UNIVERSITÄT BASEL

University of Basel

Dozent
Prof. Dr. Thomas Vetter
Departement
Mathematik und Informatik
Spiegelgasse 1
CH – 4051 Basel

Assistenten
Marcel Lüthi

Tutoren
Pascal Mafli
Loris Sauter
Linard Schwendener
Clemens Büchner
Florian Spiess
Jonathan Aellen
Lukas Stöckli

Erweiterte Grundlagen der Programmierung (45398-01)

Blatt 3 [11 Punkte]

Vorbesprechung	8.-12. Okt
Abgabe	15. - 19. Okt (vor dem Tutorat)

Wir empfehlen Ihnen, dass Sie im Buch "Sprechen Sie Java" bis und mit Kapitel 7 lesen, bevor Sie beginnen die Übungen zu lösen.

Aufgabe 1 - Binär und Hexadezimaldarstellung [4 Punkte]

In dieser Aufgabe sollen Sie Zahlen von einer Basis in eine andere umschreiben.

(a) Umwandlung einer Dezimalzahl in eine Hexadezimalzahl:
Unsere gewohnten Zahlen sind die Dezimalzahlen. Das sind Zahlen zur Basis zehn mit den Ziffern 0 bis 9. Jede Stelle i hat die Wertigkeit 10^{i-1}. Die Hexadezimalzahlen sind Zahlen zur Basis 16. Sie bestehen aus den Ziffern 0 bis 9 und den Buchstaben A, B, C, D, E und F für die Dezimalwerte 10, 11, 12, 13, 14 und 15. Jede Stelle i hat die Wertigkeit 16^{i-1}. Um Hexadezimalzahlen von Dezimalzahlen zu unterscheiden, notieren wir sie mit dem Präfix '`0x`'.
Beispiel:

$$\texttt{0x01B7} = 0 \cdot 16^3 + 1 \cdot 16^2 + 11 \cdot 16^1 + 7 \cdot 16^0 = 0 + 256 + 176 + 7 = 439$$

Schreiben Sie ein Programm, das eine positive ganzzahlige Dezimalzahl in eine Hexadezimalzahl umwandelt und ausgibt. Verwenden sie keine von Java bereitgestellte Umwandlungsmethode. Lesen Sie die Dezimalzahl von der Kommandozeile ein. Verwenden Sie zur Umwandlung der Eingabe in eine Zahl vom Typ `int` folgende Methode: `Integer.parseInt(args[0])`
Ihr Programm sollte sich also wie folgt verhalten:

```
$ java Dec2Hex 439
0x01B7
```

(b) Umwandlung einer Binärzahl in eine Dezimalzahl:
Schreiben Sie ein Programm, das eine positive ganzzahlige Binärzahl in eine Dezimalzahl umwandelt und ausgibt. Lesen Sie die Binärzahl Stelle für Stelle von der Kommandozeile ein (Anzahl der Stellen = Anzahl der Parameter = `args.length`).
Ihr Programm sollte sich also wie folgt verhalten:

```
$ java Bin2Dec 1 0 0 1
9
```

Aufgabe 2 - Arrays [4 Punkte]

In dieser Aufgabe sollen Sie ein Bild aus dem `ImageWindow` in einem `Array` speichern und wieder anzeigen.

Farben werden in einem Monitor durch Mischung aus den Grundfarben Rot, Grün und Blau erzeugt. Ein Bild wird durch ein zwei-dimensionales Gitter von *Picture Elements*, kurz *Pixel*, repräsentiert. Jeder Pixel beschreibt die Farbe an einem Punkt des Bildes. Wir verwenden hier RGB-Bilder, bei denen die Pixel durch die Grundfarben Rot, Grün und Blau beschrieben werden. Rot, Grün und Blau werden typischerweise verwendet, weil diese Farben einerseits ungefähr den drei Farben entsprechen, auf die die drei unterschiedlichen Rezeptoren im menschlichen Auge am stärksten reagieren, und andererseits historisch bedingt, weil diese Farben mit der Phosphorbeschichtung von Röhrenmonitoren gut erzeugt werden können. Die Intensität der Grundfarben wird als Zahl zwischen 0 und 255 kodiert, so kann z.B. ein dunkles Blau als (0,0,64) dargestellt werden, und ein helles Lila als (255, 128, 255).

Ein Bild kann also mit Breite × Höhe × Anzahl Farben vielen Zahlen kodiert werden. Diese Zahlen können verschieden im Speicher angeordnet sein. Die zwei am häufigsten verwendeten Methoden sind, hintereinander alle roten Intensitäten, dann alle grünen Intensitäten und dann alle blauen Intensitäten zu speichern oder jeweils aufeinanderfolgend Rot, Grün und Blau eines Pixels.

Die Numerierung in den Bildern entspricht den Speicherindizes für beide Methoden, wenn man das Bild an aufeinanderfolgenden Speicherpositionen in einem ein-dimensionalen Array speichert. Überlegen Sie sich, welchen Index der zu der Bildposition (Zeile, Spalte, Farbkanal) gehörende Farbwert unter beiden Speicherungsmethoden in dem Bild-Array hat.

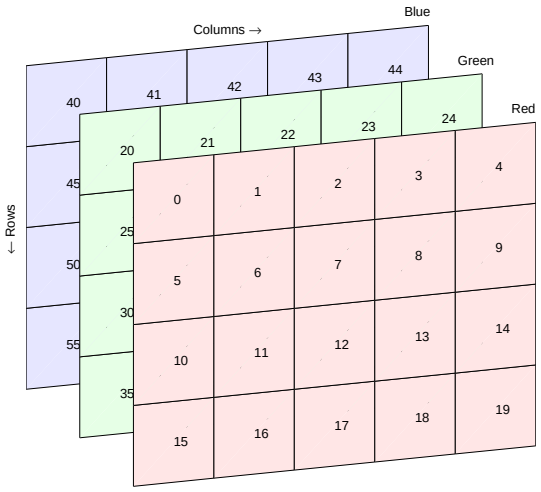
Im Zip Archiv `Blatt03.zip` für Übung 3 finden Sie einen Ordner `ImageArrays`. Dieser enthält die für diese Aufgabe benötigten Klassen. Sie können die Klasse ImageArrays.java als Vorlage für Ihr Programm verwenden.

Solange Sie noch nichts implementiert haben lädt das Program ein Bild aus einer Datei in das `ImageWindow` und zeigt drei Fenster an. Eines zeigt das geladene Bild, die anderen beiden Fenster sollten leer sein. Die Stellen an denen Sie Ihren Code schreiben sollen sind jeweils mit `TODO:` markiert.

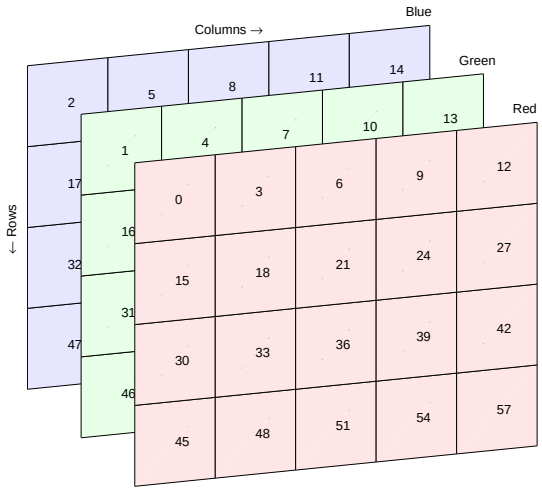
Erstellen sie mit dem Befehl

```
javadoc -sourcepath . -subpackages ch.unibas.informatik.cs101 -d doc
```

eine Dokumentation der Klasse `ImageWindow`. Nun finden sie im Verzeichnis doc eine Webseite `index.html` die die Javadoc-Dokumentation enthält.



Speicheranordnung 1



Speicheranordnung 2

(a) Erstellen Sie ein *ein-dimensionales* Array welches genau so viel Platz bereitstellt wie für die Speicherung der gesamte Farbinformation des Bildes benötigt wird.

(b) Füllen Sie das Array mit der Farbinformation des Bildes.

(c) Schreiben Sie die Farbinformation vom Array in das neue Bild so dass es um 90 Grad gedreht erscheint.

(d) Schreiben Sie eine statische Methode welche das Array in das neue Bild schreibt und dabei:

- Den roten Farbkanal horizontal spiegelt.
- Den grünen Farbkanal vertikal spiegelt.
- Den blauen Farbkanal an einer der beiden Diagonalen spiegelt.

Überlegen Sie sich dafür einen geeigneten Methodenkopf und rufen Sie die statische Methode aus der `main`-Methode heraus auf.

Aufgabe 3 - Bubble Sort I [3 Punkte]

Die Lösung dieser Aufgabe wird auf einem späteren Blatt wieder verwendet. Versuchen Sie deshalb diese Aufgabe wenn möglich zu lösen.
Mit dem Verfahren *Austauschsortieren* (Bubble Sort) kann man die Elemente eines Arrays sortieren. Dazu wandert man von vorne nach hinten durch das Array und betrachtet jeweils benachbarte Elemente. Wenn diese in der falschen Reihenfolge stehen, vertauscht man sie. Ist man am Ende des Arrays angelangt, beginnt man wieder von vorne - solange bis keine Vertauschungen mehr auftreten.

Das folgende Programm nimmt ein Zeichen-Array als Parameter entgegen, sortiert es und gibt es aus. Die Methoden `swap`, `sort` und `displayArray` sind allerdings noch nicht implementiert. Implementieren Sie die fehlenden Methoden und verwenden Sie dabei das oben beschriebene Verfahren. Den unvollständigen Source-Code können Sie von der Vorlesungsseite herunter laden.

Listing 1: File BubbleSort.java

```
1  class BubbleSort {
2
3    /**
4     * Vertauscht zwei Werte in einem Array an den gegebenen Positionen.
5     **/
6    public static void swap(int i, int j, char[] characters) {
7      /* Diese Methode muss implementiert werden */
8    }
9
10   /**
11    * Sortiert das Eingabearray und aendert das Array in place
12    **/
13   public static void sort(char[] characters) {
14     /* Diese Methode muss implementiert werden */
15   }
16
17   /**
18    * Schreibt das Array auf die Ausgabekonsole
19    **/
20   public static void displayArray(char[] characters) {
21     /* Diese Methode muss implementiert werden */
22   }
23
24   /**
25    * Die Hauptfunktion liest das Character Array und ruft die Sortierfunktion
26    * und die Ausgabefunktion auf
27    **/
28   public static void main(String[] args) {
29     if (args.length != 1) {
30       System.out.println("Bitte rufen Sie das Programm mit einem Eingabewert auf");
31       System.out.println("  java BubbleSort 'dies ist ein text'");
32       System.exit(-1);
33     }
34     char[] characters = args[0].toCharArray();
35
36     sort(characters);
37
38     displayArray(characters);
39   }
40 }
```

Aufgabe 4 - Bubble Sort II - Denksport (Fakultativ) [0 Punkte]

Gegeben sei die gleiche Problemstellung wie in der vorherigen Aufgabe. Nehmen Sie jedoch an, dass die Eingabe jeweils nur aus einzelnen Kleinbuchstaben `a-z` besteht. Können Sie eine Methode angeben, die das gleiche Ergebnis mit weniger Rechenschritten erzeugt?