Informatik 2 für Regenerative Energien

Klausur vom 28. Januar 2019

Jörn Loviscach

Versionsstand: 28. Januar 2019, 12:14

15 Punkte für die erste Aufgabe; 3 Punkte für alle weiteren Aufgaben. Mindestpunktzahl zum Bestehen: 20 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 (auch nicht wearable), kein Handy und Ähnliches.

Name	Vorname	Matrikelnummer	E-Mail-Adresse

1. Im C#-Programmlisting im Anhang sind 15 Fehler, darunter keine Tippfehler und höchstens ein Fehler pro Zeile. Erstellen Sie eine Liste mit 15 Zeilen aus den Fehlern und ihren jeweiligen Korrekturen, nach dem folgenden Muster:

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

2. Die Methode `Test.Teste` des korrigierten Code aus dem Programmlisting im Anhang wird heute ausgeführt. Was steht am Ende in den Variablen `x`, `y` und `z`? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. Die Methode `FügeBaumHinzu` der Klasse `Wald` soll eine Exception werfen, wenn der übergebene `Baum` schon im `Wald` enthalten ist. Was ändern Sie dazu wie an dem korrigierten Code aus dem Programmlisting?

4. Schreiben Sie für die Klasse `Wald` aus dem korrigierten Code aus dem Programmlisting eine parameterlose öffentliche Methode `ZähleNester`, deren Rückgabewert die Anzahl aller Nester aus den jeweils letzten Untersuchungen aller Bäume ist.

5. Schreiben Sie für die Klasse `Wald` des korrigierten Code aus dem Programmlisting eine parameterlose öffentliche Methode `FindeZuUntersuchendeBäume`, die eine `List` aller Bäume zurückliefert, die im aktuellen Jahr noch nicht untersucht worden sind. Welche Änderungen sind dazu gegebenenfalls an anderen Klassen nötig? Hinweis: `DateTime` besitzt die Property `Year`.

6. Leiten Sie von der Klasse `Untersuchung` des korrigierten Code aus dem Programmlisting eine Klasse `EingehendeUntersuchung` ab, die zusätzlich einen `string` für einen Beschreibungstext enthält. Geben Sie dieser Klasse einen entsprechenden Konstruktor und sorgen Sie dafür, dass die Methode `ToString` für diese Klasse folgende Zeichenkette liefert: `"Eingehende Untersuchung:"` und daran angehängt der Beschreibungstext.

7. Zeichnen Sie ein UML-Klassendiagramm für die folgenden drei Klassen. Kennzeichnen Sie Kursivschrift zum Beispiel durch Farbe.

```
abstract class A
{
    public abstract bool f(int z);
    int x;
}
class B : C
{
    public override bool f(int z)
    {
        return true;
    }
}
abstract class C : A
{
    double y;
}
```

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<List<int>> a = new List<List<int>>();
List<List<int>> b = a;
a.Add(new List<int>());
a.Add(new List<int>());
a.Add(a[0]);
a[0].Add(7);
a[0].Add(8);
Queue<int> c = new Queue<int>();
c.Enqueue(9);
c.Enqueue(10);
int x = a[2][1];
int y = b.Count;
int z = c.Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dies ist der Anfang eines Programms zur Verwaltung eines Forstbetriebs. Die Methode `Teste` der Klasse `Test` macht die Benutzung der Klassen vor. Dies ist der Programmcode der Klassen:

```
1  class Test
2  {
3      public static void Teste()
4      {
5          Wald w = new Wald();
6          Baum b1 = new Fichte(new DateTime(2000, 1, 1),
7                                       new Koordinaten(52.0129414, 8.4921373));
8          w.FügeBaumHinzu(b1);
9          Baum b2 = b1;
10         Baum b3 = new Fichte(new DateTime(2015, 1, 1),
11                                      new Koordinaten(52.0129157, 8.4921628));
12         w.FügeBaumHinzu(b3);
13         Untersuchung u = new Untersuchung(6.0, 30.0, Baumzustand.Gesund, 1);
14         Vogelnest v = new Vogelnest(Elster(), 5.0);
15         u.SetzeNest(v);
16         b1.FügeUntersuchungHinzu(u);
17         bool x = b1.IstNaheBei(b3);
18         double y = w.SchätzeFestmeter();
19         int z = b2.LetzteZahlDerNester();
20     }
21 }
22
23 abstract class Vogelart
24 {}
25
26 class Elster
27 {}
28
29 class Vogelnest
30 {
31     Vogelart vogelart;
32     double höheÜberGrund;
33
34     Vogelnest(Vogelart vogelart, double höheÜberGrund)
35     {
36         this.vogelart = vogelart;
37         this.höheÜberGrund = höheÜberGrund;
38     }
39 }
40
41 enum Baumzustand { Unbekannt, Gesund, Geschädigt, Gefällt }
42
43 class Untersuchung
44 {
45     DateTime wann;
46     double höhe;          // in Metern
```

```
47      double durchmesser;    // in Zentimetern
48      Baumzustand zustand;
49      Vogelnest[] vogelnester;
50
51      public void Untersuchung(double höhe, double durchmesser,
52                               Baumzustand zustand, int anzahlNester)
53      {
54          this.wann = DateTime.Now;
55          this.höhe = höhe;
56          this.durchmesser = durchmesser;
57          this.zustand = zustand;
58          vogelnester = Vogelnest[anzahlNester];
59      }
60
61      public void SetzeNest(int i, Vogelnest n)
62      {
63          vogelnester[i] = n;
64      }
65
66      public int ZahlDerNester
67      {
68          get { vogelnester.Length; }
69      }
70
71      public double SchätzeFestmeter()    // erwartete Kubikmeter Holz
72      {
73          double radius = durchmesser / 200.0;     // Radius in Metern
74          return höhe * Math.PI * radius * radius;
75      }
76  }
77
78  struct Koordinaten
79  {
80      double breite;     // GPS-Angabe des Breitengrads
81      double länge;      // GPS-Angabe des Längengrads
82
83      public Koordinaten(double breite, double länge)
84      {
85          this.breite = breite;
86          this.länge = länge;
87      }
88
89      public IstNaheBei(Koordinaten g)
90      {
91          return Math.Abs(breite - g.breite) + Math.Abs(länge - g.länge) < 0.001;
92      }
93  }
94
95  abstract class Baum
96  {
97      DateTime setzjahr;
```

```
98
99      Koordinaten ort;
100     public Koordinaten Koordinaten
101     {
102         get { return ort; }
103     }
104
105     List<Untersuchung> untersuchungen = new List<Untersuchung>;
106
107     public Baum(DateTime setzjahr, Koordinaten ort)
108     {
109         this.setzjahr = setzjahr;
110         this.ort = ort;
111     }
112
113     public void FügeUntersuchungHinzu(Untersuchung u)
114     {
115         untersuchungen.Add(u);
116     }
117
118     public int Alter        // in Jahren
119     {
120         get { return (int) ((DateTime.Now - setzjahr).TotalDays / 365.0); }
121     }
122
123     public Untersuchung GibLetzteUntersuchung
124     {
125         if(untersuchungen.Count == 0)
126         {
127             return null;
128         }
129         return untersuchungen[untersuchungen.Count];
130     }
131
132     public bool IstNaheBei(Baum b)
133     {
134         return Koordinaten.IstNaheBei(b.Koordinaten);
135     }
136
137     public int LetzteZahlDerNester()
138     {
139         if (untersuchungen.Count == 0)
140         {
141             return 0;
142         }
143         return untersuchungen.Last().ZahlDerNester;
144     }
145 }
146
147 abstract class Fichte : Baum
148 {
```

```
149     public Fichte(DateTime setzjahr, Koordinaten ort)
150         : base(setzjahr, ort)
151     {}
152 }
153
154 class Wald
155 {
156     List<Baum> bäume = new List<Baum>();
157
158     public void FügeBaumHinzu(Baum b)
159     {
160         bäume.Add(b);
161     }
162
163     public List<Baum> FindeBäumeInDerNähe(Koordinaten g)
164     {
165         return bäume.FindAll(b => b.Koordinaten.IstNaheBei(g));
166     }
167
168     public double SchätzeFestmeter()
169     {
170         double summe;
171
172         for (int i = 0; i < bäume.Count; i++)
173         {
174             if(bäume[i].Alter > 30)   // nur ab 30 Jahren berücksichtigen
175             {
176                 Untersuchung u = bäume.GibLetzteUntersuchung();
177                 if(u != null)
178                 {
179                     summe += SchätzeFestmeter();
180                 }
181             }
182         }
183
184         return summe;
185     }
186 }
```