Informatik 2 für Regenerative Energien

Klausur vom 28. September 2016

Jörn Loviscach

Versionsstand: 28. September 2016, 10:58

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. Was steht am Ende in den Variablen `x`, `y` und `z`? Beschreiben Sie gegebenenfalls, wie Sie zu Ihrer Antwort kommen.

3. Noch ist es mit dem korrigierten Code aus dem Programmlisting möglich, dass dieselbe Person zwei Exemplare desselben Buchs ausleiht. Verhindern Sie das mit einer Exception.

4. In der Methode `NehmeZurück` der Klasse `Buch` des korrigierten Code aus dem Programmlisting ist es möglich, dass die Person, die das Buch zurückgibt, nicht diejenige ist, die das Buch ausgeliehen hat. Ist das ein Problem? Falls ja, beschreiben Sie grob (ca. drei Sätze) eine Möglichkeit, das Programm diesbezüglich zu verbessern.

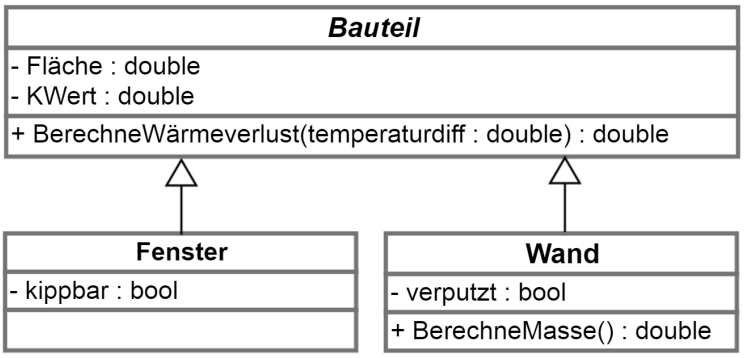
5. Schreiben Sie für die Klasse `Bibliothek` im korrigierten Code aus dem Anhang diese öffentliche Methode:

   ```
   List<string> SucheKomplettEntlieheneBücher()
   ```

   Sie soll eine Liste der Titel aller Buchtitel zurückgeben, von denen aktuell alle Exemplare ausgeliehen sind. Gegebenenfalls sind in anderen Klassen Änderungen nötig.

6. Schreiben Sie von der Klasse `Ausleihvorgang` im korrigierten Code aus dem Anhang eine abgeleitete Klasse `UnbefristeterAusleihvorgang`. Hier soll die Ausleihe beliebig lang sein dürfen. Gegebenenfalls sind auch in der Klasse `Ausleihvorgang` Änderungen nötig. Hinweis: `int.MaxValue` ist die größtmögliche `int`-Zahl und `DateTime.MaxValue` ist der Zeitpunkt, der nach allen anderen Zeitpunkten liegt.

7. Schreiben Sie ein fehlerfrei compilierbares C#-Programm, das diesem Diagramm entspricht. Die Methoden geben, wenn nötig, Phantasiewerte zurück.



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

Dieses Listing enthält 15 Fehler!

Die soll ein Programm zur Verwaltung der Ausleihen einer Bibliothek sein. Von jedem Buch kann es mehrere Exemplare geben. Wenn man ein Buch vormerkt, muss man dazu nicht ein bestimmtes Exemplar des Buchs wählen.

```
1  class Test
2  {
3      public static void Teste()
4      {
5          Buch b1 = new Buch("Elektrotechnik", 10);
6          Buch b2 = new Buch("Englisch", 1);
7          Buch b3 = new NichtVerlängerbaresBuch("Chemie", 2);
8          Bibliothek bib = new Bibliothek(3);
9          bib.FügeBuchHinzu(b1);
10         bib.FügeBuchHinzu(b2);
11         bib.FügeBuchHinzu(b3);
12         Person p1 = new Person("Anton");
13         Person p2 = new Person("Berta");
14         Person p3 = new Person("Carla");
15         b2.LeiheAusFallsMöglich(p1);
16         bool x = b2.LeiheAusFallsMöglich(p2);
17         b1.MerkeVor(p1);
18         b2.MerkeVor(p3);
19         bool y = b2.VerlängereFallsMöglich(p1);
20         b2.NehmeZurück(p1);
21         string z = bib.FindeErfüllbareReservierungen();
22     }
23 }
24
25 abstract class Buch
26 {
27     static string titel;
28     public string Titel { get { return titel; } }
29     int anzahlExemplare;
30     tageLeihfrist = 30;
31     List<Ausleihvorgang> ausleihen = new List<Ausleihvorgang>();
32     List<Person> vormerkungen = new List<Person>();
33
34     public void Buch(string titel, int anzahlExemplare)
35     {
36         this.titel = titel;
37         this.anzahlExemplare = anzahlExemplare;
38     }
39
40     public bool LeiheAusFallsMöglich(Person person)
41     {
42         List<Person> erfüllbar = FindeErfüllbareReservierungen();
43         if(erfüllbar.Contains(person))
44         {
45             ausleihen.Add(new Ausleihvorgang(person, tageLeihfrist));
46             vormerkungen.Remove(person);
```

```
47              return true;
48          }
49          else if(ausleihen.Count + vormerkungen.Count < anzahlExemplare)
50          {
51              ausleihen.Add(new Ausleihvorgang(person, tageLeihfrist));
52              return true;
53          }
54          else
55          {
56              return false;
57          }
58      }
59
60      // Nur dann verlängerbar, wenn genügend Exemplare nicht vorgemerkt.
61      public virtual bool VerlängereFallsMöglich(Person person)
62      {
63          if (anzahlExemplare >= ausleihen.Count + vormerkungen.Count)
64          {
65              foreach (var ausleihe in ausleihen)
66              {
67                  if (ausleihe.DiePerson == person)
68                  {
69                      ausleihe.Verlängere(tageLeihfrist);
70                      return true;
71                  }
72              }
73              return false;
74          }
75          else
76          {
77              return false;
78          }
79      }
80
81      public void NehmeZurück(Person person)
82      {
83          Ausleihvorgang a = 0;
84          foreach (var ausleihe in ausleihen)
85          {
86              if(ausleihe.DiePerson = person)
87              {
88                  a = ausleihe;
89                  break;
90              }
91          }
92
93          if(a == null)
94          {
95              ausleihen.Remove(a);
96          }
97      }
```

```
98
99      public void MerkeVor(Person person)
100     {
101         vormerkungen.Add(person);
102     }
103
104     public void EntferneVormerkung(Person person)
105     {
106         vormerkungen.Remove(person);
107     }
108
109     public List<Person> FindeErfüllbareReservierungen()
110     {
111         List<Person> erfüllbar = List<Person>();
112
113         for (int i = 0; i < vormerkungen.Count
114                 && i < anzahlExemplare - ausleihen.Count; i++)
115         {
116             erfüllbar.Add(vormerkungen[i]);
117         }
118
119         return;
120     }
121 }
122
123 class NichtVerlängerbaresBuch : Buch
124 {
125     public NichtVerlängerbaresBuch(string titel, int anzahlExemplare)
126         : base(titel, anzahlExemplare)
127     { }
128
129     public bool VerlängereFallsMöglich(Person person)
130     {
131         return false;
132     }
133 }
134
135 class Person
136 {
137     string name;
138     public string Name { get { return name; } }
139
140     public Person(string name)
141     {
142         this.name = name;
143     }
144 }
145
146 class Ausleihvorgang
147 {
148     Person person;
```

```
149     public Person DiePerson { get { return person; } }
150     DateTime endeLeihfrist;
151     public DateTime EndeLeihfrist { get { return endeLeihfrist; } }
152 
153     public Ausleihvorgang(Person person, int tageLeihfrist)
154     {
155         this.person = person;
156         endeLeihfrist = DateTime.Now.AddDays(tageLeihfrist); // korrekt
157     }
158 
159     public void Verlängere(int tage)
160     {
161         endeLeihfrist = endeLeihfrist.AddDays(tage);
162     }
163 }
164 
165 class Bibliothek
166 {
167     List<Buch> bücher = new List<Buch>;
168 
169     public void FügeBuchHinzu(buch)
170     {
171         Add(buch);
172     }
173 
174     public string FindeErfüllbareReservierungen()
175     {
176         string resultat = "";
177 
178         foreach (buch in bücher)
179         {
180             List<Person> erfüllbar = buch.FindeErfüllbareReservierungen();
181             foreach (var person in erfüllbar)
182             {
183                 resultat += person.Name + ":␣" + buch.Titel + ";␣";
184             }
185         }
186 
187         return resultat;
188     }
189 }
```