# Fragen und Antworten: Übungen



# Übungen

F: **Gemeinsamkeiten und Unterschiede von Java und C# ? Prognose ?**
A: Gemeinsamkeiten: Abstammung von C++, Virtual Machine, Zwischencode, JustInTime-Compilierung, Sandbox, Garbage Collection. Beide Sprachen besitzen Entwicklungs- und RunTime-Umgebungen auf Windows, Mac OS X, Red Hat, Suse, Fedora (für die 3 letzteren: MonoDevelop und Mono). Die Java2EE- und die .NET-Technologien sind etwa gleich mächtig und funktionell gleichwertig.
Unterschiede:
1) C# ist anders als Java radikal objektorientiert: auch primitive Datentypen sind Objekte.
2) C# erlaubt schnelle Zeigerprogrammierung mit den Keywords `fixed` und `unsafe`.
3) C# erlaubt schnelle Zugriffe auf Graphik- und Soundbibliotheken und deren Hardwaretreiber.
4) Java ist reifer und altmodischer, C# ist komfortabler und schneller.
Prognose: Wahrscheinlich werden Java und C# auf Jahre hinaus koexistieren, Java auf Back End und Server, C# auf Front End und Multimedia.

F: **Vorteile und Nachteile von C++ gegenüber C# ?**
A: Vorteil von C++: unschlagbar schnelle Programme durch Zeigerprogrammierung mit und ohne Typprüfung, daher unersetzlich für Hardwarenähe und für Betriebssystemprogrammierung
Nachteile von C++:
1) schlechter und/oder bösartiger Code kann Betriebssystem+Harddisk zerstören.
2) Die Installation der EXEs, DLLs ist gefährlich, kompliziert und führt zu Registry- und Versionskonflikten: Schlagwort: DLL-Hölle.
3) Der einmal aus C++-Code compilierte Maschinencode muss auf allen (auch alten) Prozessoren lauffähig sein und deshalb dürfen die C++-Compiler die modernen Befehle neuer Prozessoren nicht in den Maschinencode einbauen.
Vorteile von C#:
1) Windows- und Internet-Programmierung ist unter C# einfacher und eleganter als unter C++ und MFC.
2) Der vom .NET-Compiler erzeugte Zwischencode ist ungefährlich und einfach zu installieren.
3) Die .NET-JustInTime-Compilierung erlaubt es, lokal den Maschinencode so zu erzeugen, dass er auf den lokalen Prozessor optimiert ist (und die Befehlssätze modernen Prozessoren nutzt). Das erklärt, warum C#-Code (auf modernen Prozessoren) manchmal schneller läuft als C++-Code.

F: **Wenn eine Animation ruckig und/oder langsam ist, muss man diese schneller machen. Welche Möglichkeiten gibt es ?**
A: 1) Wenn genügend Rechenleistung vorhanden: Timer Intervall verkürzen. (Hat aber keinen Sinn unter 1000/refresh = ca. 1000/75 = 13 msec.)
2) Wenn nicht genügend Rechenleistung vorhanden: 2a) Schrittweiten `dx,dy,dz,dzoomx,dzoomy,dzoomz,dpitch,dyaw,droll` vergrößern 2b) Anzahl der Ecken des Polygons bzw. der Polygone vermindern 2c) Anzahl der Lichtquellen vermindern 2d) neuen Treiber für die Graphikkarte suchen und installieren 2e) neue Graphikkarte mit neuem Treiber

F: **Bedeutung von Windows-`Form` Ereignissen mit den üblichen Reaktionen darauf**

| Ereignis | ausgelöst vom Betriebssystem durch | Reaktion des Programmierers |
|---|---|---|
| `OnCreate` | | |
| `OnPaint` | | |
| `OnResize` | | |
| `OnTimer` | | |
| `OnMouseDown` | | |
| `OnMouseMove` | | |
| `OnMouseUp` | | |

A:

| Ereignis | ausgelöst vom Betriebssystem durch | Reaktion des Programmierers |
|---|---|---|
| `OnCreate` | Neuöffnung eines Fensters | Graphikbefehle zur Einleitung, Begrüßung |
| `OnPaint` | `OnCreate`, `Invalidate()` | Graphikbefehle mit Dauerwirkung |
| `OnResize` | `OnCreate`, User ändert Fenstergröße | Graphikbefehle, die sich an die Fenstergröße anpassen; bei DirectX: `new Device ...` |
| `OnTimer` | `Timer`-Objekt | Graphikbefehle von Animationen |
| `OnMouseDown` | User drückt Maustaste | Malprogramm: erste Mauskoordinate speichern |
| `OnMouseMove` | Maus wird bewegt | Malprogramm: Strich von der alten zur neuen Koordinate ziehen |
| `OnMouseUp` | User läßt Maustaste los | Malprogramm: Länge, Fläche, umschr. Rechteck, Mitte berechnen |

F: **Ein minimales Hello World!-Programm in einem verschieblichen Fenster (entweder C# oder Java)**

```
//C# mit Windows.Forms ************************************************************
using System; using System........; using System..............;
public class Form1 : Form
{ static void Main() { ............................ ); }
  protected override void .......( .............. e )
  { e.Graphics...........("Hello World!",new Font("Arial",20),new SolidBrush(Color.Red),0,0); }
}

//alternativ Java mit awt+swing ***************************************************
import java.awt.*; import ............; import ...........; import ............;
class Hello1 ............... {
  public static void main(...........) { new Hello1(); }
  public Hello1() {
   ................;
   ................;
  }
  public void paint(Graphics g) {
   Graphics2D g2 = ............;
   ..............................;
   g2............("Hello World !", 10,  40);
  }
}
```

```
A: C# mit Windows.Forms ************************************************
using System; using System.Drawing; using System.Windows.Forms;
public class Form1 : Form
{ static void Main() { Application.Run( new Form1() ); }
  protected override void OnPaint( PaintEventArgs e )
  { e.Graphics.DrawString("Hello World!",new Font("Arial",20), new SolidBrush(Color.Red),0,0); }
}
A: Java mit awt+swing **************************************************
import java.awt.*; import java.awt.event.*; import java.util.*; import javax.swing.*;
class Hello1 extends JFrame {
  public static void main(String[] args) { new Hello1(); }
  public Hello1() {
   setSize(800,600);
   setVisible(true);
  }
  public void paint(Graphics g) {
   Graphics2D g2 = (Graphics2D)g;
   super.paint(g2);
   g2.drawString("Hello World !", 10,  40);
  }
}
```

F: **Minimalprogramm: Text "`Hello`" in ein File `C:\temp\Text.txt` zuerst schreiben dann lesen in C# oder Pseudocode**

A:

```
StreamWriter sw = new StreamWriter( "C:\\temp\\Text.txt" );
sw.WriteLine( "Hello" );
sw.Close();
StreamReader sr = new StreamReader( "C:\\temp\\Text.txt" );
String s = sr.ReadToEnd();
sr.Close();
```

F: **Erklärung der Datentypen**

| | |
|---|---|
| SByte, Int16, Int32, Int64 | |
| Byte, UInt16, UInt32, UInt64 | |
| Single, Double | |
| Char | |
| String | |
| Boolean | |
| Decimal | |
| Object | |

A:

| | |
|---|---|
| SByte, Int16, Int32, Int64 | 8-, 16-, 32-, 64-Bit-Wert incl. Vorzeichenbit |
| Byte, UInt16, UInt32, UInt64 | 8-, 16-, 32-, 64-Bit-Wert ohne Vorzeichen |
| Single, Double | IEEE-32- bzw. 64-Bit-float |
| Char | 16-Bit-Unicode-Zeichen |
| String | Array von Zeichen |
| Boolean | `True` oder `False` |
| Decimal | 128-Bit-Gleitkommawert ohne Rundungsfehler |
| Object | Basistyp für alle Typen |