TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK
Lehrstuhl für Angewandte Informatik / Kooperative Systeme WS 15/16
Praktikum Grundlagen der Programmierung **Übungsblatt 4**
Prof. Dr. A. Brüggemann-Klein, R. Palenta, J. Kranz, Dr. A. Reuss 06.11.2015

Abgabe: 16.11.2015 (vor 5:00 Uhr)

Hinweis: Implementieren Sie alle Hausaufgaben in einem Package namens **blatt04**.

Aufgabe 4.1 (P) Kontrollflussdiagramm

Zeichnen Sie den Kontrollflussgraphen für das folgende `Java`-Fragment. Markieren Sie im Graphen die Scopes der Variablen und geben Sie die Zeilennummern der Anweisungen und Bedingungen jeweils mit an.

```
1   int x;
2   int y = new java.util.Scanner(System.in).nextInt();
3   x = new java.util.Scanner(System.in).nextInt();
4   for (int i = 0; x != y; i = 1) {
5      if (x < y) {
6         y = y - x;
7      } else {
8         x = x - y;
9      }
10  }
11  System.out.println("ggT: " + x);
```

Sie dürfen dabei

- `new java.util.Scanner(System.in).nextInt` durch `read` ersetzen.
- `System.out.println` durch `write` ersetzen.

Aufgabe 4.2 (P) Methoden mit Parametern

a) Schreiben Sie eine Methode `int ggT(int x, int y)` zur Berechnung des größten gemeinsamen Teilers (ggT) zweier positiver Zahlen. Eine einfache Möglichkeit, den ggT zu berechnen, besteht darin, so lange die kleinere von der größeren Zahl zu subtrahieren, bis beide Zahlen gleich geworden sind.

b) Damit wir den ggT zweier beliebiger eingegebener Zahlen berechnen können, müssen wir diese Zahlen einlesen. Schreiben Sie dazu eine Methode `int readInt(String msg, int low, int high)`, welche zunächst die Zeichenkette `msg` ausgibt und dann eine Zahl einliest. Ist die eingelesene Zahl nicht im Bereich zwischen `low` und `high` (jeweils einschließlich), so soll eine entsprechende Meldung ausgegeben werden, welche den Benutzer über den geforderten Wertebereich informiert. Dies wird dann wiederholt, bis eine Zahl im geforderten Bereich eingegebenen wird.

c) Benutzen Sie nun die beiden Methoden aus den vorigen Teilaufgaben in der `main`-Methode. Zuerst soll der ggT von zwei eingegebenen positiven Zahlen berechnet und ausgegeben werden. Mit welchen Parametern muss `readInt` hierfür aufgerufen werden? Anschließend soll der Benutzer nach einer einstelligen Zahl `z` gefragt und sodann eine zweidimensionale Tabelle mit den ggTs für alle Kombinationen von Zahlen im Bereich zwischen 1 und `z` ausgegeben werden.

d) Diskutieren Sie, wofür globale statische Variablen bzw. Parameter (und Rückgabewerte von Methoden) jeweils besonders geeignet sind. Lassen sich die Methoden in Ihrem Programm leicht ändern, so dass statt Parametern globale Variablen verwendet werden?

Aufgabe 4.3 (P) Zahlenbasen

a) Führen Sie die folgenden Additionen und Multiplikationen schriftlich aus. Wandeln Sie die Basis der Zahlen dabei nicht um, sondern rechnen direkt mit den Zahlen zu der jeweils gegebenen Basis.

i) $11110000111_2 * 101010_2$

ii) $badcab1e_{16} * deadda7a_{16}$

iii) $2221_3 * 112_3$

iv) $a5435435a_{11} + a222_{11}$

v) $1000110101_2 + 10111110_2$

vi) $abad1dea_{16} + deadbeef_{16}$

b) Führen Sie die folgenden Basisumwandlungen durch.

i) Stellen Sie 1010111011_2 zur Basis 10 dar.

ii) Stellen Sie 1010101011_2 zur Basis 16 dar.

iii) Stellen Sie 2542_{10} zur Basis 2 dar.

iv) Stellen Sie $c0deba5e_{16}$ zur Basis 2 dar.

v) Stellen Sie 7237467_8 zur Basis 17 dar.

Aufgabe 4.4 (P) Mastermind

Für die nächste Aufgabe muss eine Zahl von der Kommandazeile eingelesen werden. Bisher ist der Befehl `new java.util.Scanner(System.in)).nextLine()` zum einlesen eines Strings bekannt. Dieser String soll nun in einen Integer umgewandelt werden. Dazu wird der eingegebenen String von rechts nach links Zeichen für Zeichen durchlaufen, um aus den einzelnen Ziffern die komplette Zahl zu berechnen.

a) Schreiben Sie eine Methode `static int digit(char s)`, die den Charakter `s` in einen Integer umwandelt und zurückgibt, wenn es sich um eine Ziffer (0-9) handelt. Falls der Eingabecharacter keine Ziffer ist, soll standardmäßig 0 zurückgegeben werden. Testen Sie Ihre Methode anhand des Strings `"0123456789abc"`.

b) Schreiben Sie eine Methode `static int pow10(int n)`, die bei Eingabe eines Integers `n`, 10^n zurückgibt. Machen Sie sich sich mithilfe Ihres Tutors klar, dass in einem 32-Bit Integer neun Ziffern vollständig gespeichert werden können, d.h. eine neunstellige Zahl, wobei die einzelnen Positionen der Zahl beliebige Ziffern haben können. Geben Sie im Fall eines drohenden Integer Overflows eine Warnung aus! Diskutieren Sie mit Ihrem Tutor, wie mit diesem Fall umgegangen werden kann. Testen Sie Ihre Methode mit $n = 0$, $n = 3$ und $n = 8$.

c) Schreiben Sie eine Methode `static int getDigit(int data, int pos)`, die die Ziffer an Position `pos` (von rechts nach links mit 0 beginnend gezählt) im Integer `data` zurückgibt. Kleiner Hinweis anhand eines Beispiels:

$8572 : 1$	$= 8572$	8572	$\mod 10 = 2$
$8572 : 10$	$= 857.2$	857.2	$\mod 10 = 7$
$8572 : 100$	$= 85.72$	85.72	$\mod 10 = 5$
$8572 : 1000$	$= 8.572$	8.572	$\mod 10 = 8$

Testen Sie die Methode; lassen Sie sich für den Integer `123456789` die Positionen 0 bis 8 zurückgeben.

d) Schreiben Sie eine Methode `static int resetDigit(int data, int pos)`, die die Ziffer an der Stelle `pos` (von rechts nach links mit 0 beginnend gezählt) im Integer `data` auf Null setzt und den daraus resultierenden Integer zurückgibt. Testen Sie Ihre Methode; setzen Sie im Integer `123456789` die erste und letzte Position auf Null.
e) Schreiben Sie eine Methode `static int putDigit(int data, int digit, int pos)`, die die Ziffer an der Stelle `pos` (von rechts nach links mit 0 beginnend gezählt) im Integer `data` auf den Wert `digit` setzt. Setzen Sie dazu die Ziffer an der Position `pos` auf Null und addieren anschließend $digit * 10^{pos}$ zum ursprünglichen Wert `data`. Stellen Sie dabei sicher, dass `digit` eine Ziffer ist. Testen Sie Ihre Methode; setzen Sie im Integer `123456789` die zweite und vorletzte Position auf 3.
f) Erstellen Sie schließlich die Methode `static int stringToNumber(String s)`, die mithilfe der zuvor erstellten Methoden den Eingabe-String `s` Zeichen für Zeichen in einen Integer umwandelt. Beachten Sie dabei, dass das erste Zeichen ein Minuszeichen sein kann. Testen Sie Ihre Methode mit `"0123456789"` und `"-23456789"`.

Jetzt programmieren wir das Spiel Mastermind. Dabei spielt der Benutzer gegen den Computer und hat mehrere Versuche um einen Zahlencode zu erraten. Nach jedem Rateversuch erhält der Nutzer Hinweise, wie viele Ziffern des letzten Rateversuchs an der richtige Position im Code stehen und wieviele Ziffern des letzten Rateversuchs zwar im Code vorkommen, aber nicht an der richtigen Stelle stehen. Dabei kann jede Ziffer im Code für maximal eine Übereinstimmung mit Ziffern im Rateversuch verwendet werden und Positionsübereinstimmungen haben höhere Präzedenz als Ziffernübereinstimmungen.

Dazu gehen wir wie folgt vor:

a) Legen Sie Variablen an, welche die Anzahl der zu spielenden Runden, die Länge des zu erratenden Codes und die Menge der im Code verwendeten Ziffern festlegen.
b) Begrüssen Sie den Nutzer und teilen Sie ihm die zuvor festgelegten Werte mit.
c) Berechnen Sie Zufallsziffern für den Code im festgelegten Bereich und speichern Sie alle Ziffern des Codes mittels `putDigit` in einem Integer.
d) Schreiben Sie eine `for`-Schleife, welche die gegebene Anzahl von Runden abläuft und nur dann etwas tut, wenn der Nutzer den Code noch nicht erraten hat.
e) Geben Sie hinter der `for`-Schleife den Code aus, falls der Nutzer ihn nicht erraten hat.
f) Jetzt füllen wir die `for`-Schleife mit Inhalt:
    i) Lesen Sie den Rateversuch des Nutzers als String von der Konsole.
    ii) Wandeln Sie den String in einen Integer um.
    iii) Überprüfen Sie mittels String und Integer, ob der Rateversuch innerhalb der geforderten Parameter (Codelänge, verwendete Ziffern, etc.) liegt und geben Sie eine Warnung aus falls nicht.
    iv) Schreiben Sie eine Funktion `static boolean getResult(int codeLength,int code,int guess)`:
        1. Die Funktion gibt zurück, ob der übergebene Rateversuch `guess` mit dem `code` übereinstimmt.
        2. Zusätzlich gibt die Funktion auf der Konsole aus, wieviele Ziffern aus `guess` an der richtigen Stelle sind und wie viele Ziffern aus `guess` in `code` vorkommen, aber nicht an der richtigen Stelle sind.
        3. Achten Sie dabei darauf, dass Ziffern aus dem Code die einmal für eine Übereinstimmung (in Position oder Ziffer) verwendet wurden, keine weiter Übereinstimmung liefern und Positionsübereinstimmung höhere Präzedenz als Ziffernübereinstummng haben. **Beispiel:** guess=1001, code=2221 ergibt eine richtige Position (die rechtestete 1) und keine richtige Ziffer, da die rechteste 1 aus `code` schon für die Position verwendet wurde und nicht mehr für die Ziffernübereinstimmung mit der linkesten 1 im `guess` verwendet werden kann. Sie müssen sich also merken, welche Positionen aus dem `code` und aus dem `guess` schon verwendet wurden. guess=3002, code=2221 ergibt keine

richtige Position und eine richtige Ziffer.

v) Verwenden Sie `getResult` um den Rateversuch zu bewerten und zu ermitteln ob der Spieler gewonnen hat.

Aufgabe 4.5 [5,5 Punkte] **(H) Zahlenbasen**

a) Führen Sie die folgenden Additionen und Multiplikationen schriftlich aus. Wandeln Sie die Basis der Zahlen dabei nicht um, sondern rechnen direkt mit den Zahlen zu der jeweils gegebenen Basis.

i) $1010101100_2 * 11000111_2$

ii) $c01dc0ffe_{16} * deadaffe_{16}$

iii) $120022_3 * 22210_3$

iv) $323478_9 + 111202337_9$

v) $01010100011_2 + 10000111_2$

vi) $badeaffe_{16} + badf00d_{16}$

b) Führen Sie die folgenden Basisumwandlungen durch.

i) Stellen Sie 1010101100_2 zur Basis 10 dar.

ii) Stellen Sie 1010101100_2 zur Basis 16 dar.

iii) Stellen Sie 354347357_{10} zur Basis 2 dar.

iv) Stellen Sie $deadaffe_{16}$ zur Basis 2 dar.

v) Stellen Sie 561644_7 zur Basis 19 dar.

Aufgabe 4.6 [8 Punkte] **(H) Blackjack im Casino**

Blackjack[1] ist ein Kartenspiel, bei dem es darum geht, die Augensumme von 21 durch wiederholtes Ziehen von Karten möglichst exakt zu erreichen. In dieser Aufgabe soll ein Besuch im Casino programmiert werden, bei dem Sie einen bestimmten Geldbetrag (in ganzen Euros ohne Cent) verspielen möchten. Der verfügbare Finanzrahmen soll gleich zu Beginn vom Benutzer eingelesen werden, der Einsatz pro Spiel wird vor jeder Runde erfragt. Ein Spiel läuft ab wie folgt:

a) Zu Beginn erhält der Dealer eine, der Spieler zwei offene Karten.

b) Der Spieler darf nun so lange weitere Karte nehmen, bis er entweder die Augensumme von 21 überschreitet oder keine weitere Karte mehr nehmen möchte. Werden 21 Augen überschritten, verliert der Spieler.

c) Hat der Spieler nicht verloren, ist nun der Dealer an der Reihe, weitere Karten zu ziehen. Erreicht dieser so 17 oder mehr Punkte, zieht er keine weitere Karte. Bei 16 oder weniger Augen zieht der Dealer jeweils eine weiter Karte. Zieht der Dealer ein Ass, so zählt dieses nur dann mit 11 Augen, wenn der Dealer die 21 so nicht überschreitet; ansonsten zählt es mit einem Auge. Überschreitet der Dealer die 21 Augen, so verliert auch er.

d) Haben weder der Dealer noch der Spieler verloren, so gewinnt derjenige, der näher an 21 Augen liegt. Haben beide die gleiche Augenzahl, ist das Spiel unentschieden.

[1] `https://de.wikipedia.org/wiki/Black_Jack`

Verliert der Spieler, so wird sein Einsatz einbehalten. Gewinnt der Spieler, erhält er seinen Einsatz und einen Gewinn in Höhe seines Einsatzes zurück, muss allerdings eine Gewinnabgabe in Höhe von 25% (abgerundet, es handelt sich um ein kundenfreundliches Casino) seine Einsatzes an das Casino bezahlen. Endet das Spiel unentschieden, erhält der Spieler seinen Einsatz zurück. Es werden vom Spieler so viele Spiele gespielt, bis er pleite ist und das Casino verlässt; in diesem Fall wird das Programm beendet.

Nutzen Sie folgendes Rahmenprogramm und fügen Sie Ihre Lösung an den markierten Stellen ein. Sie dürfen auch, soweit sinnvoll, eigene Methoden implementieren. Achten Sie darauf, fehlerhafte Eingaben abzufangen!

```java
import java.util.Random;
import java.util.Scanner;

public class Casino {
  static Random random = new Random();

  /**
   * Diese Methode zieht eine zufällige Karte. Es wird die
   * ID der Karte zurückgegeben. Die IDs bedeuten dabei:
   *
   * 0: Ass
   * 1: 1
   * 2: ..
   * 10: 10
   * 11: Bube
   * 12: Dame
   * 13: König
   *
   * @return die ID der Karte
   */
  static int zieheKarte() {
    return random.nextInt(14);
  }

  static String kartenbeschriftung(int kartenID) {
    /*
     * Todo
     */
  }

  /**
   * Diese Methode berechnet die Augenzahl zu einer bestimmten
   * Karte.
   *
   * @param kartenID die ID der Karte
   * @return die Augenzahl
   */
  static int augenzahl(int kartenID) {
    /*
     * Todo
     */
  }

  /**
   * Diese Methode liest eine Zahl vom Benutzer ein; der Benutzer
   * soll dabei durch eine Nachricht zu einer Eingabe aufgefordert
   * werden.
   *
   * @param nachricht die Nachricht an den Benutzer
   * @return die vom Benutzer eingegebene Zahl
   */
  static int leseZahl(String nachricht) {
    /*
     * Todo
     */
  }

  public static void main (String[] args) {
    /*
     * Todo
     */
  }
}
```

Aufgabe 4.7 [3,5 Punkte] **(H) Kontrollflussdiagramm**

Zeichnen Sie einen Kontrollflussgraphen für das folgende `Java`-Fragment. Markieren Sie im Graphen die Scopes der Variablen und geben Sie die Zeilennummern der Anweisungen und Bedingungen jeweils an.

```
1       int x = new java.util.Scanner(System.in).nextInt();
2       int y = new java.util.Scanner(System.in).nextInt();
3       boolean equal =x== y;
4       do {
5          switch (x-y) {
6             case 0 : equal = true;
7             default : System.out.println(x+"!="+y); break;
8          }
9          if (x < y) {
10            y = y - x;
11         } else {
12            x = x - y;
13         }
14      } while (!equal);
15      System.out.println("ggT: " + y);
```

Sie dürfen dabei

- `new java.util.Scanner(System.in).nextInt` durch `read` ersetzen.
- `System.out.println` durch `write` ersetzen.

Hinweis: Es gibt keine strikt einzuhaltende Stilvorgabe zum Zeichnen des Graphen. Aus dem Graphen muss jedoch eindeutig der Kontrollfluss hervorgehen, der zwischen Anweisungen oder dem Auswerten von Ausdrücken im Programm vorkommen kann. Anfang und Ende des Kontrollflusses sind zu markieren. (Hierzu können Sie jeweils einen extra Start- bzw. Endknoten einfügen, der keine Anweisung oder Bedingung enthält.) Zudem muss ggf. klar gekennzeichnet werden, unter welchen Bedingungen der Kontrollfluss wohin verzweigt. Recherchieren Sie dazu den Kontrollfluss zum Switch-Statement und zur Do-While-Schleife selbständig und wenden Sie die Ergebnisse Ihrer Nachforschungen für das gegebene Programm an.