Übungsblatt 5 - Teil 2

Ausgabe: 11.6.2018
Abgabe: 20.6.2018 (23:59)
Testat: 21.6.2018 (13:40 –16:50 Uhr)

Aufgabe 1: IO-Filter — 40 Punkte

Die Caesar-Chiffre ist ein einfaches symmetrisches Verschlüsselungsverfahren, das auf der monographischen und monoalphabetischen Substitution basiert. Sie kennen das Verfahren von Übungsblatt 2.
Bei der Verschlüsselung wird jeder Buchstabe des Klartexts auf einen Geheimtextbuchstaben abgebildet. Diese Abbildung ergibt sich, indem man die Zeichen eines geordneten Alphabets um eine bestimmte Anzahl zyklisch nach rechts verschiebt (rotiert). Die Anzahl der verschobenen Zeichen bildet den Schlüssel, der für die gesamte Verschlüsselung unverändert bleibt.

Implementieren Sie die Klassen `CaesarWriter` und `CaesarReader`, welche folgende Anforderungen erfüllen sollen:

`CaesarWriter`

- Diese Klasse soll den zu schreibenden Text mittels Caesar-Chiffre verschlüsseln.
- Sie soll einen Konstruktor besitzen, dem die Anzahl der Verschiebungen übergeben werden kann. Dieser Konstruktor soll selbstverständlich ebenfalls die Verschachtelung verschiedener Writer-Implementierungen unterstützen.
- `CaesarWriter` soll von der Klasse `FilterWriter` aus dem Paket `java.io` erben. Überlegen Sie sich, welche Methoden Sie überschreiben müssen, sodass eine korrekte Verschlüsselung zu jeder Zeit garantiert ist.

`CaesarReader`

- Diese Klasse soll den zu lesenden Text mittels Caesar-Chiffre entschlüsseln.
- Sie soll einen Konstruktor besitzen, dem die Anzahl der Verschiebungen übergeben werden kann. Dieser Konstruktor soll selbstverständlich ebenfalls die Verschachtelung verschiedener Reader-Implementierungen unterstützen.
- `CaesarReader` soll von der Klasse `FilterReader` aus dem Paket `java.io` erben. Überlegen Sie sich, welche Methoden Sie überschreiben müssen, sodass eine korrekte Entschlüsselung zu jeder Zeit garantiert ist.

Im Unterschied zu Übungsblatt 2 werden alle Buchstaben (A-Z, a-z, Ä, Ö, Ü ,ä, ö, ü) verschlüsselt. Sonderzeichen und Leerzeichen sollen unverändert bleiben. Ihr Alphabet könnte also wie folgt aussehen:

Position:	0	1	...	25	26	27	...	51	52	53	54	55	56	57
Buchstabe:	A	B	...	Z	a	b	...	z	Ä	Ö	Ü	ä	ö	ü

Beachten Sie: wird bei einer Verschiebung das Ende des Alphabets erreicht, wird wieder von vorne begonnen. Bei einer Verschiebung um 5 würde also beispielsweise Ü auf B abgebildet werden.

Aufgabe 2: File-/Directory Handling - Zusatzaufgabe - — 40 Punkte

In dieser Aufgabe sollen die in Aufgabe 1 implementierten Caesar-Reader/Writer dazu verwendet werden, *Textdateien* (.txt) auf der Festplatte zu ver-/entschlüsseln. Ein Benutzer soll den Pfad zu einem beliebigen Ordner angeben können, woraufhin alle darin enthaltenen Textdateien verschlüsselt bzw. entschlüsselt werden. Die gesamte Struktur des Ordners soll dabei erhalten bleiben.

Erstellen Sie dazu zuerst das Interface `IFileEncryptor` mit folgenden Methoden:

```
public File encrypt(File sourceDirectory)
public File decrypt(File sourceDirectory)
```

Anmerkungen: `File` kommt aus dem Paket `java.io` und kann eine Datei oder ein Ordner sein. Der Rückgabewert ist der verschlüsselte/entschlüsselte Ordner.

Anschließend implementieren Sie das Interface in der Klasse `CaesarFileEncryptor`, welche in der encrypt-Methode den CaesarWriter verwendet, um den Inhalt aller Dateien des übergebenen Ordners zu verschlüsseln und in der decrypt-Methode den CaesarReader, um den Inhalt aller Dateien des übergebenen Ordners zu entschlüsseln.

- Der verschlüsselte bzw. entschlüsselte Ordner soll im selben Verzeichnis wie der Quellordner gespeichert werden und am Ende der Methode als Rückgabewert zurückgegeben werden.
- Als Name des verschlüsselten bzw. entschlüsselten Ordners wählen Sie den Namen des Quellordners und hängen _encrypted bzw. _decrypted an.
- Beachten Sie, dass die Zielordner keine bereits existierenden Ordner überschreiben dürfen. Falls der Ordner bereits existieren sollte, stellen Sie zusätzlich eine Zahl dahinter (z.B. SecretFolder_encrypted(1), bzw. 2, 3, ... bis ein freier Ordnername gefunden ist).

Schreiben Sie zusätzlich einen geeigneten Programmrahmen, in dem ein Benutzer den Pfad zu einem Ordner eingeben kann, der daraufhin nach Wunsch ver- oder entschlüsselt wird.

Beispiel: Nach dem Verschlüsseln könnte eine Verzeichnisstruktur folgendermaßen aussehen:

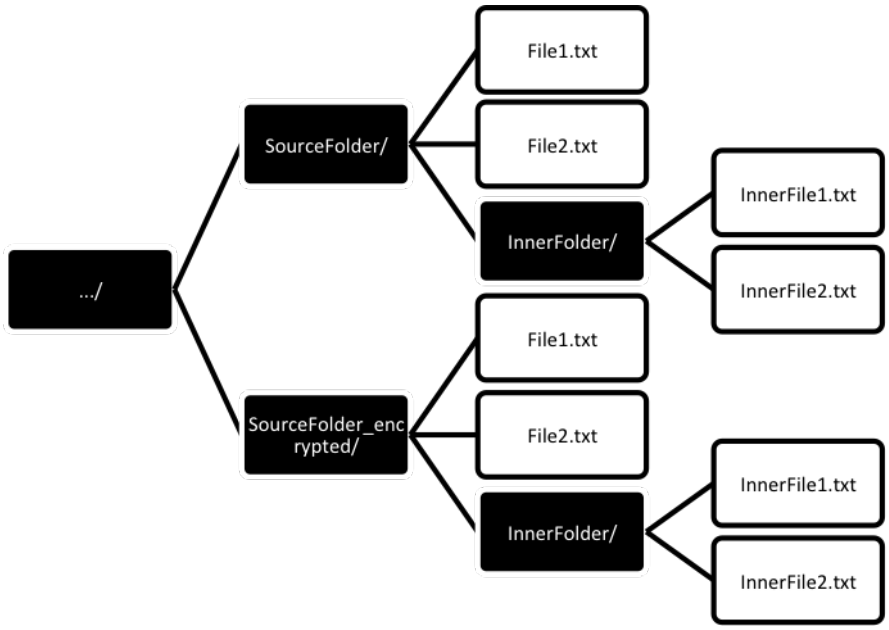

Abbildung 1: Mögliche Ausprägung eines Verzeichnisbaums