Lehrstuhl für Angewandte Informatik I
Prof. Dr. Bernhard Westfechtel
Sandra Greiner

WS 2017/18
16. November 2017

Konzepte der Programmierung

Übung 5 – Abgabe: 30.11.2017

Aufgabe 5.1

Wir betrachten die kontextfreie Sprache L_T, die den Verlauf einer *Theateraufführung* in vereinfachter Form beschreibt. Wir nehmen an, dass ein Theaterstück aus zwei Akten besteht. Ein Akt beginnt mit der Regieanweisung „Vorhang Auf“ (`VA`) und endet mit „Vorhang Zu“ (`VZ`). Jeder Akt besteht aus mindestens drei Szenen. Zwischen zwei Szenen, aber nicht am Anfang oder Schluss des Akts, findet ein `Szenenwechsel` (`SW`) statt. Innerhalb einer Szene gibt es mehrere `Sprechereignis`se (`S`) oder Gesänge (`G`) in beliebiger Reihenfolge, aber mindestens eines von beiden einmal.

Ein Beispiel-Theaterstück könnte folgendermaßen ablaufen:

```
VA S S G S S S SW S S S G S SW G VZ
VA S S S S SW G SW S VZ
```

Geben Sie eine **BNF** G_T an, die alle Worte in der Sprache L_T erzeugt. Leerzeichen und Zeilenumbrüche müssen nicht berücksichtigt werden.

(**3 Punkte**)

Aufgabe 5.2 (Objektorientierte Programmierung)

Beschreiben Sie in je maximal zwei Sätzen mit **eigenen Worten** die folgenden **OOP**-Konzepte:

- Klasse und Objekt
- Objektfeld
- Konstruktor
- Klassenmethode vs. Objektmethode

(**3 Punkte**)

Aufgabe 5.3 (Ableitungen)

Gegeben sei folgende kontextfreie Grammatik $G = (N, T, P, S)$ für arithmetische Ausdrücke:
N = {expression, term, primary}, T = {NUMBER, (,), *, /, +, -}
S = expression, $P = \{p_1, ...p_9\}$ mit

p_1: expression → expression + term
p_2: expression → expression - term
p_3: expression → term
p_4: term → term * primary
p_5: term → term / primary
p_6: term → primary
p_7: primary → NUMBER
p_8: primary → - primary
p_9: primary → (expression)

(a) Geben Sie an, ob die folgenden Wörter in der durch G beschriebenen Sprache der arithmetischen Ausdrücke enthalten sind (keine Begründung nötig):

w_1: 13

w_2: (NUMBER ( ) * NUMBER)

w_3: (- (- NUMBER))

w_4: NUMBER + + NUMBER

w_5: primary $\rightarrow$ NUMBER

w_6: - NUMBER * - NUMBER * - NUMBER

Gegeben ist das Wort w_7: NUMBER * ((NUMBER - NUMBER) / -NUMBER) + NUMBER

(b) Geben Sie einen Ableitungs**baum** für w_7 an.

(c) Geben Sie eine Ableitungs**folge** (s. Kapitel *Syntax*) für w_7 an.

Hinweis: Kürzen Sie Terminal- und Nichtterminalsymbole zugunsten der Übersichtlichkeit ab.

(**3+4+3=10 Punkte**)

Aufgabe 5.4 (Aufzählungstypen und Wahrheitswerte)
Erstellen Sie eine Klasse `Messenger`. Definieren Sie innerhalb dieser Klasse einen **Aufzählungstypen** `MessengerStatus` und vergeben Sie jeweils ein geeignetes **Literal** für die folgenden Zustände:

- Benutzer ist anwesend.
- Benutzer ist beschäftigt.
- Benutzer will nicht gestört werden.
- Benutzer ist ausgeloggt.

Schreiben Sie außerdem in der Klasse `Messenger` eine `main`-Methode, in der mindestens drei Variablen vom Typ `MessengerStatus` initialisiert werden.

Anschließend soll jeweils ein positiver und ein negativer Vergleich (`==` bzw. `!=`) zwischen zwei unterschiedlichen Variablen stattfinden. Die Ergebnisse der Vergleiche sollen in zwei weiteren Variablen vom Typ `boolean` festgehalten werden.

Geben Sie schließlich den Wert der drei Variablen sowie das Ergebnis der beiden Vergleiche auf der Konsole aus. was stellen Sie dabei fest?

(**4 Punkte**)

Aufgabe 5.5 (Zeichen – Symmetrische Kryptographie)
Symmetrische Kryptographie ist ein seit den Tagen Caesars bekanntes Verfahren zur Verschlüsselung geheimer Botschaften. Gaius Julius Caesar selbst benutzte meist das folgende Verfahren:

- Aufschreiben des gesamten Alphabets
- Verschieben des Alphabets um 3 Buchstaben, sodass sich folgende „Tabelle“ ergibt:

  ```
  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  DEFGHIJKLMNOPQRSTUVWXYZABC
  ```

- Benutzen der so entstandenen „Tabelle“ als Schlüssel. Damit wird aus dem Wort „UEBUNG“ die Chiffre „XHEXQJ“

- Leerzeichen bleiben.

Mehr Infos zum Caesar-Verfahren finden Sie z.B. bei Wikipedia (→ Cäsar-Verschlüsselung).

Tipp zur Bearbeitung der folgenden Aufgaben: Character werden intern als Zahl repräsentiert und können daher mit Vergleichsoperatoren verglichen und mit Rechenoperationen verändert werden. Die Zuordnung Zeichen zu Zahl kann man in ASCII-Tabellen[1]

```
char c ='A';
c =(char) (c + 1); // in c ist nun der Buchstabe 'B' gespeichert.
```

Erstellen Sie eine Klasse `Zeichenbehandler` mit den folgenden Eigenschaften:

(a) Zur Vereinfachung gehen wir nur von Großbuchstaben aus. Schreiben Sie dafür eine Klassenmethode, der ein einzelner Buchstabe übergeben wird. Handelt es sich dabei um einen Großbuchstaben, gibt die Methode `true` zurück, sonst `false`.

(b) Implementieren Sie eine Klassenmethode `verschluessle(...)` zur Umwandlung eines einzelnen characters (Datentyp: char) gemäß eines Schlüssels. Entsprechend werden der Methode ein character und der Schlüssel übergeben und der konvertierte Buchstabe zurückgegeben.

(c) Schreiben Sie eine Klassenmethode `verschluessleWort(...)`, der die Länge des Wortes und ein Schlüssel übergeben wird. Sie soll die einzelnen Buchstaben vom Benutzer nacheinander eingeben lassen. Wenn es sich um Großbuchstaben handelt, sollen Sie mittels der Methode `verschluessle()` umgewandelt werden und in einem entsprechend großen Array gespeichert werden. Das Array wird von der Methode zurückgegeben. Handelt es sich nicht um einen Großbuchstaben, bricht die Methode vorzeitig ab, gibt einen Fehler auf der Konsole aus und gibt `null` zurück.

(d) Schreiben Sie nun ein Hauptprogramm in einer neuen Klasse (`Main`) mit main-Methode. Sie soll zunächst vom Benutzer den gewünschten Schlüssel erfragen. Dann soll Sie ihn um die Länge des übergebenen Wortes bitten. Mithilfe der beiden Informationen wird das Verschlüsselungsverfahren (`verschluessleWort()`) gestartet. Das Ergebniswort soll anschließend durch die main-Methode auf der Konsole ausgegeben werden.

Hinweis:

a) Beachten Sie, dass eventuell der Wert `null` für das Wortarray zurückgegeben wird. In diesem Fall soll nochmals ein Fehler ausgegeben werden.

b) Sie können auf Klassenmethoden einer anderen Klasse mit der Punktnotation zugreifen, also `<Klassenname>.<Methodenname>`. Beispiel:
Angenommen Klasse `A` bietet die öffentliche Klassenmethode `b()` an, dann erfolgt die Ausführung in einer anderen Klasse so: `A.b();`

(e) **[optional]** Implementieren Sie analog Methoden zur Entschlüsselung eines Wortes.

Eine Beispielausgabe einer zweimaligen Ausführung des Hauptprogramms könnte wie auf der nächsten Seite aussehen.

(1+3+4+4=12 Punkte)

[1] z.B. hier: http://exe.fam-trachsel.ch/InfoGL/der_asciicode__bung.html

```
Geben Sie den gewünschten Schluessel ein:
5
Geben Sie jetzt die Laenge des zu verschluesselnden Wortes ein:
3
Geben Sie den 1.  Buchstaben ein
z
Es handelt sich bei z nicht um einen Grossbuchstaben!
Sie haben ungueltige Buchstaben eingegeben
Geben Sie den gewünschten Schluessel ein:
5
Geben Sie jetzt die Laenge des zu verschluesselnden Wortes ein:
3
Geben Sie den 1.  Buchstaben ein
Z
Geben Sie den 2.  Buchstaben ein
O
Geben Sie den 3.  Buchstaben ein
H
Das Wort lautet verschluesselt:
ETM
```