Informatik 2 für Regenerative Energien

Klausur vom 13. Juli 2016

Jörn Loviscach

Versionsstand: 13. Juli 2016, 00:09

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

3. Zeichnen Sie für die drei Klassen `Tätigkeit`, `Projekttätigkeit` und `Pause` aus dem Code im Anhang ein UML-Klassendiagramm. Zeichnen Sie nur Felder (Attribute) und Methoden ein, keine Konstruktoren und Properties. Damit gegebenenfalls Kursivschrift zu erkennen ist, umkringeln Sie die oder benutzen Sie eine andere Farbe dafür.

4. In der Klasse `Projekt` des korrigierten Code aus dem Programmlisting ist es möglich, dass `kunde` die Nullreferenz ist. Verhindern Sie das.

5. Jede Instanz der Klasse `Tätigkeit` im korrigierten Code aus dem Anhang soll eine Methode `GibProjekt` besitzen. Instanzen der Klasse `Projekttätigkeit` sollen hier das Projekt zurückliefern; Instanzen der Klasse `Pause` sollen `null` zurückliefern. Lösen Sie dies mit Polymorphie.

6. Schreiben Sie für die Klasse `Zeiterfassung` im korrigierten Code aus dem Anhang diese öffentliche Methode:

   ```
   TimeSpan ZähleProjektstundenVonMitarbeiter(Mitarbeiter m)
   ```

   Sie soll Dauern der gesamten Projekttätigkeiten des Mitarbeiters aufsummieren und zurückliefern. Gegebenenfalls sind auch anderswo Änderungen am Code nötig.

7. Leiten Sie von der Klasse `Tätigkeit` eine Klasse `Urlaub` ab und schreiben Sie dafür folgenden öffentlichen Konstruktor:

   ```
   Urlaub(DateTime von, int anzahlTage)
   ```

   Hinweis: `TimeSpan` hat eine statische Methode `FromDays`.

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<int> q = new Queue<int>();
   Queue<int> r = new Queue<int>();
   List<Queue<int>> s = new List<Queue<int>>();
   s.Add(q);
   s.Add(q);
   s.Add(r);
   q.Enqueue(1);
   q.Enqueue(2);
   q.Enqueue(3);
   r.Enqueue(q.Dequeue());
   int x = s[0].Dequeue();
   int y = s[1].Dequeue();
   int z = s[2].Dequeue();
   ```

Dieses Listing enthält 15 Fehler!

Dies soll ein Programm für eine Zeiterfassung sein, die verschiedene Projekte und Kunden unterscheiden kann.

```
1  class Test
2  {
3      public static void Teste()
4      {
5          Zeiterfassung z = new Zeiterfassung;
6          Mitarbeiter m1 = new Mitarbeiter("Schmidt", 100.0, 200.0);
7          Mitarbeiter m2 = new Mitarbeiter("Müller", 150.0, 300.0);
8          Kunde k1 = new Kunde();
9          Kunde k2 = new Kunde();
10         Projekt p1 = new Projekt(k1);
11         Projekt p2 = new Projekt(k2);
12         // DateTime(Jahr, Monat, Tag, Stunde, Minute, Sekunde)
13         Projekttätigkeit pt1 = new Projekttätigkeit(
14             new DateTime(2016, 7, 11, 11, 0, 0),
15             new DateTime(2016, 7, 11, 12, 0, 0), p1);
16         Tätigkeit t1 = new Pause(
17             new DateTime(2016, 7, 11, 12, 0, 0),
18             new DateTime(2016, 7, 11, 13, 0, 0));
19         Tätigkeit t2 = new Projekttätigkeit(
20             new DateTime(2016, 7, 11, 20, 0, 0),
21             new DateTime(2016, 7, 11, 23, 0, 0), p2);
22         z.FügeAufgabeHinzu(m1, pt1, "Planungstreffen");
23         z.FügeAufgabeHinzu(m2, pt1, "Planungstreffen");
24         z.FügeAufgabeHinzu(m1, t1, "Mittag");
25         z.FügeAufgabeHinzu(m1, t2, "Notfall");
26         double a = z.SammleProjekte().Count();
27         double b = z.BestimmeKostenImProjekt(p1);
28         double c = z.BestimmeKostenFürKunde(k2);
29     }
30 }
31
32 class Zeiterfassung
33 {
34     List<Aufgabe> aufgaben = new List<Aufgabe>();
35
36     public void FügeAufgabeHinzu(Mitarbeiter mitarbeiter,
37         Tätigkeit tätigkeit, string kommentar)
38     {
39         aufgaben.Add(Aufgabe(mitarbeiter, tätigkeit, kommentar));
40     }
41
42     public double BestimmeKostenImProjekt(Projekt projekt)
43     {
44         double summe = 0.0;
45         foreach (Aufgabe a in aufgaben)
46         {
47
```

```
48              if(a.DieTätigkeit is Projekttätigkeit)
49              {
50                  Projekttätigkeit p = (Projekttätigkeit)a.DieTätigkeit;
51                  if(p.DasProjekt == projekt)
52                  {
53                      summe += a.BerechneKosten();
54                  }
55              }
56          }
57          return summe;
58      }
59
60      public SammleProjekte()
61      {
62          public List<Projekt> projekte = new List<Projekt>();
63          foreach (a in aufgaben)
64          {
65              if (a.DieTätigkeit is Projekttätigkeit)
66              {
67                  Projekt p = ((Projekttätigkeit)a.DieTätigkeit).DasProjekt;
68                  if (!projekte.Contains(p))  // Contains: enthalten?
69                  {
70                      projekte.Add;
71                  }
72              }
73          }
74          return projekte;
75      }
76
77      public double BestimmeKostenFürKunde(kunde)
78      {
79          double summe = 0.0;
80          List<Projekt> projekte = SammleProjekte();
81          foreach (Projekt p in projekte)
82          {
83              if (p.DerKunde == kunde)
84              {
85                  summe += BestimmeKostenImProjekt();
86              }
87          }
88          return summe;
89      }
90  }
91
92  class Aufgabe
93  {
94      Mitarbeiter mitarbeiter;
95      Tätigkeit tätigkeit;
96      public Tätigkeit DieTätigkeit { get { return tätigkeit; } }
97      string kommentar;
98
```

```
99      public Aufgabe(Mitarbeiter mitarbeiter, Tätigkeit tätigkeit,
100                                                 string kommentar)
101     {
102         this.mitarbeiter = mitarbeiter;
103         this.tätigkeit = tätigkeit;
104         this.kommentar = kommentar;
105     }
106
107     public double BerechneKosten()
108     {
109         // TotalHours ist eine Property von DateTime.
110         return tätigkeit.Dauer.TotalHours * Stundensatz
111             + (tätigkeit.HatNachtzuschlag() ? mitarbeiter.Nachtzuschlag : 0.0);
112     }
113 }
114
115 class Mitarbeiter
116 {
117     string nachname;
118     double stundensatz;
119     public double Stundensatz { get { return stundensatz; } }
120     double nachtzuschlag; // maximal ein Mal pro Aufgabe
121     public double Nachtzuschlag { get { return nachtzuschlag; } }
122
123     public void Mitarbeiter(string nachname, double stundensatz,
124                                             double nachtzuschlag)
125     {
126         this.nachname = nachname;
127         this.stundensatz = stundensatz;
128         this.nachtzuschlag = nachtzuschlag;
129     }
130 }
131
132 abstract class Tätigkeit
133 {
134     DateTime von;
135     DateTime bis;
136
137     public Tätigkeit(DateTime von, DateTime bis)
138     {
139         this.von = von;
140         this.bis = bis;
141     }
142
143     public Dauer
144     { get { bis - von; } }
145
146     public bool HatNachtzuschlag()
147     {
148         bis.Hour >= 20 || von.Hour <= 6 || von.Date != bis.Date;
149     }
```

```
150 }
151
152 class Projekttätigkeit : Tätigkeit
153 {
154     Projekt projekt;
155     public Projekt DasProjekt { get { return projekt; } }
156
157     public Projekttätigkeit(DateTime von, DateTime bis, Projekt p)
158         : base(von, bis)
159     {
160         projekt = p;
161     }
162 }
163
164 class Pause : Tätigkeit
165 {
166     public Pause(DateTime von, DateTime bis)
167         : base(von, bis)
168     { }
169 }
170
171 abstract class Projekt
172 {
173     Kunde kunde;
174     public Kunde DerKunde
175     { get { return kunde; } }
176
177     public Projekt(Kunde kunde)
178     {
179         this.kunde = kunde;
180     }
181 }
182
183 class Kunde
184 { }
```