Test Modul II - C# Grundlagen

Aufgabe:	1	2	3	4	5	Summe
Punkte max.:	20	20	20	20	20	100
Punkte:						

Vom Teilnehmer auszufüllen

Name: _____

Datum:_____

Fehlersuche

In folgendem Listing sind 5 Fehler enthalten. Bitte markieren Sie die Stellen mit 1 bis 5 und notieren Sie welche das sind.

```
using System;

namespace Test
{
    class Berechne
    {
        private float flaeche;
        private float breite;

        public Berechne() { }
        public ~Berechne() { }

        public void SetzeWerte(float l, float b)
        {
            laenge = l;
            breite = b;
        }

        private float Flaechen()
        {
            return (laenge * breite)
        }
    }

    class Program {
        static void Main(string[] args) {
            Berechne rechteck;
            rechteck.Setze(20, 2);
            Console.WriteLine("Die Flaeche lautet: " + rechteck.Flaechen() );
        }
    }
}
```

Lösung:

1. Länge nicht deklariert
2. Es wird kein neues Objekt der Klasse Berechne instanziiert
3. Methode `Flaechen()` ist private und kann somit nicht aufgerufen werden
4. Methode `setze()` ist nicht implementiert
5. ";" fehlt bei `Flaechen()`

Was ist richtig

Kreuzen Sie alle **korrekten** Aussagen (4 Stück) an.

☐ Aus einer sealed markierten Klasse darf ich weiter ableiten.

☑ Von einer abstrakten Klasse darf ich keine Instanz (Objekt) erzeugen.

☐ Einem Array der Größe 5 kann nach dem Initialisieren eine beliebige Anzahl an Werte aufnehmen.

☑ Die main-Methode muss immer static sein.

☐ Eine void-Methode muss eine return-Anweisung haben.

☑ unsigned-Datentypen enthalten ca. doppelt so viele positiven Werte wie ihre signed-Pendanten.

☐ Konstruktoren dürfen keine Parameter enthalten.

☐ Konstruktoren haben immer den Rückgabewert void.

☑ Eine Methode darf durch eine gleichnamige Methode mit identischer Parameteranzahl, aber mit unterschiedlichen Parameterdatentypen überladen werden.

☐ Eine als protected deklarierte Variable kann von einer anderen, unabhängigen Klasse aufgerufen werden.

Erklären Sie

Erklären Sie kurz die Unterschiede.

1. Konstruktor vs Methode (mindestens drei Unterschiede).

2. Zugriffsmodifizierer private vs public.

Lösung:

1.
   - Ein Konstruktor kann nicht von hand aufgerufen werden
   - Ein Konstruktor darf/kann keinen Wert zurückliefern
   - Ein Konstruktor muss/sollte public sein
   - Ein Konstruktor muss gleich heißen wie die Klasse
   - Ein Konstruktor wird nur aufgerufen beim erstellen des Objektes

2.
   - Private Methoden oder Variablen sind nur innerhalb der eigenen Klasse aufrufbar
   - Public kann von überall aufgerufen werden

Programmieren

Implementieren Sie die Klasse **Wetter** zur Darstellung von Wetterdaten mit folgenden Eigenschaften:

- Attribute für Temperatur und Luftfeuchtigkeit (Keine Eigneschaften (Properties) notwendig)
- Ein Konstruktor zum setzen der beiden Werte
- Methode GetTempC welche die Temperatur in Celsius liefert
- Methode GetTempF welche die Temperatur in Fahrenheit liefert
- Methode GetLuftfeuchtigkeit die die Luftfeuchtigkeit zurück gibt

Formel:

- Celsius in Fahrenheit = ((TCelsius * 9) / 5) + 32
- Fahrenheit in Celsius = (TFahrenheit - 32) * 5 / 9

Schreiben Sie die nötigen Codezeilen um ein **Wetter**-Objekt zu instanziieren und die genannten Methoden mindestens einmal aufzurufen.

Es muss nur die Wetter Klasse voll ausprogrammiert werden. Namespaces und die Main Methode müssen nicht vollständig sein.

Lösung:

```
Wetter wetter = new Wetter(21, 80);
wetter.GetLuftfeuchtigkeit();
wetter.GetTempC();
wetter.GetTempF();

  class Wetter
  {
      private int temperatur; // in Celsius
      private int luftfeuchtigkeit;

      public Wetter(int temp, int luftf)
      {
          temperatur = temp;
          luftfeuchtigkeit = luftf;
      }

      public int GetLuftfeuchtigkeit()
      {
          return luftfeuchtigkeit;
      }

      public int GetTempC()
      {
          return temperatur;
      }

      public int GetTempF()
      {
          return ((temperatur * 9) /5 ) + 32;
      }
  }
```

Ausgaben

Welche Ausgaben werden bei den beiden Aufrufen erzeugt wenn diese in der Main Methode aufgerufen werden?

1. `Radfahrzeug auto = new Radfahrzeug(4);`
2. `Kettenfahrzeug panzer = new Kettenfahrzeug();`

```
class Fahrzeug
{
    public string motor;
    public string antrieb = "na";

    public Fahrzeug() : this("verbrenner") {
        Console.WriteLine("Fahrzeug");
    }

    public Fahrzeug(string motor) {
        Console.WriteLine("Fahrzeug {0}", motor);
    }
}

class Radfahrzeug : Fahrzeug
{
    public Radfahrzeug() {
        Console.WriteLine("Radfahrzeug");
    }

    public Radfahrzeug(int anzahlSitze) {
        Console.WriteLine($"Radfahrzeug Sitze: {anzahlSitze}");
    }

    private void Antrieb() {
        Console.WriteLine("Kette");
    }
}

class Kettenfahrzeug : Fahrzeug
{
    public Kettenfahrzeug() {
        this.Antrieb("Kette");
        Console.WriteLine("Kettenfahrzeug Antrieb " + antrieb);
    }

    private void Antrieb(string temp) {
        Console.WriteLine(antrieb);
        antrieb = temp;
    }
}
```

Lösung:

1.
   - Fahrzeug verbrenner
   - Fahrzeug
   - Radfahrzeug Sitze: 4

2.
   - Fahrzeug verbrenner
   - Fahrzeug
   - na
   - Kettenfahrzeug Antrieb Kette