Informatik 2 für Regenerative Energien

Klausur vom 7. Juli 2011

Jörn Loviscach

Versionsstand: 13. Juli 2011, 20:59

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[c1] eine Liste dieser Art mit allen Fehlern auf:

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

   [c1] *text added by jl*

2. Mit den Klassen aus dem Anhang wird Folgendes ausgeführt. Welche Werte stehen danach in den Variablen `a`, `b` und `c`?

   ```
   Stack<Mitspieler> s = new Stack<Mitspieler>();
   s.Push(new Mitspieler("Doris"));
   s.Push(new Mitspieler("Egon"));
   string a = s.Pop().Name;
   s.Push(new Mitspieler("Friedrich"));
   int b = s.Pop().Punktestand;
   string c = s.Pop().Name;
   ```

3. Von der Klasse `Spiel` im Anhang soll eine Ableitung `KurzesSpiel` gebildet werden, in der man schon mit weniger Punkten gewonnen hat. Welche Änderungen wären dafür in der Klasse `Spiel` sinnvoll? Schreiben Sie außerdem einen Konstruktor für `KurzesSpiel` auf.

4. Welche der fünf Methoden der Klasse `Würfelbecher` im Anhang wird am ehesten eine Exception werfen? Unter welchen Voraussetzungen wird das passieren?

5. Es sollen Stunde:Minute:Sekunde-Zeitangaben aus Zeichenketten wie `42:13:7` oder `123:4:56` in die jeweilige Gesamtzahl an Sekunden verwandelt werden. Schreiben Sie eine Methode, die eine solche Zeichenkette annimmt und die gesamte Zahl der Sekunden als `int` zurückliefert. Verwenden Sie `int.Parse` und die `Split`-Methode der Zeichenketten.

6. Es gibt drei Klassen `GeometrischeFigur`, `Kreis` und `Dreieck`, die sinnvoll voneinander erben sollen. Alle sollen die jeweils geometrisch nötigen Attribute haben, wobei die Werte von Koordinaten als `double` angegeben sind. Alle drei Klassen sollen eine Methode zur Flächenberechung haben. Zeichnen Sie ein UML-Klassendiagramm mit den drei Klassen. (Damit Kursivschrift zu erkennen ist, umkringeln Sie die oder benutzen Sie eine andere Farbe dafür.)

7. Gegeben ist folgende Struct:

```
struct Messung
{
    public double Geschwindigkeit; // in Metern pro Sekunde
    public double Zeitpunkt; // in Sekunden
}
```

   Eine `List<Messung> messungen` enthalte solche Messungen in der korrekten zeitlichen Reihenfolge. Schreiben Sie einige Zeilen C#, welche die gesamte zurückgelegte Strecke schätzen: die Geschwindigkeit aus der ersten Messung mal die Zeit von der ersten bis zur zweiten Messung, dazu addiert die Geschwindigkeit aus der zweiten Messung mal die Zeit von der zweiten bis zur dritten Messung usw.

8. Angenommen, Sie hätten den Quelltext der Klasse `string` und müssten dort noch eine öffentliche Methode `Parse` zum Umwandeln von Strings wie `"123"` in eine `int`-Ganzzahl hinzufügen. Sie können dazu die vorhandene Methode `int.Parse` verwenden. Formulieren Sie die `???` in diesem Code aus:

```
class string
{
    // ...

    ??? Parse ???
    {
        ???
    }
}
```

Dieses Listing enthält 15 Fehler!

```
1  class MainWindow : Window
2  {
3      // ... diverses Anderes
4
5      private void buttonClick(object sender, RoutedEventArgs e)
6      {
7          Mitspieler[] dieMitspieler = new Mitspieler;
8          dieMitspieler[0] = new Mitspieler("Anton");
9          dieMitspieler[1] = new Mitspieler("Berta");
10         dieMitspieler[2] = new Mitspieler("Carla");
11         Spiel = new Spiel(dieMitspieler, 2);
12         int ergebnis = dasSpiel.SpieleBisZumEnde();
13     }
14 }
15
16 class Zufallsgenerator
17 {
18     abstract public int Werfe(); // Zufallszahl erzeugen
19 }
20
21 class Würfel : Zufallsgenerator
22 {
23     static Random zufallsgenerator = new Random();
24     public int Werfe() // soll eine Zahl von 1 bis 6 ergeben
25     {
26         // Next() liefert eine Zufallszahl ab 0 aufwärts.
27         return zufallsgenerator.Next() % 6;
28     }
29 }
30
31 class Würfelbecher : Zufallsgenerator
32 {
33     List<Würfel> enthalteneWürfel = new List<Würfel>();
34     public Würfelbecher(int würfelAnzahl)
35     {
36         for (int i = 0; i < würfelAnzahl; i++)
37         {
38             NehmeEinenWürfelMehr();
39         }
40     }
41     public override int Werfe()
42     {
43         int Summe = 0;
44         foreach (Würfel w in enthalteneWürfel)
45         {
46             Summe += w.Werfe;
47         }
48         return Summe;
49     }
```

```
50      public void NehmeEinenWürfelMehr()
51      {
52          enthalteneWürfel.Add();
53      }
54      public void NehmeEinenWürfelWeniger()
55      {
56          // Den vordesten der Liste entfernen,
57          // die anderen rücken vor.
58          enthalteneWürfel.RemoveAt(0);
59      }
60      public int ZahlDerWürfel
61      {
62          get { return enthalteneWürfel.Count; }
63      }
64  }
65
66  class Mitspieler
67  {
68      string name;
69      public string Name
70      {
71          get { return name; }
72      }
73      int punktestand;
74      public int Punktestand
75      {
76          get { return punktestand; }
77      }
78      Mitspieler(string name)
79      {
80          this.name = name;
81      }
82      public void BuchePunkte(int punkte)
83      {
84          punktestand + punkte;
85      }
86  }
87
88  class Spiel : Würfel
89  {
90      Zufallsgenerator zufallsgenerator;
91      Mitspieler[] dieMitspieler;
92      bool istBeendet = false;
93      bool IstBeendet
94      {
95          get { return istBeendet; }
96      }
97      public Spiel(Mitspieler[] dieMitspieler)
98      {
99          this.dieMitspieler = dieMitspieler;
100         zufallsgenerator = new Würfel();
```

```
101         }
102         public Spiel(Mitspieler[] dieMitspieler, int zahlDerWürfel)
103         {
104             this.dieMitspieler = dieMitspieler;
105             zufallsgenerator = new Würfelbecher();
106         }
107         public void SpieleEineRunde()
108         {
109             foreach (Mitspieler m in dieMitspieler)
110             {
111                 int z = zufallsgenerator.Werfe();
112                 m.BuchePunkte(z);
113                 if (Punktestand >= 1000)
114                 {
115                     MessageBox.Show(m.Name + "␣hat␣gewonnen!");
116                     istBeendet = false;
117                     break;
118                 }
119             }
120         }
121         public void SpieleBisZumEnde()
122         {
123             while(true)
124             {
125                 SpieleEineRunde();
126             }
127         }
128     }
```