Informatik 2 für Regenerative Energien

Klausur vom 5. Juli 2013

Jörn Loviscach

Versionsstand: 13. Juli 2013, 18:12

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 `r`, `z` und `c` in den Zeilen 16 bis 18?

3. Am Anfang des Konstruktors von `EMail` soll eine Abfrage eingefügt werden, ob `absender` und `empfänger` ein `@`-Zeichen enthalten. Falls nein, soll eine Exception geworfen werden. Hinweis: Die Methode `int IndexOf(char c)` der Klasse `string` liefert die Position des ersten Vorkommens von `c` oder, falls es nicht vorkommt, -1.

4. Angenommen, man hätte in `Test.Teste()` noch eine Sammlung von `EMail`s:

   ```
   List<EMail> posteingang = ... // stammt irgendwo her
   ```

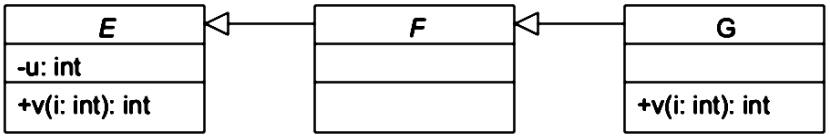
   Nun soll bestimmt werden, wie viele Mails im `posteingang` *ungelesen* sind (angehängte Mails nicht mitgezählt). Formulieren Sie den nötigen Code aus, einschließlich der nötigen Änderungen an den (korrigierten) Klassen im Anhang der Klausur[c1].

   [c1] *text added by jl*

5. Schreiben Sie eine Klasse `Visitenkarte` folgender Art: Sie erbt von `Datenelement`. Sie besitzt einen parameterlosen Konstruktor, der als Titel immer `"Visitenkarte"` benutzt. Sie liefert als Beschreibung `"Visitenkarte"` zurück. Sie liefert als Größe[c2] 42 zurück.

   [c2] jl: Länge

6. Schreiben Sie C#-Klassen, die diesem UML-Diagramm entsprechen:



7. Eine Textdatei hat einen Aufbau wie den folgenden, mit beliebig vielen Zeilen:

   ```
   42;13;"abc"
   123;456;"defgh"
   234;567;"ijklmn"
   ```

   Schreiben Sie Stück C#-Code, das diese Datei namens `"test.txt"` einliest und die Summe der Zahlen in der Mitte der Zeilen berechnet, hier also 13 + 456 + 567. Hinweis: Benutzen Sie zum Beispiel `string[] System.IO.File.ReadAllLines(string path)`, die statische Methode `Parse` von `int` und die Methode `Split(';')` von `string`. Letztere liefert die Teile der Zeichenkette, von einem Semikolon getrennt sind, als Array von Zeichenketten.

8. Welche Zahlen stehen nach Ausführung dieses C#-Programmfragments in den Variablen `c`, `d` und `e`? 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);
   Stack<List<int>> b = new Stack<List<int>>();
   b.Push(a);
   b.Push(a);
   b.Push(new List<int>());
   int c = a[2];
   int d = b.Pop().Count;
   int e = b.Pop()[2];
   ```

Dieses Listing enthält 15 Fehler!

Dies soll ein Programm werden, das E-Mails und deren Anhänge (Attachments) verwaltet. Solche Anhänge können auch weitergeleitete E-Mails sein. Die Methode `Teste` macht die Handhabung der Klassen vor.

```
1  class Test
2  {
3      public static void Teste()
4      {
5          EMail e = new EMail("Re:␣Anfrage", "ab@cd.com",
6              "ef@gh.de", "Tach!");
7          byte[] d1 = new byte[100000]; // müsste man aus Bilddatei einlesen
8          e.FügeAnhangHinzu(new Bilddatei("Bestellung",
9              Bilddatei.Bildtyp.JPG, d1));
10         e.FügeAnhangHinzu(new EMail("Anfrage", "ef@gh.de",
11             "ab@cd.com", "Hi!"));
12         EMail f = (EMail)e.GibAnhang(1);
13         byte[] d2 = new byte[100000]; // müsste man aus Bilddatei einlesen
14         f.FügeAnhangHinzu(new Bilddatei("Angebot",
15             Bilddatei.Bildtyp.JPG, d2));
16         bool r = e.EntferneAnhang("Test");
17         int z = e.ZahlDerAnhänge();
18         int c = e.HoleGrößeInZeichen();
19     }
20 }
21
22 abstract class Datenelement
23 {
24     string titel;
25     public string Titel
26     {
27         get { return titel;  }
28     }
29
30     public Datenelement(string titel)
31     {
32         this.titel = titel;
33     }
34
35     public static string HoleBeschreibung();
36
37     public virtual int HoleGrößeInZeichen();
38 }
39
40 class EMail : Datenelement
41 {
42     string absender;
43     string empfänger;
44     DateTime erstellungsZeitpunkt;
45     string text;
46     bool gelesen;
```

```
47      List<Datenelement> anhänge = new Datenelement();
48
49      public EMail(string titel, string absender,
50          string empfänger, string text)
51          : base(titel)
52      {
53          this.absender = absender;
54          this.empfänger = empfänger;
55          this.text = text;
56          erstellungsZeitpunkt = DateTime.Now;
57      }
58
59      public string HoleBeschreibung()
60      {
61          string resultat = Titel + "␣mit␣" + anhänge.Count + "␣Anhängen:␣";
62          bool istDerErste = true;
63          foreach (var item in anhänge)
64          {
65              if (!istDerErste);
66              {
67                  resultat = resultat + ",␣";
68              }
69              istDerErste = false;
70              resultat = resultat + item.HoleBeschreibung;
71          }
72          return resultat;
73      }
74
75      public override HoleGrößeInZeichen()
76      {
77          resultat = text.Length;
78          foreach (var item in anhänge)
79          {
80              resultat += item.HoleGrößeInZeichen(resultat);
81          }
82          return resultat;
83      }
84
85      public void FügeAnhangHinzu(Datenelement anhang)
86      {
87          Add(anhang);
88      }
89
90      public Datenelement GibAnhang(int i)
91      {
92          anhänge[i];
93      }
94
95      public bool EntferneAnhang(string titel)
96      {
97          Datenelement a = anhänge.Find(x => x.Titel == titel);
```

```
98          if (a != null)
99          {
100             anhänge.Remove(a);
101             return true;
102         }
103         else
104         {
105             return false;
106         }
107     }
108
109     public int ZahlDerAnhänge
110     {
111         get { return anhänge.Count; }
112     }
113
114     public string Text
115     {
116         get
117         {
118             gelesen = true;
119             return text;
120         }
121     }
122 }
123
124 class Bilddatei
125 {
126     public enum Bildtyp { JPG, PNG, GIF }
127     Bildtyp typ;
128     byte[] daten;
129
130     public Bilddatei(string titel, Bilddatei.Bildtyp typ, byte[] daten)
131         : base(daten)
132     {
133         this.typ = typ;
134         this.daten = daten;
135     }
136
137     public override string HoleBeschreibung()
138     {
139         // Die folgende Zeile ist korrekt: "JPG" usw. als string.
140         return Titel+"␣("+System.Enum.GetName(typeof(Bildtyp),typ)+")";
141     }
142
143     public override int HoleGrößeInZeichen()
144     {
145         return daten;
146     }
147 }
```