Institut für Angewandte und
Numerische Mathematik

Dr. Mathias J. Krause
Dr. Stefan Findeisen
M. Sc. Albert Mink
M. Sc. Zoltán Veszelka

2.2.2017

Einstieg in die Informatik und Algorithmische Mathematik (WS 2016/2017)

Klausur

Bearbeitungszeit: **90 min**

Gesamtpunktzahl: **90 Punkte**

Tragen Sie Ihren Vor- und Nachnamen, Ihren Studiengang und Ihre Matrikelnummer ein.

Vorname: _____

Nachname: _____

Studiengang: ☐ Bachelor Mathematik (alle) ☐ Sonst: _____

☐ Lehramt

Matrikelnummer: _____

Hinweise:

- Grundlage für diese Klausur ist die in der Vorlesung gelehrte Sprache `Java`.
- Es sind keine Hilfsmittel zugelassen.
- Jedes Lösungsblatt ist mit Namen und Matrikelnummer zu versehen.
- Schreiben Sie leserlich und mit einem dokumentenechten Stift.
  (Lösungen in Bleistift, löschbarer Tinte usw. werden bei der Bewertung nicht berücksichtigt)
- Durchgestrichene Abschnitte werden bei der Bewertung ebenfalls nicht berücksichtigt.

Bewertung:

1	2	3	4	5	Σ	Note

Name, Matrikelnummer:

Aufgabe 1 *Ausdrücke* (10 Punkte)

Geben Sie das Ergebnis und den Datentyp der nachfolgenden Ausdrücke an:

(a) `0b1100-0b101`

Ergebnis: .....**Datentyp:** .....

(b) `0x25`

Ergebnis: .....**Datentyp:** .....

(c) `12/5`

Ergebnis: .....**Datentyp:** .....

(d) `(char)('C'+2)`

Ergebnis: .....**Datentyp:** .....

(e) `(false || true) || (!false && true)`

Ergebnis: .....**Datentyp:** .....

(f) `4/2.0`

Ergebnis: .....**Datentyp:** .....

(g) `3.5f`

Ergebnis: .....**Datentyp:** .....

(h) `"Ergebnis ="+"?"`

Ergebnis: .....**Datentyp:** .....

(i) `('a'!='b') && ('C'=='c')`

Ergebnis: .....**Datentyp:** .....

(j) `Math.exp(0)`

Ergebnis: .....**Datentyp:** .....

Name, Matrikelnummer:

Aufgabe 2 *Zahlensysteme* (14 Punkte)

Geben Sie die entsprechende Lösung **mit Rechnung** an. Gesucht ist:

(a) Die Dezimaldarstellung der Oktalzahl 132_8.

(b) Die Oktaldarstellung der Dezimalzahl 101.

(c) Die Binärdarstellung der Oktalzahl 123_8.

(d) Die Hexadezimaldarstellung der Binärzahl 11111111_2.

(e) Die Dezimaldarstellung der Hexadezimalzahl $\mathrm{A0C}_{16}$.

(f) Die Hexadezimaldarstellung der Oktalzahl 313_8.

(g) Die Dezimaldarstellung des Binärbruchs 1100.011_2.

Aufgabe 3 *Klassenhierarchie* (6 Punkte)

Betrachten Sie die nachfolgenden Klassendefinitionen:

```
public class Sport {}

public class Kampfsport extends Sport {}

public class Ballsport extends Sport {}

public class Motorsport extends Sport {}

public class Automobilsport extends Motorsport {}

public class Motorradsport extends Motorsport {}
```

(a) Erstellen Sie ein Klassendiagramm, welches die Beziehungen der Klassen untereinander veranschaulicht.

Fortsetzung auf der nächsten Seite

(b) In der `main`-Methode eines Java-Programms stehen die Zeilen

```
Sport s;
Ballsport bs;
Automobilsport as;
```

Welche der folgenden Ausdrücke sind (unabhängig voneinander) gültig? Begründen Sie Ihre Entscheidung kurz.

(b1) `s = new Ballsport();`

(b2) `as = new Automobilsport();`

(b3) `bs = new Sport();`

(b4) `s = (Kampfsport) new Ballsport();`

Aufgabe 4 *Matrixpotenz* (25 Punkte)

In der Vorlesung und der Übung haben Sie den Alogithmus von Pingala kennengelernt. Dieser ermöglicht die effiziente Berechnung einer Potenz $x^e, e \in \mathbb{N}, x \in \mathbb{R}$ oder $x \in \mathbb{C}$. Analog dazu lässt sich auch die Potenz $A^e, e \in \mathbb{N}$ einer Matrix $A \in \mathbb{R}^{n \times n}$ effizient berechnen:

$$A^e = \begin{cases} I, & \text{falls } e = 0, \\ (A^2)^{\frac{e}{2}}, & \text{falls } e > 0 \text{ gerade}, \\ A\,A^{e-1}, & \text{falls } e > 0 \text{ ungerade}. \end{cases} \tag{1}$$

Hierbei bezeichnet $I \in \mathbb{R}^{n \times n}$ die Einheits– bzw. Identitätsmatrix. Die Diagonaleinträge dieser Matrix sind eins, während alle anderen Einträge null sind, d.h.

$$I_{i,j} = \begin{cases} 1, & \text{falls } i = j, \\ 0, & \text{falls } i \neq j. \end{cases}$$

Schreiben Sie ein kompilier– und ausführbares Java–Programm, das die Potenz einer Matrix mithilfe des Pingala–Algorithmus (1) **rekursiv** berechnet. Gehen Sie wie folgt vor:

Erstellen Sie eine Klasse `Matrixpotenz` die folgende Elemente enthält:

- Eine Klassenmethode `matrixEinlesen` ohne formale Parameter.
  Diese soll zunächst die Anzahl der Zeilen und Spalten n einer Matrix $A \in \mathbb{R}^{n \times n}$ von der Konsole einlesen. Anschließend sollen die Komponenten A_{ij} der Matrix zeilenweise eingelesen werden. Geben Sie die Matrix als Variable vom Typ **`double`**`[][]` zurück.

- Eine Klassenmethode `matrixMult` mit zwei formalen Parametern `matrixA` und `matrixB` vom Typ **`double`**`[][]`.
  Hierbei bezeichnen `matrixA` und `matrixB` zwei übergebene Matrizen $A, B \in \mathbb{R}^{n \times n}$. Schreiben Sie einen Methodenrumpf, in dem das Matrixprodukt AB berechnet wird und das Ergebnis in einer Variablen vom Typ **`double`**`[][]` zurück gegeben wird.

  Hinweis: Verwenden Sie dazu drei **`for`**-Schleifen.

- Eine Klassenmethode `matrixPow` mit einem formalen Parameter `matrixA` vom Typ **`double`**`[][]` und einen ganzzahligen formalen Parameter `e`.
  Hierbei bezeichnet `matrixA` eine Matrix $A \in \mathbb{R}^{n \times n}$ und `e` einen Exponenten $e \in \mathbb{N}$. Berechnen Sie innerhalb des Methodenrumpfs die Potenz A^e der übergebenen Matrix A nach dem Pingala–Algorithmus (1). Rufen Sie dazu die Klassenmethode `matrixPow` **rekursiv** auf und verwenden Sie die Klassenmethode `matrixMult`, um das Produkt zweier Matrizen zu berechnen. Geben Sie das Ergebnis in einer Variablen vom Typ **`double`**`[][]` zurück.

- Das Hauptprogramm.
  Lesen Sie zu Beginn eine Matrix $A \in \mathbb{R}^{n \times n}$ und einen Exponenten $e \in \mathbb{N}$ von der Konsole ein. Berechnen Sie anschließend die Matrixpotenz A^e. Verwenden dabei Sie die zuvor erstellten Klassenmethoden.

Aufgabe 5 *Mobilfunk* (35 Punkte)

Schreiben Sie ein kompilier– und ausführbares Java–Programm, das ein einfaches Mobilfunknetz modelliert.

Verwenden Sie dazu die folgenden Klassen:

- Eine Klasse `Teilnehmer`, die die einzelnen Teilnehmer eines Betreibers simuliert. Diese können Textnachrichten an andere Teilnehmer versenden sowie die zuletzt empfangene Textnachricht im Speicher ihres Mobilfunkgerätes ablegen.
- Eine Klasse `Betreiber`, die den Betreiber eines Mobilfunknetzes simuliert. Der Betreiber verwaltet die Teilnehmer des Mobilfunknetzes.
- Eine Klasse `Mobilfunk`, die einen Markt mit einem Mobilfunkbetreiber simuliert.

Gehen Sie wie folgt vor:

Schreiben Sie eine Klasse `Teilnehmer`, welche die folgenden Elemente enthält:

- Eine ganzzahlige Variable `nummer` für die Rufnummer des Teilnehmers und zwei Zeichenketten `name` und `speicher` vom Typ `String` für den Namen des Teilnehmers und den Inhalt der zuletzt empfangenen Textnachricht. Außerdem eine Variable `bt` vom Typ `Betreiber` für den Betreiber des Mobilfunknetzes, welches der Teilnehmer nutzt.
  Hierbei soll die Variable `nummer` als öffentliche Instanz–Variable deklariert werden, während alle anderen Variablen als private Instanz–Variablen deklariert werden.
- Einen öffentlichen Konstruktor mit zwei Parametern für die ganzzahlige Rufnummer des Teilnehmers und den Betreiber des Mobilfunknetzes vom Typ `Betreiber`. Initialisieren Sie die Instanz–Variablen mit den übergebenen Werten bzw. belegen Sie diese mit sinnvollen Werten. Lesen Sie dabei den Namen des Teilnehmers von der Konsole ein.
- Eine öffentliche Instanz–Methode `sendeNachricht` mit zwei formalen Parametern für die ganzzahlige Rufnummer des Empfängers sowie für den Mobilfunkbetreiber des Empfängers vom Typ `Betreiber`. Lesen Sie dazu die Textnachricht von der Konsole ein. Geben Sie dabei eine Meldung auf dem Bildschirm aus, aus der ersichtlich wird, wer Sender und Empfänger der Nachricht ist. Speichern Sie die Textnachricht in der Komponente `speicher` des Empfängers ab.

  Hinweis: Über die Variable `bt` können Sie auf alle gespeicherten Teilnehmer `tn` des gleichen Betreibers zugreifen.
- Eine öffentliche Instanz–Methode `toString` ohne formale Parameter mit einem Rückgabe–Parameter vom Typ `String` zur Ausgabe eines Objektes vom Typ `Teilnehmer`. Geben Sie dabei den Namen des Teilnehmers, seine Rufnummer und die zuletzt empfangene Textnachricht in einem `String` an die aufrufende Methode zurück.

Schreiben Sie eine weitere Klasse `Betreiber`, welche die folgenden Elemente enthält:

- Eine ganzzahlige Variable `anzahlTN` in der die Anzahl der Teilnehmer des Betreibers gespeichert werden. Des Weiteren ein eindimensionales Feld `tn` vom Typ `Teilnehmer[]`, in dem alle Teilnehmer des Betreibers gespeichert werden. Deklarieren Sie diese Variablen als öffentliche Instanz–Variablen.

- Einen öffentlichen Konstruktor mit einem formalen Parameter für die ganzzahlige Anzahl der Teilnehmer des Betreibers, in dem die Instanz–Variablen mit dem übergebenenen Wert sinnvoll initialisiert werden. Iterieren Sie anschließend in einer Schleife über die Anzahl aller Teilnehmer und legen Sie im i-ten Schleifendurchlauf den i-ten Teilnehmer explizit durch Aufruf des entsprechenden Konstruktors an. Speichern Sie dabei die Instanzen der Klasse `Teilnehmer` im Feld `tn` ab.

  Hinweis: Die Teilnehmer eines Betreibers werden, beginnend mit der Zahl Null, fortlaufend durchnummeriert. Die Nummer entspricht dabei gleichzeitig der Rufnummer des Teilnehmers.

- Eine öffentliche Instanz–Methode `bericht` ohne formale Parameter, in der der aktuelle Status des Betreibers und aller Teilnehmer auf der Konsole ausgegeben wird. Dieser besteht aus der Anzahl der Teilnehmer und den Informationen jedes Teilnehmers, die mithilfe der `toString` Methode der Klasse `Teilnehmer` ausgegeben werden können.

Schreiben Sie eine Klasse `Mobilfunk`, welche nur das Hauptprogramm enthält:

- Erstellen Sie zu Beginn einen Betreiber `fminus` mit drei Teilnehmern und führen Sie anschließend die folgenden Aktionen aus:

  (1) Teilnehmer 1 sendet eine Textnachricht an Teilnehmer 2,

  (2) Teilnehmer 2 sendet eine Textnachricht an Teilnehmer 0.

  Geben Sie im Anschluß daran den aktuellen Status des Betreibers mithilfe der Methode `bericht` auf dem Bildschirm aus.