TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Computer Vision Group
Einführung in die Informatik I für Ingenieure (MSE)
Prof. Dr. Daniel Cremers, Mohamed Souiai

WS 2012
Weihnachtsblatt
23.12.2012

Aufgabe 1 Reguläre Ausdrücke

Schreiben Sie reguläre Ausdrücke für die folgenden Mengen:

a) Die Menge der nichtleeren Zeichenketten, die nur aus den Zeichen a und b bestehen und die gerade länge haben.
b) Die Menge der Zeichenreihen aus Nullen und Einsen, die mit dem gleichen Zeichen anfangen und enden.
c) Die Menge aller Zeichenreihen aus a,b,c deren Länge mindestens 1 ist und höchstens 3.
d) Die Menge der Zeichenreihen aus Nullen und Einsen, deren Anzahl von Null ein vielfaches von 3 ist.

Aufgabe 2 Einfache Funktionen

Implementieren Sie die folgenden Funktionen:

a) Schreiben Sie eine Funktion

```
public static double calculateMean(int[] array)
```

die den Mittelwert (arithmetisches Mittel) aller Elemente eines Integer-Arrays berechnet. Die Berechnung erfolgt anhand der Formel

$$\frac{1}{N}\sum_{i=0}^{N-1} a_i.$$

Für ein leeres Array soll 0 zurückgegeben werden.

b) Die Funktion

```
public static double iterativePower(double base, int exponent)
```

soll beliebige Potenzen von `base` zu ganzzahligen und nichtnegativen Exponenten `exponent` *iterativ* berechnen. Für negative Exponenten soll eine Fehlermeldung auf der Konsole angezeigt und der Wert 0 zurückgegeben werden.

c) Implementieren Sie in

```
public static double recursivePower(double base, int exponent)
```

eine Funktion mit der gleichen Spezifikation wie in Teilaufgabe (b) *rekursiv*!

d) Schreiben Sie eine Funktion

```
public static void divisionChecker(int max, int value1, int
    value2)
```

mit der Spezifikation, alle Zahlen von 1 bis (einschließlich) `max` zu durchlaufen und zu überprüfen, ob die entsprechende Zahl durch `value1`, `value2` oder beide ganzzahlig teilbar ist.

Entsprechend des Ergebnisses der Überprüfung sollen (verschiedene) Nachrichten auf der Konsole ausgegeben werden, falls es sich bei mindestens einem der Parameter um einen Teiler handelt. Ihre Ausgabe für `divisionChecker(10,2,3)` soll wie folgt aussehen:

```
2 can be divided by 2!
3 can be divided by 3!
4 can be divided by 2!
6 can be divided by 2 and 3!
8 can be divided by 2!
9 can be divided by 3!
10 can be divided by 2!
```

Hinweis: Wenn die Zahl `a` durch die Zahl `b` teilbar ist, so bleibt beim Teilen kein Rest. Dies kann durch den Modulo-Operator "%" wie folgt getestet werden:

```
if (a % b == 0) {
  System.out.println("a ist durch b teilbar.");
} else {
  System.out.println("a ist nicht durch b teilbar.");
}
```

e) Testen Sie Ihre Funktionen durch beispielhafte Berechnungen, insbesondere auch aller Spezialfälle, in der `main`-Methode.

Aufgabe 3 Char Histogramm

Schreiben Sie ein Mini-Java Programm `CharHistogram.java`, dass für einen vorgegebenen Text die Häufigkeiten der enthaltenen Buchstaben bestimmt. Zum Beispiel ergibt sich für den Satz:

„*The art of flying is learning how to throw yourself at the ground and miss*“

folgendes Buchstaben-Histogramm:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
4	0	0	2	4	3	3	4	4	0	0	3	1	5	6	0	0	5	4	6	2	0	2	0	2	0

Zur Vereinfachung sollten Sie zunächst nur Eingabe-Texte mit Großbuchstaben betrachten und Ihr Programm dann schrittweise erweitern unter Berücksichtigung von Leerzeichen, Groß-/Kleinschreibung, Umlauten, Ziffern und andere Zeichen (nicht gezählte Zeichen können ausgefiltert werden).

Die Eingabe kann zunächst per `String eingabe = "Das ist ein Eingabetext "` fest vorgegeben werden. Später soll aus einer Datei eingelesen werden, deren Name als Argument beim Aufruf des Programms übergeben wird. Dazu liegt die Funktion `String readFile(String file)` in der Klasse `FileReaderMiniJava` auf der Übungshomepage (Blatt 5) bereit.

Aufgabe 4 ToDo Liste

In dieser Aufgabe sollen Sie eine ToDo-List auf Basis einer *verketteten Liste* erstellen, in der die verschiedenen Einträge stets *absteigend nach ihrer Priorität sortiert* gespeichert werden.

Hinweis:

Achten Sie im Folgenden darauf, *alle* Attribute Ihrer Klassen bzw. Objekte vor dem direkten Zugriff von außerhalb der jeweiligen Klasse durch entsprechende Deklarationen zu schützen!

Teilaufgaben:

a) **ToDoItem**

Implementieren Sie zunächst die Klasse `ToDoItem`, die neben dem Namen und einer Beschreibung der Aufgabe eine ganzzahlige Priorität (je größer die Zahl, desto wichtiger) speichert und den folgenden Konstruktor zur Verfügung stellt:

```
public ToDoItem(String name, String description, int priority)
```

Darüberhinaus sollen (ausschließlich) die folgenden Methoden implementiert werden:

- `public String getName()`
- `public String getDescription()`
- `public int getPriority()`
- `public void setPriority(int priority)`
- `public String toString()`

Ihre `toString()`-Methode soll alle relevanten Informationen, d.h. sowohl Name als auch Beschreibung und Priorität, umfassen.

b) **ToDoListItem**

Erstellen Sie nun die Klasse `ToDoListItem` zur Repräsentation eines Eintrags der verketteten Liste. Neben einer Referenz auf ein `ToDoItem`-Objekt soll sie außerdem auf das nächste Element der Liste verweisen. Implementieren Sie dazu folgenden Konstruktor und die genannten Methoden:

- `public ToDoListItem(ToDoItem value, ToDoListItem next)`
- `public ToDoItem getValue()`
- `public ToDoListItem getNext()`
- `public void setNext(ToDoListItem next)`

c) **ToDoList**

Die Liste selbst soll durch die Klasse `ToDoList` repräsentiert werden. Implementieren Sie diese Klasse inklusive der benötigten Membervariablen und der folgenden Methoden:

- `public ToDoList()`
  Erzeugt eine leere Liste.
- `public boolean contains(String name)`
  Prüft, ob eine Aufgabe mit dem eingegeben Namen in der Liste enthalten ist.

- `public boolean add(String name, String description, int priority)`
  Fügt ein neues `ToDoListItem` so in die bestehende Liste ein, dass diese dabei sortiert bleibt. Das Einfügen eines neuen Elements soll nur dann möglich sein, wenn noch keine Aufgabe mit gleichem Namen existiert. Der Rückgabewert gibt entsprechend an, ob das Element hinzugefügt werden konnte oder nicht.
- `public boolean remove(String name)`
  Entfernt das Element, falls es enthalten ist.
- `public boolean setPriorityFor(String name, int priority)`
  Setzt die Priorität für das gesuchte Element neu, sofern dieses existiert. Die Liste muss danach weiterhin korrekt sortiert sein. Der Rückgabewert gibt an, ob das Element in der Liste enthalten ist.
- `public int size()`
  Gibt die Anzahl der Einträge zurück.
- `public ToDoItem[] toArray()`
  Liefert ein Array aller enthaltenen `ToDoItems` in der richtigen Reihenfolge zurück, d.h. je höher die Prioritätszahl, desto weiter vorne muss sich das `ToDoItem` befinden.