Informatik 2 für Regenerative Energien

Klausur vom 29. September 2011

Jörn Loviscach

Versionsstand: 29. September 2011, 00:55

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. Welche Werte stehen danach in den Variablen `c`, `d` und `e`?

```
Vokabel a = new Vokabel("Pferd", "horse");
a.ZähleFehler();
Vokabel b = a;
a.ZähleFehler();
int c = a.Übehäufigkeit;
int d = b.Übehäufigkeit;
a.SetzeLernzustand(new Unsicher());
int e = a.Übehäufigkeit;
```

3. Schreiben Sie einen zweiten, öffentlichen Konstruktor für die (korrigierte) Klasse `Wörterbuch` im Anhang. Dieser Konstruktor soll als Parameter eine Angabe wie `C:\bla\blubb\vokabeln.txt` erwarten und dann die Liste

`vokabeln` aus der so angegebenen Datei einlesen. Die Datei ist nach Art dieses Musters aufgebaut:

```
haben,have
geben,give
sagen,say
Pferd,horse
Auto,car
Erdbeere,strawberry
```

Hinweis: `System.IO.File.ReadAllLines` und die `Split`-Methode der Zeichenketten.

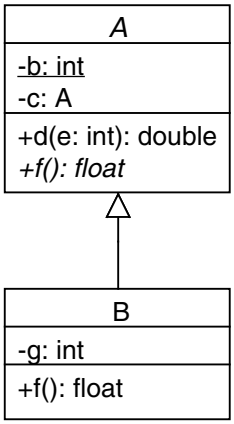
4. Schreiben Sie eine öffentliche Methode `GibSchwierigeWörter` für die (korrigierte) Klasse `Wörterbuch` im Anhang. Diese Methode soll einen `string` liefern, der alle Wörter der ersten Sprache mit einer Übehäufigkeit über 5 enthält, jeweils durch Komma getrennt, Beispiel: `"geben,Pferd,Auto"`. Vor dem ersten Wort und nach dem letzten Wort soll kein Komma stehen.

5. Erweitern Sie den Konstruktor der (korrigierten) Klasse `Vokabel` im Anhang so, dass er eine Exception wirft, wenn eine oder beide der übergebenen Zeichenketten gleich der leeren Zeichenkette ist/sind.

6. Welche Zahl steht nach Ausführung dieses C#-Programmfragments in der Variablen `d`? Geben Sie möglichst auch Zwischenschritte an, damit Ihr Gedankengang nachvollziehbar ist.

```
List<int> a = new List<int>();
a.Add(10);
a.Add(11);
a.Add(12);
List<int> b = new List<int>();
b.Add(20);
b.Add(21);
b.Add(22);
Queue<List<int>> c = new Queue<List<int>>();
c.Enqueue(a);
c.Enqueue(b);
int d = c.Dequeue()[2];
```

7. Schreiben Sie in C# Klassen mit Attributen und Methoden, die diesem UML-Diagramm entsprechen:



8. Das Array `a` wird mit dem folgenden Code aufgebaut. Wieviele Speicherstellen für `int`-Zahlen gibt es in diesem Array? Geben Sie möglichst auch Zwischenschritte an, damit Ihr Gedankengang nachvollziehbar ist.

```
int[][] a = new int[3][];
for (int i = 0; i < 3; i++)
{
    a[i] = new int[i+1];
}
```

Dieses Listing enthält 15 Fehler!

Dies soll ein einfaches Vokabelübungsprogramm werden. In einem Ausgabefeld wird eine Vokabel in der einen Sprache ausgegeben; der Benutzer muss die Übersetzung in das Eingabefeld eingeben und dann den OK-Eingabeknopf klicken. Gut gelernte Wörter sollen seltener abgefragt werden als schlecht gelernte.

```
1  class Vokabeltrainer : Window
2  {
3      // Der Code für die Bedienoberfläche ist hier ausgelassen.
4      // Hier stünden inbesondere die Definitionen und Initialisierungen
5      // von eingabefeld, ausgabefeld, eingabeknopf.
6
7      Wörterbuch wörterbuch = new Wörterbuch;
8      Vokabel aktuelleVokabel;
9
10     public Vokabeltrainer()
11     {
12         InitializeComponent(); // für die Oberfläche
13         ladeNeueVokabel(0);
14     }
15
16     void eingabeknopf_Click(object sender, RoutedEventArgs e)
17     {
18         // .Text ist die Zeichkette aus dem Eingabefeld
19         if (eingabefeld.Text != aktuelleVokabel.WortInZweiterSprache)
20         {
21             aktuelleVokabel.ZähleKorrekt();
22             MessageBox.Show("Korrekt!");
23         }
24         else
25         {
26             aktuelleVokabel.ZähleFehler();
27             MessageBox.Show("Oh␣je!");
28         }
29         ladeNeueVokabel();
30     }
31
32     string ladeNeueVokabel()
33     {
34         aktuelleVokabel = wörterbuch.WähleEineVokabel();
35         // .Content ist die Zeichenkette im Ausgabefeld
36         ausgabefeld.Content = aktuelleVokabel.WortInErsterSprache;
37     }
38 }
39
40 class Wörterbuch
41 {
42     vokabeln = new List<Vokabel>();
43     static Random zufall = new Random();
44
```

```
45      public Wörterbuch()
46      {
47          vokabeln.Add(new Vokabel("haben", "have"));
48          vokabeln.Add(new Vokabel("geben", "give"));
49          vokabeln.Add(new Vokabel("sagen", "say"));
50          vokabeln.Add(new Vokabel("Pferd", "horse"));
51          vokabeln.Add(new Vokabel("Auto", "car"));
52          vokabeln.Add(new Vokabel("Erdbeere", "strawberry"));
53      }
54
55      public Vokabel WähleEineVokabel()
56      {
57          List<Vokabel> vokabelnMitHäufigkeit = new List();
58          for (int i = 0; i < vokabeln.Count; i++)
59          {
60              Vokabel v = vokabeln[i];
61              for (int j = 0; j < v.Übehäufigkeit; j++)
62              {
63                  vokabelnMitHäufigkeit.Add(v);
64              }
65          }
66
67          // Next(n) liefert eine Zufallszahl von 0 bis n-1.
68          r = zufall.Next(vokabelnMitHäufigkeit.Count);
69          return vokabeln[r];
70      }
71  }
72
73  abstract class Vokabel
74  {
75      string wortInErsterSprache;
76      string wortInZweiterSprache;
77      Lernzustand lernzustand = new NichtGelernt();
78
79      private Vokabel(string wortInErsterSprache,string wortInZweiterSprache)
80      {
81          this.wortInErsterSprache = wortInErsterSprache;
82          this.wortInZweiterSprache = wortInZweiterSprache;
83      }
84      public void SetzeLernzustand(Lernzustand lernzustand)
85      {
86          lernzustand = lernzustand;
87      }
88      public string WortInErsterSprache
89      {
90          get { return wortInErsterSprache;  }
91      }
92      public string WortInZweiterSprache
93      {
94          get { return wortInZweiterSprache; }
95      }
```

```
96      public void ZähleFehler()
97      {
98          lernzustand.ZähleFehler(this);
99      }
100     public void ZähleKorrekt()
101     {
102         lernzustand.ZähleKorrekt(this);
103     }
104     public int Übehäufigkeit
105     {
106         get { lernzustand.Übehäufigkeit; }
107     }
108 }
109
110 abstract class Lernzustand
111 {
112     virtual public void ZähleFehler(Vokabel v)
113     { }
114     virtual public void ZähleKorrekt(Vokabel v)
115     { }
116     public int Übehäufigkeit
117     {
118         get { return 10; }
119     }
120 }
121
122 class Gelernt : Lernzustand
123 {
124     override public void ZähleFehler(Vokabel v)
125     {
126         v.SetzeLernzustand(new Unsicher());
127     }
128     override public int Übehäufigkeit
129     {
130         get { return 1.0; }
131     }
132 }
133
134 class NichtGelernt : Lernzustand
135 {
136     override public void ZähleKorrekt(Vokabel v)
137     {
138         v.SetzeLernzustand(new Unsicher());
139     }
140     override public int Übehäufigkeit
141     {
142         get { return 10; }
143     }
144 }
145
146 class Unsicher
```

```
147 {
148     override public void ZähleFehler(Vokabel v)
149     {
150         v.SetzeLernzustand(new NichtGelernt());
151     }
152     override public void ZähleKorrekt(Vokabel v)
153     {
154         v.SetzeLernzustand(new Gelernt());
155     }
156     override public int Übehäufigkeit
157     {
158         get { return 3; }
159     }
160 }
```