Informatik 2 für Regenerative Energien

Klausur vom 8. Juli 2019

Jörn Loviscach

Versionsstand: 8. Juli 2019, 10:53

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 ausgeführt. Welche Werte stehen am Ende in den Variablen `x`, `y`, `z`? **Beschreiben Sie in jeweils einem Satz, wie Sie zu diesen drei Werten kommen.**

3. Die Methode `FügeBeobachtungHinzu` der Klasse `Pflanze` soll eine Exception werfen, wenn die neu beobachtete Art eine andere ist als die der allerersten Beobachtung. Was ändern Sie an dazu dieser Klasse?

4. Ergänzen Sie die `Pflanze` des korrigierten Code aus dem Programmlisting um eine öffentliche Methode `double BerechneWachstumsrate()`, die aus der ersten und der letzten Beobachtung eine Wachstumsrate bestimmt (cm/Tag). Nehmen Sie an, dass die Beobachtungen in der Liste bereits aufsteigend nach der Zeit geordnet sind. Gibt es weniger als zwei Beobachtungen für die Pflanze, soll diese Methode den Wert `double.NaN` zurückliefern. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

5. Schreiben Sie eine von der Klasse `GeoPunkt` des (korrigierten) Code aus dem Programmlisting abgeleitete Klasse `GeoPunktMitHöhe`, die als Datenelement zusätzlich die geographische Höhe des Punkts enthält. Geben Sie dieser Klasse einen sinnvollen Konstruktor. Kann man dann `p1.BerechneAbstandZu(p2)` für Instanzen `p1` und `p2` von `GeoPunktMitHöhe` aufrufen? Falls ja: Wieso könnte das verwirrend sein? Falls nein: Warum geht das nicht? (ein Satz)

6. Die Klasse `Acker` des korrigierten Code aus dem Programmlisting soll eine öffentliche Methode `FindeNördlichstePflanze` erhalten, welche eine Pflanze zurückliefert, deren geographische Breite maximal groß ist. Falls es keine Pflanze gibt, soll die Methode `null` liefern. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

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

```
class A
{
    double f(int x)
    {
        return 2.7 * x;
    }
    int y;
}
abstract class B
{
    double z;
    public abstract int g();
}
class C : B
{
    public override int g()
    {
        return 23;
    }
    public int h()
    {
        return 42;
    }
}
```

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(6);
a.Add(7);
List<int> b = new List<int>();
b.Add(8);
b.Add(9);
List<int> c = b;
c[0] = 10;
Stack<List<int>> d = new Stack<List<int>>();
d.Push(a);
a.Add(11);
d.Push(a);
d.Push(b);
int x = d.Pop()[0];
int y = d.Pop()[1];
int z = d.Pop()[2];
```

Dieses Listing enthält 15 Fehler!

Dieses Programm soll für einen hochgradig digitalisierten Landwirtschaftsbetrieb die Daten einzelner Pflanzen verwalten. Für den Acker werden Bodenproben genommen. Diese beziehen sich auf die gesamte Fläche. Außerdem wird regelmäßig festgestellt, an welchen Stellen sich Pflanzen befinden. Die Pflanzen werden mit Hilfe genauer Koordinatenangaben (geographische Breite/Länge) früheren Beobachtungen zugeordnet. Wird an einer bestimmten Stelle zum ersten Mal eine Pflanze beobachtet, legt das Programm den Datensatz für diese an.

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          GeoPunkt p1 = new GeoPunkt(52.0472882, 8.5162395);
6          GeoPunkt p2 = new GeoPunkt(52.0473243, 8.5163462);
7          GeoPunkt p3 = new GeoPunkt(52.0473284, 8.5163451);
8          Beobachtung beob1 = new Beobachtung(new DateTime(2019, 7, 4, 12, 0, 0),
9                                              p1, "Möhre", 8.0);
10         Beobachtung beob2 = new Beobachtung(new DateTime(2019, 7, 4, 12, 5, 0),
11                                             p2, "Steckzwiebel", 7.0);
12         Acker a = new Acker;
13         a.FügeBeobachtungHinzu(beob1);
14         a.FügeBeobachtungHinzu(beob2);
15         Bodenprobe bp = new Bodenprobe(Lehm, 8.0);
16         bool x = false;
17         if (p1.BerechneAbstandZu(p3) > p2.BerechneAbstandZu(p3))
18         {
19             x = true;
20         }
21         bool y = a.GibNächstliegendePflanze(p3).IstVerträglichMit(bp);
22         bool z = a.IstBodenMitAllenPflanzenVerträglich(bp);
23     }
24 }
25
26 abstract class GeoPunkt
27 {
28     double geoBreite;
29     double geoLänge;
30
31     public GeoPunkt(double geoBreite, double geoLänge)
32     {
33         this.geoBreite = geoBreite;
34         this.geoLänge = geoLänge;
35     }
36
37     static double BogenmaßAusGrad(double phi)
38     {
39         phi * Math.PI / 180.0;
```

```
40      }
41
42      public BerechneAbstandZu(GeoPunkt p) // in m
43      {
44          double b1 = BogenmaßAusGrad(geoBreite);
45          double b2 = BogenmaßAusGrad(p);
46          // In dem folgenden Ausdruck ist kein Fehler!
47          return 6.37e6 * Math.Acos(Math.Sin(b1) * Math.Sin(b2) + Math.Cos(b1) *
48              Math.Cos(b2) * Math.Cos(BogenmaßAusGrad(geoLänge - p.geoLänge)));
49      }
50  }
51
52  class Beobachtung
53  {
54      DateTime wann;
55      GeoPunkt wo;
56      public GeoPunkt Wo { get { return wo; } }
57      string art;
58      public string Art { get { return art; } }
59      double höhe; // in cm
60
61      Beobachtung(DateTime wann, GeoPunkt wo, string art, double höhe)
62      {
63          this.wann = wann;
64          this.wo = wo;
65          this.art = art;
66          this.höhe = höhe;
67      }
68  }
69
70  enum Bodenart { Sand, Lehm, Ton }
71
72  class Bodenprobe
73  {
74      Bodenart bodenart;
75      public Bodenart Bodenart { get { return bodenart; } }
76      double phosphorgehalt; // in mg pro 100 g Boden
77      public double Phosphorgehalt { get { return phosphorgehalt; } }
78
79      public Bodenprobe(Bodenart bodenart, double phosphorgehalt)
80      {
81          this.bodenart = bodenart;
82          this.phosphorgehalt = phosphorgehalt;
83      }
84  }
85
86  class Pflanze
87  {
88      List<Beobachtung> beobachtungen = new List<Beobachtung>;
89      GeoPunkt wo;
90      public GeoPunkt Wo { get { return wo; } }
```

```
91
92      public Pflanze(GeoPunkt wo)
93      {
94          this.wo = wo;
95      }
96
97      public void FügeBeobachtungHinzu(Beobachtung b)
98      {
99          beobachtungen.Add(b);
100     }
101
102     public abstract IstVerträglichMit(Bodenprobe b);
103 }
104
105 class Zwiebel
106 {
107     bool istSteckzwiebel;
108
109     public Zwiebel(GeoPunkt wo, bool istSteckzwiebel)
110         : base(wo)
111     {
112         this.istSteckzwiebel = istSteckzwiebel;
113     }
114
115     public override bool IstVerträglichMit(Bodenprobe b)
116     {
117         // Der Phosphorgehalt soll über 5 mg pro 100 g liegen.
118         return b.Phosphorgehalt > 5.0;
119     }
120 }
121
122 class Möhre : Pflanze
123 {
124     public Möhre(GeoPunkt wo)
125         : base(wo)
126     {
127     }
128
129     public override bool IstVerträglichMit(Bodenprobe b)
130     {
131         // Der Boden soll sandig sein.
132         return b == Bodenart.Sand;
133     }
134 }
135
136 class Acker
137 {
138     List<Pflanze> pflanzen = new List<Pflanze>();
139
140     public Pflanze GibNächstliegendePflanze(GeoPunkt p)
141     {
```

```
142         Pflanze pflanze = null;
143         double bisherKleinsterAbstand = double.PositiveInfinity;
144         for (int i = 0; i < pflanzen.Count; i++)
145         {
146             double abstand = pflanzen.Wo.BerechneAbstandZu(p);
147             if (abstand < bisherKleinsterAbstand)
148             {
149                 pflanze = pflanzen[i];
150                 bisherKleinsterAbstand = abstand;
151             }
152         }
153         return pflanze;
154     }
155 
156     public void FügeBeobachtungHinzu(Beobachtung b)
157     {
158         Pflanze pf = GibNächstliegendePflanze(b.Wo);
159         if(pf == null || pf.BerechneAbstandZu(b.Wo) > 0.02)
160         {
161             if(b.Art == "Zwiebel" || b.Art == "Steckzwiebel")
162             {
163                 pf = new Zwiebel(b.Wo, b.Art == "Steckzwiebel");
164             }
165             else if (b.Art == "Möhre")
166             {
167                 pf = new Möhre(b.Wo);
168             }
169             else
170             {
171                 pf = null;
172             }
173 
174             if (pf != null)
175             {
176                 pflanzen.Add(pf);
177             }
178         }
179 
180         if(pf != null)
181         {
182             pf.FügeBeobachtungHinzu(b);
183         }
184     }
185 
186     public bool IstBodenMitAllenPflanzenVerträglich(Bodenprobe b)
187     {
188         // Exists gibt true zurück, wenn der Lambda-Ausdruck
189         // für mindestens ein Element der Liste true liefert.
190         return !( pflanzen.Exists(pf => pf.IstVerträglichMit(b)) );
191     }
192 }
```