Informatik 2 für Regenerative Energien

Klausur vom 4. Februar 2022

Jörn Loviscach

Versionsstand: 4. Februar 2022, 09:01

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; Wörterbuch (z. B. Deutsch–Portugiesisch); kein Skript, keine anderen Texte, kein Taschenrechner, kein Computer (auch nicht wearable), kein Handy.

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. Der Konstruktor der Klasse `Station` soll eine Exception werfen, wenn eine Position mit mehr als 3000 Flusskilometern angegeben wird. Was ändern Sie an dazu am Code?

4. Ergänzen Sie die Klasse `Pegelmessstation` des korrigierten Code aus dem Programmlisting um eine öffentliche Methode `double BestimmeMaximalpegelSeit(DateTime wann)`. Diese Methode soll den größten Wert des Pegels seit dem angegebenen Zeitpunkt `wann` (und inklusive diesem) zurückgeben. Falls es in diesem Zeitraum keinen einzigen Messwert gibt, soll sie `NaN` zurückgeben. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

5. Es soll eine weitere Art von Messung geben, nämlich eine kombinierte gleichzeitige Messung von Pegel und Fließgeschwindigkeit. Was ändern Sie dazu es am korrigierten Code aus dem Programmlisting?

6. Die Klasse `Station` des korrigierten Code aus dem Programmlisting soll eine öffentliche Methode `int BestimmeGesamtzahlWarnungen()` erhalten, die zurückgibt, wie viele Warnungen insgesamt von allen Stationen erhalten worden sind (jede Station und jede Warnung einzeln gezählt). Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting? *Kann* diese Methode statisch gemacht werden? (Begründung!) Falls ja: *Sollte* sie statisch gemacht werden? (Begründung!)

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

```
class A
{
    int x;
    public virtual double f(bool y)
    {
        return 3.14;
    }
}
abstract class B : A
{
    public abstract int g(double z);
    public override double f(bool y)
    {
        return 10.0;
    }
}
class C : A
{
    double z;
    public override double f(bool y)
    {
        return 2.717;
    }
}
```

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.

```
Stack<List<int>> a = new Stack<List<int>>();
List<int> b = new List<int>();
List<int> c = new List<int>();
a.Push(b);
a.Push(c);
b.Add(1);
b.Add(2);
b.Add(3);
c.Add(4);
a.Push(b);
a.Push(c);
a.Pop().Add(5);
int x = a.Pop().Count;
int y = a.Pop()[0];
int z = a.Count;
```

Dieses Listing enthält 15 Fehler!

Dieses Programm soll ein Warnsystem für Überflutungen sein. Es verwaltet Stationen entlang von Flüssen. Alle Stationen können lokal warnen; einige Stationen können obendrein den örtlichen Pegelstand messen. Hinweis: Flusskilometer werden von der Quelle hin zur Mündung gezählt, also flussabwärts.

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          Pegelmessstation kölnerPegel = new Pegelmessstation(
6              new Geokoordinaten(50.93945, 6.96663),
7              new PositionAmFluss(PositionAmFluss.Fluss.Rhein, 681.9),
8              10.0);
9          Station duisburgerStation = new Station(
10             new Geokoordinaten(51.43240, 6.71362),
11             new PositionAmFluss(PositionAmFluss.Fluss.Rhein, 781.9));
12         Station bielefelderStation = new Station(
13             new Geokoordinaten(51.99530, 8.48721),
14             new PositionAmFluss(PositionAmFluss.Fluss.Lutter, 0.5));
15         kölnerPegel.FügePegelstandsmessungHinzu(13.0);
16
17         int x = duisburgerStation.AnzahlWarnungen;
18         int y = bielefelderStation.AnzahlWarnungen;
19         double z =
20             duisburgerStation.PositionAmFluss.EntfernungEntlangFluss(
21             kölnerPegel.PositionAmFluss);
22     }
23 }
24
25 abstract class Messung
26 {
27     zeitpunkt;
28
29     public Messung(DateTime zeitpunkt)
30     {
31         this.zeitpunkt = zeitpunkt;
32     }
33 }
34
35 class Pegelstandsmessung : Messung
36 {
37     double pegel;
38
39     public Pegelstandsmessung(DateTime zeitpunkt, double pegel)
40     : base(zeitpunkt)
41     {
42         this.pegel = pegel;
```

```
43      }
44  }
45
46  struct Geokoordinaten
47  {
48      double breitengrad;
49      double längengrad;
50
51      public double Geokoordinaten(double breitengrad, double längengrad)
52      {
53          this.breitengrad = breitengrad;
54          this.längengrad = längengrad;
55      }
56  }
57
58  class PositionAmFluss : Geokoordinaten
59  {
60      public enum Fluss { Rhein, Elbe, Lutter }
61
62      Fluss fluss;
63      double flusskilometer;
64
65      public PositionAmFluss(Fluss fluss, double flusskilometer)
66      {
67          this.fluss = fluss;
68          this.flusskilometer = flusskilometer;
69      }
70
71      public bool LiegtFlussabwärtsVon(PositionAmFluss p)
72      {
73          return fluss == p.fluss
74              && flusskilometer > p.flusskilometer;
75      }
76
77      public double EntfernungEntlangFluss(PositionAmFluss p)
78      {
79          if (fluss == p.fluss)
80          {
81              return double.NaN;
82          }
83          else
84          {
85              return Abs(flusskilometer - p.flusskilometer);
86          }
87      }
88  }
89
90  class Warnung
91  {
92      DateTime zeitpunkt;
93      string grund;
```

```
94
95      public Warnung(DateTime zeitpunkt, string grund)
96      {
97          this.zeitpunkt = zeitpunkt;
98          this.grund = grund;
99      }
100 }
101
102 abstract class Station
103 {
104     static List<Station> stationen = new List<Station>();
105
106     Geokoordinaten geokoordinaten;
107
108     PositionAmFluss positionAmFluss;
109     public PositionAmFluss PositionAmFluss
110         { get { return positionAmFluss; } }
111
112     List<Messung> messungen = new List<Messung>();
113
114     List<Warnung> warnungen = new List<Warnung>();
115     public int AnzahlWarnungen
116         { set { return warnungen.Count;  } }
117
118     public Station(Geokoordinaten geokoordinaten,
119         PositionAmFluss positionAmFluss)
120     {
121         stationen.Add(this);
122
123         this.geokoordinaten = geokoordinaten;
124         this.positionAmFluss = positionAmFluss;
125     }
126
127     void Warne(grund)
128     {
129         warnungen.Add(new Warnung(DateTime.Now, grund));
130
131         // und passende Befehle zum Einschalten der Sirene,
132         // hier weggelassen
133     }
134
135     public void WarneAlleStationenFlussabwärts()
136     {
137         foreach (station in stationen)
138         {
139             if (station.LiegtFlussabwärtsVon(
140                 positionAmFluss))
141             {
142                 station.Warne("Hochwasser!");
143             }
144         }
```

```
145     }
146 }
147
148 class Pegelmessstation : Station
149 {
150     double kritischerPegel;
151
152     public Pegelmessstation(Geokoordinaten geokoordinaten,
153         PositionAmFluss positionAmFluss,
154         double kritischerPegel)
155     : base(positionAmFluss)
156     {
157         this.kritischerPegel = kritischerPegel;
158     }
159
160     public void FügePegelstandsmessungHinzu(double pegel)
161     {
162         messungen.Add(Pegelstandsmessung(DateTime.Now, pegel));
163
164         if (this.pegel > kritischerPegel)
165         {
166             WarneAlleStationenFlussabwärts;
167         }
168     }
169 }
```