PR 213_01

Prüfungsarbeit

C#-Software-Entwicklung

1120K02

PR 213_01

Prüfungsarbeit

C#-Software-Entwicklung

Autor: Christoph Siebeck
Fachlektor: Torsten Schreiber

Die in unseren Studienheften verwendeten Personenbezeichnungen schließen ausdrücklich alle Geschlechtsidentitäten ein. Wir distanzieren uns ausdrücklich von jeglicher Diskriminierung hinsichtlich der geschlechtlichen Identität.

Falls wir in unseren Studienheften auf Seiten im Internet verweisen, haben wir diese nach sorgfältigen Erwägungen ausgewählt. Auf die zukünftige Gestaltung und den Inhalt der Seiten haben wir jedoch keinen Einfluss. Wir distanzieren uns daher ausdrücklich von diesen Seiten, soweit darin rechtswidrige, insbesondere jugendgefährdende oder verfassungsfeindliche Inhalte zutage treten sollten.

Vorwort

Sehr geehrte Teilnehmerin, sehr geehrter Teilnehmer,

vor Ihnen liegt die Prüfungsarbeit, mit der Sie über Ihr Zeugnis hinaus das höherwertige Zertifikat erlangen können.

In dem Zertifikat wird neben der Durchschnittsnote Ihrer bisherigen Einsendeaufgaben auch das Ergebnis der Prüfungsarbeit ausgegeben. Das Zertifikat bescheinigt Ihnen, dass Sie ein breites Wissen erworben haben und es setzt Ihrem Abschluss ein „Geprüfte/r“ voran. Damit besitzen Sie einen Qualifikationsnachweis, der Ihren beruflichen Marktwert deutlich erhöht.

Für die Prüfung wünschen wir Ihnen viel Erfolg.

Ihre Studienleitung

Bearbeitungshinweise

Bitte beachten Sie, bevor Sie mit dem Lösen der Aufgaben beginnen, die folgenden Bearbeitungshinweise:

Zu jeder Aufgabe ist nur eine Lösung zulässig! Wenn Sie nicht genau wissen, was in einer Aufgabe von Ihnen gefordert wird, fragen Sie bitte vor dem Einsenden nach.

Halten Sie sich jeweils peinlichst genau an die Aufgabenstellung und an eventuell formulierte Randbedingungen. Achten Sie bitte darauf, dass jede Antwort genau der betreffenden Aufgabe zugeordnet werden kann.

Hinweis:

> Die genaue Einhaltung der Aufgabenstellung und der Randbedingungen ist Teil der Aufgabe. Wenn Sie davon abweichen, dann muss Ihr Prüfer die Aufgabe als nicht vollständig beziehungsweise nicht ordnungsgemäß gelöst bewerten und Punkte abziehen.

Erstellen Sie zu allen Programmieraufgaben jeweils ein eigenes Projekt in einem eigenen Ordner. Für die Bezeichnung der Projekte und der Ordner verwenden Sie bitte folgendes Muster: Aufgabe_Aufgabennummer – zum Beispiel Aufgabe_02, Aufgabe_03 und so fort.

Ob Sie bei Programmen mit grafischer Oberfläche Windows Forms oder die WPF verwenden, können Sie selbst entscheiden. Die Hinweise bei den Aufgaben beziehen sich jeweils auf Windows Forms.

Nach Fertigstellung senden Sie die Projekte auf einer CD oder einem USB-Stick zusammen mit dieser Prüfungsarbeit und eventuellen weiteren Lösungsblättern per Briefpost an Ihre Fernschule ein. Verwenden Sie dafür bitte folgende Adresse:

Fernstudienzentrum Hamburg

Einsendeaufgaben

Doberaner Weg 18–22

22143 Hamburg

Tragen Sie bitte auf jedem Lösungsblatt, das Sie zusätzlich zur Prüfungsarbeit einreichen, die folgenden Daten ein:

Prüfungsarbeit: PR 213_01

Vor- und Zuname

Vertragsnummer

Überprüfen Sie bitte vor dem Einsenden noch einmal, ob die Dateien lesbar und vollständig sind.

Wir archivieren Ihre Prüfungsarbeit inklusive aller eingereichten Datenträger hier im Hause. Sie bekommen evtl. eingesendete USB-Sticks also nicht zurückgesendet.

Bedenken Sie bitte, dass die Korrektur einer Prüfungsarbeit bis zu vier Wochen dauern kann.

Wir wünschen Ihnen viel Erfolg.

Ihr Fernschulteam

Bewertungsbogen zur Prüfungsarbeit

Nachfolgend finden Sie eine Tabelle, die das Themengebiet, die Nummer der Aufgabe und die erreichbare Punktezahl wiedergibt.

Themengebiet	Aufgabe	Maximale Punktzahl	Erreichte Punktzahl
Programme für die Eingabeaufforderung	1	2	
	2	3	
	3	5	
	4	15	
	5	5	
Programme mit grafischen Oberflächen	6	10	
	7	20	
	8	20	
	9	20	
Gesamtpunktzahl		100	

Kommentar des Fernlehrers zur Benotung:

Notenschlüssel:

Note:	Maximale Punktzahl: 100
1 = sehr gut	92,0 – 100,0
2 = gut	81,0 – 91,9
3 = befriedigend	67,0 – 80,9
4 = ausreichend	50,0 – 66,9
5 = mangelhaft	30,0 – 49,9
6 = ungenügend	0 – 29,9

Fernlehrer/in:
Datum:
Note:
Unterschrift Fernlehrer/in:

Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die beiliegenden Prüfungsaufgaben selbstständig ohne Rat und Hilfe anderer Personen gelöst habe. Vor der Einsendung wurden sie nicht von einer anderen Person auf Fehler überprüft. Zusätzlich erkläre ich, dass ich alle Hilfsmittel (z.B. Bücher, Zeitschriften, Internetquellen) vollständig und exakt angegeben habe. Alle Stellen, die den verwendeten Hilfsmitteln im Wortlaut oder dem Sinn nach entnommen wurden, sind mit einer Quellenangabe gekennzeichnet. Alle wörtlich entnommenen Stellen sind zudem als Zitate gekennzeichnet.

Es ist mir außerdem bekannt, dass diese Aufgaben dem Copyright unterliegen. Die Vervielfältigung und Weitergabe ist nicht gestattet.

Es ist mir bekannt, dass mir bei falschen Angaben das Abschlusszertifikat verweigert bzw. nachträglich wieder aberkannt werden kann.

_____ _____

(Ort und Datum) (Unterschrift)

Name: _____

Vorname: _____

Studien- bzw.
Vertrags-Nr.: _____

Straße: _____

PLZ und Ort: _____

Prüfungsaufgaben des Studienlehrgangs

Code:

Name:	Vorname:
Postleitzahl und Ort:	Straße:
Studien- bzw. Vertrags-Nr.:	Lehrgangs-Nr.:

Fernlehrer/in:
Datum:
Note:
Unterschrift Fernlehrer/in:

Beachten Sie bitte die Bearbeitungshinweise auf S. 2.

Programme für die Eingabeaufforderung:

1. Aufgabe

Schreiben Sie ein Programm, das die Primzahlen von 1 bis 1 000 ermittelt. Eine Zahl ist eine Primzahl, wenn sie größer als 1 ist und nur durch 1 und sich selbst zu teilen ist.

2 Pkt.

2. Aufgabe

Schreiben Sie ein Programm, das so lange Zahlen einliest, bis die Ziffer 0 eingegeben wird.

Berechnen Sie die Summe der eingegebenen Zahlen und den Mittelwert der Zahlen. Die Ziffer 0 soll dabei nicht berücksichtigt werden.

Hinweis:

Den Mittelwert berechnen Sie, indem Sie die Summe der Zahlen durch die Anzahl der Eingaben teilen.

3 Pkt.

3. Aufgabe

Erstellen Sie ein Programm, das eine Zeichenkette einliest und dann die Anzahl der Vokale sowie die Anzahl der Konsonanten in der Zeichenkette ermittelt. Als Konsonant dürfen dabei alle Zeichen gezählt werden, die kein Vokal sind – also zum Beispiel auch Zahlen und Sonderzeichen.

Berücksichtigen Sie bei der Ermittlung der Vokale und Konsonanten sowohl Groß- als auch Kleinbuchstaben.

Benutzen Sie für die Lösung dieser Aufgabe eine `switch`-Konstruktion.

5 Pkt.

4. Aufgabe

Erstellen Sie ein Programm, das die Ziehung der Lottozahlen simuliert und das Ergebnis auf dem Bildschirm ausgibt.

Es darf keine Zahl doppelt verwendet werden!

Hinweise:

Eine Zufallszahl ermitteln Sie über die Klasse `Random` und die Methode `Next()`. Erzeugen Sie dazu im ersten Schritt über

```
Random zufall = new Random();
```

eine neue Instanz `zufall`.

Über

```
zufall.Next();
```

können Sie dann zufällige Zahlen erzeugen.

Sehen Sie bitte in der Hilfe nach, wie die Methode genau arbeitet.

Denken Sie daran: Die Zahl 0 gibt es beim Lotto nicht.

Ein Tipp:

Um zu kontrollieren, ob Ihr Programm richtig arbeitet, lassen Sie zum Beispiel fünf zufällige unterschiedliche Zahlen aus dem Zahlenbereich 1 bis 5 ermitteln. Jede Zahl darf dabei nur einmal vorkommen.

15 Pkt.

5. Aufgabe

Erstellen Sie ein Programm, das die Höhe eines gleichschenkligen Dreiecks einliest und das Dreieck dann mit Sternchen auf dem Bildschirm ausgibt.

Ein Beispiel:

Wenn die Höhe 3 eingegeben wird, sollte die Ausgabe so aussehen:

```
  *
 ***
*****
```

5 Pkt.

Programme mit grafischen Benutzeroberflächen:

6. Aufgabe

Erstellen Sie ein Programm, das den Text in zwei Labels nach einem Klick auf eine Schaltfläche tauscht. Der Tausch soll beliebig oft möglich sein.

Verwenden Sie ein Kontrollkästchen, das den Text in den beiden Labels vergrößert. Wie groß der Text wird, ist beliebig.

Beim Abschalten der Markierung soll der Text wieder verkleinert werden. Verwenden Sie dazu die ursprüngliche Größe des Textes.

Hinweise:

Beim Verkleinern soll die Textgröße nicht über eine Konstante fest gesetzt werden. Ermitteln Sie an geeigneter Stelle vor dem Vergrößern die aktuelle Größe, speichern Sie die Werte zwischen und setzen Sie diese Werte wieder beim Verkleinern.

Die Größe eines Textes in einem Label erhalten Sie über eine Untereigenschaft der Eigenschaft Font.

Sie können die Größe eines Textes nicht direkt ändern, sondern müssen der Eigenschaft Font einen neuen Font zuweisen. Dabei können Sie als erstes Argument an den Konstruktor die Eigenschaften eines bereits vorhandenen Textes übergeben und als zweites Argument die neue Größe. Ein Beispiel:

```
labelLinks.Font = new System.Drawing.Font
(labelLinks.Font.Name, 14);
```

ändert die Größe der Schriftart in dem Steuerelement `labelLinks` auf 14.

10 Pkt.

7. Aufgabe

Erstellen Sie ein „Spiel“, bei dem der Anwender eine Schaltfläche in einem Formular „fangen“ muss.

Die Schaltfläche soll regelmäßig ihre Position im Formular wechseln. Lassen Sie dazu eine zufällige Position ermitteln, an der Sie die Schaltfläche einblenden.

Denken Sie bitte daran, dass die Position der Schaltfläche so gesetzt werden muss, dass die Schaltfläche immer vollständig im Formular zu sehen ist. Wie groß Sie das Formular selbst machen, ist Ihnen freigestellt.

Damit der Anwender eine Chance hat, die Schaltfläche zu „fangen“, soll sie mindestens eine Sekunde lang angezeigt werden. Die Schaltfläche gilt als gefangen, wenn der Anwender sie anklicken konnte.

Das Spiel soll maximal zwei Minuten dauern.

Blenden Sie in dem Formular am unteren Rand eine Anzeige der aktuellen Treffer und auch die verbleibende Zeit in Sekunden ein.

Nach 10 Treffern oder nach Ablauf der Spielzeit soll das Spiel beendet werden. Wenn der Anwender 10 Treffer erzielt hat, blenden Sie am Ende des Spiels einen Dialog ein, in dem eine Meldung wie „Herzlichen Glückwunsch“ steht. Wenn der Anwender in den zwei Minuten keine 10 Treffer erzielt hat, gilt das Spiel als verloren. Lassen Sie dann eine Meldung wie „Sie haben leider verloren“ anzeigen.

In beiden Fällen soll das Programm beendet werden, wenn der Anwender den Dialog schließt.

Hinweise:

Zufallszahlen erzeugen Sie über die Klasse `Random`.

Den Dialog am Ende des Spiels können Sie mit der Klasse `MessageBox` erzeugen. Sehen Sie gegebenenfalls noch einmal in der Hilfe nach, welche Angaben Sie in welcher Form übergeben müssen.

20 Pkt.

8. Aufgabe

Programmieren Sie ein Hangman-Spiel.

Das Spiel soll so funktionieren:

- Das Programm ermittelt zufällig ein Wort aus einer Reihe vorgegebener Wörter.
- Lassen Sie für das Wort für jeden Buchstaben ein Sternchen anzeigen.
- Der Anwender hat 10 Versuche, das Wort zu erraten. Dabei darf er pro Versuch einen Buchstaben eingeben. Wenn der Buchstabe in dem Wort vorkommt, soll er angezeigt werden. Dem Anwender wird dann auch kein Versuch abgezogen.
- Findet der Anwender dagegen keinen Buchstaben, wird ihm ein Versuch abgezogen und er kann weiterraten.
- Damit der Anwender weiß, welche Buchstaben er schon probiert hat, können Sie zum Beispiel bereits ausgewählte Zeichen aus dem Kombinationsfeld löschen.

Hinweis:

Die Änderungen an dem Wort, das auf dem Bildschirm angezeigt wird, können Sie recht einfach über die Klasse `StringBuilder` erreichen. Für Vergleiche mit einem String müssen Sie eine Instanz dieser Klasse aber erst wieder mit der Methode `ToString()` konvertieren.

20 Pkt.

9. Aufgabe

Erstellen Sie ein Programm, das arabische Zahlen in römische umrechnen kann und umgekehrt. Die römischen Ziffern mit den entsprechenden Werten finden Sie in der folgenden Tabelle:

Ziffer	Wert
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Eine römische Zahl besteht aus einer Folge von römischen Ziffern. Der Wert der gesamten Zahl ergibt sich durch die Addition der Werte. Die größeren Werte stehen normalerweise immer links, die kleineren rechts.

Steht eine kleinere Ziffer links vor einer größeren, wird der Wert dieser Ziffer von der Ziffer direkt rechts daneben abgezogen. Die Zahl IX steht dann also für 10 - 1 – also für 9. Es folgen nie mehr als drei identische Ziffern aufeinander. 4 wird als IV darstellt, 9 als IX, 40 als XL, 90 als XC, 400 als CD und 900 als CM. Diese Regel gilt allerdings nur für den jeweils nächsten und übernächsten Wert. Kombinationen wie IID oder XM sind also nicht möglich.

Einige Beispiele zum Test:

1997 – MCMXCVII

2009 – MMIX

414 – CDXIV

20 Pkt.

insgesamt 100 Pkt.