12. Übungsblatt: Programmierpraktikum I (WS 2003/04)

Abgabe: 2. Februar 2004

Java: Ausnahmebehandlung I

(nur schriftliche Abgabe, 3 Punkte)

Welcher Ausnahmetyp kommt bei folgenden Problemen zum Einsatz:

a) Division durch Null.

b) Öffnen einer Datei, die nicht existiert.

c) Zugriff auf ein Array mit einem Index, der größer ist als die Arraylänge.

d) Multiplikation zweier Integer, so dass das Ergebnis zu groß ist für einen Integer.

e) Zugriff auf ein Objekt, das nicht existiert.

f) Übergabe eines unpassenden Objektes an eine Methode.

Verwende stets die speziellste Ausnahmeklasse. Beschreibe kurz ihren Anwendungsbereich und ordne sie in die Klassenhierarchie unterhalb der Klasse `Exception` ein.

Java: Ausnahmebehandlung II

(`Aufgabe2.java`, 3 Punkte)

Schreibe ein vollständiges Java-Programm, bei dem eine Methode eine Ausnahme auswirft und die `main`-Methode diese Ausnahme behandelt. Dabei sollen mehrere `catch`-Klauseln zum Einsatz kommen, und das Programmverhalten soll sich ändern oder die Compilation soll scheitern, wenn die Reihenfolge dieser `catch`-Klauseln verändert wird.

Java: Pig Latin

(`Aufgabe3.java`, 5 Punkte)

Schreibe ein Programm, das einen deutschen Satz als Eingabe von der Konsole erhält und diesen in *Pig Latin* übersetzt. Dazu werden zunächst alle Ziffern, Satz- und Sonderzeichen entfernt, nur Buchstaben und Leerzeichen bleiben erhalten. Alle Großbuchstaben werden in Kleinbuchstaben umgewandelt, Umlaute

(ä,ö,ü,ß) in ihre Zerlegungen (ae, oe, ue, ss). Dann wird jedes Wort umgeformt, indem jeweils der erste Buchstabe entfernt und an den Schluss gehängt wird. Zu guter Letzt wird noch ein zufälliger Vokal angehängt. **Bsp.:**

```
Heute ist absolutes Mistwetter.
  --->  eutehe stie bsolutesaa istwettermu
Aber was soll man machen?
  --->  berau aswa ollso anma achenmi
Naja, man kann immer noch Java ueben!
  --->  ajana anmi annke mmerie ochnu avaji ebenuo
```

Zur Lösung des Problems sollen `StringBuffer` verwendet werden!

Java: Kryptographischer Schlüsselaustausch

(`Aufgabe4.java`, 6 Punkte)

Ein kryptographischer Schlüssel ist ein geheim gehaltener Wert, den nur die berechtigten Kommunikationspartner (nennen wir sie einmal *Alice* und *Bob*) kennen. Mit diesem Wert können dann Nachrichten ver- bzw. entschlüsselt werden. Ein Problem bestand lange Zeit darin, dass sich die Kommunikationspartner ja treffen mussten, um den Schlüssel auszutauschen - ein Austausch per Brief, Mail oder Telefon wäre für wirklich sicherheitsrelevante Anwendungen zu riskant gewesen.

Eine erste Lösung für das Problem des Schlüsselaustausches wurde erst 1977 von Diffie und Hellman gefunden und funktioniert wie folgt:

- Alice und Bob vereinbaren in aller Öffentlichkeit eine große Primzahl $p > 0$ (1024 Bit lang) und einen beliebigen Zufallswert g mit $0 < g < p$.
- Alice wählt einen geheimen Zufallswert a mit $0 < a < p$ und berechnet $x = g^a \bmod p$. Sie schickt x an Bob.
- Bob wählt einen geheimen Zufallswert b mit $0 < b < p$ und berechnet $y = g^b \bmod p$. Er schickt y an Alice.
- Alice und Bob können nun beide den gleichen Schlüssel $k = g^{a \cdot b} \bmod p$ berechnen, weil für Alice

  $$y^a \bmod p = (g^b)^a \bmod p = g^{b \cdot a} \bmod p = k$$

  gilt und für Bob

  $$x^b \bmod p = (g^a)^b \bmod p = g^{a \cdot b} \bmod p = k$$

  Dagegen kann ein Dritter, der die Kommunikation beobachtet hat, den Wert k nicht berechnen!

Implementiere dieses Protokoll in Java unter Verwendung der Klasse `BigInteger`. Gib alle Zwischenergebnisse für Alice und Bob in Hexadezimalform auf der Konsole aus.

Bem.: Das Rechnen mit derart großen Zahlen kann recht lang dauern. Beachte daher:

- Das Erzeugen großer, zufälliger Primzahlen isr rechenaufwändig. Es muss sich also nicht um einen Programmierfehler handeln, wenn sich dein Programm für eine halbe Minute scheinbar "verabschiedet".
- Verwende zum Potenzieren unbedingt die Methode `modPow`, sonst bekommst du riesige Zwischenergebnisse und Rechenzeiten!