Informatik 2 für Regenerative Energien

Klausur vom 31. Januar 2014

Jörn Loviscach

Versionsstand: 30. Januar 2014, 23:32

15 Punkte für die erste Aufgabe; 3 Punkte für alle weiteren Aufgaben. Mindestpunktzahl zum Bestehen: 15 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, 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. Stellen Sie eine Liste dieser Art mit allen Fehlern auf:

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

2. Mit dem (korrigierten) Code aus dem Programmlisting im Anhang wird die Methode `Test.Teste()` ausgeführt. Welche Zeichenkette steht in der Variablen `s` in der Zeile 13? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. Am Anfang der Methode `BeladeFach` von `Verkaufsautomat` aus dem (korrigierten) Listing im Anhang soll eine Abfrage eingefügt werden, die sicherstellt, dass das Fach mit dieser Nummer überhaupt vorhanden ist und dass die angegebene Anzahl nicht null oder negativ ist. Andernfalls soll eine Exception geworfen werden.

4. Erweitern Sie in die Klasse `Fach` aus dem (korrigierten) Listing im Anhang um eine öffentliche Methode `BerechneVolumenImFach`, die aus den Abmessungen und der Anzahl von Packungen das im Fach belegte Volumen berechnet und zurückgibt. Wenn noch keine `Sorte` definiert ist, soll als Volumen der Wert 0.0 zurückgegeben werden.

5. Schreiben Sie für das (korrigierte) Listing im Anhang eine Klasse `VerderblicherArtikel`, die von `Artikelsorte` erbt, als zusätzliches Attribut die Haltbarkeit enthält (vom Typ `TimeSpan`) und einen Konstruktor hat, der neben den Angaben des Konstruktors von `Artikelsorte` auch die Haltbarkeit als Parameter hat. Die Methode `Beschreibe` soll für diese Klasse die Zeichenkette `"Verderblich"` an den Artikelnamen hängen. Gegebenenfalls sind auch an der Klasse `Artikelsorte` Änderungen nötig.

6. Geben Sie an, was man am (korrigierten) Listing aus dem Anhang ändern muss, um dies zu verwirklichen: Wenn aus einem Fach die letzte Packung entnommen worden ist, wird die Sorte für das Fach wieder auf `null` gestellt.

7. Zeichnen Sie ein UML-Diagramm: Es gibt die drei Klassen `Bestellung`, `Eilbestellung` und `Kunde`. Zeichnen Sie sinnvolle Vererbungen ein, soweit möglich. Verwenden Sie außerdem an geeigneter Stelle das Attribut `besteller` und die parameterlose Methode `versenden`. Diese Methode soll in einer der Klassen überschrieben sein. Damit Kursivschrift (falls nötig) zu erkennen ist, umkringeln Sie diese oder benutzen Sie eine andere Farbe dafür.

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.

```
Queue<int> a = new Queue<int>();
Stack<Queue<int>> b = new Stack<Queue<int>>();
a.Enqueue(1);
a.Enqueue(2);
a.Enqueue(3);
b.Push(new Queue<int>());
b.Push(a);
b.Push(a);
int x = b.Pop().Dequeue();
int y = a.Count;
int z = b.Pop().Dequeue();
```

Dieses Listing enthält 15 Fehler!

Dies soll ein Programm werden, das einen Verkaufsautomaten verwaltet, wie er zum Beispiel für Chips und Gummibären auf dem Bahnsteig steht. Die Methode `Teste` macht die Handhabung der Klassen vor.

```
1  class Test
2  {
3      public static void Teste()
4      {
5          Artikelsorte a = new Artikelsorte("Chips", 8.0, 11.0, 4.0);
6          Artikelsorte b = new Artikelsorte("Gummibären", 8.0, 11.0, 2.0);
7          Verkaufsautomat v = new Verkaufsautomat(12);
8          v.BeladeFach(3, 4, a);
9          v.BeladeFach(7, 2, b);
10         v.EntnehmePackung(3);
11         while (v.EntnehmePackung(7))
12         {}
13         string s = v.SammeleInhaltsangabe();
14     }
15 }
16
17 class Verkaufsautomat
18 {
19     List fächer;
20
21     public Verkaufsautomat(int zahlDerFächer)
22     {
23         fächer = new List<Fach>();
24         for (int i = 0; i < zahlDerFächer; i++)
25         {
26             fächer.Add(Fach());
27         }
28     }
29
30     // Der Rückgabewert ist false, wenn das Beladen fehlschlägt.
31     public bool BeladeFach(int nummerDesFachs, int anzahl,
32                                         Artikelsorte sorte)
33     {
34         return fächer[nummerDesFachs].FügeHinzu(anzahl);
35     }
36
37     // Der Rückgabewert ist false, wenn das Entnehmen fehlschlägt.
38     public bool EntnehmePackung(int nummerDesFachs)
39     {
40         return fächer[nummerDesFachs].EntferneEinen();
41     }
42
43     public string SammeleInhaltsangabe()
44     {
45         string resultat = "";
46         for(int i = 0; i < fächer.Count; i++)
```

```
47          {
48              string s = fächer[i].Beschreibe;
49              if (s != "")
50              {
51                  if (resultat != "")
52                  {
53                      resultat += ",␣";
54                  }
55                  resultat += s;
56              }
57          }
58          return;
59      }
60  }
61
62  class Fach : Packung
63  {
64      Artikelsorte sorte;
65      List<Packung> packungen = new List<Packung>();
66      double tiefeDesFachs;
67      Abmessungen maximalePackungsgröße;
68
69      public Fach()
70      {
71          maximalePackungsgröße = new Abmessungen(10.0, 12.0, 5.0);
72          tiefeDesFachs = 30.0;
73      }
74
75      public int Beschreibe()
76      {
77          if (packungen.Count > 0)
78          {
79              return packungen.Count + "␣mal␣" + sorte.Beschreibe();
80          }
81          return "";
82      }
83
84      // Der Rückgabewert ist false, wenn das Beladen fehlschlägt.
85      public bool FügeHinzu(int anzahl, Artikelsorte sorte)
86      {
87          if (packungen.Count != 0 && this.sorte != sorte
88                      // nicht verschiedene Sorten in einem Fach
89              || sorte.Größe.PasstIn(maximalePackungsgröße)
90              || (packungen.Count() + anzahl) * sorte.Größe.Tiefe
91                                                  > tiefeDesFachs)
92          {
93              return false;
94          }
95
96          this.sorte = sorte;
97          for (int i = 0; i < anzahl; i++)
```

```
98          {
99              Add(new Packung());
100         }
101         return false;
102     }
103
104     // Der Rückgabewert ist false, wenn das Entnehmen fehlschlägt.
105     public bool EntferneEinen()
106     {
107         if (packungen.Count < 0)
108         {
109             packungen.RemoveAt(0);
110             return true;
111         }
112         return false;
113     }
114 }
115
116 class Packung
117 {
118     DateTime eingelegtAm;
119
120     public Packung()
121     {
122         eingelegtAm = DateTime.Now;
123     }
124 }
125
126 class Artikelsorte
127 {
128     string artikelName;
129     Abmessungen abmessungen;
130     public Abmessungen Größe { get { return abmessungen; } }
131
132     public Artikelsorte(string artikelName,
133                         double breite, double höhe, double tiefe)
134     {
135         artikelName = artikelName;
136         abmessungen = new Abmessungen(breite, höhe, tiefe);
137     }
138
139     public static string Beschreibe()
140     {
141         return artikelName;
142     }
143 }
144
145 class Abmessungen
146 {
147     Abmessungen(double breite, double höhe, double tiefe)
148     {
```

```
149             this.breite = breite;
150             this.höhe = höhe;
151             this.tiefe = tiefe;
152         }
153         double breite;
154         public double Breite { get { return breite; } }
155         double höhe;
156         public double Höhe { get { return höhe; } }
157         double tiefe;
158         public double Tiefe { get { return tiefe; } }
159 
160         public bool PasstIn(Abmessungen a)
161         {
162             breite <= a.breite && höhe <= a.höhe && tiefe <= a.tiefe;
163         }
164     }
```