Informatik 2 für Regenerative Energien

Klausur vom 13. April 2018

Jörn Loviscach

Versionsstand: 13. April 2018, 10:54

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 der Variablen `str`? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. Jeder Mitarbeiter soll einer Schicht höchstens einmal hinzugefügt werden können. Stellen Sie das mit einer Exception sicher. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

4. Schreiben Sie für die Klasse `Schicht` aus dem korrigierten Code aus dem Programmlisting eine parameterlose öffentliche Methode `PrüfePersonalstärke`, deren Rückgabewert angibt, ob exakt so viele Mitarbeiter in der Schicht sind, wie deren `personalstärke` angibt.

5. Schreiben Sie für die Klasse `Einsatzplan` des korrigierten Code aus dem Programmlisting eine parameterlose öffentliche Methode `PrüfeAufDoppelschichten`, die den booleschen Wert `false` zurückliefert, wenn kein Mitarbeiter an keinem Tag für *mehrere* Schichten an diesem Tag eingeteilt ist, und sonst `true` liefert. Welche Änderungen sind dazu gegebenenfalls an anderen Klassen nötig?

6. Leiten Sie von der Klasse `Mitarbeiter` des korrigierten Code aus dem Programmlisting eine Klasse `Dienstagsmitarbeiter` ab, für die die Methode `IstVerfügbarAn` für alle Dienstage `true` ergibt und sonst `false`. Welche Änderungen sind dazu obendrein an der Klasse `Mitarbeiter` nötig? Hinweis: `DateTime` hat die Property `DayOfWeek`, die man auf Gleichheit mit `DayOfWeek.Tuesday` prüfen kann.

7. Zeichnen Sie ein UML-Klassendiagramm für drei Klassen, die folgende Situation modellieren: Jeder Raum hat eine Raumnummer, die auch Buchstaben enthalten kann (zum Beispiel D442). Jeder Raum hat eine bestimmte Zahl Sitzplätze. In jedem Raum lässt sich von jedermann das Licht an- und ausschalten. Es gibt immer nur spezielle Arten an Räumen, zum Beispiel Seminarraum oder Büro. Im Seminarraum kann jedermann den Beamer an- und ausschalten. (Verwenden Sie im UML-Diagramm Farben, um Kursivschrift zu markieren.)

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<Stack<int>> q = new Queue<Stack<int>>();
Stack<int> s = new Stack<int>();
s.Push(1);
s.Push(2);
Stack<int> t = new Stack<int>();
t.Push(3);
t.Push(4);
t.Push(5);
q.Enqueue(s);
q.Enqueue(t);
q.Enqueue(t);
int x = q.Dequeue().Pop();
int y = q.Dequeue().Pop();
int z = q.Dequeue().Pop();
```

Dieses Listing enthält 15 Fehler!

Dies ist der Anfang eines Programms für die Einsatzplanung von Mitarbeitern in einem Betrieb mit Tag- und Nachtschicht. 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          Mitarbeiter m1 = new Mitarbeiter("Albers", 123);
6          Mitarbeiter m2 = new Mitarbeiter("Bell", 234);
7          Mitarbeiter m3 = new Mitarbeiter("Caesar", 345);
8          m1.FügeAbwesenheitHinzu(new Zeitspanne(
9              new DateTime(2018, 7, 2, 0, 0, 0),   // Jahr, Monat, Tag
10             new DateTime(2018, 7, 6, 0, 0, 0)
11         ));
12         Einsatzplan e = new Einsatzplan;
13         Schicht s1 = new Tagschicht(new DateTime(2018, 7, 4, 0, 0, 0));
14         s1.FügeMitarbeiterHinzu(m1);
15         s1.FügeMitarbeiterHinzu(m2);
16         s1.FügeMitarbeiterHinzu(m3);
17         e.FügeSchichtHinzu(s1);
18         e.FügeSchichtHinzu(Tagschicht(new DateTime(2018, 7, 4, 0, 0, 0)));
19         string str = PrüfeAufFehler();
20     }
21 }
22
23 class Einsatzplan
24 {
25     List<Schicht> schichten = new List<Schicht>();
26
27     public void FügeSchichtHinzu(Schicht s)
28     {
29         schichten.Add(s);
30     }
31
32     public string PrüfeAufFehler()
33     {
34         string ausgabe = "";
35
36         for(int i = 0; i < schichten.Count; i++)
37         {
38             Schicht s = schichten[i];
39
40             // Ist keine Schicht doppelt?
41             for (int j = 0; j < schichten.Count; j++)
42             {
43                 if (j != i && s.IstGleich(schichten[j]))
44                 {
45                     ausgabe += s + "␣ist␣doppelt.\n";
46                 }
```

```
47              }
48
49              foreach (m in s.BestimmeFehlendeMitarbeiter())
50              {
51                  ausgabe += "In␣Schicht␣" + s + "␣fehlt␣" + m + ".\n";
52              }
53          }
54
55          return ausgabe;
56      }
57  }
58
59  class Schicht
60  {
61      DateTime tag;
62      public DateTime Tag
63      {
64          get
65          {
66              return tag;
67          }
68      }
69
70      int personalstärke = 3;
71      List<Mitarbeiter> mitarbeiter = new List<Mitarbeiter>();
72
73      public Schicht(DateTime tag)
74      {
75          this.tag = tag;
76      }
77
78      public void FügeMitarbeiterHinzu(Mitarbeiter m)
79      {
80          mitarbeiter.Add(m);
81      }
82
83      public bool IstGleich(Schicht s);
84
85      public List<Mitarbeiter> BestimmeFehlendeMitarbeiter()
86      {
87          return mitarbeiter.FindAll(m => m.IstVerfügbarAn(Tag));
88      }
89
90      public string ToString()
91      {
92          return "Schicht";
93      }
94  }
95
96  class Tagschicht : Schicht
97  {
```

```
98      public Tagschicht(DateTime tag)
99          : base(tag)
100     {
101     }
102
103     public override bool IstGleich(Schicht s)
104     {
105         return Tag == s.Tag && s is Tagschicht;
106     }
107
108     public override string ToString()
109     {
110         // Ausgabeformat "d" ist nach dem Muster 01.01.2018
111         return "Tagschicht am " + Tag.ToString("d");
112     }
113 }
114
115 class Nachtschicht : Schicht
116 {
117     public Nachtschicht(DateTime tag)
118         : base(tag)
119     {
120         personalstärke = 2;
121     }
122
123     public override bool IstGleich(Schicht s)
124     {
125         return Tag == s.Tag && s is Nachtschicht;
126     }
127
128     public override string ToString()
129     {
130         return "Nachtschicht am " + Tag.ToString("d");
131     }
132 }
133
134 class Mitarbeiter
135 {
136     string name;
137     string nummer;
138     List<Zeitspanne> abwesenheiten = new List<Zeitspanne>;
139
140     public Mitarbeiter(string name, int nummer)
141     {
142         this.name = name;
143         this.nummer = nummer;
144     }
145
146     public void FügeAbwesenheitHinzu(Zeitspanne a)
147     {
148         abwesenheiten.Add(a);
```

```
149     }
150
151     public bool IstVerfügbarAn(DateTime tag)
152     {
153         foreach (Zeitspanne a in abwesenheiten)
154         {
155             if(a.EnthältTag())
156             {
157                 return false;
158             }
159         }
160
161         return false;
162     }
163
164     public override string ToString()
165     {
166         return "Mitarbeiter " + name;
167     }
168 }
169
170 class Zeitspanne
171 {
172     DateTime vonTag;    // einschließlich dieses Tages
173     DateTime bisTag;    // einschließlich dieses Tages
174
175     Zeitspanne(DateTime vonTag, DateTime bisTag)
176     {
177         this.vonTag = vonTag;
178         this.bisTag = bisTag;
179     }
180
181     public bool EnthältTag(DateTime d)
182     {
183         return vonTag <= d || d <= bisTag;
184     }
185 }
```