Informatik 2 für Regenerative Energien

Klausur vom 8. Juli 2022

Jörn Loviscach

Versionsstand: 7. Juli 2022, 22:06

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.

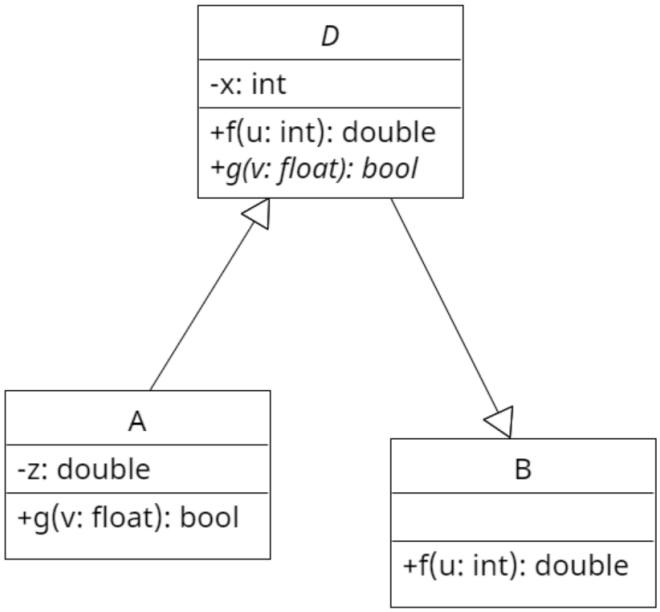
3. Wenn die Methode `SetzePosition` einer Instanz der Klasse `Messwert` zum zweiten Mal aufgerufen wird, soll sie eine Exception werfen. Was ändern Sie an dazu am (korrigierten) Code aus dem Programmlisting?

4. Ergänzen Sie die Klasse `Luftbelastungsdatenbank` des korrigierten Code aus dem Programmlisting um eine öffentliche Methode `double BestimmeMaximumVonCO2()`. Diese Methode soll den größten in den Daten vorhandenen CO_2-Gehalt liefern. Falls es keinen einzigen CO_2-Messwert gibt, soll sie `double.NaN` liefern. Was ändern Sie dazu wie an dem (korrigierten) Code aus dem Programmlisting?

5. Leiten Sie von der Klasse `Messstation` eine Klasse `MobileMessstation` ab, deren Position man mittels einer entsprechenden öffentlichen Methode verändern kann. Schreiben Sie den Code dieser Klasse `MobileMessstation` auf. Was ist außerdem am korrigierten Code der Klasse `Messstation` aus dem Programmlisting zu ändern?

6. Ergänzen Sie die Klasse `CO2Sensor` des korrigierten Code aus dem Programmlisting um eine öffentliche Methode `int ZähleÜberschreitungenDerVergangenen24h()`. Diese soll bestimmen, wie viele Aufrufe der Methode `GibAktuellenWert` in den vergangenen 24-Stunden einen Wert größer oder gleich dem `limitFürWarnungen` ergeben haben. Was muss dazu obendrein an dieser Klasse `CO2Sensor` geändert werden? Hinweis: Eine `TimeSpan` von 24 Stunden erhalten Sie mit `TimeSpan.FromHours(24.0)`.

7. Schreiben Sie compilierbaren C#-Code für die drei Klassen dieses UML-Diagramms:



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<Queue<int>> a = new List<Queue<int>>();
List<Queue<int>> b = new List<Queue<int>>();
Queue<int> c = new Queue<int>();
Queue<int> d = c;
Queue<int> e = new Queue<int>();
c.Enqueue(11);
d.Enqueue(12);
e.Enqueue(13);
a.Add(c);
a.Add(d);
a.Add(e);
b.Add(c);
b[0].Enqueue(14);
int x = a[1].Dequeue();
int y = a[2].Dequeue();
int z = b[0].Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dieses Programm soll eine Datenbank für Messungen zur Luftbelastung sein. An jeder Messstation kann es viele Sensoren geben, auch mehrfach welche vom selben Typ. 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          Geokoordinaten g1 = new Geokoordinaten(52.015, 8.527);
6          Messstation ms1 = new Messstation("Sparrenburg", g1);
7          CO2Sensor s11 = new CO2Sensor(500.0);
8          Sensor s12 = new CO2Sensor(500.0);
9          ms1.FügeSensorHinzu(s11);
10         ms1.FügeSensorHinzu(s12);
11         Messstation ms2 = new Messstation("Johannisberg",
12                                           new Geokoordinaten(52.019, 8.519)
13                                          );
14
15         Messstation stationen = new Messstation[2];
16         stationen[0] = ms1;
17         stationen[1] = ms2;
18         Luftbelastungsdatenbank ld = new Luftbelastungsdatenbank(stationen);
19
20         ld.MissUndSpeichereAlle();
21
22         int x = CO2Sensor.BestimmeDieZahlAllerCO2Messungen();
23         int y = ld.BestimmeZahlDerMessungenNahe(g1);
24         int z = ms1.ZahlDerWarnungen();
25     }
26 }
27
28 class Messstation
29 {
30     string name;
31     Geokoordinaten position;
32     List<Sensor> sensoren = new List<Sensor>();
33     int zahlDerWarnungen;
34     int ZahlDerWarnungen { get { return zahlDerWarnungen; } }
35
36     public Messstation(string name, Geokoordinaten position)
37     {
38         this.name = name;
39         this.position = position;
40     }
41
42     public void FügeSensorHinzu(sensor)
43     {
44         sensoren.Add(sensor);
45         sensor.Station = this;
```

```
46      }
47
48      public Messwert[] GibAlleAktuellenWerte()
49      {
50          Messwert[] m = new Messwert[sensoren.Count];
51          for (int i = 0; i < sensoren.Count; i++)
52          {
53              m[i] = sensoren[i].GibAktuellenWert;
54              m[i].SetzePosition(position);
55          }
56          return m;
57      }
58
59      public void Warne()
60      {
61          zahlDerWarnungen++;
62          // Hier stünde Code, um Warnungen rauszuschicken.
63      }
64  }
65
66  struct Geokoordinaten
67  {
68      public double Länge;
69      public double Breite;
70
71      public Geokoordinaten(double länge, double breite)
72      {
73          Länge = länge;
74          Breite = breite;
75      }
76
77      public bool LiegtNaheBei(Geokoordinaten g)
78      {
79          return Math.Abs(Länge - g.Länge) < 0.001
80                  || Math.Abs(Breite - g.Breite) < 0.001;
81      }
82  }
83
84  enum Sensortyp { CO2, Ozon, Feinstaub }
85
86  class Sensor
87  {
88      Sensortyp sensortyp;
89      Messstation messstation;
90      public Messstation Station
91      {
92          get { return messstation; }
93          set { messstation = value; }
94      }
95
96      static Random würfel = new Random();
```

```
97
98      public Sensor(Sensortyp sensortyp)
99      {
100         this.sensortyp = sensortyp;
101     }
102
103     public abstract Messwert GibAktuellenWert();
104 }
105
106 class CO2Sensor : Messstation
107 {
108     double limitFürWarnungen;
109     static zahlAllerCO2Messungen;
110
111     public CO2Sensor(double limitFürWarnungen)
112         : base(Sensortyp.CO2)
113     {
114         this.limitFürWarnungen = limitFürWarnungen;
115     }
116
117     public Messwert GibAktuellenWert()
118     {
119         zahlAllerCO2Messungen++;
120
121         // Generiere zum Ausprobieren zufällige Werte,
122         // NextDouble liefert eine Zahl von 0.0 bis 1.0
123         wert = 500.0 + 100.0 * würfel.NextDouble();
124
125         if(wert >= limitFürWarnungen && Station == null)
126         {
127             Station.Warne();
128         }
129
130         return Messwert(wert, Sensortyp.CO2, DateTime.Now);
131     }
132
133     static public int BestimmeDieZahlAllerCO2Messungen()
134     {
135         return zahlAllerCO2Messungen;
136     }
137 }
138
139 class Messwert
140 {
141     double wert;
142                 Sensortyp sensortyp;
143     DateTime zeitpunkt;
144     Geokoordinaten position;
145     bool hatPositionsangabe;
146
147     public Messwert(double wert, Sensortyp sensortyp, DateTime zeitpunkt)
```

```
148     {
149         this.wert = wert;
150         this.sensortyp = sensortyp;
151         this.zeitpunkt = zeitpunkt;
152     }
153 
154     public void SetzePosition(Geokoordinaten g)
155     {
156         position = g;
157         hatPositionsangabe = true;
158     }
159 
160     public LiegtNaheBei(Geokoordinaten g)
161     {
162         if (hatPositionsangabe)
163         {
164             return position.LiegtNaheBei(g);
165         }
166         else
167         {
168             return false;
169         }
170     }
171 }
172 
173 class Luftbelastungsdatenbank
174 {
175     List<Messwert> messwerte = new List<Messwert>();
176     Messstation[] messstationen;
177 
178     public Luftbelastungsdatenbank(Messstation[] messstationen)
179     {
180         this.messstationen = messstationen;
181     }
182 
183     public void MissUndSpeichereAlle()
184     {
185         foreach (Messstation messstation in messstationen)
186         {
187             // AddRange: zu der Liste alle Elemente der
188             // angegebenen Sammlung hinzufügen
189             messwerte.AddRange(messstation.GibAlleAktuellenWerte());
190         }
191     }
192 
193     public int BestimmeZahlDerMessungenNahe(Geokoordinaten g)
194     {
195         return messwerte.Count(mw => mw.LiegtNaheBei(g));
196     }
197 }
```