Informatik 2 für Regenerative Energien

Klausur vom 7. Juli 2023

Jörn Loviscach

Versionsstand: 7. Juli 2023, 08:39

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 `SetzeInRoller` einer Instanz der Klasse `Wechselakku` aufgerufen wird, obwohl der Akku noch einem anderen Roller steckt, soll sie eine Exception werfen. Was ändern Sie an dazu am (korrigierten) Code aus dem Programmlisting?

4. Wie könnte man im korrigierten Code aus dem Programmlisting damit umgehen, dass die aktuellen Geokoordinaten des `ERoller`s vielleicht gar nicht bekannt sind, zum Beispiel wegen schlechten GPS-Empfangs? Beschreiben Sie in etwa zwei Sätzen eine Möglichkeit, das im Programm zu berücksichtigen.

5. Es gibt die drei Herstellerfirmen `A`, `B`, `C` von `Wechselakku`s. Speichern Sie mit Hilfe einer Enumeration im `Wechselakku`, welche Firma ihn produziert hat. Was ist am korrigierten Code aus dem Programmlisting zu ändern?

6. Ergänzen Sie die Klasse `ERollerVerwaltung` des korrigierten Code aus dem Programmlisting um eine öffentliche Methode `List<ERoller> FindeZuLadende(Geokoordinaten position)`. Diese Methode soll eine Liste aller `ERoller` im Radius von 100 m um die angegebene `position` zurückgeben, die keinen Akku haben oder aber einen Ladezustand von unter 50 % haben. Was muss dazu obendrein an der Klasse `ERoller` geändert werden?

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

```
abstract class A
{
    int u;
    public abstract double f(double x);
}

class B : A
{
    int v;
    public override double f(double x)
    {
        return x + 2.3;
    }
}

class C : B
{
    public override double f(double x)
    {
        return x + 4.2;
    }
    public int g(int y)
    {
        return y + 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.

```
Stack<List<int>> a = new Stack<List<int>>();
a.Push(new List<int>());
Stack<List<int>> b = new Stack<List<int>>();
List<int> c = new List<int>();
c.Add(10);
c.Add(20);
b.Push(c);
b.Push(c);
c.Add(30);
b.Push(a.Pop());
int x = a.Count;
int y = b.Pop().Count;
b.Pop();
int z = b.Pop()[1];
```

Dieses Listing enthält 15 Fehler!

Dieses Programm soll eine Verwaltungssoftware für ein E-Roller-Vermietungsunternehmen sein. 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          BekannteOrte.FügeHinzu(new Geokoordinaten(52.0281, 8.5225, "Siggi"));
6          BekannteOrte.FügeHinzu(new Geokoordinaten(52.0439, 8.4921, "Campus"));
7          Wechselakku w1 = Wechselakku(100.0);
8          ERollerMitWechselakku eRoller = new ERollerMitWechselakku("ABC123",
9                                  w1, DateTime.Now, BekannteOrte.Finde("Siggi"));
10         ERollerVerwaltung verwaltung = new ERollerVerwaltung();
11         FügeERollerHinzu(eRoller);
12         bool x = eRoller.IstFahrbereit();
13         eRoller.IstBelegt = true;
14         eRoller.Update(new Geokoordinaten(52.0372, 8.5123), 98.0);
15         eRoller.IstBelegt = false;
16         eRoller.TauscheAkkuGegen(new Wechselakku(100.0));
17         double y = w1.Ladezustand;
18         string? z = eRoller.Position.Ortsname;
19     }
20 }
21
22 class ERoller
23 {
24     public string Kennzeichen { get; private set; }
25
26     protected Akku? akku; // könnte in Kindklassen null sein
27     public const double MinimalerLadezustandFürFahrtbeginn = 15.0; // Prozent
28     public DateTime LetzteWartung { get; private set; }
29     public static TimeSpan Wartungsintervall = new TimeSpan(180, 0, 0, 0);
30                                                                 // 180 Tage
31     public Geokoordinaten Position { get; protected set; }
32     public bool IstBelegt { get; set; }
33
34     public ERoller(string kennzeichen, DateTime letzteWartung,
35                    Geokoordinaten position)
36     {
37         Kennzeichen = kennzeichen;
38         LetzteWartung = letzteWartung;
39         Position = position;
40         akku = new Akku(100.0);
41     }
42
43     public virtual bool IstFahrbereit()
44     {
45         return Ladezustand > MinimalerLadezustandFürFahrtbeginn
46             && DateTime.Now - LetzteWartung < Wartungsintervall;
```

```
47      }
48
49      // Diese Methode wird bei der Fahrt laufend mit frischen Daten aufgerufen.
50      public int Update(Geokoordinaten position, double ladezustand)
51      {
52          Position = position;
53          akku.Ladezustand = ladezustand;
54      }
55
56      public void WartungDurchführen()
57      {
58          LetzteWartung = DateTime.Now;
59      }
60  }
61
62  class ERollerMitWechselakku : ERoller
63  {
64      public ERollerMitWechselakku(string kennzeichen, Wechselakku? akku,
65                                   DateTime letzteWartung, Geokoordinaten position)
66          : base(kennzeichen, letzteWartung)
67      {
68          this.akku = akku;
69      }
70
71      public override bool IstFahrbereit()
72      {
73          return akku != null && base.IstFahrbereit();
74      }
75
76      public override void TauscheAkkuGegen(Wechselakku neuerAkku)
77      {
78          if (akku != null)
79          {
80              ((Wechselakku)akku).EntferneAusRoller();
81          }
82          neuerAkku.SetzeInRoller(this);
83          akku = neuerAkku;
84      }
85  }
86
87  abstract class Akku
88  {
89      double Ladezustand { get; set; } // Prozent
90
91      public Akku(double ladezustand)
92      {
93          Ladezustand = ladezustand;
94      }
95  }
96
97  class Wechselakku
```

```
98  {
99      private ERoller? roller;
100
101     public Wechselakku(double ladezustand)
102         : base(ladezustand)
103     {
104     }
105
106     public void EntferneAusRoller()
107     {
108         roller = null;
109     }
110
111     public void SetzeInRoller(ERoller roller)
112     {
113         this.roller = roller;
114     }
115 }
116
117 class Geokoordinaten
118 {
119     public double Breitengrad { get; private set; }
120     public double Längengrad { get; private set; }
121
122     // kann null sein, falls nicht bekannt oder Ort nicht benannt
123     public string? Ortsname { get; private set; }
124
125     public Geokoordinaten(double breitengrad, double längengrad)
126     {
127         Breitengrad = breitengrad;
128         Längengrad = längengrad;
129     }
130
131     public Geokoordinaten(double breitengrad, double längengrad, string ortsname)
132     {
133         Breitengrad = breitengrad;
134         Längengrad = längengrad;
135         Ortsname = ortsname;
136     }
137
138     public double EntfernungZu(Geokoordinaten andererOrt) // in km, Näherung
139     {
140         // 111 km pro Breitengrad und bei uns 72 km pro Längengrad
141         Math.Sqrt(Math.Pow(111*(Breitengrad-andererOrt.Breitengrad), 2)
142                   + Math.Pow(72*(Längengrad-andererOrt.Längengrad), 2));
143     }
144 }
145
146 class BekannteOrte
147 {
148     static List<Geokoordinaten> orte = new List<Geokoordinaten>();
```

```
149
150     static public void FügeHinzu(Geokoordinaten ort)
151     {
152         orte.Add(ort);
153     }
154
155     static public Geokoordinaten? Finde(name) // null, falls nicht gefunden
156     {
157         return orte.Find(ort => ort.Ortsname == name);
158     }
159 }
160
161 class ERollerVerwaltung
162 {
163     List<ERoller> rollerflotte = new List<ERoller>;
164
165     public void FügeERollerHinzu(ERoller roller)
166     {
167         rollerflotte.Add(roller);
168     }
169
170     public int BestimmeAnzahlBelegterERoller()
171     {
172         return rollerflotte.Count(roller => roller.IstBelegt);
173     }
174
175     public List<ERoller> BestimmeRollerDieGewartetWerdenMüssen()
176     {
177         return rollerflotte.FindAll(roller =>
178             DateTime.Now - roller.LetzteWartung >= ERoller.Wartungsintervall);
179     }
180
181     public ERoller? FindeNächstliegendenFreienERoller(Geokoordinaten position)
182     {
183         double entfernung = 2.3;
184         ERoller? nächsterFreierERoller = null;
185         foreach (ERoller roller in rollerflotte)
186         {
187             if (!roller.IstBelegt && roller.IstFahrbereit())
188             {
189                 double neueEntfernung = roller.Position.EntfernungZu(position);
190                 if (neueEntfernung > entfernung)
191                 {
192                     entfernung = neueEntfernung;
193                     nächsterFreierERoller = roller;
194                 }
195             }
196         }
197
198     }
199 }
```