Praktikum Grundlagen der Programmierung

Prof. Dr. Harald Räcke, J. Kranz, A. Reuss
WS 2017/18

Übungsblatt 4
Abgabefrist: 20.11.2017 um 05:00 Uhr
Σ Punkte: 20

Die folgenden beiden Methoden der Klasse String werden auf diesem Blatt verwendet:

- `char charAt(int i)`: Um auf ein Zeichen innerhalb eines Strings zuzugreifen, können Sie die Methode `charAt(i)` verwenden, welche ein Character an der Position `i` zurückliefert.

  Beispiel: `String s = "Hello Students"; char c = s.charAt(7);` speichert den Character `'t'` in der Variablen `c`. **Achtung**: Java indiziert ab der Stelle 0 und nicht erst ab 1.

- `int length()`: Um die Länge eines Strings zu bestimmen, rufen Sie die Methode `length()` auf.

  Beispiel: `String s = "Hello Students"; int l = s.length();` speichert den Wert `14` in der Variablen `l`.

Eine Integer-Expression `e` können Sie durch `(char) e` in einen Buchstaben umwandeln. Zum Beispiel speichert `int i = 65; char c = (char) i;` in der `char`-Variablen `c` das Zeichen `'A'`. Machen Sie sich mit der **ASCII-Tabelle**[1] vertraut.

Zusätzlich zu den in der Vorlesung besprochenen Operatoren kennt Java noch eine Reihe sog. *bitweiser* Operatoren. Bitweise Operatoren verknüpfen jeweils zwei Bits ihrer Operanden. Java bietet folgende bitweise Operatoren an:

Funktion	Opeator	Beispiel
`bitweises und`	&	$1001_2 \,\&\, 1010_2 = 1000_2$
`bitweises oder`	|	$1001_2 \mid 1010_2 = 1011_2$
`bitweises not`	~	$\sim 1010_2 = 0101_2$
`bitweises xor` ($\oplus$)	^	$1001_2 \,\hat{}\, 1010_2 = 0011_2$

Beachten Sie, dass die Zahlen im Beispiel zur Basis 2 dargestellt sind.

Aufgabe 4.1 (P) Zahlenblasen

1. 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.

   (a) $323478_9 + 111202337_9$

   (b) $1010101100_2 * 11000111_2$

   (c) $c01dc0ffe_{16} * deadaffe_{16}$

   (d) $120022_3 * 22210_3$

[1] `http://www.torsten-horn.de/techdocs/ascii.htm`

2. Führen Sie die folgenden Basisumwandlungen durch. Wandeln Sie die gegebene Zahl nicht erst ins Dezimalsystem um.

   (a) Stellen Sie 1010101100_2 zur Basis 10 dar.

   (b) Stellen Sie 1010101100_2 zur Basis 16 dar.

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

Aufgabe 4.2 (P) Cäsar lebt

Schreiben Sie ein Programm, das einen `String` einliest und diesen mittels der Cäsar-Chiffre[2] verschlüsselt. Machen Sie sich mit der Cäsar-Verschlüsselung vertraut.

Der Einfachheit halber werden wir nur Buchstaben `a` bis `z` und `A` bis `Z` verschlüsseln. D.h. alle anderen Zeichen sollen nicht verschlüsselt werden und tauchen somit als Klartext im Geheimtext auf.

Beispiel: Der Geheimtext zum Klartext `"Hello Students! .aAbBcC? >wWxXyYzZ<"` lautet `"Khoor Vwxghqwv! .dDeEfF? >zZaAbBcC<"`, wenn der zyklische Shift 3 beträgt.

Gehen Sie wie folgt vor:

(a) Lesen Sie einen `String` mittels der MiniJava-Methode `readString()` ein, der später ver-/entschlüsselt werden soll.

(b) Lesen Sie einen `Integer` mittels der MiniJava-Methode `read()` ein, der als zyklischer Shift verwendet werden soll. Beachten Sie, dass der `Integer` sowohl negativ als auch betragsmäßig größer als 26 sein kann!

(c) Implementieren Sie nun den Algorithmus und ver-/entschlüsseln Sie den `String` aus (a) anhand des Schlüssels aus (b).

(d) Beachten sie Groß- und Kleinschreibung im Klartext wie auch Geheimtext.

(e) Geben Sie den verschlüsselten `String` mittels `write(String s)` aus.

Hinweis: Sie können Ihre Implementierung selbst testen, indem Sie den Geheimtext mit dem Schlüssel $-n$ nochmals „verschlüsseln“, sofern der Geheimtext vorher mit dem Schlüssel n verschlüsselt wurde. Die Ausgabe sollte der initiale Klartext sein.

Aufgabe 4.3 (P) Vokalersetzung

Schreiben Sie ein Programm `Vokalersetzung`, das im unten gegebenen Text alle Vokale durch einen vom Benutzer eingegebenen Vokal ersetzt. Achten Sie darauf, dass die Groß- und Kleinschreibung nach der Ersetzung mit der vor der Ersetzung übereinstimmt. Umlaute (Ä/ä, Ö/ö, usw.) sollen dabei nicht als Vokale betrachtet werden. Beispiel: Das Wort `"Exenmeister"` wird bei Eingabe von O oder o zu `"Oxonmoostor"`.

Verwenden Sie keine Bibliotheksfunktionen der Klasse `String` außer `char charAt(int i)` und `int length()`.

Verwenden Sie das folgende Codegerüst:

[2] `https://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung`

```java
public class Vokalersetzung extends MiniJava {

    public static void main(String[] args) {

        String text = "Hat der alte Hexenmeister\n" +
                      "sich doch einmal wegbegeben!\n" +
                      "Und nun sollen seine Geister\n" +
                      "auch nach meinem Willen leben.\n" +
                      "Seine Wort und Werke\n" +
                      "merkt ich und den Brauch,\n" +
                      "und mit Geistesstärke\n" +
                      "tu ich Wunder auch.\n" +
                      "Walle! walle\n" +
                      "Manche Strecke,\n" +
                      "daß, zum Zwecke,\n" +
                      "Wasser fließe\n" +
                      "und mit reichem, vollem Schwalle\n" +
                      "zu dem Bade sich ergieße.";
    }

}
```

Aufgabe 4.4 (P) Rechtschreibschwäche

Schreiben Sie ein MiniJava-Programm, das einen `String` einliest und darin jeden Kleinbuchstaben durch den jeweiligen Großbuchstaben ersetzt und umgekehrt. Alle anderen Zeichen sollen nicht ersetzt werden. Geben Sie den konvertierten `String` via der Methode `write(String s)` aus.

Beispiel: `"Hello Students!"` wird umgewandelt in `"hELLO sTUDENTS!"`.

Verwenden Sie keine Bibliotheksfunktionen der Klasse `String` außer `char charAt(int i)` und `int length()`.

Die Hausaufgabenabgabe erfolgt über Moodle. Bitte geben Sie Ihren Code als UTF8-kodierte (ohne BOM) Textdatei(en) mit der Dateiendung `.java` ab. Geben Sie **keine** Projektdateien Ihrer Entwicklungsumgebung ab. Geben Sie **keinen** kompilierten Code ab (`.class`-Dateien). Geben Sie Ihren Code **nicht** als Archiv (z.B. als `.zip`-Datei) ab. Nutzen Sie **keine** Packages. Achten Sie darauf, dass Ihr Code kompiliert. Bitte vermerken Sie aus Datenschützgründen nicht Ihren Namen oder Ihre Matrikelnummer im Code. Auf diesem Blatt gibt es auch Aufgaben, zu denen die Lösung kein Programm ist. Geben Sie diese Aufgaben in einer einzelnen, korrekt orientierten und gut lesbaren DIN-A4-PDF-Datei ab. Hausaufgaben, die sich nicht im vorgegebenen Format befinden, werden nur mit Punktabzug oder gar nicht bewertet.

Aufgabe 4.5 (H) Mehr Zahlenblasen [4 Punkte]

Geben Sie bei den folgenden Teilaufgaben Ihren Rechenweg mit ab. Lösungen ohne erkennbaren Rechenweg können nicht bewertet werden.

1. Führen Sie die folgenden Additionen, Multiplikationen und Subtraktionen schriftlich aus. Wandeln Sie die Basis der Zahlen dabei nicht um, sondern rechnen direkt mit den Zahlen zu der jeweils gegebenen Basis.
   (a) $C0FF3E_{16}$ * $57AFF5_{16}$
   (b) 24601357_{8} + 25574234_{8}
   (c) $DE1E7E_{15}$ - $ACCE55_{15}$
2. Führen Sie die folgenden Basisumwandlungen durch. Wandeln Sie die gegebene Zahl nicht erst ins Dezimalsystem um.
   (a) Stellen Sie $164791a405_{11}$ zur Basis 16 dar.

Aufgabe 4.6 (H) Stringray [4 Punkte]

Schreiben Sie ein Programm *Stringray*, das die folgenden Funktionen implementiert. Zunächst soll der Benutzer einen beliebigen Text eingeben können, danach darf der Benutzer in einem Hauptmenu verschiedene Operationen auf diesen Text anwenden.

Verwenden Sie keine Bibliotheksfunktionen der Klasse String außer `char charAt(int i)` und `int length()`.

Implementieren Sie die folgenden Funktionalitäten:

1. **Häufigkeit der Buchstaben**

   Ihr Programm soll für jeden vorkommenden Buchstaben ausgeben, wie oft er im Text vorkommt. Die Ausgabe soll in alphabetische Reihenfolge erfolgen und nicht vorkommende Buchstaben **nicht** ausgeben; zudem soll nicht zwischen Groß- und Kleinschreibung unterschieden werden. Umlaute dürfen Sie gerne als zusätzliche Herausforderung ausgeben, dies ist jedoch nicht erfordert. Alle anderen Zeichen sollen ignoriert werden.

   Beispiel:

   Eingabe “Dass ich erkenne, was die Welt // Im Innersten zusammenhält.“

   Ausgabe A: 3 C: 1 D: 2 E: 8 H: 2 I: 4 K: 1 L: 2 M: 3 N: 6 R: 2 S: 5 T: 3 U: 1 W: 2 Z: 1

2. **Buchstaben ersetzen**

   Der Benutzer soll nacheinander zwei Buchstaben eingeben; besteht die Eingabe nicht aus zwei einzelnen Buchstaben, soll sie wiederholt werden. Ihr Programm soll dann im eingegebenen Text den ersten Buchstaben überall durch den zweiten Buchstaben ersetzen. Es sollen sowohl die groß- wie die kleingeschriebenen Buchstaben unter Beibehaltung der Groß-/Kleinschreibung durch den entsprechenden Buchstaben ersetzt werden.

   Beispiel:

   Eingabe: "Acht alte Ameisen aßen am Abend Ananas"

   Buchstabe: "A"→ "o"

   Ausgabe: "Ocht olte Omeisen oßen om Obend Ononos."

3. **Wortweise Spiegeln**

   Ihr Programm soll jedes Wort eines eingegebenen Textes rückwärts ausgeben. Die einzelnen Wörter werden dabei jeweils durch ein Leerzeichen getrennt, andere Satzzeichen werden als Teil des Wortes behandelt.

   Beispiel:

   Eingabe: "Da dank ich Euch; denn mit den Toten. Hab ich mich niemals gern befangen. Am meisten lieb ich mir die vollen, frischen Wangen. Für einem Leichnam bin ich nicht zu Haus; Mir geht es wie der Katze mit der Maus."

   Ausgabe: "aD knad hci ;hcuE nned tim ned .netoT baH hci hcim slamein nreg .negnafeb mA netsiem beil hci rim eid ,nellov nehcsirf .negnaW rüF menie manhcieL nib hci thcin uz ;suaH riM theg se eiw red eztaK tim red .suaM"

Aufgabe 4.7 (H) Binnenmajuskel [4 Punkte]

In dieser Aufgabe betrachten wir Strings, die nur aus Kleinbuchstaben ('a' bis 'z'), Großbuchstaben ('A' bis 'Z') und Unterstrichen bestehen.

Gegeben sind die folgenden Regeln, um aus mehreren (≥ 1) Begriffen (z. B. Hand, Ball und Tor) entsprechend zusammengesetzte Namen (z. B. Handballtor) zu bilden:

1. Startcase: Das Wort beginnt mit einem Großbuchstaben und enthält ansonsten nur Kleinbuchstaben.
   Beispiel: Handballtor

2. UPPERCASE: Das Wort besteht nur aus Großbuchstaben.
   Beispiel: HANDBALLTOR

3. snake_case: Die einzelnen Begriffe werden klein geschrieben und durch Unterstriche getrennt.
   Beispiel: hand_ball_tor

4. PascalCase: Die einzelnen Begriffe werden zusammengeschrieben, jedoch durch Großschreiben des jeweils ersten Buchstabens jedes Begriffs „getrennt".
   Beispiel: HandBallTor

Schreiben Sie ein Programm, das es der Benutzerin erlaubt, nacheinander beliebig viele Begriffe einzugeben, die nur aus Buchstaben bestehen. Die Groß-/Kleinschreibung der Begriffe darf dabei beliebig *falsch* sein (s. u. das Beispiel). Wird ein ungültiger String eingegeben, soll der Benutzer auf seinen Fehler hingewiesen und die letzte Eingabe ignoriert werden. Die Eingabe der Begriffe soll schließlich durch Eingabe eines leeren Strings beendet werden. Als Ausgabe liefert das Programm dann für jede der oben genannten Konventionen (Startcase, UPPERCASE, snake_case, PascalCase) die entsprechenden Namen. Die jeweilige Zuordnung soll aus der Ausgabe klar hervorgehen.

Beispiel (ohne ungültige Eingaben):

Eingabe: „hand“, „BAll“, „toR“, „“.
Ausgabe: „Startcase: Handballtor“,
„UPPERCASE: HANDBALLTOR“,
„snake_case: hand_ball_tor“,
„PascalCase: HandBallTor“.

Aufgabe 4.8 (H) XOR-Verschlüsselung im CBC-Modus [8 Punkte]

In dieser Aufgabe geht es darum, einen String zu verschlüsseln. Zur Verschlüsselung soll hier die XOR-Operation[3] ($\oplus$) dienen. Ihr Programm soll dabei folgende Schritte durchführen:

1. Der Benutzer wird um einen Schlüssel und einen Initialisierungsvektor gebeten; beides ist jeweils eine Zahl zwischen 0 und 63. Gibt die Benutzerin ungültige Zahlen ein, so soll sie um erneute Eingabe gebeten werden.
2. Der Benutzer wird um die Eingabe eines Textes gebeten, der verschlüsselt werden soll.
3. Jeder Buchstabe des Textes wird auf eine Zahl zwischen 0 und 63 abgebildet. Dieser Vorgang wird im folgenden als *Kodierung* bezeichnet und darf nicht mit *Verschlüsselung* verwechselt werden.
4. Die Verschlüsselung wird durchgeführt und der verschlüsselte Text ausgegeben. Der kodierte Schlüsseltext muss dazu dekodiert und in einen String umgewandelt werden.
5. Der verschlüsselte Text wird entschlüsselt, dekodiert und ausgegeben.

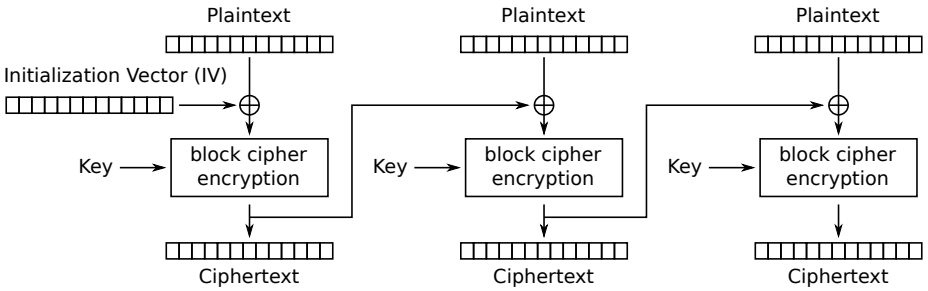
Um den Text zu verschlüsseln, soll dieser hier zunächst speziell kodiert werden. Jedem erlaubten Zeichen wird eine Zahl entsprechend folgender Tabelle zugeordnet:

Zeichen	**Kodierung**
a bis z	0 bis 25
A bis Z	26 bis 51
0 bis 9	52 bis 61
Leerzeichen	62
Punkt (.)	63

[3] vgl. `https://en.wikipedia.org/wiki/Exclusive_or` sowie die Einleitungshinweise

Gibt die Benutzerin einen Text ein, der ein Zeichen enthält, das in der Tabelle nicht vorkommt, so soll das Programm eine Fehlermeldung ausgeben und sich beenden. Durch die Kodierung wird z.B. der Text `"Hallo."` auf das Array von `int`-Zahlen `{33, 0, 11, 11, 14, 63}` abgebildet.

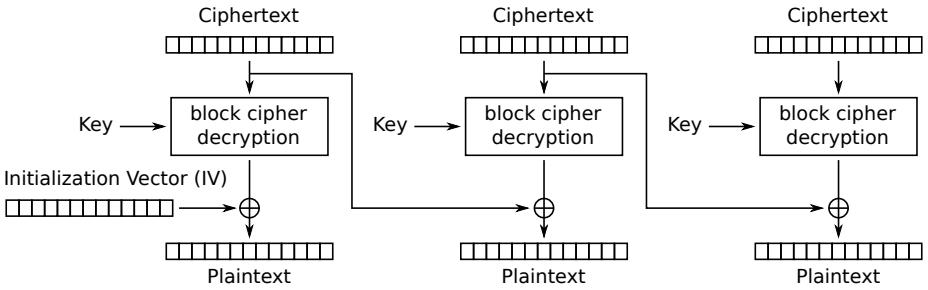
Der kodierte Text wird verschlüsselt, indem jedes kodierte Zeichen mittels XOR-Operation mit dem Schlüssel verknüpft wird. Wird jedoch jedes Zeichen separat verschlüsselt, lassen sich bei größeren Texten sehr leicht Muster erkennen[4]. Aus diesem Grund wollen wir hier das sog. *Cipher Block Chaining (CBC)* verwenden, welches die Verschlüsslung entsprechend dem folgenden Muster durchführt:



Der Intialisierungsvektor (IV) stammt hier vom Benutzer. Ein *Block* in der Abbildung entspricht in unserem Fall genau einem Zeichen. Insgesamt wird auf diese Weise also jedes Zeichen nicht nur verschlüsselt, sondern zusätzlich mit dem vorherigen Zeichen verknüpft. Für die Entschlüsselung machen wir uns zunutze, dass zweifache Anwendung der XOR-Operation die Identität ist, also die folgende Gleichheit für jedes x und y gilt:

$$(x \oplus y) \oplus y = x$$

Ein Zeichen wird dementsprechend entschlüsselt, indem es erneut mit dem Schlüssel XOR-verknüpft wird. Natürlich muss auch hier wieder beachtet werden, dass wir im CBC-Modus arbeiten, also wie folgt gezeigt vorgegangen werden:



Als Beispiel gebe der Benutzer den Schlüssel 42, den Initialisierungsvektor 22 und den Text *Pinguine sind knuffige Tiere.* ein. Das Programm muss in diesem Fall den verschlüsselten Text *v3q8cGhP9fNaP9D6eReMkKW3v7aU7* ausgeben.

Hinweis: Beginnen Sie damit, die Verschlüsselung ohne CBC-Modus zu implementieren. Im nächsten Schritt können Sie Ihre Implementierung erweitern.

[4]vgl. ECB-Modus auf `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`