Informatik 2 für Regenerative Energien

Klausur vom 26. Juli 2018

Jörn Loviscach

Versionsstand: 26. Juli 2018, 09:14

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

3. Die Methode `BucheZimmer` soll eine Exception werfen, wenn der Zeitpunkt `von` nicht vor dem Zeitpunkt `bis` liegt. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

4. Schreiben Sie für die Klasse `Reservierungssystem` aus dem korrigierten Code aus dem Programmlisting eine parameterlose öffentliche Methode `BerechneGesamteinnahmen`, deren Rückgabewert die Gesamtsumme der Einnahmen aus allen im System vorhandenen Buchungen ist.

5. Schreiben Sie für die Klasse `Reservierungssystem` des korrigierten Code aus dem Programmlisting eine Überladung der Methode `FindeZimmer`, bei der man zusätzlich angeben kann, ob man einen Kühlschrank wünscht. Welche Änderungen sind dazu gegebenenfalls an anderen Klassen nötig?

6. Leiten Sie von der Klasse `Zimmer` des korrigierten Code aus dem Programmlisting eine Klasse `ZimmerMitBalkon` ab, die eine Angabe für die Quadratmeterzahl des Balkons enthält.

7. Zeichnen Sie ein UML-Klassendiagramm für die folgenden drei Klassen:

```
class A : B
{
    int x;
}
class B
{
    float y;
    public virtual double f(int z)
    {
        return 42.0;
    }
}
class C : A
{
    public override double f(int z)
    {
        return 13.0;
    }
}
```

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>>();
Queue<int> q = new Queue<int>();
a.Add(q);
q.Enqueue(1);
a.Add(q);
q.Enqueue(2);
a.Add(q);
q.Enqueue(3);
a.Add(new Queue<int>());
a[0].Enqueue(4);
a[1].Enqueue(5);
int x = a.Count;
int y = a[0].Dequeue();
int z = a[1].Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dies ist der Anfang eines Programms für Zimmerreservierungen in einem Hotel. Die Methode `Teste` der Klasse `Test` macht die Benutzung der Klassen vor. Dies ist der Programmcode der Klassen:

```
1  class Test
2  {
3      public static void Teste()
4      {
5          List<Zimmer> zim = new List<Zimmer>();
6          zim.Add(new Zimmer(101, new Bett[] { new Doppelbett() },
7              false, 70.0));
8          zim.Add(new Zimmer(102,
9              new Bett[] { new Doppelbett(), new Einzelbett() },
10             true, 90.0));
11         zim.Add(new Zimmer(103, new Bett[] { new Einzelbett() },
12             false, 60.0));
13         Reservierungssystem res = Reservierungssystem(zim);
14         Kunde k1 = new Kunde();
15         Kunde k2 = new Kunde();
16         DateTime d1 = new DateTime(2018, 8, 1);
17         DateTime d2 = new DateTime(2018, 8, 7);
18         z = res.FindeZimmer(d1, d2, 3);
19         double p = z[0].BerechneGesamtpreis((int)((d2 - d1).TotalDays));
20         res.BucheZimmer(k1, z[0], d1, d2);
21         int c = res.FindeZimmer(d1, d2, 1).Count;
22         bool b = res.BucheZimmer(k2, zim[1], d1, d2);
23     }
24 }
25
26 class Reservierungssystem
27 {
28     List<Zimmer> zimmer;
29     List<Reservierung> reservierungen = new List<Reservierung>();
30
31     Reservierungssystem(List<Zimmer> zimmer)
32     {
33         this.zimmer = zimmer;
34     }
35
36     // Hinweis: Es soll OK sein, wenn der neue Gast am selben Tag anreist,
37     // an dem der alte Gast abreist.
38     bool IstZimmerFrei(Zimmer zimmer, DateTime von, DateTime bis)
39     {
40         frei = true;
41
42         foreach (Reservierung r in reservierungen)
43         {
44             if (r.Zimmer == zimmer || von < r.Bis && bis > r.Von)
45             {
46                 frei = false;
```

```
47                  break;
48              }
49          }
50
51          return frei;
52      }
53
54      // Hinweis: Es soll nur nach Zimmern mit
55      // der exakt richtigen Zahl an Schlafplätzen gesucht werden.
56      public List<Zimmer> FindeZimmer(DateTime von,
57                              DateTime bis, int zahlDerGäste)
58      {
59          List<Zimmer> verfügbareZimmer = List<Zimmer>();
60
61          foreach (z in zimmer)
62          {
63              if(z.ZahlDerGäste == zahlDerGäste
64                  && IstZimmerFrei(z, von, bis))
65              {
66                  Add(z);
67              }
68          }
69
70          return verfügbareZimmer;
71      }
72
73      public bool BucheZimmer(Kunde kunde, Zimmer zimmer,
74                                  DateTime von, DateTime bis)
75      {
76          if(!IstZimmerFrei(zimmer, von, bis))
77          {
78              return false;
79          }
80
81          reservierungen.Add(new Reservierung(kunde, zimmer, von, bis));
82          return;
83      }
84
85      public bool GibtEsReservierung(Kunde kunde, Zimmer zimmer,
86                                                  DateTime von, DateTime bis)
87      {
88          Reservierung r1 = new Reservierung(kunde, zimmer, von, bis);
89
90          foreach (Reservierung r in reservierungen)
91          {
92              if(r.IstGleich(r1))
93              {
94                  return true;
95              }
96          }
97
```

```
98          return false;
99      }
100 }
101
102 abstract class Bett
103 {
104     public int ZahlDerGäste();
105 }
106
107 class Einzelbett : Bett
108 {
109     public virtual int ZahlDerGäste()
110     {
111         return 1;
112     }
113 }
114
115 class Doppelbett: Bett
116 {
117     public override int ZahlDerGäste()
118     {
119         return 2;
120     }
121 }
122
123 class Zimmer : Bett
124 {
125     int nummer;
126     Bett[] betten;
127     bool mitKühlschrank;
128     double preisProNacht;
129
130     public Zimmer(int nummer, Bett[] betten, bool mitKühlschrank,
131                                                 double preisProNacht)
132     {
133         this.nummer = nummer;
134         this.betten = betten;
135         this.mitKühlschrank = mitKühlschrank;
136         this.preisProNacht = preisProNacht;
137     }
138
139     public int ZahlDerGäste
140     {
141         get { return betten.Sum(x => x.ZahlDerGäste()); }
142     }
143
144     public double BerechneGesamtpreis(int zahlDerÜbernachtungen)
145     {
146         preisProNacht * zahlDerÜbernachtungen;
147     }
148 }
```

```
149
150 class Reservierung
151 {
152     Kunde kunde;
153
154     Zimmer zimmer;
155     public Zimmer Zimmer
156     { get { return zimmer; } }
157
158     DateTime von;
159     public DateTime Von
160     { get { return von; } }
161
162     DateTime bis;
163     public DateTime Bis
164     { get { return bis; } }
165
166     public Reservierung(Kunde kunde, Zimmer zimmer,
167                                         DateTime von, DateTime bis)
168     {
169         this.kunde = kunde;
170         this.zimmer = zimmer;
171         this.von = von;
172         this.bis = bis;
173     }
174
175     public bool IstGleich(r)
176     {
177         return kunde == r.kunde && zimmer == r.zimmer
178                                     && von == r.von && bis == r.bis;
179     }
180 }
181
182 abstract class Kunde
183 {
184     string vorname; // bleibt noch ungeschrieben
185     string name; // bleibt noch ungeschrieben
186 }
```