

Informatik 2 für Regenerative Energien

Klausur vom 23. September 2019: Lösungen

Jörn Loviscach

Versionsstand: 4. Februar 2022, 15:22



This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Germany License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/de/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

1. Die Fehler:

Zeile	korrekter Programmtext
27	<code>DateTime zeitpunkt;</code>
51	<code>public Geokoordinaten(double breitengrad, ...</code>
58	<code>class PositionAmFluss</code>
79	<code>if (fluss != p.fluss)</code>
85	<code>return Math.Abs(flussskilometer - p.fluss...</code>
102	<code>class Station</code>
112	<code>protected List<Messung> messungen = new ...</code>
116	<code>{ get { return warnungen.Count; } }</code>
127	<code>void Warne(string grund)</code>
137	<code>foreach (Station station in stationen)</code>
139	<code>if (station.positionAmFluss.LiegtFlussabwärtsVon(</code>
155	<code>: base(geokoordinaten, positionAmFluss)</code>
162	<code>messungen.Add(new Pegelstandsmessung(...</code>
164	<code>if (pegel > kritischerPegel)</code>
166	<code>WarneAlleStationenFlussabwärts();</code>

2. x ist 1, weil der Kölner Pegel über dem kritischen Wert gelegen hat und flussabwärts Duisburg gewarnt worden ist. y ist 0, weil flussaufwärts an der Lutter keine Warnung gegeben worden ist. z ist 100.0 , weil dies der Betrag der Differenz der Flusskilometer ist.

3. *Zum Beispiel:* In der Klasse `Position` diese Property ergänzen:

```
public double Flusskilometer
    { get { return flussskilometer; } }
```

Und am Anfang des Konstruktors der Klasse `Station` dies ergänzen:

```
if (positionAmFluss.Flussskilometer > 3000.0)
```

```
{
throw new ApplicationException(
    "Unplausibel hohe Zahl an Flusskilometern.");
}
```

4. *Zum Beispiel:* In der Klasse `Messung` einen öffentlichen Getter für den Zeitpunkt und in der Klasse `Pegelstandsmessung` einen öffentlichen Getter für den Pegel ergänzen und dann die gefragte Methode so implementieren:

```
double BestimmeMaximalpegelSeit(DateTime wann)
{
    bool gefunden = false;
    double wert = double.MinValue;
    foreach (Messung m in messungen)
    {
        if (m.Zeitpunkt >= wann && m is Pegelstandsmessung)
        {
            gefunden = true;
            Pegelstandsmessung p = (Pegelstandsmessung)m;
            wert = Math.Max(wert, p.Pegel);
        }
    }

    if (gefunden)
    {
        return wert;
    }
    else
    {
        return double.NaN;
    }
}
```

Oder eleganter dies als Körper der obigen Funktion:

```
var ms = messungen.Where(m => m is Pegelstandsmessung
    && m.Zeitpunkt >= wann);
if (ms.Count() == 0)
{
    return double.NaN;
}
else
{
    return ms.Min(m => ((Pegelstandsmessung)m).Pegel);
}
```

5. *Zum Beispiel:*

```
class PegelUndFließgeschwindigkeitsmessung
  : Pegelstandsmessung
{
  double fließgeschwindigkeit;

  public PegelUndFließgeschwindigkeitsmessung(
    DateTime zeitpunkt, double pegel,
    double fließgeschwindigkeit)
    : base(zeitpunkt, pegel)
    {
      this.fließgeschwindigkeit = fließgeschwindigkeit;
    }
}
```

6. *Zum Beispiel:*

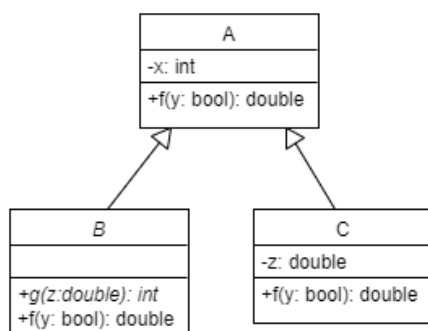
```
public static int BestimmeGesamtzahlWarnungen()
{
  int gesamtzahl = 0;
  foreach (Station station in stationen)
  {
    gesamtzahl += station.warnungen.Count;
  }
  return gesamtzahl;
}
```

Oder eleganter dies als Körper der obigen Funktion:

```
return stationen.Sum(s => s.warnungen.Count);
```

Diese Methode *kann* statisch gemacht werden, weil sie nur über die statische Variable `stationen` auf die Instanzen zugreift. Sie *sollte* statisch gemacht werden, weil sie keine bestimmte Instanz betrifft.

7. Das Diagramm:



8. Die Werte sind 3, 4, 1.