7. Übungsblatt Thema: Vererbung und Exceptions

1. Implementierung von Rectangle

Beschreibung:

Es soll eine Klasse Rectangle implementiert werden. Diese soll von der Klasse Shape erben. Die Klasse Shape soll 2 Properties, Surface und Volume, als Implementierung verlangen aber sie nicht selber liefern.

Property Surface gibt die den Umfang einer Form, Shape, aus als Getter. Property Volume gibt die Fläche/Volumen einer Form aus als Getter.

Beide Properties geben den Wert als double Wert zurück.

Zur Klasse Rectangle:

Die Rectangle Klasse soll 3 Konstruktoren haben.

1. Konstruktor bekommt keinen Parameter. Die Instance hat dann eine Länge und Breite von null.
2. Konstruktor bekommt einen Parameter. Die Instance verfällt sich wie ein Quadrat und Länge und Breite sind gleich.
3. Konstruktor bekommt zwei Parameter. Der 1. Parameter ist die Länge und der 2. Parameter ist die Breite der Instance.

Property Surface soll den Umfang eines Rechtecks zurückgeben. Property Volume soll das Fläche eines Rechtecks zurückgeben.

Rectangle soll zusätzlich noch 2 Properties bieten

Public Property Length gibt die Länge eines Rechtecks aus als Getter und Setter. Public Property Width gibt die Breite eines Rechtecks aus als Getter und Setter.

Auch diese beiden Properties geben den Wert als double Wert zurück.

Es soll die ToString Methode überschrieben werden so, dass folgendes Beispiel gilt:

Beispiel Code:

```
var rectangle = new Rectangle(78.0, 45.234);
Console.WriteLine(rectangle.ToString());
```

Ausgabe:
Length: 78, Width: 45,234

Tipps: Deklarationen sind dieses mal nicht direkt angeben. Hier muss man selbst darauf kommen wie diese aussehen anhand der Aufgabenbeschreibung.

Die Keywords virtual, override und abstract werden eventuell wegen dem Polymorphismus gebraucht.

Test Program:

```
var rectangle = new Rectangle();
Console.WriteLine(rectangle.ToString());
rectangle = new Rectangle(2.0);
Console.WriteLine(rectangle.ToString());
rectangle = new Rectangle(5.25, 4.0);
Console.WriteLine(rectangle.ToString());

if (rectangle is Shape)
{
    Console.WriteLine($"{nameof(Rectangle)} inherits from {nameof(Shape)}");
}

Console.WriteLine($"rectangle.Surface = {rectangle.Surface}");
Console.WriteLine($"rectangle.Volume = {rectangle.Volume}");
rectangle = new Rectangle(5.0);
Console.WriteLine($"rectangle.Surface = {rectangle.Surface}");
Console.WriteLine($"rectangle.Volume = {rectangle.Volume}");
```

Ausgabe sollte sein:

Length: 0, Width: 0
Length: 2, Width: 2
Length: 5,25, Width: 4
Rectangle inherits from Shape
rectangle.Surface = 18,5
rectangle.Volume = 21
rectangle.Surface = 20
rectangle.Volume = 25

1.1 Exceptions für Rectangle Klasse

Beschreibung:

Falls der Konstruktor einen Parameter bekommt, der negative ist, soll eine ArgumentOutOfRangeException geworfen werden.

Falls das Property Height oder Width einem negativen Wert gleich gesetzt wird soll auch eine ArgumentOutOfRangeException geworfen werden.

Beim werfen der Exception soll gewährtleistet werden das:

- Die Exception soll im Property ParamName den Name der Property haben unter den die Exception passierte.
- Die Exception soll im Property ActualValue den negativen Wert haben, der die Exception auslöste.
- Die Exception soll im Property Message folgende Fehlermeldung haben "Parameter must not be not negative".

Tipp: Es gibt für ArgumentOutOfRangeException mehre Konstruktoren. Es sollte recherchiert werden welchen Konstruktor Aufruf notwendig ist um alle geforderten Properties in der Exception zu setzen.

Test Program:

```
try
{
    rectangle.Width= -2.0;
}
catch (ArgumentOutOfRangeException e)
{
    Console.WriteLine($"Error Message: {e.Message}");
    Console.WriteLine($"Name of wrong Parameter: {e.ParamName}");
    Console.WriteLine($"Wrong Value: {e.ActualValue}");
}
Console.WriteLine($"{new string('=', 50)}");
try
{
    rectangle = new Rectangle(-2.0);
}
catch (ArgumentOutOfRangeException e)
{
    Console.WriteLine($"Error Message: {e.Message}");
    Console.WriteLine($"Name of wrong Parameter: {e.ParamName}");
    Console.WriteLine($"Wrong Value: {e.ActualValue}");
}
Console.WriteLine($"{new string('=', 50)}");
try
{
    rectangle.Length = -2.0;
}
catch (ArgumentOutOfRangeException e)
{
    Console.WriteLine($"Error Message: {e.Message}");
    Console.WriteLine($"Name of wrong Parameter: {e.ParamName}");
    Console.WriteLine($"Wrong Value: {e.ActualValue}");
}
```

Ausgabe sollte sein:

Error Message: Parameter must not be not negative (Parameter 'Width')
Actual value was -2.
Name of wrong Parameter: Width
Wrong Value: -2
==
Error Message: Parameter must not be not negative (Parameter 'Length')
Actual value was -2.
Name of wrong Parameter: Length
Wrong Value: -2
==
Error Message: Parameter must not be not negative (Parameter 'Length')
Actual value was -2.
Name of wrong Parameter: Length
Wrong Value: -2

---

2. Implementierung von der Klasse RectangularCuboid

Beschreibung:

Diese Klasse soll von der Klasse Rectangle erben.

Es sollen folgender Konstruktoren implementiert werden.

1. Konstruktor bekommt keinen Parameter. Die Instance hat dann eine Länge, Breite und Höhe von null.
2. Konstruktor bekommt einen Parameter. Die Instance Länge, Breite und Höhe sind gleich.
3. Konstruktor bekommt drei Parameter. Der 1. Parameter ist die Länge, der 2. Parameter ist die Breite und der 3. Parameter ist die Höhe von der Instance.

Property Surface soll den Umfang eines Quaders zurückgeben. Property Volume soll das Volumen eines Quaders zurückgeben.

Es soll eine public Property Height geben welches als Getter und Setter für die Höhe des Quaders dient.

Es soll für das Property Height auch eine Exception geworfen werden wie bei Length oder Width in der Klasse Rectangle

Es soll die ToString Methode überschrieben werden so folgendes Beispiel gilt:

Beispiel Code:

```
var cuboid = new RectangularCuboid(78.0, 45.234, 7845.0);
Console.WriteLine(cuboid.ToString());
```

Ausgabe:
Length: 78, Width: 45,234, Height: 7845

Test Program:

```
var cuboid = new RectangularCuboid(10.0, 4.0, 5.0);
Console.WriteLine(cuboid);

if (cuboid is Rectangle)
{
    Console.WriteLine($"{nameof(RectangularCuboid)} inherits from
{nameof(Rectangle)}");
}

Console.WriteLine($"cuboid.Surface = {cuboid.Surface}");
Console.WriteLine($"cuboid.Volume = {cuboid.Volume}");

Console.WriteLine($"{new string('=', 50)}");

try
{
    cuboid.Height = -2.0;
}
catch (ArgumentOutOfRangeException e)
{
    Console.WriteLine($"Error Message: {e.Message}");
    Console.WriteLine($"Name of wrong Parameter: {e.ParamName}");
    Console.WriteLine($"Wrong Value: {e.ActualValue}");
}
```

Ausgabe sollte sein:

Length: 10, Width: 4, Height: 5
RectangularCuboid inherits from Rectangle
cuboid.Surface = 220
cuboid.Volume = 200
==
Error Message: Parameter must not be not negative (Parameter 'Height')
Actual value was -2.
Name of wrong Parameter: Height
Wrong Value: -2
