Informatik 2 für Regenerative Energien

Klausur vom 3. April 2017

Jörn Loviscach

Versionsstand: 3. April 2017, 00:31

15 Punkte für die erste Aufgabe; 3 Punkte für alle weiteren Aufgaben. Mindestpunktzahl zum Bestehen: 20 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 (auch nicht wearable), kein Handy und Ähnliches.

Name	Vorname	Matrikelnummer	E-Mail-Adresse

1. Im C#-Programmlisting im Anhang sind 15 Fehler, darunter keine Tippfehler und höchstens ein Fehler pro Zeile. Erstellen Sie eine Liste mit 15 Zeilen aus den Fehlern und ihren jeweiligen Korrekturen, nach dem folgenden Muster:

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

2. Die Methode `Test.Teste` des (korrigierten) Code aus dem Programmlisting im Anhang wird ausgeführt. Was steht am Ende in den Variablen `x`, `y` und `z`? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. In der Klasse `SignalAusDatei` ist es möglich, dass aus der Datei Abtastwerte größer als 1 oder kleiner als -1 ausgelesen werden. Verhindern Sie das mit einer Exception.

4. Die Klassen `KonstantesSignal` und `Sinusschwingung` des korrigierten Code aus dem Programmlisting erben von der Klasse `Signal` die Methode `BestimmeEffektivwert`. Die Schleife darin wird allerdings viel Zeit benötigen. Beschreiben Sie grob (ca. drei Sätze) eine Möglichkeit, das Programm diesbezüglich zu verbessern.

5. Die Klasse `Signal` soll eine virtuelle Methode namens `ListeVerwendeteDateien` erhalten. Diese Methode soll eine Liste der Namen verwendeter Dateien zurückgeben. (Gegebenfalls kommen Dateinamen mehrfach vor.) Eine Instanz von `Signal` verwendet keine Datei, also gibt man in dieser Klasse eine leere Liste zurück:

```
public virtual List<string> ListeVerwendeteDateien()
{
    return new List<string>();
}
```

   Beschreiben Sie grob (ca. drei Sätze), wie diese Methode in der Klasse `SignalAusDatei` und in der Klasse `SummeVonSignalen` zu überschreiben ist und welche anderen Änderungen gegebenenfalls in diesen Klassen nötig sind.

6. Schreiben Sie für den korrigierten Code aus dem Anhang eine sinnvoll abgeleitete Klasse `SignalRückwärts`. Diese Klasse soll ein im Konstruktor übergebenes Signal im Zeitverlauf umkehren.

7. Ein Dateisystem wird durch folgende drei Klassen modelliert: `Datei`, `Ordner` und das abstrakte `Dateisystemobjekt`. Alle Objekte sollen ihren Namen speichern. Ordner sollen wissen, was sie enthalten. Es soll eine Methode geben, um Ordner zu erzeugen. Zeichnen Sie zu diesen Angaben ein sinnvolles UML-Klassendiagramm.

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

```
List<Queue<int>> a = new List<Queue<int>>();
List<Queue<int>> b = new List<Queue<int>>();
Queue<int> c = new Queue<int>();
Queue<int> d = c;
c.Enqueue(1);
c.Enqueue(2);
a.Add(c);
a.Add(d);
b.Add(c);
d.Enqueue(3);
int u = a[0].Dequeue();
int v = a[0].Dequeue();
int w = b[0].Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dies ist der Anfang eines Programms zur Erzeugung und Verarbeitung von Signalen. Ein Signal ist dabei eine Zeitreihe von Messwerten mit einer bestimmten Abtastfrequenz (eine ganzzahlige Anzahl von Abtastungen pro Sekunde). Signale können rechnerisch erzeugt sein oder aus Dateien stammen. In der Datei `test.txt` stehen folgende drei Zeilen:

```
1,0
2,0
-3,0
```

Dies ist der Programmcode der Klassen:

```
1  class Test
2  {
3      public static void Teste()
4      {
5          Signal s1 = new KonstantesSignal(1000, 23.0);
6          Signal s2 = new VerstärktesSignal(s1, 2.0);
7          Signal s3 = new SignalAusDatei(1000, "test.txt");
8          Signal s4 = new SummeVonSignalen(s2, s3);
9          int x = s4.Länge();
10         double y = s4.GibAbtastwert(1);
11         string z = s4.Name;
12     }
13 }
14
15 class Signal
16 {
17     int abtastfrequenz;
18     public int Abtastfrequenz
19     {
20         get { return abtastfrequenz; }
21     }
22
23     protected int länge = int.MaxValue; // korrekt
24     public int Länge
25     {
26         get { return länge; }
27     }
28
29     string name = "Nullsignal";
30     public string Name
31     {
32         get { return name; }
33     }
34
35     public double GibAbtastwert(int nummer)
36     {
37         return 0.0;
```

```
38      }
39
40      public Signal(int abtastfrequenz)
41      {
42          this.abtastfrequenz = abtastfrequenz;
43      }
44
45      public BestimmeEffektivwert()
46      {
47          double summe;
48          for (int i = 0; i < Länge; i++)
49          {
50              double d = Signal.GibAbtastwert(i);
51              summe += d * d;
52          }
53          return Math.Sqrt(summe / Länge);
54      }
55  }
56
57  class KonstantesSignal
58  {
59      double konstante;
60
61      public KonstantesSignal(int abtastfrequenz, double konstante)
62          : base(abtastfrequenz)
63      {
64          this.konstante = konstante;
65          name = "Konstante␣" + konstante;
66      }
67
68      public override double GibAbtastwert(int nummer)
69      {
70          return konstante;
71      }
72  }
73
74  class Sinusschwingung : Signal
75  {
76      static double frequenz;
77
78      public Sinusschwingung(int abtastfrequenz, double frequenz)
79          : base(abtastfrequenz)
80      {
81          this.frequenz = frequenz;
82          name = "Sinusschwingung␣mit␣" + frequenz + "␣Hz";
83      }
84
85      public override double GibAbtastwert(int nummer)
86      {
87          return Sin(2.0 * Math.PI * frequenz * nummer / Abtastfrequenz);
88      }
```

```
89  }
90
91  class VerstärktesSignal : Signal
92  {
93      Signal originalsignal;
94      double verstärkungsfaktor;
95
96      public VerstärktesSignal(Signal s, double faktor)
97      : base (s.Abtastfrequenz)
98      {
99          länge = s.Länge;
100         originalsignal = s.Name;
101         verstärkungsfaktor = faktor;
102         name = s.Name + ", verstärkt um Faktor " + verstärkungsfaktor;
103     }
104
105     public override double GibAbtastwert(int nummer)
106     {
107         return verstärkungsfaktor * GibAbtastwert(nummer);
108     }
109 }
110
111 abstract class SignalMitDatenspeicher : Signal
112 {
113     protected double[] abtastwerte;
114
115     public SignalMitDatenspeicher(int abtastfrequenz)
116         : base(abtastfrequenz)
117     {
118     }
119
120     public override double GibAbtastwert(int nummer)
121     {
122         return abtastwerte(nummer);
123     }
124 }
125
126 class Rauschen : SignalMitDatenspeicher
127 {
128     static Random zufallsgenerator = new Random();
129
130     public Rauschen(int abtastfrequenz, int länge)
131         : base(abtastfrequenz)
132     {
133         this.länge = länge;
134         abtastwerte = new double[Länge];
135         for (int i = 0; i < Länge; i++)
136         {
137             abtastwerte[i] = zufallsgenerator.NextDouble(); // korrekt
138         }
139         name = "Rauschen";
```

```
140     }
141 }
142
143 abstract class SignalAusDatei : SignalMitDatenspeicher
144 {
145     public SignalAusDatei(int abtastfrequenz, string dateiname)
146         : base(abtastfrequenz)
147     {
148         string[] zeilen = File.ReadAllLines(dateiname);
149         länge = zeilen.Length;
150         abtastwerte = new double[Länge];
151         for (int i = 0; i < Länge; i++)
152         {
153             abtastwerte[i] = double.Parse(zeilen[i]);
154         }
155         name = "Signal␣aus␣Datei␣" + dateiname;
156     }
157 }
158
159 class SummeVonSignalen : Signal
160 {
161     Signal s1;
162     Signal s2;
163
164     public SummeVonSignalen(Signal s1, Signal s2)
165         : base(s1)
166     {
167         this.s1 = s1;
168         this.s2 = s2;
169         länge = Math.Min(s1.Länge, s2.Länge);
170         name = "Summe␣von␣" + s1.Name + "␣und␣" + s2.Name;
171     }
172
173     public double GibAbtastwert(int nummer)
174     {
175         return s1.GibAbtastwert(nummer) + s2.GibAbtastwert(nummer);
176     }
177 }
```