# Informatik 2 für Regenerative Energien

**Klausur vom 6. Juli 2012**

Jörn Loviscach

Versionsstand: 5. Juli 2012, 23:34



*15 Punkte für die erste Aufgabe; 3 Punkte für alle weiteren Aufgaben. Mindestpunktzahl zum Bestehen: 15 Punkte. Hilfsmittel: maximal vier einseitig oder zwei beidseitig beschriftete DIN-A4-Spickzettel beliebigen Inhalts, möglichst selbst verfasst oder zusammengestellt; kein Skript, keine anderen Texte, kein Taschenrechner, kein Computer, kein Handy und Ähnliches.*

| Name | Vorname | Matrikelnummer | E-Mail-Adresse, falls nicht in ILIAS |
|---|---|---|---|
| | | | |

1. Im C#-Programmlisting im Anhang sind 15 Fehler, darunter keine Tippfehler und höchstens ein Fehler pro Zeile. Stellen Sie eine Liste dieser Art mit allen Fehlern auf:

| Zeile | korrekter Programmtext |
|---|---|
| 123 | `public void foo()` |
| 543 | `int a = 42;` |

2. Mit den (korrigierten) Klassen aus dem Anhang wird die Methode `Test.Testen()` ausgeführt. Welcher Wert steht in der Variablen `g` in Zeile 20? Welchen Wert hat danach die Variable `nächsteEindeutigeNummer` in Zeile 106? Was steht in der Variablen `z` in Zeile 22?

3. Schreiben Sie eine öffentliche Methode `ZähleStücke` für die (korrigierte) Klasse `Abspieler` im Anhang. Diese Methode soll als `int` liefern, wie viele *verschiedene* (!) Musikstücke abgespielt worden sind. Benutzen Sie dazu die Methode `IstGleich` der Klasse `Musikstück`.

4. Am Anfang der Methode `Starte` von `Abspieler` werden diese Zeilen ergänzt:

```
if (aktuellesMusikstück == null)
{
    throw new ApplicationException("kein Musikstück geladen");
}
```

   Geben Sie an, was man in der Methode `Test.Testen` ändern kann, damit diese Exception dann wirklich auftritt.

5. Schreiben Sie eine Klasse `Jingle`, die von `Musikstück` erbt, aber grundsätzlich eine Dauer von fünf Sekunden hat. Geben Sie dieser Klasse einen sinnvollen Konstruktor, dem man Titel, Interpret und Gebühr übergeben kann.

6. Zeichnen Sie ein UML-Klassendiagramm für die (korrigierten) Klassen `Zustand`, `HatKeinMusikstück` und `Gestoppt` aus dem Anhang.

7. Schreiben Sie Stück C#-Code, das eine Datei namens `"test.txt"` einliest und berechnet, welcher Anteil der Zeilen (von 0 = keine bis 1 = alle, angegeben als `double`-Zahl) Leerzeilen sind. Für eine leere Datei (also eine mit null Zeilen) soll als Anteil der Wert 0 herauskommen. Hinweis: `string[] System.IO.File.ReadAllLines(string path)`

8. Welche Zahlen stehen nach Ausführung dieses C#-Programmfragments in den Variablen `c`, `d` und `e`? Geben Sie möglichst auch Zwischenschritte an, damit Ihr Gedankengang nachvollziehbar ist.

```
Queue<int> a = new Queue<int>();
a.Enqueue(1);
a.Enqueue(2);
a.Enqueue(3);
Queue<int> b = a;
int c = a.Dequeue();
b.Enqueue(4);
int d = b.Dequeue();
int e = a.Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dies soll ein Musik-Abspielprogramm werden; es zählt auch gleich die Gebühren z. B. für öffentliches Abspielen. Die Methode `Testen` macht die Handhabung der Klassen vor.

```
1  class Test
2  {
3      public static void Testen()
4      {
5          Abspieler a = Abspieler();
6          TimeSpan ts = FromMinutes(3.5);
7          Musikstück m1 = new Musikstück("Easy", "DJ␣Solo", ts, 0.2);
8          Musikstück m2 = new Musikstück("Real", "J-Lo", ts, 0.1);
9          a.ÄndereMusikstück(m1);
10         a.Starte();
11         bool b = a.IstAmSpielen;
12         for (int i == 0; i < 2; i++)
13         {
14             a.ÄndereMusikstück(m2);
15             a.Starte();
16             a.ÄndereMusikstück(m1);
17             a.Starte();
18         }
19         a.Stoppe();
20         double g = a.Gesamtkosten;
21         a.ÄndereMusikstück(m2);
22         Zustand z = a.Spielerzustand;
23     }
24 }
25
26 class Abspieler
27 {
28     Zustand abspielerzustand = new HatKeinMusikstück();
29     Musikstück aktuellesMusikstück = 0;
30     List<Musikstück> gespielteStücke = new List<Musikstück>();
31     double gesamtkosten;
32     double Gesamtkosten { get { return gesamtkosten; } }
33
34     public void ÄndereMusikstück(Musikstück m)
35     {
36         abspielerzustand.ÄndereMusikstück(this);
37         aktuellesMusikstück = m;
38     }
39     public void Starte()
40     {
41         abspielerzustand.Starte(this);
42     }
43     public void Stoppe()
44     {
45         abspielerzustand.Stoppe(this);
46     }
```

```
47      public Zustand Spielerzustand
48      {
49          get { return abspielerzustand; }
50          set { abspielerzustand = value; }
51      }
52      public int AktuellesStückAlsGespieltMerken()
53      {
54          gespielteStücke.Add(aktuellesMusikstück);
55          gesamtkosten += aktuellesMusikstück.Gebühr;
56      }
57      public bool IstAmSpielen()
58      {
59          return abspielerzustand.IstAmSpielen();
60      }
61  }
62
63  abstract class Zustand
64  {
65      public virtual void ÄndereMusikstück(Abspieler abspieler)
66      {
67          abspieler.Spielerzustand = new Gestoppt();
68      }
69      public virtual void Starte(Abspieler abspieler)
70      {}
71      public virtual void Stoppe(Abspieler abspieler)
72      {}
73      public virtual bool IstAmSpielen()
74      {
75          return false;
76      }
77  }
78
79  class HatKeinMusikstück : Zustand
80  {}
81
82  class Gestoppt : Zustand
83  {
84      public override void Starte(Abspieler abspieler)
85      {
86          abspieler.Spielerzustand = new Spielt();
87          abspieler.AktuellesStückAlsGespieltMerken(0);
88      }
89  }
90
91  class Spielt : Zustand
92  {
93      public override void Stoppe(Abspieler abspieler)
94      {
95          abspieler.Spielerzustand = new Spielt();
96      }
97      public override bool IstAmSpielen(Abspieler abspieler)
```

```
98      {
99          return false;
100     }
101 }
102
103 class Musikstück : Zustand
104 {
105     int eindeutigeNummer;
106     static int nächsteEindeutigeNummer;
107
108     string titel;
109     public string Titel { get { return titel; } }
110     double gebühr;
111     public Gebühr { get { return gebühr; } }
112     string interpret;
113     TimeSpan dauer;
114
115     public void Musikstück(string titel, string interpret,
116                         TimeSpan dauer, double gebühr)
117     {
118         eindeutigeNummer = nächsteEindeutigeNummer;
119         nächsteEindeutigeNummer++;
120
121         this.titel = titel;
122         this.interpret = interpret;
123         this.dauer = dauer;
124         this.gebühr = gebühr;
125     }
126
127     public bool IstGleich(Musikstück m)
128     {
129         return eindeutigeNummer == m.eindeutigeNummer;
130     }
131 }
```