Informatik 2 für Regenerative Energien

Klausur vom 4. Februar 2015

Jörn Loviscach

Versionsstand: 4. Februar 2015, 09:14

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, 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. Mit dem (korrigierten) Code aus dem Programmlisting im Anhang wird die Methode `Test.Teste()` ausgeführt. Was steht in den Variablen `b`, `s`, `z` in den Zeilen 26 bis 28? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. Wenn durch Aufruf von `FügeTeilnehmerHinzu` mehr Teilnehmer in einem Raum sein sollen, als Plätze vorhanden sind, soll eine eine Exception geworfen werden. Welche Änderungen nehmen Sie dazu im (korrigierten) Code aus dem Programmlisting im Anhang vor?

4. Schreiben Sie von der Klasse `Raum` eine Kindklasse `Büro`. Diese soll eine Telefonnummer vom Typ `string` enthalten und einen öffentlichen Konstruktor besitzen, dem man Raumnummer und Telefonnummer übergibt und der die Zahl an Plätzen fest auf 3 setzt.

5. Schreiben Sie in der Klasse `Raumplanung` eine öffentliche Funktion, die die Gesamtdauer (also die Summe) der Dauern aller Raumbelegungen eines Arrays von Terminen ermittelt. Gegebenenfalls sind weitere Änderungen anderswo im Programm nötig.

6. Die bisherige Klasse `Termin` für einzelne Termine soll eine Kindklasse erhalten. Jedes Objekt dieser Kindklasse beschreibt einen Termin, der dauerhaft jede Woche zur selben Zeit im selben Raum mit den selben Teilnehmern stattfindet. Beschreiben Sie in wenigen Sätzen, welche Methoden der Klasse `Termin` Sie in der Kindklasse warum überschreiben würden.

7. Ein Programm soll Ordner und Dateien auf der Festplatte verwalten und dazu entsprechende Klassen besitzen. Sowohl Ordner als auch Dateien haben Namen und besitzen Methoden zum Löschen und Umbenennen. Zeichnen Sie dafür ein sinnvolles UML-Diagramm mit Klassen, Methoden, Attributen und Vererbung. Damit Kursivschrift (falls nötig) zu erkennen ist, umkringeln Sie diese oder benutzen Sie eine andere Farbe dafür.

8. Welche Zahlen stehen nach Ausführung dieses C#-Programmfragments in den Variablen `u`, `v` und `w`? Geben Sie möglichst auch Zwischenschritte an, damit Ihr Gedankengang nachvollziehbar ist.

```
List<Stack<int>> a = new List<Stack<int>>();
a.Add(new Stack<int>());
a.Add(new Stack<int>());
Stack<int> b = a[1];
b.Push(7);
b.Push(42);
a.Add(b);
a[0].Push(13);
int u = a[0].Pop();
int v = a[1].Pop();
int w = a[2].Pop();
```

Dieses Listing enthält 15 Fehler!

Dies soll ein Programm zur Verwaltung von Raumbuchungen werden. Die Methode `Teste` macht die Handhabung der Klassen vor.

```
1 class Test
2 {
3     public static void Teste()
4     {
5         Person p1 = new Person("Müller", "Alice", "amueller@bla.bla");
6         Person p2 = new Person("Meier", "Bert", "bmeier@bla.bla");
7         Person p3 = new Person("Schmitz", "Carla", "cschmitz@bla.bla");
8         Raum r1 = new Raum("234b", 42);
9         r1.FügeAusstattungHinzu(Gerät.Tafel);
```

```
10          Raum r2 = new Raum("Aula", 500);
11          r2.FügeAusstattungHinzu(Gerät.Beamer);
12          r2.FügeAusstattungHinzu(Gerät.Beamer);
13          r2.FügeAusstattungHinzu(Gerät.Tageslichtschreiber);
14          // Zur Erinnerung:
15          // new DateTime(Jahr, Monat, Tag, Stunde, Minute, Sekunde)
16          // new TimeSpan(Stunden, Minuten, Sekunden)
17          Termin t1 = new Termin("Besprechung",
18              new DateTime(2015,2,4,11,0,0), new TimeSpan(0,90,0), r1, p1);
19          t1.FügeTeilnehmerHinzu(p1);
20          t1.FügeTeilnehmerHinzu(p2);
21          t1.FügeTeilnehmerHinzu(p3);
22          Termin t2 = new Termin("Kaffetrinken",
23              new DateTime(2015,2,4,11,30,0), new TimeSpan(0,90,0), r1, p2);
24          t2.FügeTeilnehmerHinzu(p2);
25          Termin[] t = new Termin[] {t1, t2};
26          bool b = Raumplanung.GibtEsEineTerminkollision(t);
27          string s = Raumplanung.DruckeBelegungInZeichenkette(t, r1);
28          int z = r2.WievieleGibtEs(Gerät.Beamer);
29      }
30  }
31
32  class Raumplanung
33  {
34      public static bool GibtEsEineTerminkollision(Termin termine)
35      {
36          for (int i = 0; i < termine.Length - 1; i++)
37          {
38              for (int j = i + 1; j < termine.Length; j++)
39              {
40                  if(termine[i].KollidiertMit(j))
41                  {
42                      return true;
43                  }
44              }
45          }
46          return false;
47      }
48
49      public static string DruckeBelegungInZeichenkette(Raum r,
50                                                        Termin[] termine)
51      {
52          string s = "";
53          foreach (Termin t in termine)
54          {
55              if(t.Raum == r)
56              {
57                  if(s != "")
58                  {
59                      s += "\n";
60                  }
```

```
61                  s += t.DruckeInZeichenkette;
62              }
63          }
64          return t;
65      }
66  }
67
68  class Termin
69  {
70      string bezeichnung; // z.B. "Besprechung Jahresbericht"
71      DateTime start;
72      TimeSpan dauer;
73      Raum raum;
74      public Raum Raum
75      { get { return raum; } }
76      Person bucher; // wer den Raum bucht
77      DateTime buchungszeitpunkt;
78      List<Person> teilnehmerliste = new List<Person>();
79
80      public Termin(string bezeichnung, DateTime start, TimeSpan dauer,
81                                                  Raum raum, Person bucher)
82      {
83          this.bezeichnung = bezeichnung;
84          this.start = start;
85          this.dauer = dauer;
86          this.raum = raum;
87          this.bucher = bucher;
88          this.buchungszeitpunkt = DateTime.Now;
89      }
90
91      public void FügeTeilnehmerHinzu(p)
92      {
93          if (teilnehmerliste.Contains(p))
94          {
95              teilnehmerliste.Add(p);
96          }
97      }
98
99      public bool KollidiertMit(Termin t)
100     {
101         // Das + und das <= von DateTime und TimeSpan funktionieren!
102         // Aber stimmt die Logik?
103         if(this.raum != t.raum
104            || t.start + t.dauer <= this.start
105            || this.start + this.dauer >= t.start)
106         {
107             return false;
108         }
109         return true;
110     }
111
```

```
112     public string DruckeInZeichenkette()
113     {
114         // DateTime.Date liefert z.B. "04.02.2015 11:00:00"
115         // TimeSpan liefert z.B. "01:30:00"
116         bezeichnung + "␣" + start + ",␣Dauer␣" + dauer;
117     }
118
119     public VerschiebeAuf(DateTime neuerAnfangszeitpunkt)
120     {
121         start = neuerAnfangszeitpunkt;
122     }
123 }
124
125 enum Gerät {Tafel, Beamer, Tageslichtschreiber}
126
127 class Raum
128 {
129     string raumnummer; // auch z.B. "123a"
130     int zahlDerPlätze;
131     List<Gerät> ausstattung = new List<Gerät>();
132
133     public void Raum(string raumnummer, int zahlDerPlätze)
134     {
135         this.raumnummer = raumnummer;
136         this.zahlDerPlätze = zahlDerPlätze;
137     }
138
139     public static void FügeAusstattungHinzu(Gerät g)
140     {
141         ausstattung.Add(g);
142     }
143
144     public bool WievieleGibtEs(Gerät g)
145     {
146         return ausstattung.Count(x => x == g);
147     }
148 }
149
150 abstract class Person
151 {
152     string name;
153     string vorname;
154     string eMailAdresse;
155
156     Person(string name, string vorname, string eMailAdresse)
157     {
158         this.name = name;
159         this.vorname = vorname;
160         this.eMailAdresse = eMailAdresse;
161     }
162 }
```