Test

Kurs:	Software Developer IHK
Modul:	Programmieren in C# (Grundlagen)
Hilfsmittel:	Keine
Erstellungsdatum:	07.2019

Bewertung						
Vom Prüfer auszufüllen						
Aufgabe	1	2	3	4	5	Summe
Punkte max.	20	20	20	20	20	100
Punkte						
Bemerkungen						
Prüfer Datum / Unterschrift						

Vom Teilnehmer auszufüllen	
Name / Vorname	
Datum	

Aufgabe 1

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

- Attribute für Temperatur und Luftfeuchtigkeit
- Sinnvoller Satz an Konstruktoren mit Default-Parametern
- 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 eine **Main()**-Methode, in der ein **Wetter**-Objekt instanziiert und die genannten Methoden mindestens einmal aufgerufen werden.
Es muss nur die Wetter Klasse voll ausprogrammiert werden. Namespaces und die Main Methode müssen nicht vollständig sein.

Lösung:

```
namespace Test_Probe_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Wetter wetter = new Wetter(21, 80);
            Console.WriteLine("Luftfeuchtigkeit:␣" + wetter.
               GetLuftfeuchtigkeit());
            Console.WriteLine("Temp␣in␣C␣" + wetter.GetTempC());
            Console.WriteLine("Temp␣in␣F␣" + wetter.GetTempF());

            Console.ReadKey();
        }
    }

    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;
        }
    }
}
```

Aufgabe 2

Kreuzen Sie alle **korrekten** Aussagen an.

- ☑ Über eine Eigenschaft (Property) kann ich auf eine Variable einer Klasse zugreifen.
- ☐ Auf eine private Variable kann ich von allen Klassen zugreifen.
- ☐ Ein Array kann die Größe dynamisch verändern.
- ☑ Mit der Methode Replace der Klasse string kann ich Leerzeichen entfernen.
- ☐ Jede Methode muss einen Rückgabewert haben.
- ☑ Jede Klasse hat mindestens einen Konstruktor.
- ☐ Eine Klasse darf keine zwei Konstuktoren haben.
- ☑ Von einer mit sealed markierten Klasse darf ich nicht weiter ableiten.

Aufgabe 3

3.1 Was ist eine Collection? Vorteile, Nachteile...

Lösung:

- Eine Collection ist ein Container welcher Elemente von verschiedenen Typen oder auch nur vom gleichen Typ speichern kann.
- Es muss keine Größe beim erzeugen angegben werden.
- Eine Collection passt sich dynamisch an Änderungen an, beim hinzufügen oder entfernen werden autom. Indizes entfernt/hinzugefügt.
- Bei nicht typsicheren Collections müssen die Objekte in den passenden Datentyp konvertiert werden.

3.2 Welche Collections kennen Sie? (mind 3)

Lösung: ArrayList, List, Hashtable, Dictionary, HashSet, Queue, SortedList, Stack, ...

3.3 Schreiben Sie ein kleines Codefragement, wie Sie eine Collection erzeugen, diese mit Inhalt befüllen und danach den Inhalt ausgeben.

Lösung:

```
Dictionary<string, string> dictionary = new Dictionary<string,
    string>();

dictionary.Add("tier1", "Hase");
dictionary.Add("tier2", "Hund");
dictionary.Add("tier3", "Esel");

foreach (KeyValuePair<string, string> pair in dictionary)
{
          Console.WriteLine("Key:␣" + pair.Key + "␣Wert:␣" + pair.
              Value);
}
```

Aufgabe 4

In folgendem Listing sind 5 Fehler enthalten. Bitte markieren Sie die Stellen.

```
class Sekunden {
    private ulong sekunden;
    private ulong minutenInSek = 60;

    public Sekunden(ulong sek)  {
        sekunden = sek;
    }

    private ulong LeseSekunden()  {
        return sekunden;
    }

    public ulong LeseMinuten()  {
        return sekunden / minutenInSek;
    }

    public ulong LeseStunden()  {
        return sekunden / minutenInStd;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Sekunden sek;
        sek = new Sekunden(2222)

        Console.WriteLine("Aktuelle␣Sekunden␣sind:␣" +
           sek.LeseSekunden() + "In␣Minuten␣"  sek.
           LeseMin() + "␣In␣Stunden␣" + sek.LeseStunden()
            );
    }
}
```

Lösung:

- ";"bei new Sekunden(2222)
- LeseSekunden() ist private und kann nicht aus der Main aufgerufen werden
- minutenInStd ist nicht definiert

- sek.LeseMin() Methode existiert nicht
- + Operator fehlt vor sek.LeseMin

Aufgabe 5

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

a) RaeumlicheFiguren wuerfel = new RaeumlicheFiguren();
b) Dreieck dreieck = new Dreieck(4);

```
class GeometrischeFigur {
    public GeometrischeFigur() {
        Console.WriteLine("GeometrischeFigur");
    }

    public GeometrischeFigur(double flaeche) {
        Console.WriteLine("GeometrischeFigur␣" + flaeche);
    }
}

class EbeneFiguren : GeometrischeFigur {
    public EbeneFiguren() {
        Console.WriteLine("EbeneFiguren");
    }

    public EbeneFiguren(int flaeche) : this() {
        Console.WriteLine("EbeneFiguren␣" + flaeche);
    }

    public EbeneFiguren(double laengeSeiteA) {
        Console.WriteLine("EbeneFiguren␣" + laengeSeiteA);
    }
}

class RaeumlicheFiguren : GeometrischeFigur{
    public RaeumlicheFiguren() {
        Console.WriteLine("RaeumlicheFiguren");
    }
}

class Dreieck : EbeneFiguren {
    public Dreieck(double laengeSeiteA) : base (50.2) {
        Console.WriteLine("Dreieck␣" + laengeSeiteA);
     Output(laengeSeiteA);
    }

    public void Output(int laenge) {
     Console.WriteLine(laenge);
   }
}
```

Lösung:

a)

- GeometrischeFigur
- RaeumlicheFiguren

b)

- GeometrischeFigur
- EbeneFiguren 50,2
- Dreieck 4
- 4