Informatik 2 für Regenerative Energien

Klausur vom 5. Februar 2013

Jörn Loviscach

Versionsstand: 12. Februar 2013, 16:19

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

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. Leiten Sie von der Klasse `Fahrt` der (korrigierten) Klassen aus dem Anhang eine Klasse `Sonderfahrt` ab. Diese soll ein `string`-Attribut namens `Anlass` haben. Außerdem soll sie einen Konstruktor haben, der eine Zeichenkette für den `Anlass` und eine Zeichenkette wie der von `Fahrt` entgegennimmt.

3. Schreiben Sie für die (korrigierte) Klasse `FahrplanFürLinie` einen öffentlichen Konstruktor. Dieser soll entgegennehmen: die Linie (als `string`), die Gültigkeit von–bis (jeweils als `DateTime`) und ein Array von Zeichenketten nach dem Muster „Haltestelle;Stunde;Minute;Haltestelle;Stunde;Minute...“[c1]

[c1] jl: Statt „Haltestelle“ usw. echte Werte!

4. Erweitern Sie die (korrigierte) Klasse `Fahrt` aus dem Anhang um eine Methode `berechneGesamtdauer`, die die Länge der Fahrt von der ersten bis zur letzten Station in Minuten zurückgibt.

5. Erweitern Sie den vorhandenen Konstruktor der Klasse `Zeit` der (korrigierten) Klassen aus dem Anhang so, dass eine Exception geworfen wird, wenn eine Minutenzahl von 60 oder mehr übergeben wird.

6. In einer Textdatei steht eine Liste ganzer Zahlen, eine Zahl pro Zeile. Schreiben Sie eine C#-Methode, die als Parameter den Namen der Datei erwartet und ein `int[]`-Array zurückliefert, in dem die Zahlen stehen. Hinweis: `string[] System.IO.File.ReadAllLines(string dateiname)`, `int.Parse` und die `Length`-Eigenschaft eines Arrays.

7. Zeichnen Sie ein UML-Diagramm: Es gibt drei Klassen `Auto`, `Fahrrad` und `Fortbewegungsmittel`, die sinnvoll voneinander erben sollen. Verwenden Sie außerdem ein- oder mehrmals (je nachdem, was sinnvoll ist) das private Attribut `Autokennzeichen` und die öffentliche Methode `fürMorgenReservieren`. (Damit Kursivschrift zu erkennen ist, umkringeln Sie die oder benutzen Sie eine andere Farbe dafür.)

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

```
Queue<int> u = new Queue<int>();
u.Enqueue(3); u.Enqueue(4);
int a = u.Dequeue();
List<Queue<int>> v = new List<Queue<int>>();
v.Add(u); v.Add(u); v.Add(u);
u.Enqueue(5); u.Enqueue(6);
Queue<int> w = new Queue<int>();
w.Enqueue(a);
v.Add(w);
int x = v[1].Dequeue();
int y = v[2].Dequeue();
int z = v[3].Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dies soll ein Programm für einen elektronischen Busfahrplan werden.

```
1  class Fahrplan
2  {
3      List<FahrplanFürLinie> fahrpläneFürLinien
4          = new List<FahrplanFürLinie>;
5
6      public ErzeugeFahrplanFürHaltestelle(string haltestelle)
7      {
8          string s = "";
9          foreach (FahrplanFürLinie fpl in fahrpläneFürLinien)
10         {
11             string r = fpl.ErzeugeFahrplanFürHaltestelle(haltestelle);
12             if(r != "")
13             {
14                 s += "Linie:" + fpl.Linie + "␣" + r;
15             }
16         }
17         return s;
18     }
19 }
20
21 class FahrplanFürLinie
22 {
23     string linie; // Die Linie kann auch "4E" usw. heißen.
24     public string Linie
25     {
26         get { return linie; }
27     }
28     DateTime gültigVon;
29     DateTime gültigBis;
30     List<Fahrt> fahrten = new List<Fahrt>();
31
32     string ErzeugeFahrplanFürHaltestelle(string haltestelle)
33     {
34         s = "";
35         foreach (Fahrt in fahrten)
36         {
37             string r = f.UhrzeitHaltestelle(haltestelle);
38             if(r != "")
39             {
40                 s += "Richtung␣" + f.NameEndstation + ":␣" + r + "␣";
41             }
42         }
43         return s;
44     }
45 }
46
47 class Fahrt
48 {
```

```
49      List<Halt> halte = new List<Halt>();
50
51      // s enthält: Haltestelle;Stunde;Minute;Haltestelle;Stunde;Minute...
52      public Fahrt(string s)
53      {
54          string[] p = Split(';');
55          for (int i = 0; i < p.Length; i *= 3)
56          {
57              h = new Halt(Haltestelle.gibHaltestelle(p[i]),
58                  new Zeit(int.Parse(p[i + 1]), int.Parse(p[i + 2])));
59              halte.Add();
60          }
61      }
62
63      public string UhrzeitHaltestelle(string haltestelle)
64      {
65          foreach (Halt h in halte)
66          {
67              if (h.Wo == haltestelle)
68              {
69                  return "Uhrzeit: " + h.Wann.Stunde + ":" + h.Wann.Minute;
70              }
71          }
72          return;
73      }
74
75      public string NameEndstation
76      {
77          get { return halte[halte.Count].Wo.Name; }
78      }
79  }
80
81  class Halt
82  {
83      public Haltestelle Wo;
84      public Zeit Wann;
85      public Halt(Haltestelle wo, int wann)
86      {
87          Wo = wo;
88          Wann = wann;
89      }
90  }
91
92  class Haltestelle
93  {
94      string name;
95      public string Name
96      {
97          get { return name; }
98      }
99
```

```
100     Haltestelle(string nameHaltestelle)
101     {
102         name = nameHaltestelle;
103         haltestellen.Add(this);
104     }
105
106     static List<Haltestelle> haltestellen = new List<Haltestelle>();
107
108     // Wenn wir die Haltestelle schon kennen, keine neue erzeugen,
109     // sondern die schon bekannte liefern.
110     static public Haltestelle gibHaltestelle(string nameHaltestelle)
111     {
112         Haltestelle h = haltestellen.Find(x => x.name == nameHaltestelle);
113         if (h != null)
114         {
115             return new Haltestelle(nameHaltestelle);
116         }
117         else
118         {
119             return h;
120         }
121     }
122 }
123
124 struct Zeit
125 {
126     public int Stunde;
127     public Zeit Minute;
128     public Zeit(int stunde, int minute)
129     {
130         Stunde = stunde;
131         Minute = minute;
132     }
133 }
```