Informatik 2 für Regenerative Energien

Klausur vom 21. September 2012

Jörn Loviscach

Versionsstand: 21. September 2012, 16:30

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 den (korrigierten) Klassen aus dem Anhang wird folgendes ausgeführt:

```
Kühlschrank k1 = new Kühlschrank();
DateTime d = new DateTime(1800, 1, 1); // 1. Januar 1800
k1.LegHinein(new Milch(d), Fach.Normal);
k1.LegHinein(new Milch(d), Fach.Normal);
string s1 = k1.FindeAbgelaufeneInhaltsstücke();
```

   Was steht danach in `s1`?

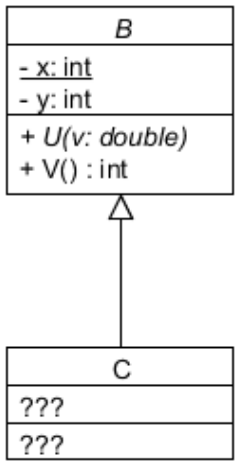
3. Ändern Sie die Methode `FindeAbgelaufeneInhaltsstücke` der (korrigierten) Klasse `Kühlschrank` aus dem Anhang so, dass hinter dem letzten Namen in der zurückgegebenen Zeichenkette kein Komma steht. Diese Zeichenkette soll also zum Beispiel (je nach Inhalt des Kühlschranks) so aussehen: `"großer Salat, kleiner Schokoriegel"`

4. Erweitern Sie die Methode `LegHinein` der Klasse `Kühlschrank` der (korrigierten) Klassen aus dem Anhang so, dass eine Exception geworfen wird, wenn der Name bereits in der Liste `inhalt` vorkommt.

5. Schreiben Sie eine Klasse `Butter`, die von der (korrigierten) Klasse `Inhaltsstück` aus dem Anhang erbt und einen Konstruktor hat, dem man den Namen und die Packungsgröße in Gramm übergibt – und sonst nichts. Machen Sie dabei plausible Annahmen für unbekannte Größen.

6. Erweitern Sie die (korrigierte) Klasse `Kühlschrank` aus dem Anhang so, dass sie eine Variable enthält, in der immer aktuell steht, wie viele Sachen erst nach Ablauf des Verfallsdatums aus dem Kühlschrank genommen wurden.

7. Implementieren Sie dieses Klassendiagramm in C#. Was muss die untere Klasse mindestens enthalten?



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(1); a.Add(2); a.Add(3);
List<int> b = new List<int>();
b.Add(4); b.Add(5); b.Add(6);
Queue<List<int>> c = new Queue<List<int>>();
c.Enqueue(a);
c.Enqueue(b);
c.Enqueue(b);
a[2] = 7;
b[2] = 8;
int x = c.Dequeue()[2];
int y = c.Dequeue()[2];
int z = c.Dequeue()[2];
```

Dieses Listing enthält 15 Fehler!

Dies soll das Programm für einen „intelligenten“ Kühlschrank werden, der sich merkt, was in ihm ist. Die Klasse `Test` macht die Handhabung der Klassen vor.

```
1  class Test
2  {
3      public void Main()
4      {
5          k = new Kühlschrank();
6
7          k.LegHinein(Salat("dicker␣Kopfsalat"), Fach.Gemüse);
8          string s = k.FindeAbgelaufeneInhaltsstücke();
9          Inhaltsstück i = k.NimmHeraus("dicker␣Kopfsalat");
10
11         if (k.IstAbtauenNötig)
12         {
13             // Abtauen und dann:
14             k.WurdeAbgetaut();
15         }
16     }
17 }
18
19 class Kühlschrank
20 {
21     List<Inhaltsstück> inhalt = new List<Inhaltsstück>;
22
23     public void LegHinein(Inhaltsstück i, Fach f)
24     {
25         if (i.Wohin == f)
26         {
27             throw new ApplicationException("Falsches␣Fach!");
28         }
29
30         inhalt.Add(i);
31     }
32
33     public Inhaltsstück NimmHeraus(string name)
34     {
35         foreach (Inhaltsstück i in inhalt)
36         {
37             if (i.Name == name)
38             {
39                 inhalt.Remove(i);
40                 return i;
41             }
42         }
43
44         return 0;
45     }
46
```

```
47      public FindeAbgelaufeneInhaltsstücke()
48      {
49          string abgelaufeneInhaltsstücke = "";
50          foreach (Inhaltsstück i in inhalt)
51          {
52              if(i.Verfallsdatum > DateTime.Now)
53              {
54                  abgelaufeneInhaltsstücke += Name + ",␣";
55              }
56          }
57          return abgelaufeneInhaltsstücke;
58      }
59
60      DateTime letztesAbtauen = DateTime.Now;
61
62      public bool IstAbtauenNötig()
63      {
64          return DateTime.Now > letztesAbtauen + TimeSpan.FromDays(150);
65      }
66
67      public WurdeAbgetaut()
68      {
69          letztesAbtauen = DateTime.Now;
70      }
71  }
72
73  abstract Inhaltsstück
74  {
75      string name;
76      public string Name
77      {
78          get {return name;}
79      }
80
81      DateTime verfallsdatum;
82      public DateTime Verfallsdatum
83      {
84          get {return verfallsdatum;}
85      }
86
87      Fach wohin;
88      public Fach Wohin
89      {
90          get {return wohin;}
91      }
92
93      private Inhaltsstück(string name, DateTime verfallsdatum, Fach wohin)
94      {
95          this.name = name;
96          this.verfallsdatum = verfallsdatum;
97          this.wohin = wohin;
```

```
98      }
99  }
100
101 enum Fach
102 {Normal, Gemüse, Gefriergut}
103
104 class Pizza : Inhaltsstück
105 {
106     public Pizza(string name, DateTime verfallsdatum)
107         : base(name, Fach.Gefriergut)
108     {
109     }
110 }
111
112 class Salat : Inhaltsstück
113 {
114     public Salat(string name)
115         : base(name, DateTime.Now + TimeSpan.FromDays(5), Gemüse)
116     {
117     }
118 }
119
120 class Milch : Inhaltsstück
121 {
122     static nummer = 1;
123     public Milch(DateTime verfallsdatum)
124         : base("Milch␣" + nummer, verfallsdatum, Fach.Normal)
125     {
126         nummer++;
127     }
128 }
```