Albert-Ludwigs-Universität Freiburg
Informatik 1
Wintersemester 2011/12

Prof. Dr. Wolfram Burgard
Dipl. Ing. Jörg Röwekämper
Dipl. Inf. Markus Kuderer

Übungsblatt 14

Abgabe bis Donnerstag, 16.02.12, 12:00 Uhr

Aufgabe 14.1

Implementieren Sie eine Haskell-Funktion `palindrome`, die die Signatur
`palindrome :: [Char] -> Bool`
besitzt. Diese Funktion soll genau dann `true` zurückgeben, wenn die eingegebene Liste ein Palindrom ist, d.h. von hinten und von vorn gelesen gleich ist.

Aufgabe 14.2

Implementieren Sie eine Haskell-Funktion `eliminateDuplicates`, die die Signatur
`eliminateDuplicates :: [Char] -> [Char]`
besitzt. Diese Funktion soll aufeinander folgende Duplikate in der eingegebenen Liste löschen.
Beispiel: `eliminateDuplicates "aavbbb"` ergibt `"avb"`

Aufgabe 14.3

Implementieren Sie eine Haskell-Funktion `rotateList`, die die Signatur
`rotateList :: [Char] -> Int -> [Char]`
besitzt. Diese Funktion soll die eingegebene Listd um n Stellen nach links rotieren.
Beispiel: `rotateList "Informatik1" 3` ergibt `"ormatik1Inf"`

Aufgabe 14.4

Implementieren Sie eine Haskell-Funktion `primeFactor`, die die Signatur
`primeFactor :: Int -> [Int]`
besitzt. Diese Funktion soll die Primfaktoren einer positiven Integer-Zahl bestimmen.
Beispiel: `primeFactor 66` ergibt `[2,3,11]`

Aufgabe 14.5

Im Folgenden soll ein Java-Programm entwickelt werden, das Texte verschlüsseln bzw. entschlüsseln kann. Dafür soll es möglich sein, verschiedene Algorithmen zur Verschlüsselung zu verwenden. Alle Klassen, die einen solchen Algorithmus implementieren, besitzen die Methoden:
`boolean setKey(String key)`: Setzt den Schlüssel und gibt true zurück, wenn ein gültiger Schlüssel eingegeben wurde.
`String encode(String text)`: Kodiert den eingegebenen Text.
`String decode(String text)`: Dekodiert den eingegebenen Text.
Hinweis: Sie können davon ausgehen, dass die verwendeten Schlüssel und Texte nur aus Großbuchstaben `A-Z` ohne Sonderzeichen bestehen.

1. Schreiben Sie ein Interface `ChryptographicAlgorithm`, welches die Methoden eines chryptographischen Algorithmus spezifiziert.

2. Schreiben Sie eine Klasse `SubstitutionChiffre`, die das Interface implementiert. Der Schlüssel besteht dabei aus einer Folge aus 26 Zeichen, bei der jeder Buchstabe des Alphabets genau einmal vorkommt. Jeder Buchstabe des Eingabetextes wird zur Verschlüsselung mit dem entsprechenden Buchstaben des Schlüssels ersetzt. Mit folgendem Beispielschlüssel:

Alphabet:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	$\cdots$
Schlüssel:	X	Z	U	F	A	E	L	I	G	Y	R	O	M	S	$\cdots$

   wird der Text "GEHEIMEEINGABE" zu "LAIAGMAAGSLXZA" verschlüsselt.

3. Schreiben Sie eine Klasse `VigenereChiffre`, die das Interface implementiert. Der Schlüssel besteht dabei aus einer beliebigen Buchstabenfolge, die wiederholt über den Eingabetext geschrieben wird. Jeder Buchstabe des Eingabetextes wird dann im Alphabet um so viele Stellen verschoben, wie der korrespondierende Eintrag im Schlüssel angibt. Dabei ist $A = 0, B = 1, \ldots$ wobei die Addition $\mod 26$ ausgeführt wird. Mit dem Schlüssel "ZEBRA" ergibt sich somit z.B.

Schlüssel:	Z	E	B	R	A	Z	E	B	R	A	Z	E	B	R	$\cdots$
Eingabetext:	G	E	H	E	I	M	E	E	I	N	G	A	B	E	$\cdots$
Verschlüsselt:	F	I	I	V	I	L	I	F	Z	N	F	E	V	V	$\cdots$

   - $G\hat{=}6, Z\hat{=}25 \Rightarrow (6 + 25) \mod 26 = 5\hat{=}F$
   - $E\hat{=}4, E\hat{=}4 \Rightarrow (4 + 4) \mod 26 = 8\hat{=}I$
   - $H\hat{=}7, B\hat{=}1 \Rightarrow (7 + 1) \mod 26 = 8\hat{=}I$
   - $\cdots$

4. Schreiben Sie JUnit-Tests für die in 2. und 3. implementierten Klassen.