Datentypen			
	ObjectPascal	Java	Beispiel
Ganzzahl	integer	int	32
Gleitkommazahl	real	double	75.7
Wahrheitswert	boolean	boolean	true/false
einzelnes Zeichen	char	char	'\$'
Zeichenkette	string	String	"Hallo"
Merke	Primitive Typen → Wertetypen → Kleinschreibung Objekttypen → Referenzen → Klassen (Großschreibung)		

© T. Hempel

9

Syntax: Deklaration		
	ObjectPascal	Java
Wo?	im Deklarationsteil nach var	überall möglich
Wie?	variable: typ; const konstante = wert;	typ variable; typ variable = startwert; final typ konstante = wert;
z. B.	nummer: integer; const PI = 3.14; const taste = 'A';	int nummer; String ort = "HRO"; final double PI = 3.14; final char taste = 'A';

© T. Hempel

10

Syntax: Arithm. Operatoren		
	ObjectPascal	Java
Zuweisung	:=	=
Grundrechenarten	+ - * / div mod	+ - * / / %
Inkrement/ Dekrement	inc(zahl); dec(zahl);	zahl++; zahl--;
Erhöhen/ Vermindern	inc(zahl, wert); dec(zahl, wert);	zahl += wert; zahl -= wert;

© T. Hempel

11

Syntax: Boolesche Operatoren		
	ObjectPascal	Java
identisch	=	==
ungleich	<>	!=
AND/OR/XOR	and or xor	&&    ^
weitere Vergleiche	< > <= >=	< > <= >=
NICHT	not	!
Vergleich von Zeichenketten	a = 'Hallo'; if a = 'Hallo' then ...	a = "Hallo"; if (a.equals("Hallo")) ...

© T. Hempel

12

Syntax: Sequenz		
	ObjectPascal	Java
begin		{
anweisung 1;		anweisung 1;
anweisung 2;		anweisung 2;
...		...
anweisung n;		anweisung n;
end		}

© T. Hempel

13

## Struktur: Alternative

ObjectPascal	Java
if bedingung then	if (bedingung)
begin	{
anweisungA;	anweisungA;
end	}
else	else
begin	{
anweisungB;	anweisungB;
end;	}
Einseitige Auswahl analog!	

© T. Hempel 14

## Struktur: Alternative

ObjectPascal	Java
case selektor of	switch (selektor)
	{
wert1: anweisung1;	case wert1: anweisung1; break;
wert2: anweisung2;	case wert2: anweisung2; break;
...	...
else anweisung;	default: anweisung;
end;	}
Es wird genau ein Fall abgearbeitet.	Beginn bei Übereinstimmung bis zum break!
	selektor kann auch vom Typ String sein (ab Java 7)!

© T. Hempel 15

## Struktur: Schleife

ObjectPascal	Java
while bedingung do	while (bedingung)
begin	{
anweisungen;	anweisungen;
end;	}
while weiter = 'j' do	while (weiter == 'j')
begin	{
...	...
end;	}

© T. Hempel 16

## Struktur: Schleife

ObjectPascal	Java
repeat	do
anweisung;	anweisung;
until abbruchbedingung;	while (bedingung);
repeat	do
...	...
until weiter <> 'j';	while (weiter == 'j');
fußgesteuerte Schleife:	fußgesteuerte Schleife:
Bedingung: wann raus aus Schleife	Bedingung: wie lange in Schleife
derzeit nicht mehr im Abitur	

© T. Hempel 17

Struktur: Schleife	
ObjectPascal	Java
for i:=start to ziel step s do	for (init; bedingung; schritt)
begin	{
anweisungen;	anweisungen;
end;	}
for i := 100 to 200 do	for (int i = 100; i <= 200; i++)
begin	{
summe := summe + i;	summe = summe + i;
end;	}

© T. Hempel

18

Struktur: Schleife	
ObjectPascal	Java
for i := 2 to 100 step 2 do	for (int i = 2; i <= 100; i+=2)
begin	{
summe := summe + i;	summe = summe + i;
end;	}
for i := 100 downto 1 do	for (int i = 100; i >= 1; i--)
begin	{
summe := summe + i;	summe = summe + i;
end;	}

© T. Hempel

19

Syntax: Typumwandlungen	
ObjectPascal	Java
<b>Ganzzahl → Gleitkommazahl</b>	
automatisch	automatisch
<b>Gleitkommazahl → Ganzzahl</b>	
ganzzahl := trunc(zahl)	ganzzahl = (int) zahl;
<b>ASCII-Wert eines Charakters</b>	
ascii := ord('A');	ascii = (int) 'A';
<b>Cast-Operator</b>	

© T. Hempel

20

Syntax: Typumwandlungen	
ObjectPascal	Java
<b>Zahlen in Zeichenkette</b>	
IntToStr(zahl)	String.valueOf(zahl)
FloatToStr(zahl)	
kette := IntToStr(zahl);	kette = String.valueOf(zahl);
	kette = "" + zahl;
<b>Zeichenkette als Zahl</b>	
StrToInt(text)	Integer.parseInt(text)
StrToFloat(text)	Double.parseDouble(text)
zahl := StrToInt(kette);	zahl = Integer.parseInt(kette);

© T. Hempel

21

## Syntax: Aus- und Eingaben

ObjectPascal	Java
<b>Ausgaben</b>	
<code>write(text)</code>	<code>System.out.print(text)</code>
<code>writeln(text)</code>	<code>System.out.println(text)</code>
<code>ShowMessage(text);</code>	<code>JOptionPane.showMessageDialog(null, text)</code>
<code>uses Dialogs; notwendig</code>	<code>import javax.swing.*; notwendig</code>
<b>Eingaben</b>	
<code>readln(variable)</code>	<code>JOptionPane.showInputDialog(null, text)</code>
	<code>import javax.swing.*; notwendig</code>
<b>Alternative IO-Klasse nutzen!</b>	

© T. Hempel 22

## Syntax: Aus- und Eingaben

ObjectPascal	Java
<b>Alternative: IO-Klasse</b>	
<b>Ausgaben</b>	
<code>writeln()</code>	<code>IO.show(text)</code>
<b>Eingaben</b>	
<code>readln()</code>	<code>variable = IO.getString(text)</code>
	<code>variable = IO.getInt(text)</code>
	<code>variable = IO.getDouble(text)</code>
	<code>variable = IO.getChar(text)</code>
	<code>variable = IO.getBoolean(text)</code>

© T. Hempel 23

## Syntax: eindim. Felder

ObjectPascal	Java
<b>Deklaration</b> <code>feld: array[a..e] of typ;</code>	<code>typ feld[] = new typ[n];</code> <code>typ[] feld = new typ[n];</code>
<code>a ... Startindex</code> <code>e ... Endindex</code>	<code>n ... Anzahl der Elemente</code>
<b>Zugriff</b> <code>feld[index]</code>	<code>feld[index]</code> <b>Feldindex beginnt stets bei 0</b>
<b>Beispiel</b> <code>w: array[1..7] of string;</code>	<code>int[] w = new int[7];</code>
<code>w[1] := 'Montag';</code>	<code>w[0] = 100;</code>

© T. Hempel 24

## Erzeugung ausführbarer Dateien

ObjectPascal	Java
<b>Kompilieren</b> → eine ausführbare Datei (*.exe)	<b>Kompilieren</b> → Byte-Code-Dateien (*.class) <b>ohne</b> Einbindung von Bibliotheken
	<b>Erzeugen/Packen einer JAR-Datei</b> → Archiv mit Byte-Code-Dateien und allen notwendigen Bibliotheken
	JAR-Dateien lassen sich bei vorhandenem Java Runtime System (JRE) ausführen

© T. Hempel 25