STUDIENSEMINAR FÜR LEHRÄMTER

AN SCHULEN KREFELD

SEMINAR FÜR DAS LEHRAMT AM BERUFSKOLLEG

Schriftlicher Unterrichtsentwurf

Fachrichtung:	Technische Informatik
Fach:	System- und Anwendungssoftware (SYA)
Lernfeld:	
Thema:	„Abfangen von falschen oder unerwarteten Anwendereingaben in einem Formular mit try-catch-Blöcken (C#) zur Vermeidung von Laufzeitfehlern“
Kurze Zusammenfassung	Die Schüler sollen heute die ihnen bekannte Software Rundstahl auf Laufzeitfehler testen, die durch falsche oder unerwartete Anwendereingaben in einer Textbox (C#) entstehen können, diese dokumentieren und sie durch Integration eines try-catch-Blockes (C#) „unschädlich“ machen. Das try-finally Konstrukt wird nicht eingeführt. Die Umsetzung der catch-Anweisung, wie sie im Auszug des Skripts zu finden ist und die darin enthaltene Parameterübergabe sowie die Erzeugung eines Objekts der Klasse Exception werden aufgrund des bisher nicht eingeführten Konzepts objektorientierter Programmiersprachen nicht im Detail thematisiert.
Datum:	Mittwoch, 04.03.2009
Bildungsgang/Stufe:	THE83 (Zweijährige Berufsfachschule Fachrichtung Technik – Fachlicher Schwerpunkt Elektrotechnik, Profilbildung Informations- und Kommunikationstechnik)
Autor: (freiwillig)	Andreas Besener

Lernvoraussetzungen im Hinblick auf die Unterrichtsstunde

1.1 Rahmenbedingungen

Die Unterrichtsstunde wird in der Klasse THE83 (Unterstufe) stattfinden. Die Schüler dieser Lerngruppe besuchen die „Zweijährige Berufsfachschule Fachrichtung Technik – Fachlicher Schwerpunkt Elektrotechnik, Profilbildung Informations- und Kommunikationstechnik“ seit August 2008. Der Unterricht findet im Fach System- und Anwendungssoftware[1] statt, das inzwischen wöchentlich mit fünf (im ersten Halbjahr nur 4) Unterrichtsstunden im Stundenplan vorgesehen ist. Das Fach ist dem fachlichen Schwerpunkt des berufsbezogenen Bereichs zuzuordnen (vgl. Bezirksregierung Düsseldorf 2001, S. 167). In der Zeit von Oktober 2008 bis Januar 2009 unterrichtete ich die THE83 in SYA wöchentlich eine Doppelstunde. Mit dem neuen Schulhalbjahr ist eine 3. Unterrichtsstunde am Mittwochmorgen dazugekommen, die im zweiwöchentlichen Rhythmus zu einer Doppelstunde gebündelt wird. Die Beteiligung der Schüler am Unterricht ist nach wie vor gut. Die Lerngruppe entspricht der, bei der auch der 1. Unterrichtsbesuch stattgefunden hat. Nähere Angaben zu den Sozialdaten sind Unterrichtsentwurf 1 zu entnehmen (siehe Besener 2008). Die einzige Schülerin der Klasse wechselte zu Beginn des zweiten Schulhalbjahrs in die gymnasiale Oberstufe. Der Wechsel hat aus meiner Sicht keinen Einfluss auf den Unterricht im Fach SYA.

Leistungsfähigkeit und Fachkompetenz sind bei den Lernenden unterschiedlich stark ausgeprägt. Drei Schüler, die auch bei der Klassenarbeit im vergangenen November unterdurchschnittliche Leistungen erbrachten, bedürfen auch zum jetzigen Zeitpunkt einer besonderen Unterstützung. Demgegenüber stehen drei Lernende, deren Leistungen weit über dem Durchschnitt liegen. Die Bearbeitungszeit von Arbeitsaufträgen und auch die Qualität der Lösungen differieren zwischen diesen beiden „Extremgruppen“ zum Teil sehr stark.

Die Sozialkompetenz der Schüler ist, wie sie sich z. B. in der Zusammenarbeit in Gruppen zeigt, zum jetzigen Zeitpunkt gut ausgeprägt und bedarf diesbezüglich keiner besonderen Förderung. Der Umgang unter den Lernenden ist respektvoll und es kommt ganz selten zu Konflikten. Bei der Bearbeitung von Arbeitsaufträgen unterstützen sich die Lernenden gegenseitig. Nur befriedigend ist zum jetzigen Zeitpunkt hingegen die Fehlerkultur in der Klasse ausgeprägt. Es fällt den Lernenden noch sehr schwer, über ihre Lernmisserfolge vor den Mitschülern Auskunft zu geben. – Wenn etwas nicht verstanden wurde, spricht man lieber nicht darüber. Bisher habe ich die Schüler ermutigt, offen über ihre Probleme in Bezug auf den eigenen Lernerfolg zu reden. Aus diesen Impulsen sind sehr gute Lernsituationen entstanden, die den meisten Schülern der Lerngruppe einen Lernfortschritt ermöglichten. Allerdings waren bisher nur zwei Schüler so mutig, offen über ihren Lernprozess Auskunft zu geben.

Weitere Informationen:

- Zwei Schüler sind dafür verantwortlich, alle funktionsfähigen Programme in einem Austauschordner zu sammeln, auf den alle Schüler zugreifen können. Die Schüler haben die Möglichkeit, Programme von diesem Ordner aus in ihr privates Verzeichnis zu kopieren, auf das sie auch von zu Hause aus zugreifen können. Die Möglichkeit, erstellte Programme auf einem USB-Stick zu speichern, besteht ebenfalls.
- Der Raum 1221 ist als Computerraum ausgestattet (12 Schüler PCs, Lehrer-PC mit Beamer, Drucker).

[1] Im Folgenden mit SYA abgekürzt.

- Für das Fach SYA wurde kein Schulbuch eingeführt. Stattdessen nutzt die Lerngruppe ein Skript, das von einem Lehrer des Berufskollegs Geldern erstellt wurde[2].

1.2 Vorkenntnisse

In Bezug auf das heutige Unterrichtsthema haben die Lernenden am 04.02.2009 eine Variante kennengelernt, die eine fehlende Eingabe, nicht aber unerwartete oder falsche Eingaben in einer Textbox (C#) abfangen kann. Sie haben das bestehende Programm „Rundstahl“ (C#) um eine zweiseitige Verzweigung ergänzt (Aufgabenstellung, realisiertes Struktogramm und dazugehöriges C#-Programm sind dem Anhang beigefügt; siehe Anlage 4). Die default-Anweisung der Mehrfachauswahl programmierten die Schüler so, dass eine fehlende Auswahl aus der Liste einer Combobox (C#) zu einer Fehlermeldung führte. Die Möglichkeiten und Grenzen der bisher eingeführten Realisierung zum Abfangen fehlender Anwendereingaben in einer Textbox (C#) wurden auf Initiative der Schüler diskutiert. Die Lernenden waren mit der Umsetzung unzufrieden, da Falscheingaben mit der erarbeiteten Lösung nicht „unschädlich“ gemacht werden konnten. Bisher entwickelte Programme, die Formularfelder enthalten (z. B. Textboxen oder Comboboxen (C#)), können durch unsachgemäße Eingaben nach wie vor zum Absturz des Programmes führen. Die Schüler artikulierten ihr Interesse, eine Möglichkeit kennen zu lernen, Falscheingaben „professioneller“ zu behandeln: Sie haben sich mit dem Problem der Laufzeitfehler auf der Basis persönlicher Erfahrungen auseinandergesetzt, ohne dass bisher der Terminus „Laufzeitfehler“ im Unterricht eingeführt wurde. Die anfänglichen Probleme (vgl. Besener 2008) der Schüler im Fach SYA sind größtenteils überwunden. Die Lernenden sind es inzwischen gewohnt, die Entwicklungsumgebung (Microsoft Visual C# 2005 Express Edition) zu bedienen, Lösungen eigenständig zu erarbeiten und diese auch zu präsentieren. Schwierigkeiten haben die meisten Lernenden allerdings damit, die Syntax fehlerfrei umzusetzen. Offensichtliche Syntaxfehler (z. B. fehlendes Semikolon) werden zum Teil nicht erkannt, da Fehlermeldungen der Entwicklungsumgebung nicht zur Kenntnis genommen oder falsch interpretiert werden. Auftretende Syntaxfehler werden häufig mit dem Versuch-Irrtum Prinzip angegangen. Nur durch Intervention des Lehrers sind die Schüler zu einer Fehlersuche zu bewegen, die auf die Klärung der Ursache des Syntaxfehlers abzielt.

2 Didaktisch/methodischer Schwerpunkt

2.1 Curriculare Anbindung

Für den betrachteten Bildungsgang liegt im Land Nordrhein-Westfalen zurzeit kein Lehrplan vor, sondern es gibt nur sogenannte „Curriculare Hinweise zu den Bildungsgängen der zweijährigen Berufsfachschule“ die zu erweiterten beruflichen Kenntnissen und der Fachhochschulreife führen (vgl. Bezirksregierung Düsseldorf 2001, S. 167). Zu den Fächern des fachlichen Schwerpunkts des berufsbezogenen Bereichs gehört das Fach SYA, in dem der Unterricht am 04.03.2009 stattfinden wird. Die Lernenden sollen in diesem Fach u. a. Qualifikationen zur Programmierung betrieblicher Anwendungen in einer objektorientierten Sprache unter Verwendung marktgängiger Systeme und Software erwerben (vgl. Bezirksregierung Düsseldorf 2001, S. 167). In den curricularen Hinweisen ist diesbezüglich für die Klasse 11 die strukturierte Programmierung unter Berücksichtigung der Projektierung (Softwareentwicklungsprozess) vorgesehen (vgl. Bezirksregierung Düsseldorf 2001, S. 169). Die Phase des Testens (vgl. Bezirksregierung Düsseldorf 2001, S. 169) wird am 04.03.2009 simuliert.

In der didaktischen Jahresplanung des Berufskollegs Geldern sind für das Fach SYA u. a. die Einführung in die strukturierte Programmierung und das Erstellen einer Windowsanwendung mit dem Visu-

[2] Und zwar: Lentz 2007.

al Studio vorgesehen (vgl. Berufskolleg Geldern 2008; Anlage 1). Der Softwareentwicklungsprozess mit der Phase des Testens kann diesem Teil der didaktischen Jahresplanung zugeordnet werden.

2.2 Einordnung der Unterrichtsstunde in den unterrichtlichen Kontext

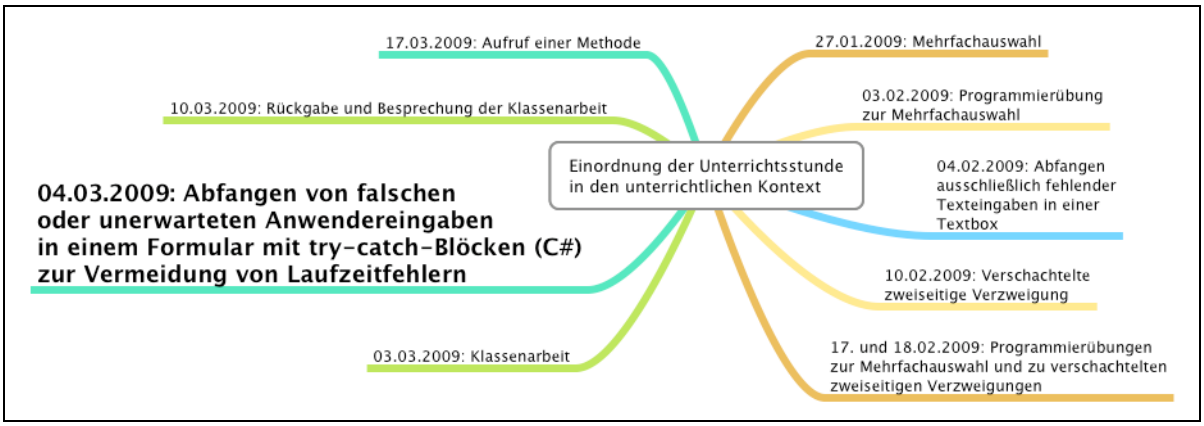

Abbildung 1: Einordnung der Unterrichtsstunde in den unterrichtlichen Kontext.

2.3 Fachlich-, methodischer Schwerpunkt der Unterrichtsstunde

Die Schüler sollen heute die ihnen bekannte Software Rundstahl auf Laufzeitfehler testen, die durch falsche oder unerwartete Anwendereingaben in einer Textbox (C#) entstehen können, diese dokumentieren und sie durch Integration eines try-catch-Blockes (C#) „unschädlich“ machen. Das try-finally Konstrukt wird nicht eingeführt. Die Umsetzung der catch-Anweisung, wie sie im Auszug des Skripts zu finden ist und die darin enthaltene Parameterübergabe sowie die Erzeugung eines Objekts der Klasse Exception werden aufgrund des bisher nicht eingeführten Konzepts objektorientierter Programmiersprachen nicht im Detail thematisiert.[3] Die Unterscheidung zwischen Syntax- und Laufzeitfehlern ist ebenfalls nicht Inhalt der Unterrichtsstunde. Der Begriff des Laufzeitfehlers wird eingeführt, der Begriff Syntaxfehler wird hingegen nicht weiter behandelt. Die Bandbreite möglicher Ursachen für Laufzeitfehler (fehlerhafte Speicherreservierung; Dateizugriffe auf Dateien, die nicht vorhanden sind…) wird nicht weiter aufgegriffen. Die Schüler durchlaufen mit dem Testen des Programms eine wichtige Phase des Softwareentwicklungsprozesses. Die weiteren Phasen (Problemanalyse, Entwurf, Implementierung, Leistungsüberprüfung, Installation und Abnahme (vgl. Meyers Lexikonredaktion 2001, S. 606)) werden heute weder durchlaufen noch sind sie Gegenstand der Betrachtung. Beim Testen wird das Ein-/Ausgabeverhalten eines Programms u. a. mit Experimenten überprüft (vgl. Meyers Lexikonredaktion 2001, S. 662). Dieses Experimentieren wird in der Unterrichtsphase als Problemstellung simuliert, aber nicht zum Objekt des Unterrichts.

Um den Lernprozess der Schüler in Zukunft noch besser unterstützen zu können, gilt es, ein Unterrichtsklima zu schaffen, in dem die Lernenden offen ihre Fragen, ihre Vermutungen, aber auch ihre Unsicherheiten und Ängste in Bezug auf die Unterrichtsinhalte und den eigenen Lernprozess zum Ausdruck bringen können (siehe dazu Kap. 2.1). Das verlangt allerdings gegenseitiges Vertrauen[4] und von den Lernenden den Mut, sich den Mitschülern auch dann offen gegenüber zu äußern, wenn etwas noch unklar ist, oder man etwas offensichtlich nicht verstanden hat. Wie in Kapitel 2.1 dargestellt, gelingt diese Offenheit bisher nur wenigen Schülern der THE83. Mit der Methode Bienenkorb (vgl. MSW 2009) werden die Schüler in der heutigen Unterrichtsstunde ermutigt, offene Fragen zum

[3] Die objektorientierte Programmierung ist als Inhalt erst für die Oberstufe vorgesehen (vgl. Bezirksregierung Düsseldorf 2001, S. 170).
[4] Und das nicht nur zwischen den Schülern, sondern auch zwischen Lehrer und Schülern.

Ausdruck zu bringen. Der Vorteil dieser Methode liegt in Bezug auf die heutige Unterrichtstunde darin, dass sich die Schüler zunächst in Einzelarbeit in einen neuen Inhalt (try-catch-Blöcke) einarbeiten und im Anschluss daran die Gelegenheit bekommen, offene Fragen mit nur einem Partner zu diskutieren. Die Explikation offener Fragen ist fester Bestandteil der Methode. Der Mut, Fragen zu stellen, wächst insbesondere dann, wenn bemerkt wird, dass der Partner ähnliche oder gleiche Fragen hat (vgl. MSW 2009). Die Methode Bienenkorb könnte auch in Gruppen mit drei oder vier Schülern ihre Anwendung finden. Die Entscheidung zur Partnerarbeit erfolgte im Hinblick darauf, dass die Schwelle, offene Fragen zu äußern, größer wird, je mehr Personen in einer Gruppe zusammenarbeiten. Mit der gewählten Methode wird vermieden, dass Schüler offene Fragen direkt vor dem Klassenverband äußern müssen. Die Zusammensetzung der Teams folgt dem Grundgedanken des binnendifferenzierten Unterrichts (vgl. dazu Kapitel 2). Für die Stunde am 04.03.2009 habe ich mich für eine zielhomogene und themengleiche Vorgehensweise entschieden[5]. Die Einteilung der Kleinteams folgt vor allem dem Ansatz der personalen Differenzierung auf der Grundlage der vermuteten Leistungsfähigkeit und der Fachkompetenz der Schüler. Je ein leistungsfähiger und/oder fachkompetenter Schüler wird mit einem schwächeren Schüler kombiniert. Die Zusammenarbeit in wechselnder Partnerarbeit ist den Schülern vertraut und verlief bisher ohne Komplikationen. Die Einteilung der Gruppen wird in der Unterrichtsstunde vor dem Unterrichtsbesuch umgesetzt.

Die Integration des try-catch-Blockes in das bestehende Programm Rundstahlrechner wird in der Erarbeitungsphase nicht durch jedes Team am PC umgesetzt. Die Modifizierung des Quelltextes am PC erfolgt erst in der Auswertungsphase, und zwar gemeinsam im Plenum. Dieses Vorgehen wird in der heutigen Stunde bewusst gewählt, um bei eventuellen Syntaxfehlern Impulse für eine systematische Fehlersuche geben zu können (siehe dazu Kap. 2.2).

3 Ziele des Unterrichts

3.1 Gesamtziel der Unterrichtsstunde

Die Schüler testen ein Programm auf mögliche Laufzeitfehler, die durch falsche oder unerwartete Anwendereingaben entstehen können und nutzen zu ihrer Behebung einen try-catch-Block (C#).

3.2 Angestrebte Fachkompetenzerweiterungen

Die Schüler erweitern ihre Fachkompetenz, Programme so zu entwickeln, dass falsche oder unerwartete Anwendereingaben zur Vermeidung von Laufzeitfehlern abgefangen werden, indem sie Programme systematisch auf ihre Funktion hin testen und auftretende Laufzeitfehler mit try-catch-Blöcken abfangen, die durch falsche oder unerwartete Anwendereingaben bedingt sind.

3.3 Angestrebte Sozialkompetenzerweiterungen

Die Schüler erweitern ihre Sozialkompetenz, indem sie offene Fragen von Mitschülern ohne negative Kommentare aufnehmen und sich konstruktiv an der Beantwortung beteiligen.

[5] Meyer (2004, S. 101 ff.) schlägt als mögliche innere Differenzierungsformen die personale, die didaktische und die pragmatische Differenzierung des Unterrichts vor (Letztere nennt er „Daddeln im Schulalltag“. Ich gebrauche stattdessen den Begriff „pragmatisch“). Ziel- und themenhomogen bedeutet, dass für alle Schüler die gleichen Unterrichtsziele und das gleiche Unterrichtsthema vorgesehen sind (vgl. Meyer, S. 101 ff.).

4 Verlaufsplan

Unterrichtsphasen	Inhalte	Methodische Hinweise	Medien/Materialien
Problemstellung/ Motivation und Problemdefinition	Laufzeitfehler durch falsche oder unerwartete Anwendereingaben in einem Formular (hier: in einer Textbox (C#)).	Im Plenum: Welche Eingaben könnte ein Anwender in einer Textbox tätigen? Wie verhält sich das Programm? Schüler-Schüler Diskussion.	Programm Rundstahl (Version ohne Fehlerbehandlung; Anlage 3), Lehrer-PC mit Beamer, Arbeitstransparent (Anlage 2).
Spontane Verarbeitung	Schülervorschläge zu geeigneten Lösungsmöglichkeiten (z. B. zweiseitige Verzweigung mit Textvergleich zum Abfangen von fehlenden Eingaben).	Im Plenum.	Arbeitstransparent (Anlage 2).
Logische Erarbeitung	Try-catch-Blöcke.	Bienenkorb (vgl. MSW 2009):	Arbeitsblätter 1 und 2 (Anlage 5), erweitertes Programm Rundstahl als erwartetes Ergebnis (Anlage 7), modifizierter Auszug aus dem Skript (Anlage 6), Karten.
	Try-catch-Blöcke.	1. Phase (Einzelarbeit): Informationsaufnahme.	
	Try-catch-Blöcke und in Bezug darauf die offenen Fragen der Kleinteams.	2. Phase (Partnerarbeit): Schüler tauschen sich über den vorliegenden Text aus und klären offene Fragen. Fragen, die in den Kleinteams nicht beantwortet werden können, sollen auf Karten festgehalten werden. Bearbeitung von Aufgabe 2 c).	
Präsentation und Auswertung	• Try-catch-Blöcke zur Vermeidung von Laufzeitfehlern am Beispiel des Programms Rundstahl. • Offene Schülerfragen.	Im Plenum, Schüler-Schüler Diskussion.	Karten, Magnete, Arbeitsblatt 2 (Anlage 5); um try-catch-Block ergänztes Programm als erwartetes Ergebnis (Anlage 7).
Sicherung der Ergebnisse/Verallgemeinerung	• Diskussion der Zweckdienlichkeit von try-catch-Blöcken. • Vereinbarung über ihre zukünftige Verwendung treffen.	Im Plenum; Programm wird vom verantwortlichen Schüler im Austauschordner gespeichert.	Lehrer-PC, Austauschordner auf Schulserver.

Abbildung 2: Synopse

4.1 Anlage 2: Arbeitstransparent

Test des Programms Rundstahl mit unterschiedlichen Anwendereingaben für r und h in der Textbox:

Eingabebeispiele für r und/oder h zu Testzwecken:	**Verhalten des Programms:**		

Tabelle 1: Arbeitstransparent für die Phase der Problemstellung.

Erwartetes Ergebnis:

Eingabebeispiele für r und/oder h zu Testzwecken:	**Verhalten des Programms:**	**Wir kümmern uns heute um die Behandlung von:**	**Lösungsidee?**
fehlende Eingabe in der Textbox	Programm stürzt ab. Fehlermeldung: Methodenargumente besitzen nicht das richtige Format.	✕	zweiseitige Verzweigung mit Textvergleich
23	Berechnung erfolgt.		
0	Berechnung erfolgt.		
23,145	Berechnung erfolgt.		
99999999999999999999999999999	Berechnung erfolgt. Ausgabebereich aber für die Ergebnisanzeige zu klein.		
-25,87	Berechnung erfolgt. Negative Werte für r und h machen keinen Sinn.		
abcd	Programm stürzt ab. Fehlermeldung: Methodenargumente besitzen nicht das richtige Format.	✕	keine
/&%§	Programm stürzt ab. Fehlermeldung: Methodenargumente besitzen nicht das richtige Format.	✕	keine
12a	Programm stürzt ab. Fehlermeldung: Methodenargumente besitzen nicht das richtige Format.	✕	keine

Tabelle 2: Erwartetes Ergebnis.

Die Spalten „Wir kümmern uns heute um die Behandlung von:“ und „Lösungsidee“ werden vom Lehrer ausgefüllt.

4.2 Anlage 3: Programm Rundstahl (Version ohne Fehlerbehandlung)

```
private void button1_Click(object sender, EventArgs e)
        {
            double r, h, volumen, oberflaeche, gewicht;
            string auswahl;

            r = Convert.ToDouble(textBox1.Text);
            h = Convert.ToDouble(textBox2.Text);
            auswahl = comboBox1.Text;
            switch (auswahl)
            {
                case "Oberfläche":
                        oberflaeche = 2 * r * 3.14 * h + 2 * r * r * 3.14;
                        label4.Text = "Die Oberfläche des Rundstahls beträgt: " +
                        oberflaeche.ToString("F2") + " cm².”; break;
                    case "Volumen":
                        volumen = 3.14 * r * r * h;
                        label4.Text = "Das Volumen des Rundstahls beträgt: " +
                        volumen.ToString("F2") + " cm³."; break;
                    case "Gewicht":
                        gewicht = 3.14 * r * r * h * 7.85 / 1000;
                        label4.Text = "Das Gewicht des Rundstahls beträgt: " +
                        gewicht.ToString("F2") + " kg."; break;
                    default:
                        label4.Text = "Bitte eine Auswahl treffen!"; break;
            }
        }
```

4.3 Anlage 4: Programm Rundstahl, das fehlende Eingaben in einer Textbox (C#) abfängt

Aufgabenstellung aus einer vergangenen Unterrichtsstunde:

Aufgabe 5:

Das Programm „Rundstahl“ soll um die Berechnung des Gewichts erweitert werden (siehe Abb. 1). Für die Erweiterung des Programms nutzen Sie bitte die **Mehrfachauswahl**! Die Ausgabe des Gewichts soll in kg erfolgen. Das spezifische Gewicht von Stahl beträgt ca. 7,85 kg/dm^3.

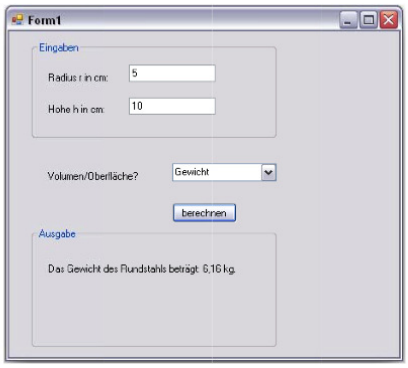

Abbildung 1

a) Leiten Sie zunächst die Formel zur Berechnung des Gewichts in **kg** her!
b) Erzeugen Sie das Struktogramm!
c) Erweitern Sie die grafische Benutzeroberfläche!
d) Geben Sie den Quelltext ein und testen Sie Ihr Programm!
e) Welche Vorteile und welche Nachteile hat eine Mehrfachauswahl? Stellen Sie Vor- und Nachteile tabellarisch gegenüber (siehe Tab. 1)!

Mehrfachauswahl	
Vorteile	Nachteile
…	…

Tabelle 1

f) Das Programm soll zusätzlich so ergänzt werden, dass eine fehlende Eingabe von r und h sowie eine fehlende Auswahl in der ComboBox nicht zum Absturz des Programms führen. Stattdessen soll wie den Abbildungen 2 und 3 zu entnehmen ist, ein Hinweis über die fehlende Eingabe erfolgen. **Hinweis:** Die fehlende Auswahl in der ComboBox können Sie sehr einfach mit Hilfe der Mehrfachauswahl abfangen. **Tipp:** Für das Abfangen der fehlenden Eingabe in TextBox1 und TextBox2 können Sie zwei Hilfsvariablen (pruefenEins, pruefenZwei) und den Vergleichsoperator (==) nutzen! Wie das funktionieren kann, das müssen Sie selbst austüfteln!

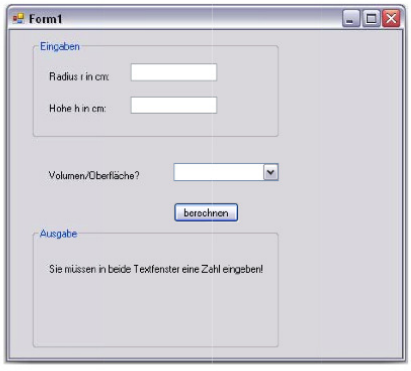

Abbildung 2

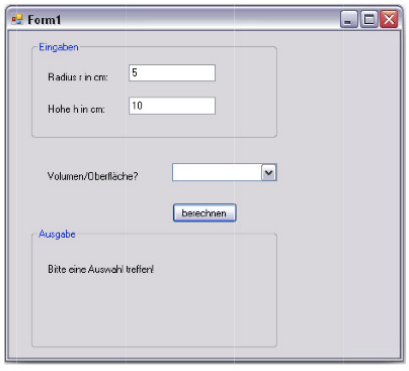

Abbildung 3

Abbildung 3: Aufgabenstellung aus einer vergangenen Unterrichtsstunde.

Struktogramm zu „Aufgabe 5“:

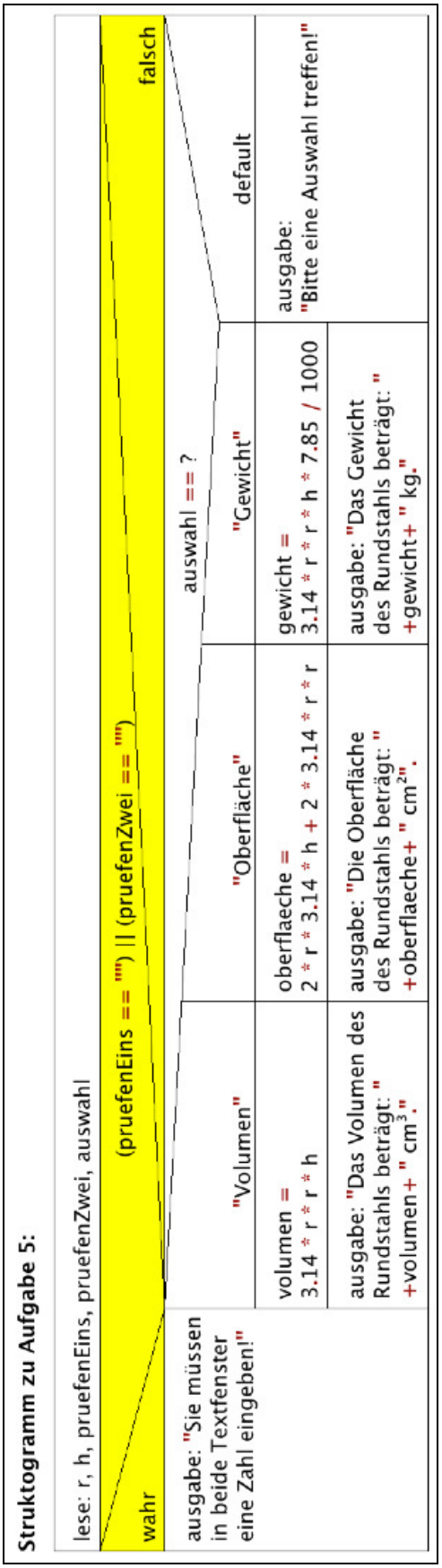

Abbildung 4: Struktogramm zu „Aufgabe 5“.

Fertiges Programm zu „Aufgabe 5“:

```
private void button1_Click(object sender, EventArgs e)
        {
            double r, h, volumen, oberflaeche, gewicht;
            string auswahl, pruefenEins, pruefenZwei;
            pruefenEins = textBox1.Text;
            pruefenZwei = textBox2.Text;
            if (pruefenEins == "" || pruefenZwei == "")
            {
                label4.Text = "Sie müssen in beide Textfenster eine Zahl
                eingeben!";
            }
            else
            {
                r = Convert.ToDouble(textBox1.Text);
                h = Convert.ToDouble(textBox2.Text);
                auswahl = comboBox1.Text;
                switch (auswahl)
                {
                    case "Oberfläche":
                        oberflaeche = 2 * r * 3.14 * h + 2 * r * r * 3.14;
                        label4.Text = "Die Oberfläche des Rundstahls beträgt: " +
                        oberflaeche.ToString("F2") + " cm².";  break;
                    case "Volumen":
                        volumen = 3.14 * r * r * h;
                        label4.Text = "Das Volumen des Rundstahls beträgt: " +
                        volumen.ToString("F2") + " cm³."; break;
                    case "Gewicht":
                        gewicht = 3.14 * r * r * h * 7.85 / 1000;
                        label4.Text = "Das Gewicht des Rundstahls beträgt: " +
                        gewicht.ToString("F2") + " kg."; break;
                    default:
                        label4.Text = "Bitte eine Auswahl treffen!"; break;
                }
            }
        }
```

4.4 Anlage 5: Arbeitsblätter 1 und 2

Arbeitsblatt 1 vom 04.03.2009 (THE83)

Try-catch-Blöcke

Aufgabe 1 (Bearbeitungszeit: 10 Minuten):

a) Bitte lesen Sie in **Einzelarbeit** den vorliegenden Text und markieren Sie die aus Ihrer Sicht wichtigen Stellen!

b) Notieren Sie Fragen zum Text, die beim Lesen entstanden sind:

Abbildung 5: Arbeitsblatt 1.

Arbeitsblatt 2 vom 04.03.2009 (THE83)
Try-catch-Blöcke

Aufgabe 2 (Bearbeitungszeit: 12 Minuten):

a) Bitte diskutieren Sie kurz mit Ihrem Sitznachbarn die offen gebliebenen Fragen, die Sie auf dem Arbeitsblatt 1 notiert haben!

b) Fragen, die Sie auch mit Hilfe des Sitznachbarn nicht beantworten konnten, bitte **groß und gut leserlich** auf Karten schreiben!

c) Ergänzen Sie mit Ihrem Sitznachbarn das Programm Rundstahl um einen try-catch-Block!
Zeichnen Sie dazu Pfeile ausgehend von den drei unten abgebildeten Kästen hin zu der Position im Programm, wo der Inhalt des jeweiligen Kastens aus Ihrer Sicht stehen müsste!

Programm Rundstahl (C#):

```
private void button1_Click(object sender, EventArgs e)
        {
            double r, h, volumen, oberflaeche, gewicht;
            string auswahl;

            r = Convert.ToDouble(textBox1.Text);
            h = Convert.ToDouble(textBox2.Text);
            auswahl = comboBox1.Text;
            switch (auswahl)
            {
                case "Oberfläche":
                        oberflaeche = 2 * r * 3.14 * h + 2 * r * r * 3.14;
                        label4.Text = "Die Oberfläche des Rundstahls beträgt: " +
                        oberflaeche.ToString("F2") + " cm².";  break;
                    case "Volumen":
                        volumen = 3.14 * r * r * h;
                        label4.Text = "Das Volumen des Rundstahls beträgt: " +
                        volumen.ToString("F2") + " cm³."; break;
                    case "Gewicht":
                        gewicht = 3.14 * r * r * h * 7.85 / 1000;
                        label4.Text = "Das Gewicht des Rundstahls beträgt: " +
                        gewicht.ToString("F2") + " kg."; break;
                    default:
                        label4.Text = "Bitte eine Auswahl treffen!"; break;


            }


        }
```

```
try
        {
```

```
}
```

```
catch (Exception ausnahme)
            {

MessageBox.Show(ausnahme.Message, "Fehler");
            }
```

Andreas Besener: Arbeitsblatt 2 vom 04.03.2008 (THE83)

Abbildung 6: Arbeitsblatt 2.

4.5 Anlage 6: Modifizierter Auszug aus dem Skript

Kapitel 7: Fehlerbehandlung in C#[56] [57] [58]

Einführung

Nachdem alle Syntaxfehler und hoffentlich alle logischen Fehler[59] mit dem Debugger aufgespürt wurden, enthält ein Programm trotzdem noch eine ganze Reihe weiterer potenzieller Fehlermöglichkeiten, auf die Sie als Programmierer gefasst sein müssen! Einige Beispiele:

- Anwender geben unzulässige Werte in einem Formular ein.
- Es wird versucht, eine nicht vorhandene oder gesperrte Datei zu öffnen.
- Es steht zu wenig Speicher zur Verfügung.
- Eine Netzwerkverbindung ist instabil.

Abbildung 33: Unbehandelter Fehler

Fehler dieser Art, die auch als **Exceptions** bezeichnet werden, haben eins gemeinsam: Ist Ihr Programm auf einen Fehler nicht vorbereitet, wird der bedauernswerte Anwender zum Beispiel mit der in Abbildung 33 dargestellten Meldung schockiert. Um einen solchen Anblick nicht zum Markenzeichen Ihrer zukünftigen Programme werden zu lassen, bietet C# genügend Möglichkeiten zur Fehlerbehandlung.
Die **Fehlerbehandlung** hat dabei die Zielsetzung, dem Anwender beispielsweise durch eine Eingabekorrektur die Fortsetzung des Programms zu ermöglichen oder zumindest alle notwendigen Daten zu sichern, bevor das Programm beendet wird.

Try-catch-Blöcke

Programmblöcke, die einen Fehler auslösen können, werden in so genannten Blöcken gekapselt. Tritt innerhalb des geschützten Blocks, d. h. im **`try`-Block** ein Fehler auf, wird die Programmausführung in diesem Block unterbrochen und der **`catch`-Block** ausgeführt.[60] Tritt kein Fehler auf, wird der `catch`-Block nie durchlaufen.

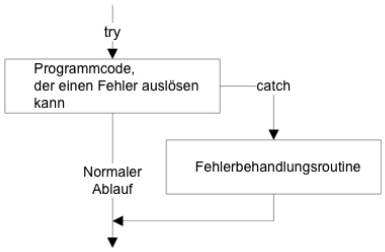

Abbildung 34: Prinzip der Fehlerbehandlung

[56] Vgl. Doberenz, Walter und Kowalski, Thomas: Visual C#.NET. Grundlagen und Profiwissen. Hanser, München, Wien, 2003, S. 692 ff.

[57] Vgl. Kühnel, Andreas: Visual C# 2005. Das umfassende Handbuch. 3. Auflage, Galileo Computing, 2006, *openbook* (HTML-Version)

[58] Vgl. Louis, Dirk und Strasser, Shinja: C# in 21 Tagen. Markt+Technik, München, 2002, S. 669 ff.

[59] Hinweis: Logisch korrekte Programme zu schreiben ist sehr schwierig. Es ist tatsächlich so schwierig, dass der Großteil der kommerziell vertriebenen Software bekanntermaßen eine signifikante Anzahl von Fehlern enthält. Wenn man Software schreibt, sollte man deshalb damit rechnen, dass diese logische Fehler enthält. Aus diesem Grund ist es wichtig, sowohl das Testen als auch die Fehlerbeseitigung als normale Aktivitäten im gesamten Entwicklungsprozess anzusehen.

[60] Hinweis: Im `catch`-Block wird sozusagen die Exception „aufgefangen".

Abbildung 7: Modifizierter Ausdruck aus dem Skript (Seite 1/2).

Beispiel:

```
private void button1_Click(object sender, EventArgs e)
{
      double tempFahrenheit, tempCelsius;

      try
      {
            // Lese: Temperatur in °Celsius
            tempCelsius = Convert.ToDouble(textBox1.Text);

            // Berechnung der Temperatur in °Fahrenheit
            tempFahrenheit = 9D/5*tempCelsius+32;

            // Ausgabe: Temperatur in °Fahrenheit
            label3.Text = "= "+tempFahrenheit.ToString("0.00")+" °F";
      }
      catch(Exception ausnahme)
      {
            MessageBox.Show(ausnahme.Message, "Fehler");
      }
}
```

Tritt im `try`-Block ein Fehler auf, werden alle Anweisungen im `catch`-Block ausgeführt. In diesem Fall wird der Fehler über ein `Exception`-Objekt ausgewertet und die systeminterne Fehlermeldung angezeigt, siehe Abbildung 35.[61]

Abbildung 35: Meldungsfenster zum Beispiel

[61] Hinweis: Die `Convert.ToDouble(String)`-Methode kennt neben der allgemeinen Ausnahme `Exception` spezifische Ausnahmen `FormatException` und `OverflowException`. Um auf die verschiedenen Ausnahmen spezifisch reagieren zu können, kann es sinnvoll sein, mehrere `catch`-Blöcke zu verwenden. Wird eine Ausnahme ausgelöst, werden die `catch`-Zweige zur Laufzeit so lange der Reihe nach angesteuert, bis der Typ gefunden wird, der die ausgelöste Ausnahme beschreibt. Optional kann ein `finally`-Block unmittelbar an den letzten `catch`-Block angehängt werden. Der `finally`-Block wird unabhängig davon ausgeführt, ob eine Ausnahme ausgelöst worden ist oder nicht, z. B. um abschließende Arbeiten und Bereinigungen auszuführen, z. B. eine Datenbankverbindung freigeben.

Abbildung 8: Modifizierter Ausdruck aus dem Skript (Seite 2/2).

4.6 Anlage 7: Modifiziertes Programm mit try-catch-Block (erwartetes Ergebnis)

```
private void button1_Click(object sender, EventArgs e)
        {
            double r, h, volumen, oberflaeche, gewicht;
            string auswahl;

            try
```

```
        {
            r = Convert.ToDouble(textBox1.Text);
            h = Convert.ToDouble(textBox2.Text);
            auswahl = comboBox1.Text;
            switch (auswahl)
            {
                case "Oberfläche":
                    oberflaeche = 2 * r * 3.14 * h + 2 * r * r * 3.14;
                    label4.Text = "Die Oberfläche des Rundstahls beträgt: " +
                    oberflaeche.ToString("F2") + " cm².";  break;
                case "Volumen":
                    volumen = 3.14 * r * r * h;
                    label4.Text = "Das Volumen des Rundstahls beträgt: " +
                    volumen.ToString("F2") + " cm³."; break;
                case "Gewicht":
                    gewicht = 3.14 * r * r * h * 7.85 / 1000;
                    label4.Text = "Das Gewicht des Rundstahls beträgt: " +
                    gewicht.ToString("F2") + " kg."; break;
                default:
                    label4.Text = "Bitte eine Auswahl treffen!"; break;
            }
        }
        catch (Exception ausnahme)
        {
            MessageBox.Show(ausnahme.Message, "Fehler");
        }
    }
```

5 Literatur

Berufskolleg Geldern (Hrsg.) (2008): Didaktische Jahresplanung – Zweijährige Berufsfachschule Fachrichtung Technik – Fachlicher Schwerpunkt Elektrotechnik, Profilbildung Informations- und Kommunikationstechnik. Geldern.

Besener, A. (2008): Unterrichtsentwurf 1 in der beruflichen Fachrichtung Technische Informatik. Thema der Unterrichtsstunde: „Programmierung einer zweiseitigen Auswahl (C#)“. Straelen.

Bezirksregierung Düsseldorf (2001): 62. Ergänzende Vorgaben und Hinweise zur Einrichtung und Durchführung von Bildungsgängen gemäß § 2 Abs. 2 Nr. 2 Anlage C (Zweijährige Berufsfachschule „Erweiterte berufliche Kenntnisse und Fachhochschulreife“) und gemäß § 2 Abs. 3 Anlage D (Fachoberschule Klasse 13) der Verordnung über die Ausbildung und Prüfung in den Bildungsgängen des Berufskollegs (APO-BK). Düsseldorf, S. 167-178.

Lentz, M. (2007): C# 2.0 - Sprachelemente. Windows-Programmierung mit dem .NET Framework 2.0. 2., erweiterte und überarbeitete Auflage. Geldern.

Meyer, H. (2004): Was ist guter Unterricht? Berlin.

Meyers Lexikonredaktion (Hrsg.) (2001): Duden Informatik. Ein Fachlexikon für Studium und Praxis. 3. Auflage. Mannheim u. a.

Ministerium für Schule und Weiterbildung des Landes Nordrhein-Westfalen (MSW) (2009): Methodensammlung – Anregungen und Beispiele für Moderatoren. Bienenkorb. Düsseldorf. Online unter: http://www.learn-line.nrw.de/angebote/methodensammlung/karte.php?karte=006. Stand 01.03.2009.