hochschule mannheim

Fakultät für Informatik
Prof. Dr. Peter Knauber

Bachelor-Studiengang Informatik
Übungen zur Vorlesung “Grundlagen der Informatik” (GDI), WS 2011/2012

Übungsblatt 5: Die Bacon-Chiffre (nach D. Marschall)

Ausgabe am: 28.10.2011
Abgabe am: 11.11.2011

Aufgabe 1: **4 + 3 + 6 + 5 + 5 + 2 + 5 = 30 Punkte**

Steganografie ist die Kunst des Versteckens einer geheimen Nachricht (Ursprungstext) in einem unscheinbaren Trägermedium. Das Ergebnis wird Steganogramm genannt. Ein sehr bekanntes klassisches Steganografiesystem ist z.B. das Schreiben mit Zitronensäure auf (das Medium) Papier.

Der Bacon-Chiffre ist ein auf Francis Bacon (1561 - 1626) zurückgehendes klassisches Steganographieverfahren, bei dem der Ursprungstext in einen Binärcode umgewandelt und anschließend in einen anderen Text z.B. mittels Groß- und Kleinschreibung untergebracht wird.

Vorgehensweise:

1. Jedem Buchstaben des Ursprungstextes wird ein fünfstelliger Binärcode zugeordnet:

Buchstabe	Code	Buchstabe	Code	Buchstabe	Code
A	kkkkk	I, J	kgkkk	R	gkkkk
B	kkkkg	K	kgkkg	S	gkkkg
C	kkkgk	L	kgkgk	T	gkkgk
D	kkkgg	M	kgkgg	U, V	gkkgg
E	kkgkk	N	kggkk	W	gkgkk
F	kkgkg	O	kggkg	X	gkgkg
G	kkggk	P	kgggk	Y	gkggk
H	kkggg	Q	kgggg	Z	gkggg

2. Diese Kodierung wird nun in einen anderen größeren Text, der als Trägermedium fungiert, mittels Anpassung der Groß- und Kleinschreibung versteckt. Für "k" wird ein **K**leinbuchstabe und für "g" ein **G**roßbuchstabe verwendet.[1]
   Die Entschlüsselung findet analog statt. Die Groß- und Kleinschreibung des Steganogramms wird untersucht und als k/g-Binärcode aufgeschrieben. Dieser Binärcode wird nun anhand der obigen Tabelle in den Ursprungstext zurückgewandelt.

   Beispiel: Verstecken der Botschaft "Wikipedia":

```
Ursprungstext    W     i     k      i    p    e     d    i       a
Binärkodierung   gkgk kkg kkkk gkkg kgkkkkgggkkk gkkkkggk  gkkk kkkk k....
Trägermedium     Dies ist eine fast unauffällige Nachricht, oder etwa nicht?
Steganogramm     DiEs isT eine FasT uNauffÄLLige NachriCHt, Oder etwa nicht?
```

1.Bacon selbst verwendete eine weitaus unauffälligere Variante: Er behielt die Groß- und Kleinschreibung des Mediums bei und verwendete bei allen "g"-kodierten Zeichen eine leicht abgeänderte Variante des jeweiligen kleinen bzw. großen Buchstabens in seiner Handschrift.

Aufgabenstellung

Legen Sie eine Klasse *BaconChiffre* in dem Package *uebung05* an.

a) Schreiben Sie eine Funktion *static String reinigeUrsprungstext(String text)*, die aus einem geheimen Ursprungstext *alle* Zeichen entfernt, die *keine* Buchstaben sind (also auch Leerzeichen und Satzzeichen). Zusätzlich sollen alle Kleinbuchstaben in Großbuchstaben umgewandelt und deutsche Sonderzeichen (Ä, Ö, Ü, ß) sinnvoll ersetzt werden (AE, OE, UE, SS).
   Beispiel:
```
Eingabe: "Hallo Welt!"
Ausgabe: HALLOWELT
```

b) Schreiben Sie eine Funktion *static String kodiereUrsprungstext(String text)*, die einen zuvor gereinigten Ursprungstext gemäß dem Binärcode aus der Tabelle kodiert.
   Für den Fall, dass die Funktion falsch verwendet wird (nicht kodierbare Zeichen bzw. keine Buchstaben), soll das betreffende Zeichen ignoriert und eine Warnmeldung auf der Konsole ausgegeben werden. Die Funktion soll ansonsten normal weiterarbeiten.
   Beispiel:
```
Eingabe: HALLOWELT
Ausgabe: kkgggkkkkkkgkgkkgkkggkggkgkkkkgkkkgkgkgkkgk
```

c)) Schreiben Sie eine Funktion *static String dekodiereUrsprungstext(String binaerCode)*, die die Dekodierung analog zu Teilaufgabe b durchführt.
   Zusätzliche Anforderungen:
   - Ist ein 5er-Block unbekannt, soll "#" als Zeichen in die Ausgabe eingesetzt werden. (Es soll keine Warnmeldung auf der Konsole ausgegeben werden).
   - Verwenden Sie bei den doppeldeutigen Codes "I, J" und "U, V" jeweils den ersten Buchstaben, also "I" bzw. "U".
   - Es ist möglich, dass die Länge des Binärcodes nicht durch 5 teilbar ist. In diesem Fall soll die Funktion den unvollständigen 5er-Block am Ende stillschweigend ignorieren.

   Beispiel:
```
Eingabe: kkgggkkkkkkgkgkkgkkggkggkgkkkkgkkkgkgkgkkgk
Ausgabe: HALLOWELT
```

d) Schreiben Sie eine Funktion *static String versteckeText(String ursprungsText, String mediumText)*, die einen beliebigen Ursprungstext (muss nicht gereinigt sein) in einem Trägermedium versteckt. Nutzen Sie die zuvor implementierten Funktionen *reinigeUrsprungstext* und *kodiereUrsprungstext*. Die ursprüngliche Groß- und Kleinschreibung des Trägermediums geht logischerweise verloren.
   Vorsicht:
   - Leerzeichen und Satzzeichen (bzw. alle Nicht-Buchstaben) des Trägermediums sowie das "ß" sollen erhalten bleiben. Die Chiffrierung mittels Änderung der Groß- und Kleinschreibung soll dann beim nächsten Buchstaben fortgeführt werden.
   - Die Funktion soll eine Warnmeldung auf der Konsole ausgeben, wenn das Trägermedium zu kurz ist und der Binärcode nicht vollständig in ihm untergebracht werden kann. (Das Trägermedium muss mindestens so viele *Buchstaben* (außer "ß") haben, wie die Binärkodierung des Ursprungstextes lang ist). Die Funktion soll trotz Fehler ein (wenn auch unvollständiges) Steganogramm zurückliefern.

   Beispiel:
```
ursprungsText: "Hallo Welt!"
mediumText:    Es ist traurig zu denken, die Natur spricht und keiner hört zu.
Ausgabe:       es IST trauriG zU deNkEn, dIE nATuR spriCht uNd KeIneR hört zu.
```

e) Schreiben Sie eine Funktion *static String zeigeText(String steganogramm)*, die die Groß- und Kleinschreibung eines Steganogramms analysiert und dadurch den Ursprungstext zurückliefert. Nutzen Sie die zuvor implementierten Funktionen reinigeUrsprungstext() und dekodiereUrsprungstext().
Beispiel:
```
Eingabe: es IST trauriG zU deNkEn, dIE nATuR spriCht uNd KeIneR hört zu.
Ausgabe: HALLOWELT
```

f) Schreiben Sie eine Methode *main*, die die Botschaft "Treffen uns um drei Uhr am Bahnhof!" im ersten Kapitel von 1. Mose, Vers 1-3 der Bibel ("Am Anfang schuf Gott Himmel und Erde. Und die Erde war wüst und leer, und es war finster auf der Tiefe; und der Geist Gottes schwebte auf dem Wasser. Und Gott sprach: Es werde Licht! und es ward Licht.") versteckt, den Code ausgibt und anschließend die Botschaft wieder sichtbar macht.
Anmerkung:
Aufgrund der Überlänge des Trägermediums werden bei der Dechiffrierung 3 zusätzliche Zeichen ermittelt und 1 unvollständiger 5er-Block mit 1 Bit ignoriert.

g) Erweitern Sie die Main-Routine aus Teilaufgabe f, sodass eine interaktive konsolenbasierte Anwendung entsteht. Zu Beginn soll der Benutzer in einem Hauptmenü zwischen folgenden Punkten auswählen können:
   1. Botschaft verstecken
      Der Benutzer soll einen beliebigen Ursprungstext sowie ein beliebiges Trägermedium eingeben können. Nach der Ausgabe des Steganogramms soll das Programm zurück ins Hauptmenü wechseln.
   2. Botschaft sichtbar machen
      Der Benutzer soll ein beliebiges Steganogramm eingeben können. Nach der Ausgabe des Ursprungstextes soll das Programm zurück ins Hauptmenü wechseln.
   3. Demo
      Es soll das Beispiel aus Teilaufgabe f mit anschaulicherer Darstellung und Zwischenschritten ausgegeben werden.
   4. Programm beenden
      Das Programm soll beendet werden.

Hinweise

- Die Aufgaben sind in Eclipse zu bearbeiten.
- Von allen Aufgaben sind Programmausdrucke (Listings) abzugeben, *keine* Ausdrucke von Testläufen. Die Aufgaben sind im Labor mit Eclipse vorzuführen.
- Erlaubt sind *MakeItSimple*-Funktionen (keine nicht besprochene Funktionalität aus der Java Standard Bibliothek) und das bisher erworbene Wissen aus den GDI-Vorlesungen. Fragen Sie, bevor Sie Java-Konstrukte verwenden, die noch nicht behandelt wurden! Zusätzliche eigene Hilfsfunktionen (keine fremden oder externen) sind ausdrücklich erlaubt.
- In den Laborstunden sollen Ihre Programme automatisch getestet werden. Damit Sie vorab prüfen können, ob Ihr jeweiliges Programm äußerlich korrekt ist, finden Sie im Wiki für jede geforderte Methode ein JUnit-Testprogramm.