1 Beispiel

Die Firma Good GmbH will von Ihnen Eine RSA Engine programmiert haben. Lesen Sie sich in das Thema ein.

2 Beispiel

Die Firma Good GmbH stellt folgende Anforderungen an die RSA Engine Klasse:

1. Als Good GmbH möchte ich ...
    1. ... den Konstruktor `RSAEngine(int p, int q)` verwenden können, welcher die benötigten Werte für RSA als `BigInteger` berechnet.
    2. ... die Methoden `getN()`, `getE()` und `getD()` verwenden können, um den öffentlichen `(e, N)` und privaten Schlüssel `(d, N)` abfragen zu können.
    3. ... die Methoden `BigInteger encryptNumber(int plain)` und `int decryptNumber(BigInteger encrypted)` verwenden können, um eine ganze Zahl ver- und entschlüsseln zu können.
    4. ... die Methoden `BigInteger encryptChar(char plain)` und `char decryptChar(BigInteger encrypted)` verwenden können, um ein Zeichen ver- und entschlüsseln zu können.
    5. ... die Methoden `BigInteger[] encryptString(String plain)` und `String decryptString(BigInteger[] encrypted)` verwenden können, um einen Text ver- und entschlüsseln zu können.
    6. ... eine Testabdeckung der Methoden durch Unit-Tests von 70% haben, damit die Korrektheit garantiert ist. Was ist eine Testabdeckung? Wie erreicht man diese? Erkläre das bei der Korrektur der Hausübung deinem Übungsleiter!

3 Beispiel

Sie haben den Auftrag der Firma Good GmbH angenommen und erfolgreich abgeschlossen. Nun Nehmen Sie einen Auftrag der Firma EVIL GmbH an welche in direkter Konkurrenz zur Firma Good GmbH steht.

Als Firma EVIL GmbH möchte ich ...

1. ... den Konstruktor `RSAAttack(BigInteger n)` verwenden können.
2. ... die Methode `determinePrimesSerial()` verwenden können, um aus n vom Konstruktor die Primzahlen p und q berechnen zu können. Dazu sollte der Primzahlgenerator welcher bereits in einer vorherigen Übung implementiert wurde, verwendet werden. Verwendet als Rückgabetyp eine selbst zu schreibende generische Klasse Pair, welche p und q in den eigenen Objektvariablen first und second speichert.
3. ..., die Methode `determinePrimesParallel()` verwenden können, welche die serielle Methode aus 2. beschleunigt/parallelisiert, damit die abgefangene Nachricht so schnell wie möglich

entschlüsselt werden kann. Verwendet dafür einen `ExecutorService`, `Callables` und `Futures`!

4. ... die Laufzeit der beiden Methoden für $n = 997241904391$ messen, damit man den Geschwindigkeitsunterschied gut erkennen kann. Schreibt die beiden Messungen als Kommentar in den Code (im Stub gibt es eine vordefinierte Stelle mit einem eigenen HTML-Tag).
5. ... die Methode BigInteger `determineD()` verwenden können, um den unbekannten Teil des privaten Schlüssels (d) direkt bestimmen zu können.

Programmieren Sie die Lösung in den Student Stub welcher im Moodle verfügbar ist! Es müssen die im Stundent Stub mitgelieferten Unit-Tests laufen!