KRY Serie 2

(1) Laden Sie vom Moodle-Kurs KRY das ZIP-File
`/Software/JAVA-Dateien zu KryptoTrainer/KRY_Eclipse.zip`
herunter und kopieren Sie je nach Entwicklungsumgebung alle Dateien und Unterverzeichnisse daraus in ein lokales Projektverzeichnis auf Ihrem Rechner.
Das Verzeichnis enthält das Projekt `KryptoTrainer`. Bringen Sie das Programm `KryptoTrainer` auf Ihrem Rechner und Ihrer JAVA-Entwicklungsumgebung (z.B. Eclipse) zum Laufen.
Hinweis: Das Programm `KryptoTrainer` benötigt die Packages `mybiginteger` (im Unterverzeichnis `MyBigInteger`), sowie `ptolemy.plot` (im Unterverzeichnis `ptplot5.7`). Passen Sie unter `Projects -> Java Build Path -> Libraries` die Pfadangaben Ihrer Installation entsprechend an.

(2) In der Package `mybiginteger` ist eine Klasse `BigInteger` enthalten. Studieren Sie das Interface dieser Klasse. Ressourcen dazu finden Sie im Source-Code oder im Internet unter
http://docs.oracle.com/javase/7/docs/api/index.html

Hinweis: Die Package `mybiginteger` ist eine (abänderbare/erweiterbare) lokale Kopie der aus der Standard-Bibliothek von JAVA 1.4.2 entnommenen Package `java.math`.

(3) Die Klasse `mybiginteger.BigInteger` enthält das Gerüst einer Methode `public BigInteger myModPow(BigInteger exponent, BigInteger m)`.
Ersetzen Sie dieses Gerüst durch einen eigenen naiven Algorithmus zur modularen Exponentiation. Die Terminologie „naiv“ soll hier bedeuten, dass zur Bestimmung des Ausdrucks $a^b\,(mod\,m)$ die Rechnung

$$\underbrace{(\ldots((a\cdot a)\cdot a)\ldots\cdot a)}_{b\text{ mal}}\,(mod\,m)$$

ausgeführt wird. Testen Sie Ihren Algorithmus mit Hilfe des Programms KryptoTrainer „Prakt. 2“ in Bezug auf Resultate und zeitliche Effizienz. Was stellen Sie fest?

Bei sehr grossen Exponenten, kann die myModPow Methode nicht in einer vernünftigen Zeit terminieren. Bei mir geht es so lange, dass ich das Programm selbst beenden muss. Bei sehr grossen Exponten ist die Berechnung sehr zeitintensiv und nicht effizient!

(4) Speichern Sie eine Kopie Ihrer Methode `myModPow()` aus Aufgabe (3) als Kommentar.
Verbessern Sie sodann `myModPow()`, indem Sie den Trick des fortgesetzten Quadrierens anwenden: Es soll wiederum der Ausdruck $a^b\,(mod\,m)$ berechnet werden. Im Binärsystem lässt sich der Exponent als Summe von aufsteigenden Zweierpotenzen darstellen:
$b = 2^{k_1} + 2^{k_2} + 2^{k_3} + \cdots + 2^{k_r}$, mit $k_1 < k_2 < k_3 < \cdots < k_r$.
Damit lässt sich der Ausdruck $a^b\,(mod\,m)$ mit Hilfe der Potenzgesetze berechnen nach der Formel

$$\left[\underbrace{(\ldots((a^2)^2)^2\ldots)^2}_{k_1\text{ mal}}\right]\cdot\left[\underbrace{(\ldots((a^2)^2)^2\ldots)^2}_{k_2\text{ mal}}\right]\cdot\ldots\cdot\left[\underbrace{(\ldots((a^2)^2)^2\ldots)^2}_{k_r\text{ mal}}\right](mod\,m).$$

Hinweis: Die obige Formel birgt in zwei Richtungen noch grosse Optimierungsmöglichkeiten:
- Die Reduktion $mod\,m$ kann und soll nach jedem Quadrieren ausgeführt werden, statt erst ganz am Schluss.
- Hat man den Block für k_i berechnet, so muss man für den Block für k_{i+1} nicht wieder bei a beginnen mit dem Quadrieren.

Passen Sie Ihren Algorithmus entsprechend an und testen Sie ihn wiederum mit Hilfe des Programms KryptoTrainer in Bezug auf Resultate und zeitliche Effizienz.

Es ist sehr effizient und kann mit sehr grosse Exponenten rechnen. Aber Methode "modPow" ist immer noch circa doppelt so schnell wie die implementierte "myModPow" Methode.