9. Übungsblatt Thema: Advanced Interfaces and Generics

Einführung:

Um die Interfaces noch mal zu üben, implementieren wir Artikel mit verschiedenen Preis Modellen. Diese sollen später durch ein Warenhaus verwaltet werden. Damit das Warenhaus mit den Instanzen von verschiedenen Typen arbeiten kann, brauchen sie einen gemeinsamen Nenner. Dieser Nenner wird durch ein Interface geschaffen welches Gemeinsamkeiten zwischen den Objekten, die das Interface implementieren, fordert.

Implementierung von Klassen StandardArticle, DiscountArticle and BundleWare

Beschreibung:

Es sollen die 3 Klassen StandardArticle, DiscountArticle and BundleWare implementiert werden. Diese 3 Klassen solle das Interface IArticle implementieren.

Interface IArticle hat folgende Struktur

```
public interface IArticle
{
    string Name { get; }
    double SellPrice { get; }
    double BuyPrice { get; }

    bool IsSame(IArticle otherArticle);
}
```

Details zu StandardArticle:
Hat einen Constructor, welcher 3 Parameter bekommt, Name, SellPrice, und BuyPrice.

Die Methode IsSame return dann true wenn der Parameter ebenfalls gleiche im Property Name, SellPrice und BuyPrice ist.

Diese Klasse überschreibt die Methode ToString mit folgende Ausgabe:
$"Name: {this.Name}, sell price: {this.SellPrice}$, buy price: {this.BuyPrice}$"

Details zu DiscountArticle:

Hat einen Constructor, welcher 2 Parameter bekommt article vom Type StandardArticle und discountInPrecent vom Type int.

Der Parameter discountInPrecent soll Werte zwischen 0 und 100 haben. Falls der Parameter discountInPrecent außerhalb dieses Bereichs liegt, dann soll eine Exception vom Typ ArgumentOutOfRangeException geworfen werden. Diese Exception soll folgende Nachricht enthalten "Parameter discountInPrecent must be between 0 and 100"

Dieser discountInPrecent reduziert den SellPice um den Prozentsatz anhand seines Wertes.

Die Methode IsSame return dann true wenn der Parameter ebenfalls gleiche im Property Name, SellPrice und BuyPrice ist.

Diese Klasse überschreibt die Methode ToString mit folgende Ausgabe:
$"Name: {this.Name}, sell price: {this.SellPrice}$, buy price: {this.BuyPrice}$, Discount: {this._discount}%"

Details zu BundleWare:

Hat 2 Konstruktoren:

- Ein Konstruktor ohne Parameter
- Ein Konstruktor hat 2 Parameter (SellPrice, und BuyPrice)

Hat die Methode Add mit 2 Overloads

- Aufruf mit einen Parameter vom Type IArticle
- Aufruf mit 2 Parameter (SellPrice, und BuyPrice)

Die Property SellPrice soll den gesamenten Verkaufspreis, aller Artikel in der BundleWare Instanze, zurück geben. Die Property BuyPrice soll den gesamenten Einkaufspreis, aller Artikel in der BundleWare Instanze, zurück geben. Das Property Name ist dabei egal. Kann auch leer sein, dass BundleWare auch als Ramschware verstanden werden kann.

Die Methode IsSame return dann true wenn der Parameter nur gleich im Property SellPrice und BuyPrice ist !

Diese Klasse überschreibt die Methode ToString mit folgende Ausgabe:
$"Whole sell value: {this.SellPrice}$, Whole buy value: {this.BuyPrice}$"

Test Program für StandardArticle:

```
var articleNES = new StandardArticle("NES", 200.0, 100.0);
var articleNESCopy = new StandardArticle("NES", 200.0, 100.0);
var articleSNES = new StandardArticle("SNES", 400.0, 200.0);

Console.WriteLine($"articleNES.IsSame(articleSNES) =
{articleNES.IsSame(articleSNES)}");
Console.WriteLine($"articleNES.IsSame(articleNESCopy) =
{articleNES.IsSame(articleNESCopy)}");
Console.WriteLine($"{articleNES.ToString()}");
```

Ausgabe sollte sein:

articleNES.IsSame(articleSNES) = False
articleNES.IsSame(articleNESCopy) = True
Name: NES, sell price: 200$, buy price: 100$

Test Program für DiscountArticle:

```
var nes = new StandardArticle("NES", 200.0, 100.0);
var discountArticle = new DiscountArticle(nes, 10);
var tooCheap = new DiscountArticle(nes, 50);
var stillTooCheap = new DiscountArticle(nes, 50);

Console.WriteLine($"{discountArticle.ToString()}");

try
{
    var invalidItem = new DiscountArticle(nes, 200);
}
```

```
catch (ArgumentOutOfRangeException e)
{
    Console.WriteLine(e.Message);
}

Console.WriteLine($"discountArticle.IsSame(tooCheap) =
{discountArticle.IsSame(tooCheap)}");
Console.WriteLine($"tooCheap.IsSame(stillTooCheap) =
{tooCheap.IsSame(stillTooCheap)}");
```

Ausgabe sollte sein:

Name: NES, sell price: 180$, buy price: 100$, Discount: 0,1%
Specified argument was out of the range of valid values. (Parameter 'Parameter discountInPrecent must be between 0 and 100')
discountArticle.IsSame(tooCheap) = False
tooCheap.IsSame(stillTooCheap) = True

Test Program für BundleWare:

```
var bundleWare = new BundleWare();

bundleWare.Add(new StandardArticle("TV", 1000.0, 500.0));
Console.WriteLine(bundleWare.ToString());
bundleWare.Add(200.0, 100.0);
Console.WriteLine(bundleWare.ToString());

var otherBundleWare = new BundleWare();
otherBundleWare.Add(new StandardArticle("TV", 1000.0, 500.0));
Console.WriteLine($"otherBundleWare.IsSame(bundleWare) =
{otherBundleWare.IsSame(bundleWare)}");
otherBundleWare = new BundleWare();
otherBundleWare.Add(new StandardArticle("TV", 800.0, 400.0));
otherBundleWare.Add(new StandardArticle("Radio", 400.0, 200.0));
Console.WriteLine($"otherBundleWare.IsSame(bundleWare) =
{otherBundleWare.IsSame(bundleWare)}");
otherBundleWare.Add(bundleWare);
Console.WriteLine(otherBundleWare.ToString());
```

Ausgabe sollte sein:

Whole sell value: 1000$, Whole buy value: 500$
Whole sell value: 1200$, Whole buy value: 600$
otherBundleWare.IsSame(bundleWare) = False
otherBundleWare.IsSame(bundleWare) = True
Whole sell value: 2400$, Whole buy value: 1200$

Implementierung von Klasse Warehouse

Beschreibung:

WareHouse soll als Container von Objekten, die das Interface IArticle implementieren, dienen.

ToString Methode soll so überschrieben werden, dass der Return Value pro Zeile die Ausgabe von ToString eines hinzugefügten Artikels enthält.

Deklaration:

```
public void AddArticle(IArticle article) {}
public void RemoveArticle(IArticle articleToRemove) {}
public double GetWholeSellValue() {}
public double GetWholeBuyValue() {}
public double GetExpectedProfit() {}
public string GetSummary() {}
```

Hier darf und sollte man eine Datenstruktur vom Namespace System.Collections.Generic nutzten wie zum Beispiel List.

Dabei sollen Objekte über die Methoden AddArticle hingefügt werden können und über RemoveArticle soll ein Artikel entfernte werden können.

Bei RemoveArticle soll das 1. Objekte entfernt werden, welches mit seiner Methode IsSame true zurück liefert mit articleToRemove als Parameter.

GetWholeSellValue:

Gibt den gesamten Einkaufspreis aller Artikel zurück.

GetWholeSellValue:

Gibt den gesamten Verkaufspreis aller Artikel zurück.

GetWGetExpectedProfit:

Gibt den gesamten Gewinn, falls alle Artikel verkauft werden, zurück.

GetSummary:

Gibt den Zusammenfassung zurück. Eine Zusammenfassung ergibt sich aus folgenden Beispiel:

"Expected Profit: 2001, Whole Buy Value: 6002, Whole Sale value: 8003"

Test Program:

```
var wareHouse = new Warehouse();
var articleToRemoveLater = new StandardArticle("TV", 2000.0, 1000.0);
wareHouse.AddArticle(articleToRemoveLater);
wareHouse.AddArticle(new StandardArticle("Chips", 3.0, 2.0));
var discounted = new DiscountArticle(new StandardArticle("Car", 10000.0, 5000.0), 40);
wareHouse.AddArticle(discounted);
var bundleWare = new BundleWare();
Console.WriteLine($"wareHouse.GetSummary() ={wareHouse.GetSummary()}");
bundleWare.Add(new StandardArticle("TV", 1000.0, 500.0));
bundleWare.Add(200.0, 100.0);
```

```
wareHouse.AddArticle(bundleWare);
Console.WriteLine($"(Sell price, Buy price) = {(wareHouse.GetWholeSellValue(),
wareHouse.GetWholeBuyValue())}");
Console.WriteLine($"wareHouse.GetWholeSellValue() = {wareHouse.GetWholeSellValue()}");
Console.WriteLine($"wareHouse.GetWholeBuyValue() = {wareHouse.GetWholeBuyValue()}");
Console.WriteLine($"wareHouse.GetExpectedProfit() = {wareHouse.GetExpectedProfit()}");
wareHouse.RemoveArticle(articleToRemoveLater);
Console.WriteLine($"(Sell price, Buy price) = {(wareHouse.GetWholeSellValue(),
wareHouse.GetWholeBuyValue())}");
Console.WriteLine($"wareHouse.GetWholeSellValue() = {wareHouse.GetWholeSellValue()}");
Console.WriteLine($"wareHouse.GetWholeBuyValue() = {wareHouse.GetWholeBuyValue()}");
Console.WriteLine($"wareHouse.GetExpectedProfit() = {wareHouse.GetExpectedProfit()}");
Console.WriteLine($"{wareHouse.ToString()}");
```

Ausgabe sollte sein:

wareHouse.GetSummary() =Expected Profit:2001, Whole Buy Value: 6002, Whole Sale value: 8003
(Sell price, Buy price) = (9203, 6602)
wareHouse.GetWholeSellValue() = 9203
wareHouse.GetWholeBuyValue() = 6602
wareHouse.GetExpectedProfit() = 2601
(Sell price, Buy price) = (7203, 5602)
wareHouse.GetWholeSellValue() = 7203
wareHouse.GetWholeBuyValue() = 5602
wareHouse.GetExpectedProfit() = 1601
Name: Chips, sell price: 3$, buy price: 2$
Name: Car, sell price: 6000$, buy price: 5000$, Discount: 0,4%
Whole sell value: 1200$, Whole buy value: 600$
