

Zweijähriges Berufskolleg Informations- und Kommunikationstechnik

# Abschlussprüfung 2010

Fach:

## Programmiertechnik-Praktikum

<b>Fachlehrer:</b>	Geng, Götz		
<b>Prüfungsdatum:</b>	25.06.2010		
<b>Prüfungsdauer:</b>	240 Minuten		
<b>Zugelassene Hilfsmittel:</b>	KEINE		
<b>Bearbeitungshinweise:</b>	<p>Alle Aufgaben sind zu bearbeiten.</p> <p><b>2BK121</b> Legen Sie einen Ordner mit Ihrem Nachnamen auf dem Desktop an. In diesem Ordner legen Sie für jede Aufgabe ein Projekt mit Projektmappenname an. Projektmappennamen: A1, A2, A3Konsole, A3GUI. Die Projektmappen müssen so benannt werden und die zu der jeweiligen Aufgabe gehörende Lösung enthalten.</p> <p><b>2BK122</b> Legen sie <u>eine</u> Projektmappe mit Ihren Nachnamen auf dem Desktop an und erstellen Sie dort für jede Aufgabe die Projekte mit den Projektnamen A1, A2, A3Konsole und A3GUI.</p>		
<b>Schüler/in:</b>	<div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="display: flex; justify-content: space-between; padding: 0 10px;"> <span>Name</span> <span>Vorname</span> <span>Klasse</span> </div>		
<b>Ergebnis der Prüfung:</b>	<div style="display: flex; justify-content: space-between;"> <div> <b>Erstkorrektur:</b>   <b>Zweitkorrektur:</b> </div> <div> <b>Durchschnitt:</b>   <b>Note:</b> </div> </div>		

1.	<p>Da nach der aktuellen Rechtslage der Betreiber eines WLAN-Netzes für alle Zugriffe auf das Internet von diesem Netz aus haftet, hat sich ein privater Betreiber dazu entschlossen sein Netz zu verschlüsseln. Dazu erzeugt er einen 63 Zeichen langen Schlüssel. Um diesen so einfach wie möglich zu erzeugen, verwendet er mehrere Guids (Global Unique Identifier) . GUIDs können mit der statischen Methode <i>NewGuid()</i> der im .Net Framework enthaltenen Klasse <i>Guid</i> erzeugt werden.</p> <p>Beispiel für einen GUID: b6471495-45f5-47d6-a3af-2f1b13427592.</p> <p>Bevor er an den zunächst leeren Schlüssel eine neu erzeugte Guid anhängt, entfernt er aus dieser, um die Sicherheit weiter zu erhöhen, die Bindestriche. An den Schlüssel werden solange Guids angehängt bis er mindestens die gewünschte Länge hat.</p> <p>Danach werden die ersten 63 Zeichen als neu erzeugter Schlüssel unter dem Namen <i>key.txt</i> im Verzeichnis <i>c:\temp</i> gespeichert.</p>	20
2.	Für den Umgang mit Uhrzeiten soll eine Klasse <i>Uhrzeit</i> erstellt werden.	
a)	Erstellen Sie die privaten Felder <i>stunde</i> und <i>minute</i> und initialisieren Sie diese mit 0. Erstellen Sie auch die zugehörigen Eigenschaften.	4
b)	Erstellen Sie den Standardkonstruktor der die Uhrzeit auf 01:02 Uhr setzt.	2
c)	Erstellen Sie außerdem einen Konstruktor mit Parametern, der die Uhrzeit auf die übergebenen Werte setzt, falls diese eine gültige Uhrzeit bilden. Falls nicht, soll eine <i>ArgumentOutOfRangeException</i> erzeugt werden. Der Exception soll als zusätzliche Hilfe für den Anwender der Fehlertext: „Keine gültige Uhrzeit.“ mitgegeben werden.	5
d)	Erstellen Sie einen weiteren Konstruktor der als Parameter ein Objekt der Klasse <i>Uhrzeit</i> erwartet und das neue Objekt auf diese Uhrzeit setzt.	4
e)	Erstellen Sie die Methode <i>ToString()</i> welche die Uhrzeit formatiert in der Form hh:mm zurückliefert. Beispiel: 02:03 oder 13:15	4
f)	Folgende Anweisung soll im Hauptprogramm möglich sein: <code>uhrzeit3 = uhrzeit1 + uhrzeit2;</code> Dabei soll die neue Uhrzeit korrekt berechnet werden.	5
g)	Folgende Anweisung soll im Hauptprogramm möglich sein: <code>uhrzeit1 = uhrzeit2++;</code> Dabei soll der Inkrement-Operator die Uhrzeit um eine Minute erhöhen und die neue Uhrzeit korrekt berechnet werden.	5
h)	Erzeugen Sie im Hauptprogramm zwei Objekte <i>uhrzeit1</i> und <i>uhrzeit2</i> . Setzen Sie <i>uhrzeit1</i> auf 23:59 und <i>uhrzeit2</i> auf 01:10. Demonstrieren Sie jeweils die Verwendung der Methode <i>ToString()</i> . Addieren Sie anschließend <i>uhrzeit1</i> und <i>uhrzeit2</i> ohne dass sich diese Uhrzeiten ändern und geben Sie das Ergebnis aus. Danach inkrementieren Sie <i>uhrzeit1</i> und geben das Ergebnis aus.	6

### Aufgabe 3

**Chuck a Luck** (dt. etwa: „Glückswurf“) ist ein Würfel-Glücksspiel mit drei Würfeln.

Der Spieler tätigt seinen Einsatz, indem er eine der Zahlen „Eins“ bis „Sechs“ auswählt, dann wirft der Bankhalter drei Würfel.

Zeigt ein Würfel die gesetzte Augenzahl, gewinnt der Spieler den einfachen Einsatz, zeigen zwei Würfel die gesetzte Augenzahl, gewinnt der Spieler den doppelten Einsatz, zeigen alle drei Würfel die gesetzte Augenzahl, so gewinnt der Spieler den zehnfachen Einsatz.

Zeigt aber kein Würfel die gesetzte Augenzahl, ist der Einsatz verloren.

Vergl.: [http://de.wikipedia.org/wiki/Chuck\\_a\\_Luck](http://de.wikipedia.org/wiki/Chuck_a_Luck)

In dieser Aufgabe wird nach festen Vorgaben die Klasse ChuckALuck entwickelt, in einem Konsoleprojekt getestet und in einem WINDOWS-Programm mit graphischer Oberfläche (GUI) verwendet.

**Zur Vereinfachung kann bei einem Spiel immer nur ein Spieler auf eine Augenzahl tippen!**

**Der Einsatz beträgt dabei stets „1“ (Chip, ...)!**

Ein Klassendiagramm und eine tabellarische Zusammenstellung beschreiben den verbindlichen Aufbau der Klasse ChuckALuck:

```

ChuckALuck
- r: Random
- number: int
- count: int
- totalWin: int

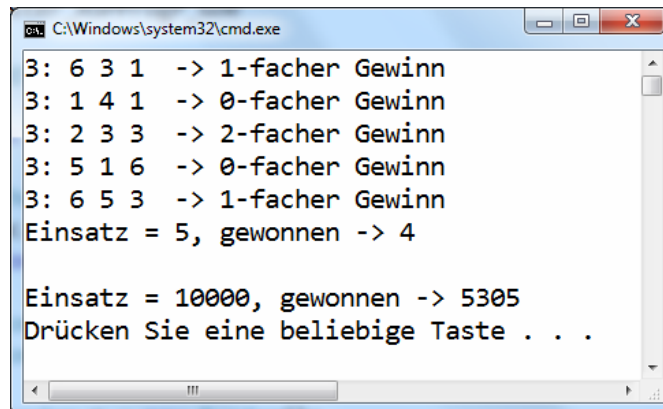
+ SetNumber(int): void
+ Play(): string
+ GetResult(): string
  
```



Member	Beschreibung
r	Der Zufallsgenerator
number	Speichert die gesetzte Zahl
count	Die Anzahl der gespielten Spiele. Der Einsatz pro Spiel ist „1“!
totalWin	Die Summe der Gewinne aller gespielten Spiele.
SetNumber(int)	Legt die gewählte Glückszahl fest. Dabei wird der zulässige Bereich (1 .. 6) überprüft, im Fehlerfall wird die Glückszahl zu 0 gesetzt.
Play()	Wenn die Glückszahl gültig ist, wird ein Spiel gespielt und das Ergebnis als String zurückgegeben. Der String beinhaltet die Glückszahl, die Würfelzahlen und den ermittelten Gewinn. → siehe Beispielausgabe Bei ungültiger Glückszahl bekommt der Rückgabestring den Wert: keine Glückszahl gewählt.  Die Felder count und totalWin werden entsprechend verändert.
GetResult()	Gibt als String die Anzahl der gespielten Spiele, somit also den Einsatz und die Gewinnsumme zurück. → Rückgabestring siehe Beispielausgabe

**Teilaufgaben:**

- a) Klassenentwicklung (30 Punkte)  
Entwickle in einem Konsolenprojekt A3Konsole die Klassen ChuckALuck nach den Vorgaben.
- b) Konsolenprogramm (10 Punkte)  
Ein Test macht zuerst 5 Spiele und zeigt alle Spiele und das Resultat an. Die Glückszahl ist dabei immer 3.  
Danach werden nochmals 10000 neue Spiele gespielt. Die Glückszahl ist bei jedem Spiel zufällig. Es wird nur am Ende das Resultat angezeigt.



```
C:\Windows\system32\cmd.exe
3: 6 3 1 -> 1-facher Gewinn
3: 1 4 1 -> 0-facher Gewinn
3: 2 3 3 -> 2-facher Gewinn
3: 5 1 6 -> 0-facher Gewinn
3: 6 5 3 -> 1-facher Gewinn
Einsatz = 5, gewonnen -> 4

Einsatz = 10000, gewonnen -> 5305
Drücken Sie eine beliebige Taste . . .
```

Hinweis: In der Klasse wurde als Seed für die Random-Instanz die Zahl 12 verwendet.

- c) GUI-Programm (25 Punkte)  
Erstelle nun in der Projektmappe A3GUI ein WINDOWS-Projekt mit der unten abgebildeten GUI.  
`play` spielt ein einzelnes Spiel. Es wird das Spiel- und das Gesamtergebn ausgegeben.  
`reset` setzt alles zurück, indem ein neues Objekt der Klasse ChuckALuck erzeugt wird.  
`AUTOPLAY` führt die in der Textbox angegebene Anzahl von Spielen mit der gewählten Glückszahl aus und zeigt nur das Resultat an (siehe unten rechts).  
Die Gültigkeit der Eingabe in der Textbox für die Anzahl der Spiele muss sichergestellt werden.

