Ü 5b.1: Verschlüsselung

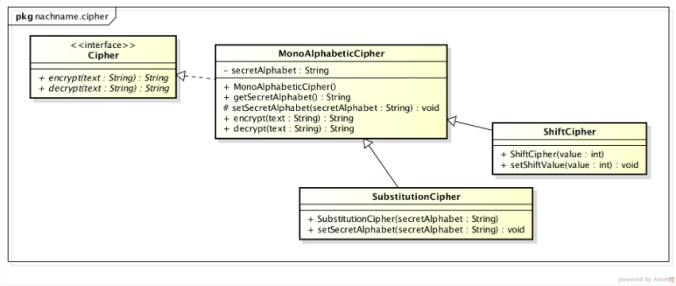
✔ **Done:** Make a submission ✔ **Done:** Receive a grade

Due: Tuesday, 27 December 2022, 11:27 PM

Die Angabe bitte bis zum Ende lesen bevor begonnen wird.

Es soll zunächst folgendes UML-Klassendiagramm umgesetzt werden. Zusätzlich soll eine einfache GUI das Verschlüsseln von Texten mit den hier vorgestellten Verfahren ermöglichen. Erstelle dazu ein Github-Projekt mit folgendem Link: https://classroom.github.com/a/o5sXmPYA

Achtung: Überprüfe vorher, ob du dein erstes Github-Projekt in Eclipse auch umbenannt hast (z.B. statt EmptyJavaProject WorttrainerGit)! Es kann nicht zwei Projekte mit dem selben Namen geben.



Hinweise zu den Klassen und Methoden:

Recherchiert zunächst im Internet die gängigen Verschlüsselungsmethoden nach dem Verfahren der *monoalphabetischen Substition* (auch Wikipedia ist erlaubt). Ihr solltet vor allem das Grundprinzip und das Prinzip der Verschiebechiffre verstehen.

Basisklassen:

Die Klasse `MonoAlphabeticCipher` stellt die Basisfunktionalitäten auf Basis des Interfaces `Cipher` zur Verfügung. Die Methoden `encrypt` und `decrypt` sollen die eigentliche Verschlüsselungs- bzw. Entschlüsselungsarbeit vornehmen. Dazu übernehmen sie als Parameter einen Text, wandeln diesen in Kleinbuchstaben um und geben den ver- bzw. entschlüsselten Text zurück. Dabei sollen sie das Geheimalphabet als Substitutionsschlüssel verwenden (Zeichen die nicht im Alphabet sind, sollen 1:1 im Geheimtext umgesetzt werden). Die Klasse `MonoAlphabeticCipher` setzt das Geheimalphabet immer auf das Ausgangsalphabet (abcdefghijklmnopqrstuvwxyzäöüß) wodurch bei der Superklasse de facto keine Verschlüsselung stattfindet. Trotzdem sollen die beiden Methoden `encrypt` und `decrypt` mit jedem Geheimalphabet arbeiten können, da ja die Subklassen diese Methoden nicht implementieren, das Geheimalphabet je nach Klasse aber anders gesetzt wird. Die Methode `getSecretAlphabet` soll einfach das Geheimalphabet zurück geben. Die Methode `setSecretAlphabet` (Achtung! diese ist nicht public) soll für Subklassen die Möglichkeit bereitstellen, das Attribut zu ändern.

Die Klasse `SubstitutionCipher` soll eine allgemeine Substitionsverschlüsselung darstellen. Dazu muss das gesamte Geheimalphabet (alle 30 Zeichen) an ein Objekt dieser Klasse im Konstruktor oder in der Methode `setSecretAlphabet` übergeben werden. Achte dabei darauf, dass ein gültiges Geheimalphabet übergeben wird (alle Zeichen sind vorhanden, aber auch nur 1x).

Die Klasse `ShiftCipher` soll einen Verschiebechiffre implementieren. Dazu muss beim Konstruktor und bei der Methode `setShiftValue` der Wert der Verschiebung angegeben werden. Das Geheimalphabet soll dann aus diesem Verschiebewert generiert werden.

Achtung! Alle Methoden sollen mit fehlerhaften Parametern umgehen können. Entweder in dem Sie eine passende Exception (eigene verwenden und ebenfalls von sinnvollen Exceptions erben lassen) auslösen oder den falschen Parameter in einen gültigen "umwandeln" können. Eine entsprechende Dokumentation ist Pflicht!

GUI:

Die GUI soll das **Prinzip der Polymorphie** möglichst ohne Umcasten umsetzen, so dass es möglich ist, eine der implementierten Verschlüsselungsvarianten auszuwählen, die notwendigen Konfigurationen für diese Verschlüsselung vorzunehmen und dann so lange Texte zu ver- und entschlüsseln, bis eine neue Verschlüsselung ausgewählt ist. Also:

1. Verschlüsselungsart auswählen und die notwendige Konfiguration (Verschiebewert, Geheimalphabet,...) setzen.
2. Beliebig oft ver- oder entschlüsseln.

Achtet darauf, dass es auch möglichst einfach sein soll, weitere Verschlüsselungen durch Hinzufügen neuer Klassen dazu zu nehmen.

Abgabe:

Eine Aufgabe gilt erst und nur dann als erfolgreich abgegeben, wenn folgende Bedingungen erfüllt sind:

- Auf dem Github-Projekt (https://classroom.github.com/a/o5sXmPYA) sind zumindest 10 commits zu dieser Aufgabe in regelmäßigen Abständen.
- die Klassen stecken alle im Package `loginname.cipher`
- die Applikation ist mitsamt Sourcecode in einem ausführbaren jar-Archiv verpackt.
- das jar-Archiv enthält einen Unterordner doc, der die JavaDocs zu allen Klassen enthält.

Submission status

Submission status	Submitted for grading
Grading status	Graded
Time remaining	Assignment was submitted 22 days 9 hours late
Last modified	Thursday, 19 January 2023, 9:20 AM
File submissions	Verschluss.jar 19 January 2023, 9:20 AM
Submission comments	Comments (0)

Grading criteria

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *3 points*	Funktion vollständig *6 points*
Abstrakte Klasse und Polymorphie	Polymorphie nicht angewendet. *0 points*	Polymorphie nicht sinnvoll angewendet. *2 points*	Polymorphie sinnvoll angewendet. *4 points*
Commits/JAR	Zu wenig Commits und/oder keine sinnvollen Messages. JAR nicht vorhanden. *0 points*	Commit-Anzahl gerade ausreichend und/oder kaum sinnvolle Messages und/oder JAR nicht vorhanden. *1 points*	Ausreichende Anzahl an Commits mit sinnvollen Messages. JAR vorhanden und ausführbar. *2 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*

Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben. *2 points*

Feedback

Grade	14.00 / 18.00
Graded on	Thursday, 19 January 2023, 9:33 AM
Graded by	HD Haselberger David
Feedback comments	Polymorphie könnte noch verbessert werden

Grade breakdown

Programmierstil	Programmierstil wenig berücksichtigt *0 points*	Einrückungen und Kommentare vorhanden aber verbesserungswürdig *1 points*	Einrückungen und Kommentare gut gemacht. *2 points*
Funktion	Funktion mangelhaft *0 points*	Funktion mit leichten Fehlern ok *3 points*	Funktion vollständig *6 points*
Abstrakte Klasse und Polymorphie	Polymorphie nicht angewendet. *0 points*	Polymorphie nicht sinnvoll angewendet. *2 points*	Polymorphie sinnvoll angewendet. *4 points*
Commits/JAR	Zu wenig Commits und/oder keine sinnvollen Messages. JAR nicht vorhanden. *0 points*	Commit-Anzahl gerade ausreichend und/oder kaum sinnvolle Messages und/oder JAR nicht vorhanden. *1 points*	Ausreichende Anzahl an Commits mit sinnvollen Messages. JAR vorhanden und ausführbar. *2 points*
Abgabegespräch	Abgabegespräch mangelhaft *0 points*	Fragen und Änderungen im Programm konnten mit leichten Unsicherheiten beantwortet werden. *1 points*	Fragen und Änderungen im Programm konnten gut beantwortet werden. *2 points*
Zeit	Nachfrist *0 points*	Knapp verspätet abgegeben. *1 points*	Im zeitlichen Rahmen abgegeben. *2 points*