Informatik 2 für Regenerative Energien

Klausur vom 4. Oktober 2018

Jörn Loviscach

Versionsstand: 3. Oktober 2018, 21:52

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. Auf welche Zeichenkette verweist die Variable `x` am Ende? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. Die Methode `Reserviere` der Klasse `Reservierungssystem` soll eine Exception werfen, wenn der Passagier bereits einen Sitzplatz in diesem Flug hat. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

4. Ändern Sie die Klasse `JetbusA42` des korrigierten Code aus dem Programmlisting so, dass es in den Reihen 1 bis 9 (Business Class) keine Sitzplätze mit dem Buchstaben B oder dem Buchstaben E gibt, also nur welche mit A, C, D oder F.

5. Die Sitzplätze mit dem Buchstaben A und die Sitzplätze mit dem Buchstaben F sind Fensterplätze. Schreiben Sie für die Klasse `Reservierungssystem`

aus dem korrigierten Code aus dem Programmlisting eine öffentliche Methode `Sitzplatz FindeFreienFensterplatz(Flug f)`. Diese soll einen freien Fensterplatz (egal welchen) auf diesem Flug zurückgeben oder aber, falls kein Fensterplatz mehr frei ist, `null` zurückgeben. Welche Änderungen sind dazu gegebenenfalls an anderen Klassen nötig?

6. Leiten Sie eine Klasse `PassagierMitKleinkind` von der Klasse `Passagier` ab. Diese Klasse `PassagierMitKleinkind` muss keine weiteren Attribute besitzen. Geben Sie dieser Klasse einen Konstruktor.

7. Man erfasst Verbrauchswerte für Strom (in kWh), Wasser (in m^3) und Gas (in m^3) zu jeweils verschiedenen Zeitpunkten. Jede Messung hat einen Zahlenwert, eine Einheit und einen Zeitpunkt. Zeichnen Sie mittels UML sinnvolle Klassen (samt Attributen und Methoden) auf, um die Verbrauchswerte zu speichern.

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<int> a = new List<int>();
a.Add(10);
a.Add(20);
List<int> b = a;
b.Add(30);
b[0] = 40;
List<int> c = new List<int>();
c.Add(50);
Stack<List<int>> d = new Stack<List<int>>();
d.Push(a);
d.Push(b);
d.Push(c);
int x = a[0];
int y = b[2];
int z = d.Pop()[0];
```

Dieses Listing enthält 15 Fehler!

Dieses Programm soll Platzreservierungen im Flugzeug verwalten.

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          Flugzeug a42 = new JetbusA42();
6          Flug f = new Flug(1234, new DateTime(2018, 10, 4), a42);
7          Passagier p1 = new Passagier("Anton");
8          Passagier p2 = new Passagier("Berta");
9          Passagier p3 = new Passagier("Carla");
10         Reservierungssystem rsys = new Reservierungssystem();
11         List<Sitzplatz> s = rsys.FindeAlleFreienPlätze(f);
12         rsys.Reserviere(p1, f, s[0]);
13         rsys.Reserviere(p2, f, s[1]);
14         rsys.EntferneReservierung(p1, f);
15         s = rsys.FindeAlleFreienPlätze(f);
16         Reserviere(p3, f, s[0]);
17         string x = rsys.GibReservierungen();
18     }
19 }
20
21 enum Buchstabe {A, B, C, D, E, F}
22
23 class Sitzplatz
24 {
25     int reihe;
26     Buchstabe buchstabe;
27
28     public Sitzplatz(int reihe, buchstabe)
29     {
30         this.reihe = reihe;
31         this.buchstabe = buchstabe;
32     }
33
34     public override string ToString()
35     {
36         return "Platz␣" + reihe + buchstabe;
37     }
38 }
39
40 abstract class Flugzeug
41 {
42     protected List<Sitzplatz> sitzplätze = new List<Sitzplatz>;
43
44     public int GibAnzahlSitzplätze()
45     {
```

```
46          return sitzplätze.Count;
47      }
48
49      public Sitzplatz GibSitzplatz(i)
50      {
51          return sitzplätze[i];
52      }
53  }
54
55  class JetbusA42 : Flugzeug
56  {
57      public JetbusA42()
58      {
59          for (int reihe = 1; reihe <= 30; reihe++) // einschließlich 30
60          {
61              if(reihe == 13) // Die Reihe 13 soll es nicht geben.
62              {
63                  break;
64              }
65
66              // Die Zeile 68 ist korrekt!
67              // Bedeutung: alle Buchstaben durchgehen
68              foreach (Buchstabe b in Enum.GetValues(typeof(Buchstabe)))
69              {
70                  sitzplätze.Add(Sitzplatz(reihe, b));
71              }
72          }
73      }
74  }
75
76  abstract class Flug
77  {
78      int flugnummer;
79      DateTime wann;
80
81      Flugzeug flugzeug;
82      public Flugzeug Flugzeug { get { return flugzeug; } }
83
84      public Flug(int flugnummer, DateTime wann, Flugzeug flugzeug)
85      {
86          this.flugnummer = flugnummer;
87          this.wann = wann;
88          this.flugzeug = flugzeug;
89      }
90
91      public override string ToString()
92      {
93          return "Flug␣" + flugnummer;
94      }
95  }
96
```

```
97  class Passagier
98  {
99      string name;
100
101     Passagier(string name)
102     {
103         this.name = name;
104     }
105
106     public override string ToString()
107     {
108         return name;
109     }
110 }
111
112 class Reservierung
113 {
114     Passagier passagier;
115     public Passagier Passagier { get { return passagier; } }
116
117     Flug flug;
118     public Flug Flug { get { return flug; } }
119
120     Sitzplatz sitzplatz;
121     public Sitzplatz Sitzplatz { get { return sitzplatz; } }
122
123     public Reservierung(Passagier p, Flug f, Sitzplatz s)
124     {
125         this.passagier = p;
126         this.flug = f;
127         this.sitzplatz = s;
128     }
129
130     public override string ToString()
131     {
132         return passagier + "␣" + flug + "␣" + sitzplatz;
133     }
134 }
135
136 class Reservierungssystem
137 {
138     List<Reservierung> reservierungen = new List<Reservierung>();
139
140     public bool IstPlatzFrei(Flug f, Sitzplatz s)
141     {
142         return reservierungen.Exists(r => r.Flug == f && r.Sitzplatz == s);
143     }
144
145     public List<Sitzplatz> FindeAlleFreienPlätze(Flug flug)
146     {
147         List<Sitzplatz> freiePlätze = new List<Sitzplatz>();
```

```
148             int n = flug.Flugzeug.GibAnzahlSitzplätze();
149             for (int i = 0; i < n; i++)
150             {
151                 Sitzplatz s = Flugzeug.GibSitzplatz(i);
152                 if(IstPlatzFrei(flug, s))
153                 {
154                     freiePlätze.Add(s);
155                 }
156             }
157             return freiePlätze;
158         }
159 
160         public void Reserviere(Passagier p, Flug f, Sitzplatz s)
161         {
162             reservierungen.Add(new Reservierung(p, f, s));
163         }
164 
165         // Gibt bei Erfolg true zurück.
166         public EntferneReservierung(Passagier p, Flug f)
167         {
168             Reservierung res = reservierungen.Find(r =>
169                                             r.Passagier == p || r.Flug == f);
170             if (res == null)
171             {
172                 return true;
173             }
174             reservierungen.Remove(res);
175             return true;
176         }
177 
178         public override string GibReservierungen()
179         {
180             string resultat = "";
181             foreach (r in reservierungen)
182             {
183                 if(resultat != "")
184                 {
185                     resultat += ",␣";
186                 }
187                 resultat += r;
188             }
189             return resultat;
190         }
191     }
```