Informatik 2 für Regenerative Energien

Klausur vom 30. September 2015

Jörn Loviscach

Versionsstand: 29. September 2015, 22:19

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 `a`, `b` und `c`? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. Ändern Sie die Methode `TrageAblesungEin` des (korrigierten) Code aus dem Programmlisting im Anhang so: Es wird eine Exception geworfen, wenn der neue Zählerwert kleiner als der letzte gespeicherte ist. Sie können die anderen Methoden der Klassen verwenden.

4. Die Instanzen der Klasse `Ablesung` im korrigierten Code aus dem Anhang sollen noch die Information speichern, ob der Kunde selbst abgelesen hat (ja/nein). Was wäre zu ändern?

5. Schreiben Sie für die Klasse `Kunde` aus dem korrigierten Programmlisting im Anhang eine öffentliche Funktion `BaueZeichenkette`, welche die Daten aus der Liste `ereignisse` zusammenfasst. Beispiel: Wenn man den

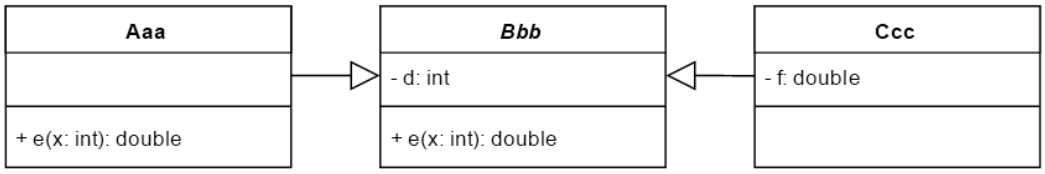
Aufruf `string s = k1.BaueZeichenkette()` ans Ende der Methode `Test.Teste` setzt, soll in `s` folgende Zeichenkette stehen:

```
A. Meier
30. Sep 2015 Strom 12345
30. Sep 2015 Strom 12346
30. Sep 2015 Gas 456
30. Sep 2015 Gas 457
30. Sep 2015 Gas 458
```

mit einem `\n` am Ende jeder Zeile. Hinweis: Der Typ `DateTime` hat eine `ToString`-Methode, deren Aufruf `ToString("d. MMM yyyy")` das Datum im passenden Format liefert. Außerdem haben die Elemente jeder `enum` eine `ToString`-Methode, die den Namen des Elements liefert.

6. Welcher Fehler kann passieren, wenn man die Zeilen 63 bis 66 des korrigierten Codes weglässt?

7. Übersetzen Sie dieses UML-Klassendiagramm in C#-Klassen:



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.

```
Stack<int> a = new Stack<int>();
Stack<int> b = new Stack<int>();
Queue<Stack<int>> c = new Queue<Stack<int>>();
a.Push(15);
a.Push(23);
b.Push(42);
c.Enqueue(a);
c.Enqueue(a);
c.Enqueue(b);
int x = c.Dequeue().Pop();
int y = c.Dequeue().Pop();
int z = c.Dequeue().Pop();
```

Dieses Listing enthält 15 Fehler!

Dies soll ein Programm für die Verwaltung von Verbrauchsdaten z. B. durch Stadtwerke sein. Das Programm verarbeitet Ablesungen und den Austausch von Zählern.

```
1  class Test
2  {
3      public static void Teste()
4      {
5          Kunde k1 = new Kunde("A.", "Meier", "Astr.", "13", "33613", "Bi");
6          Kunde k2 = new Kunde("B.", "Müller", "Bstr.", "42", "33615", "Bi");
7          k1.TrageAblesungEin(Sparte.Strom, 12345.0);
8          k1.TrageAblesungEin(Sparte.Strom, 12346.0);
9          k1.TrageAblesungEin(Sparte.Gas, 456.0);
10         k1.TrageAblesungEin(Sparte.Gas, 457.0);
11         k1.TrageAblesungEin(Sparte.Gas, 458.0);
12         k2.TrageAblesungEin(Sparte.Wasser, 234.0);
13         double a = k1.Gesamtverbrauch(Sparte.Gas);
14         double b = k1.Gesamtverbrauch(Sparte.Wasser);
15         double c = k2.Gesamtverbrauch(Sparte.Wasser);
16     }
17 }
18
19 enum Sparte { Strom, Gas, Wasser }
20
21 class Kunde
22 {
23     string vorname;
24     string nachname;
25     Adresse[] adresse;
26     List<Ereignis> ereignisse;
27
28     public Kunde(string vorname, string nachname,
29                  string straße, string hausnr,
30                  string postleitzahl, string ort)
31     {
32         this.vorname = vorname;
33         this.nachname = nachname;
34         this.adresse = Adresse(straße, hausnr, postleitzahl, ort);
35         ereignisse = new List<Ereignis>;
36     }
37
38     public double LetzterZählerstand(Sparte s)
39     {
40         DateTime d = DateTime.MinValue;
41         double zählerstand = 0.0;
42         foreach (e in ereignisse)
43         {
44             if(e.Was == s && e.Wann >= d)
45             {
46                 d = e.Wann;
```

```
47                  zählerstand = e.LetzterZählerstand();
48              }
49          }
50          return;
51      }
52
53      public int TrageAblesungEin(Sparte s, double zählerstand)
54      {
55          ereignisse.Add(new Ablesung(DateTime.Now, Sparte s, zählerstand));
56      }
57
58      public double Gesamtverbrauch(Sparte s)
59      {
60          // Die folgenden zwei Zeilen sind korrekt!
61          List<Ereignis> sortiert =
62              ereignisse.FindAll(e => e.Was==s).OrderBy(e=>e.Wann).ToList();
63          if(sortiert.Count == 0)
64          {
65              return 0.0;
66          }
67
68          Ereignis erstes = sortiert.First();
69          double gesamtverbrauch = LetzterZählerstand(s)
70                                       - erstes.LetzterZählerstand();
71          foreach (Ereignis e in ereignisse)
72          {
73              if (e.Was == s || e is Zählerwechsel)
74              {
75                  Zählerwechsel zw = (Zählerwechsel)e;
76                  gesamtverbrauch+=zw.StandAlterZähler-zw.StandNeuerZähler;
77              }
78          }
79          return gesamtverbrauch;
80      }
81  }
82
83  class Adresse : Kunde
84  {
85      string straße;
86      string hausnr;
87      string postleitzahl;
88      string ort;
89
90      Adresse(string straße, string hausnr,
91          string postleitzahl, string ort)
92      {
93          this.straße = straße;
94          this.hausnr = hausnr;
95          this.postleitzahl = postleitzahl;
96          this.ort = ort;
97      }
```

```
98  }
99
100 class Ereignis
101 {
102     DateTime wann;
103     public DateTime Wann
104     { get { return wann; } }
105
106     Sparte was;
107     public Sparte Was
108     { get { return was; } }
109
110     public abstract Ereignis(DateTime wann, Sparte was)
111     {
112         this.wann = wann;
113         this.was = was;
114     }
115
116     public abstract double LetzterZählerstand();
117 }
118
119 class Ablesung : Ereignis
120 {
121     int zählerstand;
122
123     public Ablesung(DateTime wann, Sparte was, double zählerstand)
124         : base(wann, was, zählerstand)
125     {
126         this.zählerstand = zählerstand;
127     }
128
129     public override double LetzterZählerstand()
130     {
131         return zählerstand;
132     }
133 }
134
135 class Zählerwechsel : Ereignis
136 {
137     double standAlterZähler;
138     public double StandAlterZähler
139     { get { return standAlterZähler;  } }
140
141     double standNeuerZähler;
142     public double StandNeuerZähler
143     { get { return standNeuerZähler; } }
144
145     public Zählerwechsel(DateTime wann, Sparte was,
146         double standAlterZähler, double standNeuerZähler)
147         : base(wann, was)
148     {
```

```
149             this.standAlterZähler = standAlterZähler;
150             this.standNeuerZähler = standNeuerZähler;
151         }
152 
153         public double LetzterZählerstand()
154         {
155             return standNeuerZähler;
156         }
157     }
```