

Fortgeschrittene Programmierkonzepte (FPK)

Musteraufgaben

Prof. Dr. Stefan Eicker, Tim Gottschalk

Wintersemester 2008/2009

Version 3.0



Klausurmodalitäten

- Die Klausur besteht aus zwei Alternativen. Jede Alternative besteht aus 2 Aufgaben, diese können wiederum aus mehreren Teilaufgaben bestehen. Es ist genau eine Alternative zu bearbeiten.
- Orientieren Sie sich hinsichtlich der Ausführlichkeit der Antworten an den hinter der jeweiligen Aufgabenstellung in Klammern angegebenen Punktzahl (1 Punkt ~ 1 Minute)!
- Die Bearbeitungszeit beträgt 60 Minuten plus 5 Minuten Einlesezeit. Die maximal erreichbare Punktzahl beträgt 60 Punkte.
- Mit Ausnahme von Schreibgeräten sind keine Hilfsmittel erlaubt.

Inhalt

Aufgabe A:	C# und das .NET Framework	4
Aufgabe B:	Properties, Indexer und Foreach	5
Aufgabe C:	Properties, Indexer,...	6
Aufgabe D:	Objektorientierung	7
Aufgabe E:	Operatorüberladung	8
Aufgabe F:	Vererbung	9
Aufgabe G:	Assemblies	10
Aufgabe H:	Zeichenkettenverarbeitung	11
Aufgabe I:	Zeichenkettenverarbeitung	13
Aufgabe J:	Zeichenkettenverarbeitung	15
Aufgabe K:	Speichermanagement	17
Aufgabe L:	Speichermanagement	19
Aufgabe M:	Speichermanagement	20
Aufgabe N:	Metadaten & Fehlerbehandlung	22
Aufgabe O:	Fehlerbehandlung	23
Aufgabe P:	Fehlerbehandlung	24
Aufgabe Q:	Fehlerbehandlung	25
Aufgabe R:	XML-Serialisierung	26
Aufgabe S:	XML-Serialisierung	29

Aufgabe T:	XML-Serialisierung	31
Aufgabe U:	Refactoring	33
Aufgabe V:	Vererbung	35
Aufgabe W:	Metadaten	36
Aufgabe X:	Metadaten und Delegates	37
Aufgabe Y:	Generics	38

Aufgabe A: C# und das .NET Framework**Teilaufgabe A1: (20 Punkte)**

C# ist als Sprache eng mit dem .NET Framework verbunden. Erläutern Sie die wesentlichen Komponenten des Frameworks und setzen Sie diese in Zusammenhang mit der Sprache C#. Gehen Sie mindestens auf die folgenden Begriffe ein:

- CTS
- CLR
- Basisklassen
- IL

Teilaufgabe A2: (10 Punkte)

Skizzieren Sie kurz die wesentlichen Voraussetzungen damit auf Elemente einer Klasse mittels foreach-Schleife zugegriffen werden kann. Welche Member müssen in der Klasse implementiert werden und welche Aufgabe besitzen diese?

Aufgabe B: Properties, Indexer und Foreach**Teilaufgabe B1: (20 Punkte)**

Erläutern Sie die Grundlagen des Zugriffs auf Felder einer Klasse über Properties und über Indexer. Gehen Sie auch auf die Unterschiede ein. Geben Sie jeweils ein kurzes Beispiel für die Implementation eines Properties und eines Indexers.

Teilaufgabe B2: (10 Punkte)

Skizzieren Sie kurz die wesentlichen Voraussetzungen damit auf Elemente einer Klasse mittels foreach-Schleife zugegriffen werden kann. Welche Member müssen in der Klasse implementiert werden und welche Aufgabe besitzen diese?

Aufgabe C: Properties, Indexer,...

Teilaufgabe C1: (25 Punkte)

Implementieren Sie in C# eine Klasse "Vorlesungsteilnehmer", die zur Verwaltung der C# Vorlesung ab SS04 eingesetzt werden soll.

Auf dem Klausurbogen soll lediglich der Quellcode der Datei "Vorlesungsteilnehmer.cs" zu finden sein! Es sind keine weiteren Angaben zur Konfiguration innerhalb von VS.net nötig.

Implementieren Sie die folgenden Member:

Capacity	Read-Only Property zum lesen der maximalen Anzahl von Studenten, die an dieser Veranstaltung teilnehmen können.
capacity	das zugehörige private konstante Feld, das durch den Konstruktor initialisiert werden soll.
Count Read-Only	Property zur Ausgabe der zzt. in der Vorlesung verwalteten StudentInnen
CountM	Diesmal nur die männlichen.
CountF	Diesmal nur die weiblichen.
Add(StudentIn)	fügt eine(n) StudentIn zur Vorlesung hinzu.
Remove(StudentIn)	entfernt eine(n) StudentIn aus der Vorlesung.

Zusätzlich sollen ein Konstruktor und ein Indexer implementiert werden. Weitere Member – sofern benötigt – sind eigenständig dieser Klasse hinzuzufügen. Es ist freigestellt, wie die Klasse "Vorlesungsteilnehmer" die Studenten intern verwaltet.

Eine Klasse "StudentIn" ist bereits vorhanden. Objekte dieser Klasse sollen in der Klasse "Vorlesungsteilnehmer" verwendet werden. Die Klasse StudentIn verfügt über ein Property "Gender", über das das Geschlecht ausgelesen werden kann. Ein Konstruktor der Klasse StudentIn erwartet den Namen (string) und das Geschlecht (char) als Parameter.

Teilaufgabe C2: (5 Punkte)

Implementieren Sie ein Hauptprogramm, welches ihren Code testet. Gehen Sie dabei von einer C#-Konsolenanwendung aus.

Aufgabe D: Objektorientierung

Teilaufgabe D1: (30 Punkte)

Implementieren Sie eine Assembly, die ein elektronisches Buch repräsentiert.

Innerhalb des Namespaces `ElectronicLibrary` sollen die Klasse `Book` und die Enumeration `BookType` implementiert werden.

Für die Klasse `Book` gelten die folgenden Anforderungen:

- Alle privaten Felder sollen über öffentliche Read-Write-Properties zugreifbar sein.
- In einem `Book` sollen der Autor, der Titel, das Erscheinungsjahr und der Buchtyp gespeichert sein.
- Der Inhalt des Buches soll als Collection von einzelnen Kapiteln verwaltet werden. Diese sollen über einen Indexer abgerufen und über eine Add-Methode hinzugefügt werden können.
- Als Buchtyp soll die Enumeration `BookType` genutzt werden, die 5 unterschiedliche Buchtypen umfassen soll: Roman, Gedicht, Märchen, Sage, Sachbuch
- Implementieren Sie geeignete Konstruktoren
 - Standard-Konstruktor,
 - Copy-Konstruktor,
 - mindestens 2 unterschiedliche Konstruktoren, die eine sinnvolle Erzeugung eines Buches durch direkte Parameterübergabe ermöglichen
- Ermöglichen Sie eine Ausgabe eines Buches als Zeichenkette über die in C# vorgesehenen Mechanismen.

Dabei sollen ein Buch in einen „XML-String“ serialisiert werden, der die einzelnen Bestandteile der Klasse gemäß dem folgenden Beispiel enthält, wobei die Punkte durch die Werte des Objektes gefüllt werden sollen:

```
<book> <author>...</author> <title>...</title> <type>...</type>
<chapters> <chapter>...</chapter> <chapter>...</chapter> <chapter>...</chapter> <chapter>...</chapter> <chapter>...</chapter>
<chapter>...</chapter> <chapter>...</chapter> <chapter>...</chapter> <chapter>...</chapter> </chapters> </book>
```

Erstellen Sie ein Hauptprogramm, das folgende Funktionen der Assembly testet:

- Erstellen Sie ein `Book`-Objekt und füllen Sie das Buch mit Inhalt:
 - Setzen Sie die Felder über die Properties.
 - Füllen Sie das Buch mit einigen „Kapiteln“ Text.
 - Geben Sie das Buch auf der Konsole aus.

Greifen Sie über den Indexer auf ein Kapitel zu und geben dieses erneut auf der Konsole aus.

Aufgabe E: Operatorüberladung

Teilaufgabe E1:

(30 Punkte)

Gegeben ist eine Assembly `Geometrie.dll`, in der eine Klasse `Point` implementiert wurde. Die Klasse `Point` ist eine einfache Implementierung eines Punktes im zweidimensionalen Raum. Ein Punkt wird durch seine zwei Koordinaten `x` und `y` repräsentiert, die als öffentliche Felder vom Typ `int` in der Klasse `Point` implementiert wurden. Über diese Felder hinaus wurde in der Klasse keine Funktionalität implementiert.

Implementieren Sie eine Klasse `EnhancedPoint`, die von der Klasse `Point` abgeleitet ist und um die folgenden Bestandteile erweitert wird:

- Erzeugen Sie geeignete Konstruktoren (Standard-Konstruktor, Copy-Konstruktor, Konstruktor mit `x` und `y` als Parameter)
- Sorgen Sie für eine ansprechende Ausgabe des Punktes im Konsolenfenster durch die in C# vorgesehenen Mechanismen.
- Implementieren Sie für diese Klasse die Operatoren `+` und `-` zur Verknüpfung von Punkten sowie den Operator `==`.
- Entwerfen Sie eine statische Methode um die Entfernung zwischen zwei Punkten zu berechnen.
- Definieren Sie einen expliziten Cast, der die Entfernung eines Punkts vom Ursprung in `Float` ausgibt.

Erstellen Sie ein Hauptprogramm, das alle Funktionen der Klasse ausreichend testet.

Hinweise:

Die Entfernung zweier Punkte berechnet sich nach der folgenden Formel

$$\sqrt{(\Delta x)^2 + (\Delta y)^2}$$

Die Methode `Math.Sqrt` gibt die Quadratwurzel einer angegebenen Zahl zurück.

```
public static double Sqrt(double d);
```


Aufgabe F: Vererbung

Teilaufgabe F1: (10 Punkte)

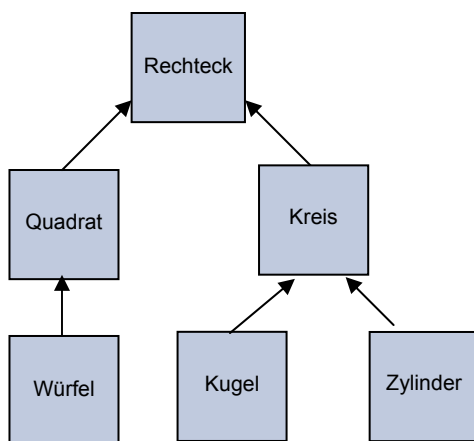
Erklären Sie den Unterschied zwischen abstrakten und virtuellen Methoden anhand einfacher Beispiele.

Teilaufgabe F2: (20 Punkte)

Erzeugen Sie eine Klasse Dimension, die die Verwaltung von graphischen Objekten erlaubt.

Die Basisklasse eignet sich zur Darstellung eines Rechtecks und hat zwei Properties, die die Ausdehnung in x- und y-Richtung enthalten und eine Methode Fläche zur Berechnung des Flächeninhalts ($x * y$).

Die folgenden Klassen sollen ebenfalls entsprechend der Hierarchie implementiert werden:



Als Hilfestellung die Formeln zur Flächenberechnung der einzelnen Klassen:

- Rechteck ($x*y$)
- Quadrat (a^2)
- Würfel ($6*a^2$)
- Kreis ($2\pi r^2$)
- Kugel ($4\pi r^2$)
- Zylinder ($2*r^2*\pi + (2r*\pi*h)$)

Legen Sie ein Array vom Typ der Basisklasse (Dimension) an und erzeugen Sie hierbei Instanzen der Kindklassen. Geben Sie anschließend den Flächeninhalt des Objekts auf der Konsole aus.

Aufgabe G: Assemblies

Teilaufgabe G1: (2 Punkte)

Definieren Sie den Begriff Assemblies!

Teilaufgabe G2: (10 Punkte)

Nennen Sie die wesentlichen Eigenschaften von Assemblies! Erläutern Sie anschließend die Probleme traditioneller Komponenten-Architekturen, die durch die Eigenschaften der Assemblies gelöst werden sollen.

Teilaufgabe G3: (5 Punkte)

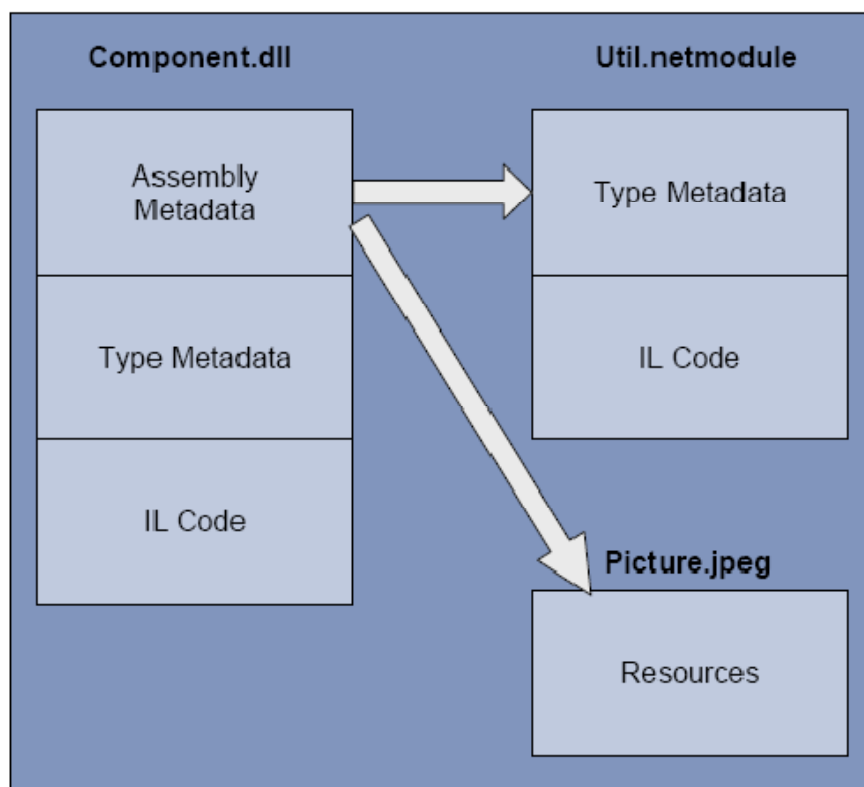
Erläutern Sie den Unterschied zwischen Private Assemblies und Shared Assemblies und nennen Sie potentielle Anwendungsgebiete für die beiden Varianten.

Teilaufgabe G4: (8 Punkte)

Erläutern Sie, wie Assemblies einen Beitrag zum Mehrsprachigkeitskonzept innerhalb des .NET Frameworks beitragen. Gehen Sie in diesem Zusammenhang auch kurz auf CTS und CLS ein.

Teilaufgabe G5: (5 Punkte)

Beschreiben Sie anhand der Abbildung die Bestandteile von Assemblies und schildern Sie, wie die einzelnen Bestandteile einer Multifile-Assembly einander zugeordnet werden.



Aufgabe H: Zeichenkettenverarbeitung

Teilaufgabe H1:

(10 Punkte)

Im Hinblick auf die Speicherverwaltung existieren wesentliche Unterschiede zwischen den Klassen `String` und `StringBuilder`! Verbessern Sie (auf dem Aufgabenblatt) das folgende Programm derart, dass die Klasse `StringBuilder` in sinnvollem Umfang eingesetzt wird. Berücksichtigen Sie bei der Verwendung des `StringBuilders` auch seine Properties, um die Speicherverwaltung optimal zu steuern.

Begründen Sie zusätzlich kurz schriftlich (auf einem Klausurbogen) die getroffenen Veränderungen.

```
1  using System;
2
3  namespace Zeichenkettenverarbeitung
4  {
5      class MainClass
6      {
7          static void Main(string[] args)
8          {
9              Encoding Encoder = new Encoding();
10
11              string klartext = "Dies ist beziehungsweise war der Klartext";
12
13              string chiffprat = Encoder.Encode(klartext);
14
15              Console.WriteLine(chiffprat);
16          }
17      }
18
19      class Encoding
20      {
21          public string Encode(string text)
22          {
23              for(int i = (int)'z'; i >= (int)'a'; i--)
24              {
25                  char alt = (char)i;
26                  char neu = (char)(i+2);
27                  text = text.Replace(alt, neu);
28              }
29
30              for(int i = (int)'Z'; i >= (int)'A'; i--)
31              {
32                  char alt = (char)i;
33                  char neu = (char)(i+2);
34                  text = text.Replace(alt, neu);
35              }
36
37              return text;
38          }
39      }
40  }
41
```

Teilaufgabe H2:

(20 Punkte)

Zum Kontext: Eine Assembly wird von einem Hauptprogramm verwendet (dieses muss **nicht** implementiert werden), welches Datensätze aus einer Datei zeilenweise einliest. Das Hauptprogramm übergibt jede Zeile der Reihe nach an die Methoden der Assembly und verarbeitet das Ergebnis weiter. Als Ergebnis wird bei den spezifischen Methoden jeweils nur ein Treffer erwartet.

Implementieren Sie unter Verwendung regulärer Ausdrücke diese Assembly, die folgende Merkmale besitzt:

- Die Assembly enthält eine Klasse „AdressParser“
- Die Klasse AdressParser enthält die folgenden statischen Methoden, deren Parameter und Rückgabewert den Typ string besitzen. Die Methoden geben jeweils das durch den Methodennamen spezifizierte Element zurück.

- SearchStreet // Straße inkl. Hausnummer
- SearchLastName // Nachname
- SearchBirthDate // Geburtsdatum

- Der Eingabestring, der für alle Methoden identisch ist, entspricht dem folgenden Aufbau:

Name;Adresse;Geburtsdatum

- Beispiele, mit denen die Assembly korrekt arbeiten soll:

Meier, Hans;Heinrichstr. 34 45332 Essen;23.02.2001
Müller-Lüdenscheidt, Otto;Berliner Str. 89 45123 Es-
sen;12.04.1987
Schmidt, Hans O.;Sommer Allee 1 17643 Lubmin;01.03.1917

- Zusätzlich ist gewünscht, dass ein fortgeschrittener Anwender über eine weitere Methode unter Angabe von Datenzeile und Suchpattern weitere Elemente aus den Textzeilen ermitteln kann. Daher soll eine allgemeine Methode bereitgestellt werden, die den Suchtext und das Pattern übergeben bekommt und mögliche Suchergebnisse (Plural!) als Rückgabewert liefert.

Als Hilfestellung sind die folgenden Sprachbestandteile regulärer Ausdrücke gegeben:

Zeichen	
.	Beliebiges Zeichen außer \n
\n	New Line
\d	Dezimal-Zeichen
\s	White-Space-Zeichen
\w	Wortzeichen
Quantifizierer	
*	0-n
+	1-n
?	0-1
{n}	N

Aufgabe I: Zeichenkettenverarbeitung

Teilaufgabe I1:

(10 Punkte)

Im Hinblick auf die Speicherverwaltung existieren wesentliche Unterschiede zwischen den Klassen `String` und `StringBuilder`! Verbessern Sie (auf dem Aufgabenblatt) das folgende Programm derart, dass die Klasse `StringBuilder` in sinnvollem Umfang eingesetzt wird. Berücksichtigen Sie bei der Verwendung des `StringBuilders` auch seine Properties, um die Speicherverwaltung optimal zu steuern.

Begründen Sie zusätzlich kurz schriftlich (auf einem Klausurbogen) die getroffenen Veränderungen.

```

1  using System;
2
3  namespace Zeichenkettenverarbeitung
4  {
5      class MainClass
6      {
7          static void Main(string[] args)
8          {
9              Encoding Encoder = new Encoding();
10
11              string klartext = "Dies ist beziehungsweise war der Klartext";
12
13              string chiffprat = Encoder.Encode(klartext);
14
15              Console.WriteLine(chiffprat);
16          }
17      }
18
19      class Encoding
20      {
21          public string Encode(string text)
22          {
23              for(int i = (int)'z'; i>=(int)'a'; i--)
24              {
25                  char alt = (char)i;
26                  char neu = (char)(i+2);
27                  text = text.Replace(alt, neu);
28              }
29
30              for(int i = (int)'Z'; i>=(int)'A'; i--)
31              {
32                  char alt = (char)i;
33                  char neu = (char)(i+2);
34                  text = text.Replace(alt, neu);
35              }
36
37              return text;
38          }
39      }
40  }
41

```

Teilaufgabe I2:

(20 Punkte)

Zum Kontext: Eine Assembly wird von einem Hauptprogramm verwendet (dieses muss **nicht** implementiert werden). Das Hauptprogramm nutzt diese Assembly zum Validieren von Benutzereingaben in einer Webanwendung. Die Validierung soll innerhalb der Assembly über Reguläre Ausdrücke erfolgen. Um die Komponente einfach nutzbar zu machen, soll nach außen nichts von der internen Technologie sichtbar sein.

Implementieren Sie unter Verwendung regulärer Ausdrücke die beschriebene Assembly, die folgende Merkmale besitzt:

Implementieren Sie unter Verwendung regulärer Ausdrücke diese Assembly, die folgende Merkmale besitzt:

- Die Assembly enthält eine Klasse „Validator“
- Die Klasse `Validator` enthält die folgenden statischen Methoden, deren Parameter die zu prüfende Zeichenkette ist und deren Rückgabewerte den Typ `Boolean` besitzen, um anzuzeigen, ob der Parameterausdruck dem gewünschten Schema entspricht.
 - `IsEmailAddress`
Schema: z.B. hugo.mueller@mail-server.de
 - `IsTelephoneNumber`
Schema: deutsche Telefonnummer inkl. Vorwahl durch „/“ getrennt.
 - `IsPlz`
Schema: Deutsche Postleitzahl mit dem optionalen Präfix „D-“
 - `IsUrl`
Schema: http://host-name.sub-domain.domain. topleveldomain/ directory/file.html

Zusätzlich soll eine weitere Methode `ParseForEmailAddresses` implementiert werden, die aus einer beliebigen gegebenen Zeichenkette Emailadressen herausfiltert und zurückgibt. Wenn in der Zeichenkette mehr als ein Treffer enthalten ist, sollen entsprechend mehrere Ergebnisse (Emailadressen) zurückgegeben werden.

Parameter und Rückgabewerte sind vom Typ `string` bzw. `string[]`.

Geben Sie in jeder Methode innerhalb eines Kommentars mindestens je ein Beispiel für einen erfolgreichen und einen erfolglosen Treffer an.

Als Hilfestellung sind die folgenden Sprachbestandteile regulärer

Ausdrücke gegeben:

Zeichen	
.	Beliebiges Zeichen außer \n
\n	New Line
\d	Dezimal-Zeichen
\s	White-Space-Zeichen
\w	Wortzeichen
Quantifizierer	
*	0-n
+	1-n
?	0-1
{n}	N

Aufgabe J: Zeichenkettenverarbeitung

Teilaufgabe J1:

(10 Punkte)

In Hinblick auf die Speicherverwaltung existieren wesentliche Unterschiede zwischen den Klassen `String` und `StringBuilder`! Verbessern Sie (auf dem Aufgabenblatt) das folgende Programm derart, dass die Klasse `StringBuilder` in **sinnvollem** Umfang eingesetzt wird. Berücksichtigen Sie bei der Verwendung des `StringBuilders` auch seine Properties, um die Speicherverwaltung optimal zu steuern.

```
1  using System;
2
3  namespace Zeichenkettenverarbeitung
4  {
5      class MainClass
6      {
7          static void Main(string[] args)
8          {
9              Encoding Encoder = new Encoding();
10
11              string klartext = "Dies ist beziehungsweise war der Klartext";
12
13              string chiffrat = Encoder.Encode(klartext);
14
15              Console.WriteLine(chiffrat);
16          }
17      }
18
19      class Encoding
20      {
21          public string Encode(string text)
22          {
23              for(int i = (int)'z'; i >= (int)'a'; i--)
24              {
25                  char alt = (char)i;
26                  char neu = (char)(i+2);
27                  text = text.Replace(alt, neu);
28              }
29
30              for(int i = (int)'Z'; i >= (int)'A'; i--)
31              {
32                  char alt = (char)i;
33                  char neu = (char)(i+2);
34                  text = text.Replace(alt, neu);
35              }
36
37              return text;
38          }
39      }
40  }
41
```

Teilaufgabe J2:

(5 Punkte)

Begründen Sie zusätzlich schriftlich (auf einem Klausurbogen) die in Aufgabenteil 1 vorgenommenen Veränderungen.

Teilaufgabe J3:**(15 Punkte)**

Erläutern Sie am Beispiel der Zeichenkettenverarbeitung, wie sich Entscheidungen innerhalb der Softwareentwicklung auf die folgenden Bereiche auswirken können:

- Entwicklung
- Ausführung (z.B. Laufzeit & Speicher)
- Wartung

Berücksichtigen Sie bzgl. der Zeichenkettenverarbeitung die Klassen String, StringBuilder und RegularExpression bzw. die mit diesen Klassen verbundenen Konzepte.

Aufgabe K: Speichermanagement**Teilaufgabe K1: (8 Punkte)**

Erläutern Sie die wesentlichen Aspekte des Speichermanagements in C#. Gehen Sie dabei insbesondere auf Begriffe wie "Managed Heap" und "Stack" ein. Unterstützen Sie Ihre Ausführungen an geeigneten Stellen durch Abbildungen.

Teilaufgabe K2: (12 Punkte)

Erläutern Sie die Grundlagen der „Pointer“! Notieren und beschreiben Sie die syntaktischen Bestandteile der Sprache C#, die beim Einsatz von Pointern verwendet werden. Begründen Sie, warum Pointer innerhalb der Sprache C# enthalten sind.

Teilaufgabe K3:

(10 Punkte)

Erläutern Sie das Vorgehen des Garbage Collectors anhand der folgenden Abbildungen und geben Sie mindestens zwei Gründe an, warum für den Garbage Collector dieses Vorgehen gewählt wurde.

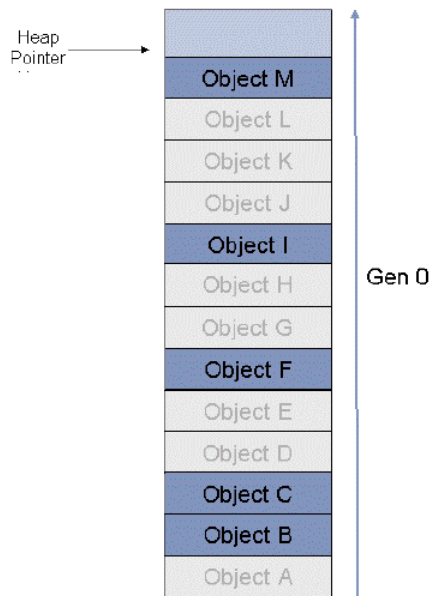


Abbildung 1

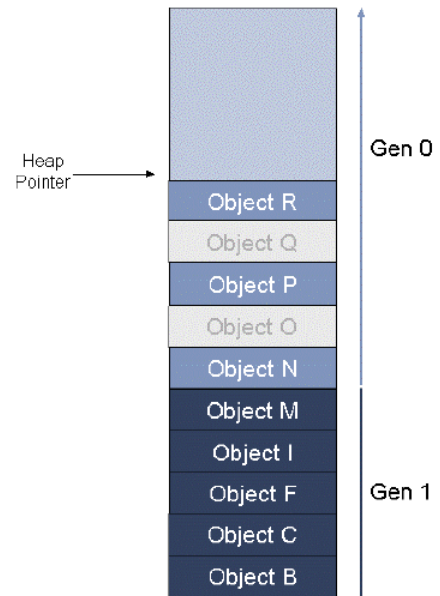


Abbildung 2

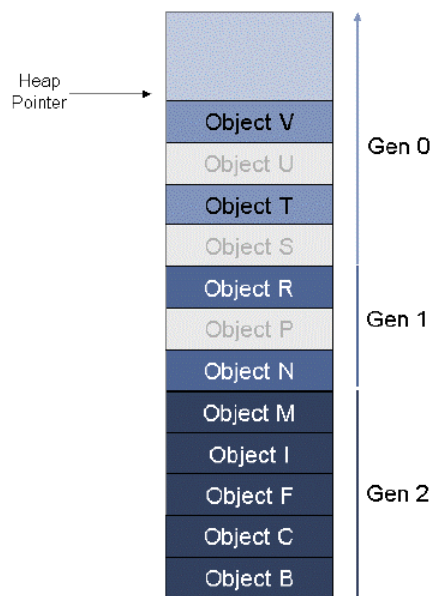


Abbildung 3

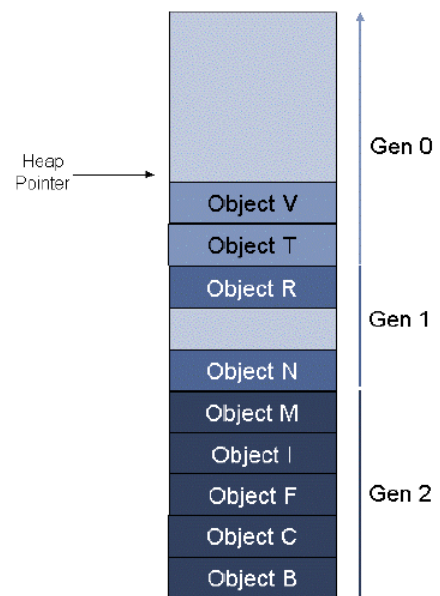


Abbildung 4

Aufgabe L: Speichermanagement

Teilaufgabe L1: (10 Punkte)

Erläutern Sie die wesentlichen Aspekte des Speichermanagements in C#. Gehen Sie dabei insbesondere auf Begriffe wie "Heap", "Stack", "managed", "unmanaged" sowie "Pointer" ein. Unterstützen Sie Ihre Ausführungen an geeigneten Stellen durch Abbildungen.

Teilaufgabe L2: (10 Punkte)

Beschreiben Sie im Detail, welche Vorgänge im Bezug auf das Speichermanagement während der Ausführung des folgenden Code-Abschnitts stattfinden.

```
int x = 20;
{
    int y = 30;
}
int * pX = &x;
*pX = 50 ;
```

Teilaufgabe L3: (10 Punkte)

Erläutern Sie die im Hinblick auf die Speicherverwaltung die Unterschiede zwischen den Klassen String und StringBuilder! Gehen Sie dabei auch auf die Properties des Stringbuilders ein!

In welchen Situationen ist der Einsatz eines Strings bzw. eines Stringbuilders vorzuziehen?

Unterstützen Sie Ihre Erläuterungen bei Bedarf durch eine Abbildung!

Aufgabe M: Speichermanagement**Teilaufgabe M1: (10 Punkte)**

Erläutern Sie die wesentlichen Aspekte des Speichermanagements in C#. Gehen Sie dabei insbesondere auf Begriffe wie "Heap", "Stack", "managed", "unmanaged" sowie "Pointer" ein. Unterstützen Sie Ihre Ausführungen an geeigneten Stellen durch Abbildungen.

Teilaufgabe M2: (10 Punkte)

Beschreiben Sie im Detail, welche Vorgänge im Bezug auf das Speichermanagement während der Ausführung des folgenden Code-Abschnitts stattfinden.

```
int x = 20;
{
    int y = 30;
}
int * pX = &x;
*pX = 50 ;
```

Teilaufgabe M3:

(10 Punkte)

Erläutern Sie das Vorgehen des Garbage Collectors anhand der folgenden Abbildungen und geben Sie mindestens zwei Gründe an, warum für den Garbage Collector dieses Vorgehen gewählt wurde.

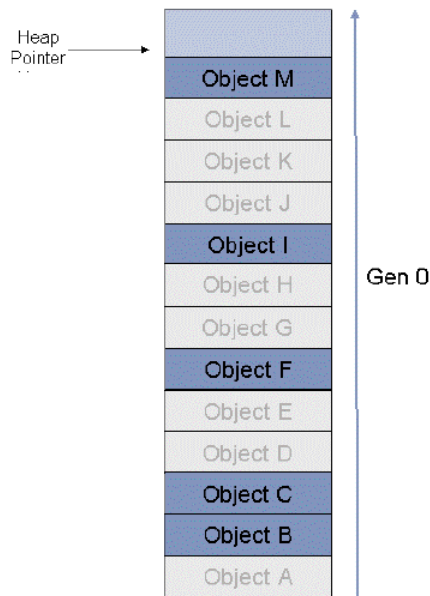


Abbildung 1

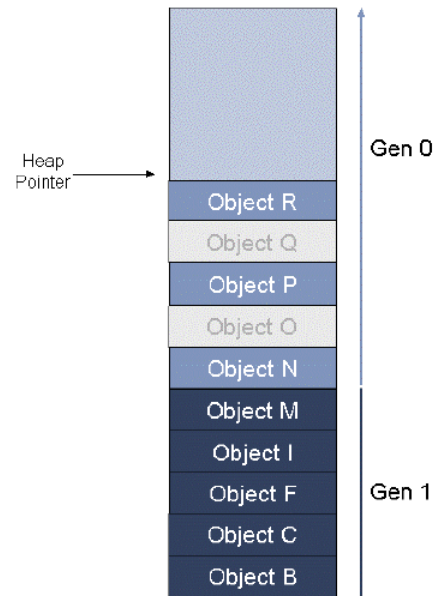


Abbildung 2

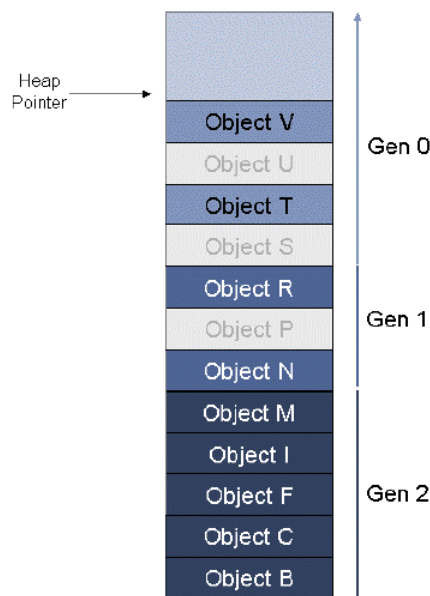


Abbildung 3

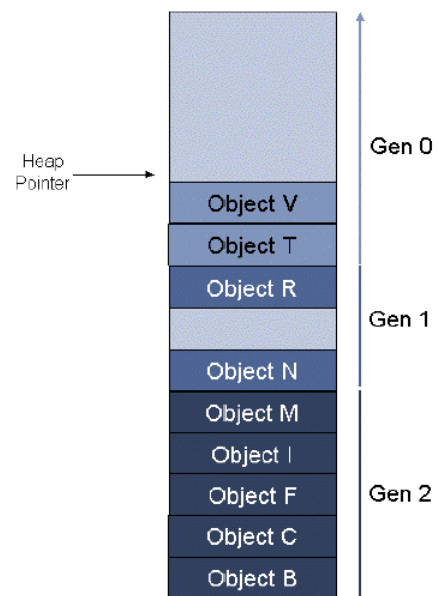


Abbildung 4

Aufgabe N: Metadaten & Fehlerbehandlung**Teilaufgabe N1: (15 Punkte)**

Erläutern Sie, wo bei der Entwicklung mit dem .NET-Framework Metadaten zum Einsatz kommen. Gehen Sie dabei u.a. die folgenden Aspekte ein:

- Welche Relevanz besitzen Metadaten im Kontext der Software-Entwicklung?
- Welche Metadaten werden bei der Entwicklung mit C# „automatisch“ bereitgestellt?
- Über welche Möglichkeiten zur Nutzung und zur Einflussnahme verfügt der Entwickler?

Teilaufgabe N2: (15 Punkte)

Im .NET-Framework wird dem Entwickler ein Mechanismus zur Fehlerbehandlung bereitgestellt.

- Skizzieren Sie die wesentlichen Aspekte dieses Mechanismus.
- Gehen Sie dabei sowohl auf systemseitige als auch auf benutzerdefinierte Fehlerbehandlung ein.
- Formulieren Sie ein kurzes Code-Beispiel zur Fehlerbehandlung, um die in C# eingesetzte Syntax darzustellen.

Aufgabe O: Fehlerbehandlung

Teilaufgabe O1: (10 Punkte)

Im .NET-Framework wird dem Entwickler ein Mechanismus zur Fehlerbehandlung bereitgestellt. Skizzieren Sie ausführlich die wesentlichen Aspekte dieses Mechanismus und erwähnen Sie dabei ebenfalls konkrete Einsatzgebiete für diese Mechanismen!

Teilaufgabe O2: (10 Punkte)

Erläutern Sie den Unterschied zwischen systemseitigen und benutzerdefinierten Fehlerbehandlungsobjekten in .NET! Nennen Sie Beispiele für beide Varianten und erläutern Sie den sinnvollen Einsatz der systemseitigen und benutzerdefinierten Variante innerhalb einer .NET Anwendung!

Teilaufgabe O3: (10 Punkte)

Formulieren Sie ein Code-Beispiel zur Fehlerbehandlung, um die in C# eingesetzte Syntax darzustellen. Das Beispiel soll im Hinblick auf die Syntax die verschiedenen zu nennenden Aspekte der Fehlerbehandlung aus den vorangegangenen Teilaufgaben abdecken.

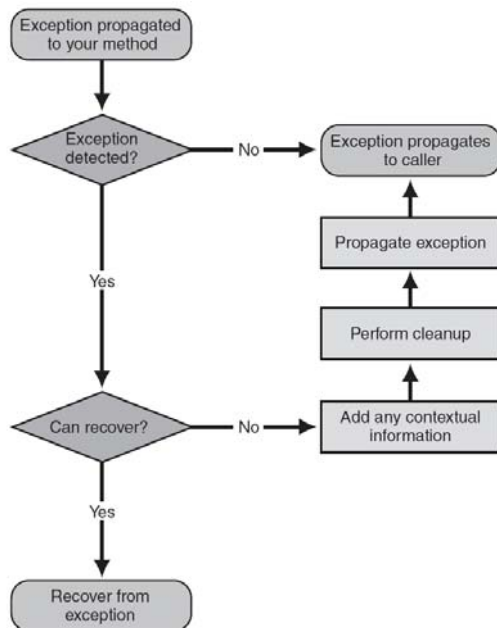
Die inhaltliche Funktion des Beispiels – also der Teil des Codes, der nichts mit der Fehlerbehandlung zu tun hat – muss nicht durch Quellcode dargestellt werden sondern kann in Form von Kommentaren angedeutet werden. Die Kommentare müssen aber entsprechend aussagekräftig sein, damit nachvollziehbar ist, warum dieser Programmteil für die Fehlerbehandlung von Bedeutung ist.

Aufgabe P: Fehlerbehandlung

Teilaufgabe P1:

(20 Punkte)

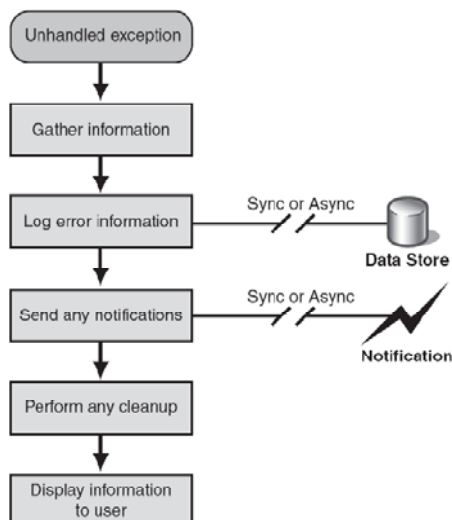
Im .NET-Framework wird dem Entwickler ein Mechanismus zur Fehlerbehandlung bereitgestellt. Skizzieren Sie zunächst kurz die grundlegenden Aspekte der Fehlerbehandlung im .NET Framework! Erläutern Sie anschließend anhand der folgenden Abbildung ausführlich die Vorgehensweise bei der Implementierung dieses Mechanismus! Erwähnen Sie dabei ebenfalls konkrete inhaltliche Anwendungsgebiete, um die Ausführungen zu stützen! (Code-Beispiele sind nicht erforderlich).



Teilaufgabe P2:

(10 Punkte)

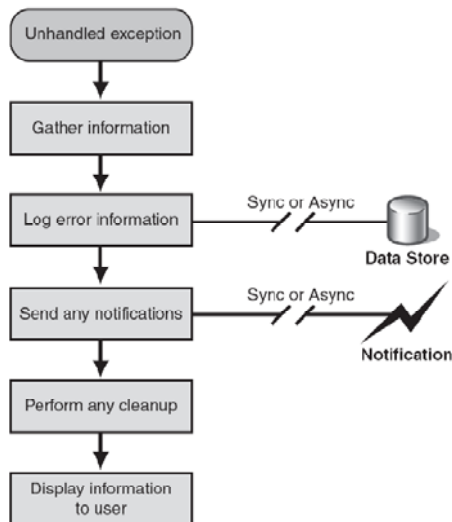
Die Abbildung zeigt die Bearbeitungsschritte im Falle eines nicht behandelten Fehlers. Erläutern Sie die einzelnen Schritte und nennen Sie einige Informationsarten, die im Fehlerfall erfasst werden, um hilfreiche Meldungen zusammenzustellen!



Aufgabe Q: Fehlerbehandlung

Teilaufgabe Q1: (10 Punkte)

Die Abbildung zeigt die Bearbeitungsschritte im Falle eines nicht behandelten Fehlers. Erläutern Sie die einzelnen Schritte und nennen Sie einige Informationsarten, die im Fehlerfall erfasst werden, um hilfreiche Meldungen zusammenzustellen!



Teilaufgabe Q2: (5 Punkte)

Welche Zielsetzung verfolgt die *Enterprise Library*?

Welche Bestandteile umfasst die *Enterprise Library*? (Bitte aggregieren und nur durch zwei charakterisierende Beispiele ergänzen. Eine Nennung jedes einzelnen Elementes ist nicht erforderlich.)

Teilaufgabe Q3: (15 Punkte)

Erläutern Sie, wie der Einsatz des *Exception Handling Building Blocks* die Fehlerbehandlung sowohl professionalisiert als auch vereinfacht.

Unterstützen Sie Ihre Ausführungen durch einen exemplarischen Catch-Block (der den *Exception Handling Building Block* nutzt) und erläutern Sie alle in diesem Beispiel verwendeten, relevanten Bestandteile.

Nennen Sie die Schritte, die erforderlich sind, um einen eigenen Logging-Handler zu implementieren und zu verwenden.

Aufgabe R: XML-Serialisierung

Teilaufgabe R1:

(10 Punkte)

Implementieren Sie eine Assembly, die einen Bestellauftrag abbildet.

Innerhalb des Namespaces `Finance` sollen die Klassen `PurchaseOrder`, `Address` und `OrderItem` implementiert werden. Beachten Sie bei der Implementierung die zweite Teilaufgabe, damit Sie die Anforderungen frühzeitig bedenken oder vor den jeweiligen Elementen entsprechend eine Zeile frei lassen.

Für die Klassen gelten die folgenden Anforderungen:

- Die Klasse `Address` ist gegeben und muss nicht implementiert werden. Sie enthält die folgenden öffentlichen Felder vom Typ `string`
 - `Name`, `Street`, `Zip`, `City`
- Die Klasse `OrderItem` enthält die folgenden Felder
 - `ItemName` vom Typ `string`
 - `Description` vom Typ `string`
 - `UnitPrice` vom Typ `decimal`
 - `Quantity` vom Typ `int`
- Die Klasse `OrderItem` enthält das folgende Property zur Berechnung des Gesamtpreises
 - `LineTotal` vom Typ `decimal`
- Die Klasse `PurchaseOrder` enthält die folgenden Felder
 - `ShipTo` vom Typ `Address`
 - `OrderDate` vom Typ `string`
 - `OrderedItems` vom Typ `Array` von `OrderItem`
 - `ShipCost` vom Typ `decimal`
- Die Klasse `PurchaseOrder` enthält die folgenden Properties zur Berechnung der Gesamtpreise
 - `SubTotal` vom Typ `decimal` (Summe aller enthaltenen `OrderItems`)
 - `TotalCost` vom Typ `decimal` (Summe von `SubTotal` und `ShipCost`)
- Alle Felder können öffentlich verfügbar sein. Es müssen keine zusätzlichen Properties implementiert werden.

Teilaufgabe R2:

(15 Punkte)

Implementieren Sie innerhalb der Klasse `PurchaseOrder` zwei Methoden zur Serialisierung bzw. Deserialisierung der Klasse. Die Methoden sollen den folgenden Signaturen entsprechen:

- `void ExportToXml(string fileName)`
- `PurchaseOrder ImportFromXml(string fileName)`

Passen Sie zusätzlich die Implementierung der Klasse an, damit die Ausgabe des XML-Dokumentes der Abbildung 1 entspricht. Ohne den Einsatz von Attributen entspricht die Ausgabe der Abbildung 2. In der Tabelle können Sie mögliche Attribute nachschlagen.

```
<?xml version="1.0" encoding="utf-8" ?>
- <PurchaseOrder xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" OrderDate="2005-08-02">
  <ShipTo Name="Hugo Müller" Street="Zum Irrtum 12" Zip="37121" City="Velbert-Neviges" />
- <Items>
  <Item ItemName="Asus MyPal 610" Description="PDA" UnitPrice="270.65" Quantity="1" />
  <Item ItemName="Rey Color A4 100" UnitPrice="8.99" Quantity="5" />
</Items>
  <ShipCost>4.55</ShipCost>
</PurchaseOrder>
```

Abbildung 1 - Zielformat der XML-Datei

```
<?xml version="1.0" encoding="utf-8" ?>
- <PurchaseOrder xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ShipTo Name="Hugo Müller" Street="Zum Irrtum 12" Zip="37121" City="Velbert-Neviges" />
  <OrderDate>2005-08-02</OrderDate>
- <OrderItems>
  - <OrderItem>
    <ItemName>Asus MyPal 610</ItemName>
    <Description>PDA</Description>
    <UnitPrice>270.65</UnitPrice>
    <Quantity>1</Quantity>
  </OrderItem>
  - <OrderItem>
    <ItemName>Rey Color A4 100</ItemName>
    <UnitPrice>8.99</UnitPrice>
    <Quantity>5</Quantity>
  </OrderItem>
</OrderItems>
  <ShipCost>4.55</ShipCost>
</PurchaseOrder>
```

Abbildung 2 - Format der XML-Datei ohne Einsatz von Attributen

Attribut	Betrifft	Bedeutung
XmlAnyAttributeAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von XmlAttribute -Objekten zurückgegeben wird.	Beim Deserialisieren wird das Array mit XmlAttribute -Objekten aufgefüllt, die für alle im Schema unbekannten XML-Attribute stehen.
XmlAnyElementAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von XmlElement -Objekten zurückgegeben wird.	Beim Deserialisieren wird das Array mit XmlElement -Objekten aufgefüllt, die für alle im Schema unbekannten XML-Elemente stehen.
XmlArrayAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von komplexen Objekten zurückgegeben wird.	Die Member des Arrays werden als Member eines XML-Arrays generiert.
XmlArrayItemAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von komplexen Objekten zurückgegeben wird.	Die abgeleiteten Typen, die in ein Array eingefügt werden können. Wird i. d. R. im Zusammenhang mit einem XmlArrayAttribute -Objekt angewendet.
XmlAttributeAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Der Member wird als XML-Attribut serialisiert.
XmlChoiceIdentifierAttribu	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Der Member kann durch Verwenden einer Enumeration eindeutig bestimmt werden.
XmlElementAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Das Feld oder die Eigenschaft wird als XML-Element serialisiert.
XmlEnumAttribute	Öffentliches Feld, das ein Enumerationsbezeichner ist.	Der Membername eines Enumerationsmembers.
XmlIgnoreAttribute	Öffentliche Eigenschaften und Felder.	Die Eigenschaft oder das Feld wird beim Serialisieren der enthaltenden Klasse ignoriert.
XmlIncludeAttribute	Öffentliche abgeleitete Klassendeklarationen und Rückgabewerte öffentlicher Methoden (für WSDL-Dokumente, Web Service Description Language).	Die Klasse wird beim Generieren von Schemas eingeschlossen (und daher bei der Serialisierung erkannt).
XmlRootAttribute	Deklarationen öffentlicher Klassen.	Steuert die XML-Serialisierung des Attributziels als XML-Stammelement. Mit diesem Attribut können Sie Namespace und Elementnamen genauer angeben.
XmlTextAttribute	Öffentliche Eigenschaften und Felder.	Die Eigenschaft oder das Feld soll als XML-Text serialisiert werden.
XmlTypeAttribute	Deklarationen öffentlicher Klassen.	Der Name und Namespace des XML-Typs.

Aufgabe S: XML-Serialisierung

Teilaufgabe S1:

(30 Punkte)

Implementieren Sie eine Assembly, die das Pendant zum abgebildeten XML-Dokument bildet. Das XML-Dokument entspricht einem Austauschformat mit einem internationalen Parterunternehmen. Im Gegensatz zum englischsprachigen XML-Dokument erfolgt die Implementierung unter Berücksichtigung der unternehmens-internen Namensgebung, die deutsche Bezeichner für Klassen, Felder, etc. vor-schreibt.

Berücksichtigen Sie bei der Implementierung die Einordnung in einen geeigneten Namespace, eine geeignete Bezeichnung der Felder, Klassen, etc. Treffen Sie für komplexe Elemente der XML-Struktur entsprechende Implementierungsentscheidungen, um eine möglichst hohe Deckung zwischen XML-Struktur und Klasse zu erreichen.

Beachten Sie bei der Implementierung, dass einige Einträge innerhalb einer Klasse als Methoden bzw. Properties bereitgestellt werden und dynamisch berechnet werden sollen. Treffen Sie angemessene Entscheidungen bzgl. der Datentypen. Es ist nicht erforderlich, einfache Felder als Properties zu kapseln.

Die Klasse soll zwei Methoden bereitstellen, die die Serialisierung und Deserialisierung kapseln. (ExportToXml(string fileName); ImportFromXml(string fileName))

Nutzen Sie XML-Attribute um zu erreichen, dass die Serialisierung dem abgebildeten Ergebnis entspricht. In der Tabelle können Sie hierzu mögliche Attribute nach-schlagen.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <PurchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.cpandl.com">
4   <ShipTo Surname="Schuler" Forename="Peter M.">
5     <Street>Universitätsstr. 9</Street>
6     <City>Essen</City>
7     <Zip>45141</Zip>
8   </ShipTo>
9   <OrderDate>2006-02-09</OrderDate>
10  <Items>
11    <OrderedItem>
12      <ItemName>Braun WK210 Sahara</ItemName>
13      <Description>Wasserkocher - kabellos</Description>
14      <UnitPrice>30.00</UnitPrice>
15      <Quantity>5</Quantity>
16      <LineTotal>150.00</LineTotal>
17    </OrderedItem>
18    <OrderedItem>
19      <ItemName>Philips HP 4841 Compact</ItemName>
20      <Description>Haartrockner - kabelgebunden</Description>
21      <UnitPrice>12.50</UnitPrice>
22      <Quantity>2</Quantity>
23      <LineTotal>25.00</LineTotal>
24    </OrderedItem>
25  </Items>
26  <SubTotal>175.00</SubTotal>
27  <ShipCost>6.95</ShipCost>
28  <TotalCost>181.95</TotalCost>
29 </PurchaseOrder>

```

Abbildung 3 - Zielformat der XML-Datei

Attribut	Betrifft	Bedeutung
XmlAnyAttributeAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von XmlAttribute -Objekten zurückgegeben wird.	Beim Deserialisieren wird das Array mit XmlAttribute -Objekten aufgefüllt, die für alle im Schema unbekannten XML-Attribute stehen.
XmlAnyElementAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von XmlElement -Objekten zurückgegeben wird.	Beim Deserialisieren wird das Array mit XmlElement -Objekten aufgefüllt, die für alle im Schema unbekannten XML-Elemente stehen.
XmlArrayAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von komplexen Objekten zurückgegeben wird.	Die Member des Arrays werden als Member eines XML-Arrays generiert.
XmlArrayItemAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von komplexen Objekten zurückgegeben wird.	Die abgeleiteten Typen, die in ein Array eingefügt werden können. Wird i. d. R. im Zusammenhang mit einem XmlArrayAttribute -Objekt angewendet.
XmlAttributeAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Der Member wird als XML-Attribut serialisiert.
XmlChoiceIdentifierAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Der Member kann durch Verwenden einer Enumeration eindeutig bestimmt werden.
XmlElementAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Das Feld oder die Eigenschaft wird als XML-Element serialisiert.
XmlEnumAttribute	Öffentliches Feld, das ein Enumerationsbezeichner ist.	Der Membername eines Enumerationsmembers.
XmlIgnoreAttribute	Öffentliche Eigenschaften und Felder.	Die Eigenschaft oder das Feld wird beim Serialisieren der enthaltenden Klasse ignoriert.
XmlIncludeAttribute	Öffentliche abgeleitete Klassendeklarationen und Rückgabewerte öffentlicher Methoden (für WSDL-Dokumente, Web Service Description Language).	Die Klasse wird beim Generieren von Schemas eingeschlossen (und daher bei der Serialisierung erkannt).
XmlRootAttribute	Deklarationen öffentlicher Klassen.	Steuert die XML-Serialisierung des Attributziels als XML-Stammelement. Mit diesem Attribut können Sie Namespace und Elementnamen genauer angeben.
XmlTextAttribute	Öffentliche Eigenschaften und Felder.	Die Eigenschaft oder das Feld soll als XML-Text serialisiert werden.
XmlTypeAttribute	Deklarationen öffentlicher Klassen.	Der Name und Namespace des XML-Typs.

Aufgabe T: XML-Serialisierung

Teilaufgabe T1:

(30 Punkte)



Implementieren Sie eine Assembly „Warehouse“, die für eine Software eingesetzt werden soll, die die oben abgebildete Struktur zur Verwaltung von Waren einsetzt.

- Die ersten 4 Ebenen sollen als Klassenhierarchie implementiert werden; die fünfte Ebene stellt die Instanzen dar; die letzte Ebene die Felder der Klasse.
- Die ‚Funktionalität‘ der einzelnen Elemente soll in der abstrakten Basisklasse „Ware“ implementiert werden.
- Alle abgeleiteten Klassen dienen lediglich der eindeutigen Typisierung und des Aufbaus der Hierarchie.
- Für die ersten drei Ebenen soll verhindert werden, dass Instanzen der Klassen erzeugt werden können.
- Für die Klassen in der Hierarchie müssen Sie keinen Konstruktor implementieren! (Gehen Sie davon aus, dass diese Aufgabe durch einen anderen Entwickler erledigt wird.)
- Neben den aus der Abbildung ersichtlichen Klassen soll eine Klasse „WarehouseAdministration“ erstellt werden, die Waren über eine Collection verwaltet. Die Klasse soll zwei Methoden bereitstellen, die die Serialisierung und Deserialisierung kapseln (`ExportToXml(string fileName); ImportFromXml(string fileName)`).
- Nutzen Sie in der Assembly „Warehouse“ XML-Attribute um zu erreichen, dass
 - der Name einer Ware als Attribut serialisiert wird und
 - die Bestandteile der Collection als „Ware“ serialisiert werden.

In der Tabelle können Sie hierzu mögliche Attribute nachschlagen.

Attribut	Betrifft	Bedeutung
XmlAnyAttributeAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von XmlAttribute -Objekten zurückgegeben wird.	Beim Deserialisieren wird das Array mit XmlAttribute -Objekten aufgefüllt, die für alle im Schema unbekannten XML-Attribute stehen.
XmlAnyElementAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von XmlElement -Objekten zurückgegeben wird.	Beim Deserialisieren wird das Array mit XmlElement -Objekten aufgefüllt, die für alle im Schema unbekannten XML-Elemente stehen.
XmlArrayAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von komplexen Objekten zurückgegeben wird.	Die Member des Arrays werden als Member eines XML-Arrays generiert.
XmlArrayItemAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert, wodurch ein Array von komplexen Objekten zurückgegeben wird.	Die abgeleiteten Typen, die in ein Array eingefügt werden können. Wird i. d. R. im Zusammenhang mit einem XmlArrayAttribute -Objekt angewendet.
XmlAttributeAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Der Member wird als XML-Attribut serialisiert.
XmlChoiceIdentifierAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Der Member kann durch Verwenden einer Enumeration eindeutig bestimmt werden.
XmlElementAttribute	Öffentliches Feld, Eigenschaft, Parameter oder Rückgabewert.	Das Feld oder die Eigenschaft wird als XML-Element serialisiert.
XmlEnumAttribute	Öffentliches Feld, das ein Enumerationsbezeichner ist.	Der Membername eines Enumerationsmembers.
XmlIgnoreAttribute	Öffentliche Eigenschaften und Felder.	Die Eigenschaft oder das Feld wird beim Serialisieren der enthaltenden Klasse ignoriert.
XmlIncludeAttribute	Öffentliche abgeleitete Klassendeklarationen und Rückgabewerte öffentlicher Methoden (für WSDL-Dokumente, Web Service Description Language).	Die Klasse wird beim Generieren von Schemas eingeschlossen (und daher bei der Serialisierung erkannt).
XmlRootAttribute	Deklarationen öffentlicher Klassen.	Steuert die XML-Serialisierung des Attributziels als XML-Stammelement. Mit diesem Attribut können Sie Namespace und Elementnamen genauer angeben.
XmlTextAttribute	Öffentliche Eigenschaften und Felder.	Die Eigenschaft oder das Feld soll als XML-Text serialisiert werden.
XmlTypeAttribute	Deklarationen öffentlicher Klassen.	Der Name und Namespace des XML-Typs.

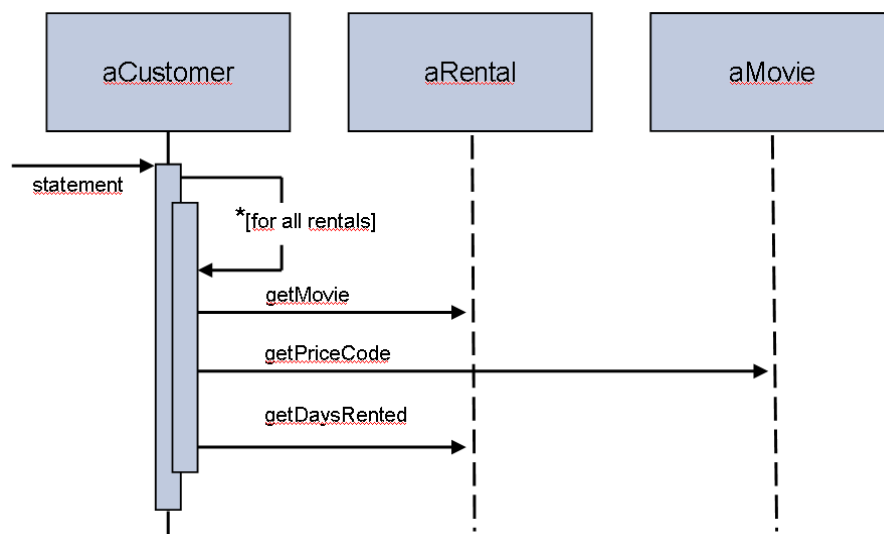
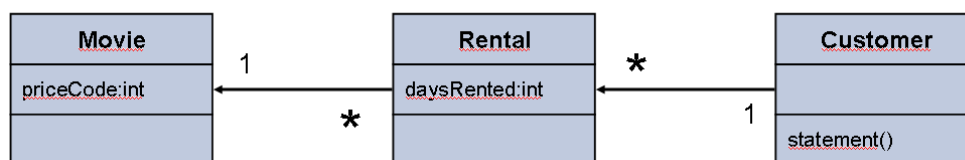
Aufgabe U: Refactoring

Teilaufgabe U1: (10 Punkte)

Erläutern, was sich hinter dem Konzept des Refaktorisierens verbirgt. Gehen Sie bei Ihren Ausführungen auch auf wichtige Regeln ein, die beim Refaktorisieren befolgt werden sollen.

Teilaufgabe U2: (10 Punkte)

Erläutern Sie anhand der gegebenen Materialien (2 Abbildungen, 1 Codeauszug), welchen Einschränkungen dieses „pseudo-objektorientierten“ Programm in Hinblick auf die Wartbarkeit und Erweiterbarkeit unterliegt.



```

11 public class Customer
12 {
13     private string name;
14     private ArrayList rentals = new ArrayList();
15
16     public Customer (string name)
17     {
18         this.name = name;
19     }
20     public void addRental(Rental arg)
21     {
22         rentals.Add(arg);
23     }
24     public string Name
25     {
26         get {return name;}
27     }
28
29     public string Statement()
30     {
31         double totalAmount = 0;
32         int frequentRenterPoints = 0;
33
34         string result = "Rental Record for " + this.Name + "\n";
35
36         foreach (Rental aRental in rentals)
37         {
38             double thisAmount = 0;
39
40             //Beträge pro Zeile ermitteln
41             switch (aRental.Movie.PriceCode)
42             {
43                 case Movie.Type.REGULAR:
44                     thisAmount += 2;
45                     if (aRental.DaysRented > 2)
46                         thisAmount += (aRental.DaysRented - 2) * 1.5;
47                     break;
48                 case Movie.Type.NEW_RELEASE:
49                     thisAmount += aRental.DaysRented * 3;
50                     break;
51                 case Movie.Type.CHILDRENS:
52                     thisAmount += 1.5;
53                     if (aRental.DaysRented > 3)
54                         thisAmount += (aRental.DaysRented - 3) * 1.5;
55                     break;
56             }
57             // Bonuspunkte aufaddieren
58             frequentRenterPoints ++;
59             // Bonuspunkte für zweitägige Ausleihe einer Neuerscheinung
60             if ((aRental.Movie.PriceCode == Movie.Type.NEW_RELEASE) &&
61                 aRental.DaysRented > 1) frequentRenterPoints ++;
62             // Zahlen für diese Ausleihe ausgeben
63             result += "\t" + aRental.Movie.Title + "\t" +
64                 thisAmount + "\n";
65             totalAmount += thisAmount;
66         }
67         // Fußzeilen einfügen
68         result += "Amount owed is " + totalAmount + "\n";
69         result += "You earned " + frequentRenterPoints +
70             " frequent renter points";
71         return result;
72     }
73 }
74

```

Teilaufgabe U3:

(10 Punkte)

Erläutern **kurz** Sie 2 Veränderungen des Anwendungsdesigns, die im Rahmen eines Refaktorisierens durchgeführt werden können. Stellen Sie für jede Veränderung kurz die Nachteile der Ausgangssituation dar, die wesentlichen Aspekte der Durchführung der Veränderung sowie die Charakteristika der Zielsituation dar.

Aufgabe V: Vererbung**Teilaufgabe V1: (20 Punkte)**

Nennen und Erläutern Sie die Schlüsselworte, die innerhalb der Vererbung in C# zum Einsatz kommen.

Erklären Sie dabei auch den Unterschied zwischen abstrakten und virtuellen Methoden anhand einfacher Beispiele.

Gehen Sie zusätzlich auf darauf ein, wie sich die Mechanismen des Überschreibens (override) und des Verbergens (hide) in C# auswirken, wenn zu einem späteren Zeitpunkt Veränderungen an der Basisklasse vorgenommen werden.

Teilaufgabe V2: (10 Punkte)

Zeichnen Sie eine frei zu wählende - für die folgende Aufgabe angemessene - Vererbungshierarchie.

Erläutern Sie anschließend anhand dieser Darstellung die Objekterzeugung in Vererbungshierarchien! Was ist bei der Implementierung von Konstruktoren zu beachten? Gehen Sie darüber hinaus kurz darauf ein, welche Restriktionen bei Type Casts innerhalb von Vererbungshierarchien zu beachten sind!

Aufgabe W: Metadaten

Sie sind an einem Projekt beteiligt, in dem ein Framework für O/R-Mapping implementiert werden soll. Ihr Vorschlag, Metadaten zur Steuerung des Frameworks einzusetzen wurde von der Projektleitung angenommen. Die Methoden zur Verarbeitung der Metadaten sind bereits von einem Kollegen auf Basis einer theoretischen Spezifikation implementiert worden.

Die Implementierung könnte u.a. folgendes SQL-Script erzeugen:

```
2 CREATE TABLE [dbo].[contact] (  
3     [id] [int] IDENTITY (1, 1) NOT NULL,  
4     [email] [varchar] (80) NOT NULL,  
5     [address] [varchar] (80) NOT NULL,  
6     [city] [varchar] (50) NOT NULL,  
7     [name] [varchar] (50) NOT NULL,  
8     [age] [int] NOT NULL  
9 ) ON [PRIMARY]  
10 GO  
11
```

Teilaufgabe W1: (18 Punkte)

Ihre Aufgabe ist es, drei für das O/R-Mapping der Klasse Contact benötigte Attribute in einer Assembly zu implementieren. Die Attribute sollen die benötigten Metadaten abbilden, um ein Mapping zwischen Objekten und relationalen Datentabellen zu erreichen.

Die Implementierung der Attribute soll den Anforderungen an Attributimplementierung genügen (Serialisierbarkeit ist nicht erforderlich) und abhängig vom Attribut die benötigten Member bereitstellen. Versäumen Sie nicht den gezielten Einsatz des AttributeUsage-Attributes.

Teilaufgabe W2: (12 Punkte)

Implementieren zusätzlich Sie eine Klasse Contact, die mit den Attributen versehen ist und die das Pendant zum SQL-Skript darstellen kann. Neben den Feldern, Properties und Attributen sind keine weiteren Funktionen erforderlich - Es genügt die Klasse. Assembly-Informationen (using, namespace, etc.) sind an dieser Stelle nicht erforderlich.

Aufgabe X: Metadaten und Delegates**Teilaufgabe X1: (15 Punkte)**

Erläutern Sie, wo bei der Entwicklung mit dem .NET-Framework Metadaten zum Einsatz kommen. Gehen Sie dabei u.a. die folgenden Aspekte ein:

- Welche Relevanz besitzen Metadaten im Kontext der Software-Entwicklung?
- Welche Metadaten werden bei der Entwicklung mit C# „automatisch“ bereitgestellt?
- Über welche Möglichkeiten zur Nutzung und zur Einflussnahme verfügt der Entwickler?

Teilaufgabe X2: (10 Punkte)

Erläutern Sie die wesentlichen Grundlagen der Delegates! Beschreiben Sie dazu zunächst, was Delegates sind. Gehen Sie anschließend auf drei Einsatzgebiete für Delegates ein.

Teilaufgabe X3: (5 Punkte)

Ergänzen Sie Ihre Ausführungen aus Teilaufgabe A1.2 mit einem kurzen Code-Snippet, das die Deklaration und den Einsatz eines Delegates aufzeigt!

Aufgabe Y: Generics

Teilaufgabe Y1: (5 Punkte)

Nennen Sie drei Vorteile von Generics.

Teilaufgabe Y2: (10 Punkte)

Notieren Sie ein kurzes Code-Snippet, das den Einsatz von Generics anhand einer Collection aus den Basisklassen veranschaulicht und sowohl

- Deklaration,
- Instanziierung der Liste,
- Hinzufügen von Elementen und
- den Aufruf einer Methode eines Elements aus der Liste

umfasst. Wählen Sie für das Beispiel einen geeigneten Typ als Element aus.

Teilaufgabe Y3: (15 Punkte)

Implementieren Sie eine Assembly, die eine generische Klasse `DocumentManager` enthält. Die Klasse `DocumentManager` soll über die Mechanismen der Generics Dokumente unterschiedlichen Typs in einer Collection verwalten können. Als öffentliche Member sollen

- eine Add-Methode zum Hinzufügen von Dokumenten
- eine Get-Methode zum Auslesen von Dokumenten
- Eine `IsDocumentAvailable`, die `false` zurückgibt, wenn die Collection leer ist (und umgekehrt)

Implementiert werden.