Informatik 2 für Regenerative Energien

Klausur vom 12. Oktober 2017

Jörn Loviscach

Versionsstand: 11. Oktober 2017, 23:55


This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Germany License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/de/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

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. Es soll unmöglich sein, unter der Kategorie „Einkommen“ einen negativen Betrag zu buchen. Stellen Sie das mit einer Exception sicher. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

4. Leiten Sie von der Klasse `Einmalig` des korrigierten Code aus dem Programmlisting eine Klasse `Kinobesuch` ab und implementieren Sie diese sinnvoll.

5. Schreiben Sie für die Klasse `Haushaltsbuch` des korrigierten Code aus dem Programmlisting eine Methode `MeldeTeures`, die alle einmaligen Ausgaben von 1000 oder mehr Euro zu einer mehrzeiligen Zeichenkette mit folgendem Aufbau zusammenstellt:

   ```
   Titel Betrag
   Titel Betrag
   Titel Betrag
   ...
   ```

   Der Betrag soll dabei positiv ausgegeben werden.

6. Man möchte im korrigierten Code aus dem Programmlisting für jede Buchung angeben, ob das Geld bar gezahlt wurde oder aber per Bankkonto geflossen ist. Geben Sie an, welche Änderungen dazu nötig sind.

7. Ihr Programm besitzt drei Klassen, um Sensoren auszulesen: eine für Temperatur, eine für Luftfeuchtigkeit, eine für den CO_2-Gehalt der Luft. Jede der drei Klassen verfügt über eine Methode, die den aktuellen Messwert zurückgibt. Wie gliedern Sie diese Klassen sinnvoll in eine Objekthierarchie ein? Welche Methoden sind mindestens nötig? Beantworten Sie dies mit Hilfe eines UML-Klassendiagramms. Kennzeichen Sie dabei Kursivschrift durch eine andere Farbe o. ä.

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.

   ```
   List<Queue<int>> a = new List<Queue<int>>();
   a.Add(new Queue<int>());
   a[0].Enqueue(10);
   a[0].Enqueue(11);
   a.Add(a[0]);
   a[1].Enqueue(12);
   int x = a[0].Dequeue();
   List<Queue<int>> b = new List<Queue<int>>();
   b.Add(a[0]);
   b.Add(a[1]);
   int y = b[1].Dequeue();
   b = a;
   a.Add(new Queue<int>());
   int z = b.Count;
   ```

Dieses Listing enthält 15 Fehler!

Dies ist der Anfang eines Programms für ein elektronisches Haushaltsbuch, in dem man seine Einnahme- und Ausgabeposten vermerkt. Die Methode `Teste` der Klasse `Test` macht die Benutzung der Klassen vor. Man sieht, dass es am 1. Oktober eine Gehaltserhöhung gegeben hat.

```
1  class Test
2  {
3      public static void Teste()
4      {
5          Haushaltsbuch h = new Haushaltsbuch();
6          DateTime d1 = new DateTime(2017, 1, 1); // Jahr, Monat, Tag
7          Monatlich g = new Monatlich("Gehalt", Kategorie.Einkommen, d1, 2000.0);
8          h.FügePostenHinzu(g);
9          h.FügePostenHinzu(
10                         new Monatlich("Miete", Kategorie.Wohnung, d1, -1000.0));
11         DateTime d2 = new DateTime(2017, 9, 15);
12         Posten k1 = new Einmalig("Kino", Kategorie.Freizeit, d2, -10.0);
13         h.FügePostenHinzu(k1);
14         Posten k2 = new k1.KopiereMitAnderemDatum(new DateTime(2017, 9, 16));
15         h.FügePostenHinzu(k2);
16         DateTime d3 = new DateTime(2017, 10, 1);
17         g.SetzeEnde(d3 - TimeSpan.FromDays(1));
18         h.FügePostenHinzu(
19                     new Monatlich("Gehalt", Kategorie.Einkommen, d3, 2300.0));
20         g.EntfernePosten(k1);
21         string x = h.BeschreibePosten(3);
22         double y = h.BerechneKassenstand(new DateTime(2017, 10, 12));
23         double z = h.Summiere(Kategorie.Wohnung, new DateTime(2016, 1, 1));
24     }
25 }
26
27 class Haushaltsbuch
28 {
29     List<Posten> posten = new List<Posten>;
30
31     public void FügePostenHinzu(Posten p)
32     {
33         posten.Add(p);
34     }
35
36     public void EntfernePosten(Posten p)
37     {
38         posten.Remove(p);
39     }
40
41     public int AnzahlPosten { get { return Count; } }
42
43     public double Summiere(Kategorie kategorie, DateTime bisWann)
44     {
45         double resultat = 0.0;
```

```
46
47          foreach (Posten p in posten)
48          {
49              if (p == kategorie)
50              {
51                  resultat += p.Summiere(bisWann);
52              }
53          }
54
55          return resultat;
56      }
57
58      public double BerechneKassenstand(DateTime bisWann)
59      {
60          double resultat = 0.0;
61
62          foreach (Posten p in posten)
63          {
64              resultat += p.Summiere(bisWann);
65          }
66
67          return resultat;
68      }
69
70      public string BeschreibePosten(int i)
71      {
72          return posten[i].Beschreibe;
73      }
74  }
75
76  enum Kategorie { Einkommen, Wohnung, Lebensmittel, Freizeit }
77
78  class Posten
79  {
80      string titel;
81      public string Titel { get { return titel; } }
82
83      Kategorie kategorie;
84      // Die folgende Zeile ist korrekt.
85      public Kategorie Kategorie { get { return kategorie; } }
86
87      DateTime wann;
88      public DateTime Wann { get { return wann; } }
89
90      double betrag;
91      public double Betrag { get { return betrag; } }
92
93      public Posten(string titel, Kategorie kategorie,
94                    DateTime wann, double betrag)
95      {
96          this.titel = titel;
```

```
97          this.kategorie = kategorie;
98          this.wann = wann;
99          this.betrag = betrag;
100     }
101
102     public abstract double Summiere(DateTime bisWann);
103
104     public double Beschreibe()
105     {
106         return Titel + "␣" + Betrag;
107     }
108
109     public Posten KopiereMitAnderemDatum(DateTime wann);
110 }
111
112 class Einmalig : Posten
113 {
114     public Einmalig(string titel, Kategorie kategorie,
115                     DateTime wann, double betrag)
116         : base(titel, kategorie, wann, betrag)
117     {}
118
119     public double Summiere(DateTime bisWann)
120     {
121         if(Wann <= bisWann)
122         {
123             return Betrag;
124         }
125         else
126         {
127             return 0.0;
128         }
129     }
130
131     public override Posten KopiereMitAnderemDatum(DateTime wann)
132     {
133         return new Einmalig(Titel, Kategorie, wann);
134     }
135 }
136
137 class Monatlich : Posten
138 {
139     DateTime ende = MaxValue;
140     public DateTime Ende { get { return ende; } }
141
142     public Monatlich(string titel, Kategorie kategorie,
143                      DateTime wann, double betrag)
144         : base(titel, kategorie, wann, betrag)
145     {}
146
147     public void SetzeEnde(DateTime ende)
```

```
148     {
149         this.ende = ende;
150     }
151
152     public override double Summiere(DateTime bisWann)
153     {
154         if(bisWann < Wann)
155         {
156             return;
157         }
158
159         DateTime zählenBis = ende;
160         if(bisWann < ende)
161         {
162             zählenBis = bisWann;
163         }
164
165         int anzahlMonate = zählenBis.Month + 12 * zählenBis.Year
166                            - (Wann.Month + 12 * Wann.Year);
167
168         if(zählenBis.Day >= Wann.Day)
169         {
170             anzahlMonate + 1;
171         }
172
173         return Betrag * anzahlMonate;
174     }
175
176     public override Posten KopiereMitAnderemDatum(DateTime wann)
177     {
178         Monatlich m = new Monatlich(Titel, Kategorie, wann, Betrag);
179         SetzeEnde(Ende + (wann - Wann)); // Ende wie Anfang verschieben
180         return m;
181     }
182 }
```