HAWK Fakultät Naturwissenschaften und Technik Göttingen

Windowsprogrammierung – WiSe17

Übung 1

Aufgabe 1.1 Properties und Exceptions, Teil 1

Schreiben Sie eine Klasse mit dem Namen **Testklasse**. **Testklasse** soll über eine öffentliche Eigenschaft (Property) mit dem Namen **Test** verfügen. **Test** soll einen **set** und einen **get** Accessor besitzen, mit deren Hilfe dem privaten Feld **test** vom Typ **int** von **Testklasse** ein Wert zugewiesen bzw. der zugewiesene Wert abgefragt werden kann. Zulässige Werte von **test** sollen sich im Bereich zwischen 0 und 100 bewegen. In dem **set**-Accessor von Test soll überprüft werden, daß die zu setzenden Werte sich innerhalb des zulässigen Bereichs bewegen. Ist dies nicht der Fall, so soll der **set** Accessor von **Test** eine Ausnahme vom Typ **ArgumentOutOfRangeException** werfen.
Überprüfen Sie **Testklasse**, indem Sie von einem Hauptprogramm aus die **Test** Eigenschaft benutzen und ihr korrekte und auch außerhalb des zulässigen Bereichs liegende Werte zuweisen bzw. den Wert der **Test** Eigenschaft abfragen. Fangen Sie in dem Hauptprogramm eine evtl. auftretende Ausnahme durch den try/catch-Mechanismus ab.

Aufgabe 1.2 Properties und Exceptions, Teil 2

Schreiben Sie eine Klasse mit dem Namen **Film**. Die Klasse soll über die folgenden öffentlichen Eigenschaften verfügen:

- **Titel** (vom Typ *string*)
- **Regisseur** (vom Typ *string*)
- **ReleaseDatum** (vom Typ *DateTime*)
- **Spielzeit** (vom Typ *TimeSpan*)
- **Mindestalter** (vom Typ *byte*)
- **Bewertung** (vom Typ *int?* (Alias für *System.Nullable<int>*); soll zwischen 1 und 5 liegen oder den *null*-Wert annehmen)
- **WieHäufigAbgespielt** (vom Typ *int*)
- **Alter** (vom Typ *TimeSpan*)
- **Gesamtspielzeit** (vom Typ *TimeSpan*)
- **Qualitätskategorie** (siehe unten)
- **IstAllzeitFavorit** (vom Typ *bool*; soll *true* zurückgeben, wenn die Gesamtspielzeit länger als 20 Stunden ist)

Die Eigenschaft **Bewertung** soll dabei über Schreib- und Lesezugriff (*get* und *set*) verfügen, alle anderen Eigenschaften sollen nur über einen Lesezugriff (*get*) verfügen.
Für jede der Eigenschaften **Titel**, **Regisseur**, **ReleaseDatum**, **Spielzeit**, **Mindestalter**, **Bewertung** und **WieHäufigAbgespielt** soll ein privates Feld existieren, dessen Wert in der zugehörigen Eigenschaft zurückgegeben werden kann, z.B.:

```
//Feld
private string regisseur;
//Eigenschaft
public string Regisseur { get { return regisseur; } }
```

Für die Eigenschaften **Alter**, **Gesamtspielzeit**, **Qualitätskategorie** und **IstAllzeitFavorit** werden keine zugehörigen Felder benötigt, da sich ihre Werte mit Hilfe von anderen Eigenschaften berechnen lassen (z.B. hängt die **Gesamtspielzeit** von den Eigenschaften **Spielzeit** und **WieHäufigAbgespielt** ab).
Als Rückgabetyp der Eigenschaft **Qualitätskategorie** soll ein selbstdefinierter Enumerationstyp beliebigen Namens verwendet werden, der über die folgenden 4 möglichen Werte verfügt: **Gut** (**Bewertung** liegt zwischen 4 und 5), **Mittelmäßig** (**Bewertung** beträgt 3), **Schlecht** (**Bewertung** liegt zwischen 1 und 2), **Unbewertet** (**Bewertung** hat den Wert *null*).
Analog zu Aufgabe 1.1 soll in dem set-Accessor der Eigenschaft **Bewertung** überprüft werden, ob der zu setzende Wert gültig ist (also zwischen 1 und 5 liegt oder den Wert *null* hat). Ist dies nicht der Fall, soll im set-Accessor von Bewertung eine Ausnahme vom Typ *ArgumentOutOfRangeException* geworden werden.
Definieren Sie ferner einen allgemeinen Konstruktor so, dass man ihm Werte für den Titel, den Regisseur, das ReleaseDatum, die Spielzeit und das Mindestalter übergeben kann. Die Bewertung soll zu Beginn den Wert *null* haben.
Weiterhin soll die Klasse über 3 öffentliche Elementfunktionen verfügen:

- **MindestalterErreicht** (nimmt das Geburtsdatum eines potentiellen Konsumenten vom Typ *DateTime* entgegen und gibt über einen *bool*schen Wert zurück, ob dieser alt genug für den Film ist)
- **Abspielen** (nimmt ebenfalls das Geburtsdatum (Typ: *DateTime*) eines potentiellen Konsumenten entgegen. Ist dieser nicht alt genug, soll eine Ausnahme vom Typ *InvalidOperationException* geworfen werden; ansonsten soll lediglich der Wert von **wieHäufigAbgespielt** um 1 erhöht werden)
- **ZeichenkettenKorrigieren** (keine Rückgabe- und Übergabeparameter. Soll die Zeichenketten von Titel und Regisseur so anpassen, dass sie nach der Anpassung mit einem Großbuchstaben anfangen und ansonsten nur noch aus Kleinbuchstaben (und sonstigen Zeichen wie z.B. Leerzeichen) bestehen. Es kann davon ausgegangen werden, dass das erste Zeichen immer ein Buchstabe ist. Beispielsweise soll der im Konstruktor eingegebene Titel „pUlP FiCtIoN“ nach der Korrektur so aussehen: „Pulp

fiction“. Hinweis: Im Gegensatz zur string-Klasse aus der C++-Standard-Bibliothek sind in C# Instanzen vom Typ *string* „immutable“, d.h. sie können nach ihrer Erzeugung nicht mehr verändert werden. Deshalb müssen Sie für die Korrektur neue *string*-Objekte erzeugen und den Feldern **titel** und **regisseur** zuweisen. Optional können Sie auch versuchen, dass jedes einzelne, durch Leerzeichen getrennte Wort mit einem Großbuchstaben anfängt.)

Schreiben Sie analog zu Aufgabe 1.1 ein Hauptprogramm, in dem Sie eine Instanz der Film-Klasse erzeugen und alle Möglichkeiten (also alle Eigenschaften und Methoden), die die Klasse bietet, demonstrieren. Weisen Sie der Eigenschaft Bewertung sowohl korrekte wie auch unzulässige Werte zu und fangen Sie evtl. auftretende Ausnahmen durch den try/catch-Mechanismus ab. Lassen Sie den erzeugten Film mit Hilfe einer Schleife zu einem Allzeit-Favoriten werden.

Aufgabe 1.3 Vererbung in C#

In Fortsetzung von Aufgabe 1.2 soll nun eine Klasse **CD** geschrieben werden, die über einen sehr ähnlichen Aufbau wie die Klasse **Film** verfügt. Konkret unterscheidet sich **CD** von **Film** nur in den folgenden Punkten:

- Statt der Eigenschaft **Regisseur** (und dem zugehörigen Feld) hat eine CD die Eigenschaft **Interpret**.
- **CD** soll zusätzlich über ein Array von Liedern (in Form von Zeichenketten) sowie über die zugehörigen Eigenschaften (nur Lesezugriff) AnzahlLieder (vom Typ *int*) und Lieder (vom Typ *string[]*) verfügen.
- Eine **CD** soll dann als Allzeit-Favorit gelten, wenn die Gesamtspielzeit länger als 40 Stunden ist.
- Die Methode **ZeichenkettenKorrigieren** soll neben Titel und Interpret auch die Schreibweise der einzelnen Lieder anpassen.
- Dem Konstruktor von **CD** soll statt einem Regisseur der Interpret übergeben werden, ebenso müssen die Lieder in Form eines string-Arrays übergeben werden.

Zur Vermeidung von Code-Verdopplung ist es sinnvoll, die Gemeinsamkeiten von **Film** und **CD** in eine gemeinsame Basisklasse auszulagern. Schreiben Sie zu diesem Zweck eine abstrakte Klasse **Abspielmedium**, in der alle Gemeinsamkeiten von **Film** und **CD** definiert sind. Deklarieren Sie in **Abspielmedium** die Eigenschaft **IstAllzeitFavorit** sowie die Methode **ZeichenkettenKorrigieren** als **abstrakt**, damit die erbenden Klassen **CD** und **Film** gezwungen sind, diese Member zu überschreiben.

Demonstrieren Sie abschließend in einem kleinen Testprogramm das polymorphe Verhalten der Methode **ZeichenkettenKorrigieren** und der Eigenschaft **IstAllzeitFavorit**. (D.h. Sie erzeugen jeweils eine Instanz der Klassen **Film** und **CD**, weisen diese je einer Referenzvariablen vom Typ **Abspielmedium** zu und rufen **ZeichenkettenKorrigieren**/**IstAllzeitFavorit** auf diesen auf.)

Aufgabe 1.4 Operatorüberladung

Entwerfen Sie eine Klasse für komplexe Zahlen. Diese Klasse soll einen überladenen Operator für die Division zweier komplexer Zahlen haben.

Zur Erinnerung: Zwei komplexe Zahlen z_1,z_2 mit

$z_1 = a + bi$

$z_2 = c + di$

werden dividiert mit Hilfe der Formel

$$\frac{z_1}{z_2} = \frac{a + b\,\mathrm{i}}{c + d\,\mathrm{i}} = \frac{(a + b\,\mathrm{i})(c - d\,\mathrm{i})}{(c + d\,\mathrm{i})(c - d\,\mathrm{i})} = A + B\,\mathrm{i}$$

$$A = \frac{a\,c + b\,d}{c^2 + d^2}, \qquad B = \frac{b\,c - a\,d}{c^2 + d^2}.$$

Fangen Sie den Fall, dass Real- und Imaginärteil von z_2 Null sind, mit Hilfe einer Ausnahme ab. Schreiben Sie ein Testprogramm für die Klasse, in das vom Benutzer interaktiv die beiden zu dividierenden komplexen Zahlen eingelesen werden.

[optional] Aufgabe 1.5 Überschreiben

Alle Klassen erben von der Klasse *object*, also auch die Klasse aus Aufgabe 1.3.
Die Klasse object besitzt eine virtuelle Methode *ToString*, die in abgeleiteten Klassen überschrieben werden kann und soll, um einen String mit Werten, der für die Klasse Bedeutung hat, zurückzugeben.
Überschreiben Sie die *ToString*-Methode in den Klassen Film und CD (aus dem Aufgaben 1.2 und 1.3) sowie in der Klasse für komplexe Zahlen (aus Aufgabe 1.4).
Die zurückgegebenen Zeichenketten sollen folgendes Format aufweisen:

Klasse Film:	*Pulp Fiction (Quentin Tarantino, 1994)*
Klasse CD:	*[1997] Radiohead – OK Computer*
Klasse für komplexe Zahlen:	*3 + 4i*

Was gibt die *ToString*-Methode zurück, wenn sie nicht überschrieben wurde? Nehmen Sie sich die überschriebene *ToString*-Methode der **DatumPlus**-Klasse aus dem Vorlesungsskript (1.CSharp) zum Vorbild.