Informatik 2 für Regenerative Energien

Klausur vom 20. September 2013

Jörn Loviscach

Versionsstand: 20. September 2013, 09:25

15 Punkte für die erste Aufgabe; 3 Punkte für alle weiteren Aufgaben. Mindestpunktzahl zum Bestehen: 15 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, falls nicht in ILIAS

1. Im C#-Programmlisting im Anhang sind 15 Fehler, darunter keine Tippfehler und höchstens ein Fehler pro Zeile. Stellen Sie eine Liste dieser Art mit allen Fehlern auf:

Zeile	korrekter Programmtext
123	`public void foo()`
543	`int a = 42;`

2. Mit dem (korrigierten) Code aus dem Anhang wird die Methode `Test.Teste()` ausgeführt. Welche Werte stehen in den Variablen `p`, `istDa1` und `istDa2` in den Zeilen 12 bis 14?

3. Wenn für denselben Mitarbeiter zweimal hintereinander `Geht` aufgerufen wird, ohne dass zwischendurch für ihn `Kommt` aufgerufen wird, soll eine Exception geworfen werden. Was schreiben Sie dazu an welcher Stelle ins Programm?

4. Schreiben Sie eine Methode der Klasse `Zeiterfassung`, die bestimmt, wie viele Mitarbeiter aktuell anwesend sind.

5. Schreiben Sie eine Klasse `Gast` folgender Art: Sie erbt von `Person`. Sie enthält einen `Mitarbeiter`, den der Gast besucht. Der Konstruktor von `Gast` hat als Parameter Vorname, Nachname, Anrede, Geburtsdatum und den Mitarbeiter, der besucht wird.

6. Schreiben Sie eine Methode der Klasse `Zeiterfassung`, die in eine Datei namens `"zeiten.txt"` zeilenweise Personalnummer und gesamte Arbeitszeit ausgibt. Benutzen Sie dafür zum Beispiel `System.IO.File.WriteAllLines(string pfad, List<string> zeilen)`. Welche Änderung(en) sind in den anderen Teilen des Programms nötig?

7. Zeichnen Sie zu diesen C#-Klassen ein UML-Klassendiagramm:

```
abstract class A
{
    int b;
    public virtual int c(int d)
    {
        return 13;
    }
}

class B : A
{
    public override int c(int d)
    {
        return 42;
    }
}

class C : B
{
    int e;
}
```

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(13);
a.Add(42);
Queue<int> b = new Queue<int>();
b.Enqueue(a[1]);
b.Enqueue(9);
List<Queue<int>> c = new List<Queue<int>>();
c.Add(b);
c.Add(new Queue<int>());
c.Add(b);
int x = b.Dequeue();
int y = c[2].Dequeue();
c[0].Enqueue(11);
int z = c[2].Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dies soll ein Programm zur Zeiterfassung in einem kleinen Betrieb werden.
Die Methode `Teste` macht die Handhabung der Klassen vor.

```
1  class Test
2  {
3      static public void Teste()
4      {
5          z = new Zeiterfassung();
6          Mitarbeiter mCurie = new Mitarbeiter("Marie", "Curie",
7                                               "Madame", new DateTime(1867, 11, 7));
8          z.fügeMitarbeiterHinzu(mCurie);
9          z.fügeMitarbeiterHinzu(Mitarbeiter("Albert", "Einstein",
10                                              "Herr", new DateTime(1879, 3, 14)));
11         mCurie.Kommt();
12         int p = z.findePersonalnummer(mCurie);
13         bool istDa1 = z.IstDa(p);
14         bool istDa2 = z.IstDa(2);
15         z.Geht(p);
16     }
17 }
18
19 class Zeiterfassung
20 {
21     List<Mitarbeiter> mitarbeiter = new List<Mitarbeiter>();
22
23     public void fügeMitarbeiterHinzu(Mitarbeiter m)
24     {
25         Add(m);
26     }
27
28     public Mitarbeiter gibMitarbeiter(int personalnummer)
29     {
30         return mitarbeiter.Find(x => x.Personalnummer == personalnummer);
31     }
32
33     public int findePersonalnummer(Mitarbeiter m)
34     {
35         return m.Personalnummer;
36     }
37
38     public void Kommt(int personalnummer)
39     {
40         gibMitarbeiter(personalnummer).Kommt();
41     }
42
43     public void Geht(int personalnummer)
44     {
45         gibMitarbeiter(personalnummer).Geht();
46     }
47
```

```
48      public bool IstDa(int personalnummer)
49      {
50          gibMitarbeiter(personalnummer).IstDa();
51      }
52  }
53
54  class KommtGehtZeit
55  {
56      DateTime kommt;
57      DateTime geht;
58
59      bool nochNichtGegangen;
60      bool NochNichtGegangen
61      {
62          get { return nochNichtGegangen; }
63      }
64
65      public KommtGehtZeit()
66      {
67          kommt = DateTime.Now;
68          nochNichtGegangen = true;
69      }
70
71      public Geht()
72      {
73          geht = DateTime.Now;
74          nochNichtGegangen = true;
75      }
76
77      public double Dauer
78      {
79          get { return (geht - kommt).TotalHours; } // Kein Fehler!
80      }
81  }
82
83  class Person
84  {
85      string vorname;
86      string nachname;
87      string anrede;
88      DateTime geburtsdatum;
89
90      public Person(string vorname, string nachname,
91                    string anrede, DateTime geburtsdatum)
92      {
93          this.vorname = vorname;
94          this.nachname = nachname;
95          this.anrede = anrede;
96          this.geburtsdatum = geburtsdatum;
97      }
98
```

```
99      public abstract bool DarfBüroAlleinBetreten()
100     {
101         return false;
102     }
103 }
104
105 class Mitarbeiter
106 {
107     int wöchentlicheRegelarbeitszeit = 38.5;
108     List<KommtGehtZeit> kommtGehtZeiten = new List();
109     double gesamteArbeitszeit;
110
111     int personalnummer;
112     static int zuletztVergebenePersonalnummer;
113     public int Personalnummer
114     {
115         get { return personalnummer; }
116     }
117
118     public Mitarbeiter(string vorname, string nachname,
119                             string anrede, DateTime geburtsdatum)
120         : base(vorname, nachname, anrede, geburtsdatum)
121     {
122         zuletztVergebenePersonalnummer++;
123         personalnummer = zuletztVergebenePersonalnummer;
124     }
125
126     public void Kommt()
127     {
128         kommtGehtZeiten.Add(new KommtGehtZeit);
129     }
130
131     public void Geht()
132     {
133         // Last() gibt den letzten Eintrag einer Liste.
134         kommtGehtZeiten.Last().Geht();
135         gesamteArbeitszeit = kommtGehtZeiten.Last().Dauer;
136     }
137
138     public bool IstDa()
139     {
140         return kommtGehtZeiten.Count != 0
141                         && !kommtGehtZeiten.Last().NochNichtGegangen;
142     }
143
144     protected override bool DarfBüroAlleinBetreten()
145     {
146         return true;
147     }
148 }
```