AUFGABE 1: BILDVERARBEITUNG

WERKZEUGE:	ECLIPSE (JAVA)
ABGABETERMIN:	31.05.2018, 23:59 UHR
ABGABEDATEI(EN):	`ImageEditor.java`
ERREICHBARE PUNKTE:	28 PUNKTE (+4 ZUSATZPUNKTE)
NOTWENDIGE PUNKTE:	14 PUNKTE

ALLGEMEINE HINWEISE UND ANFORDERUNGEN

Bereitgestellt wird ein Eclipse-Projekt einer einfachen Swing-basierten Java-Anwendung. Diese Anwendung dient als Grundlage für die Lösung der Teilaufgaben.

Geben Sie am Ende lediglich die Datei „ImageEditor.java“ ab, welche Sie entsprechend durch die in den Teilaufgaben unten beschriebenen Funktionen erweitert haben. **Tragen Sie bitte im oberen Kommentarbereich ihren Namen und Matrikelnummer ein!** Implementieren Sie die geforderten Funktionen **ausschließlich** in dieser Java-Datei und nehmen Sie **keine Änderungen** in den übrigen Java-Dateien vor. Achten Sie darauf, die Funktionen so zu benennen, wie angegeben!

Als Einstiegsbeispiel finden Sie im Quellcode eine vorimplementierte Funktion **`grayscale`**, die ein Bild (RGB) in ein Graustufenbild umwandelt. Alle zu implementierenden Funktionen sind als „Gerüste“ im Projekt vorbereitet und müssen nur mit den entsprechenden Anweisungen ergänzt werden.

Dokumentieren Sie ihren Quellcode, indem Sie in kurzen eigenen Worten/Stichpunkten durch Codekommentare beschreiben, was der Algorithmus leistet, wie er arbeitet und wofür welche Variablen gewählt wurden. Dies hilft bei der Bewertung, Ihre Lösung nachzuvollziehen.

AUFGABE 1.1 4P

PUNKTOPERATIONEN

1. RGB → BGR

Implementieren Sie eine Funktion **`swap`**, die die Farben Rot und Blau vertauscht, d. h. RGB in BGR umwandelt.

1P

2. Gammakorrektur

Erstellen Sie eine Funktion namens **`gamma`**, mit dessen Hilfe eine Gammakorrektur des Bildes mit γ=1/2,2 durchgeführt wird (zum Ausgleich eines typischen Gammawertes von 2,2 eines Monitors).

2P

3. Spiegelung

Implementieren Sie eine Funktion **`mirror`**, welche das Bild horizontal spiegelt.

1P

AUFGABE 1.2 8P

FARBKANÄLE

1. Farbkanäle RGB

Realisieren Sie eine Funktion **`rgb_tiles`**. Diese erstellt aus dem Originalbild ein zusammengesetztes Bild bestehend aus 4 Bildausschnitten, wie im folgenden zu sehen:

Die linke obere Kachel zeigt den Ausschnitt aus dem Originalbild, oben rechts ist der R-Kanal des Ausschnitts, unten links der G-Kanal und unten rechts der B-Kanal zu sehen. Die Ausschnitte der Farbkänale sollen entsprechend farblich erscheinen.

4P

2. Farbkanäle YCbCr

Ähnlich wie in der vorherigen Aufgabe soll nun eine Funktion **`ycbcr_tiles`** implementiert werden, die jedoch nicht die RGB-Zusammensetzung des Bildes zeigt, sondern die YCbCr-Komponenten.

Oben rechts soll die Helligkeit (Y) gezeigt werden, unten links bzw. recht die Cb- bzw. Cr-Komponente (in Graustufen).

Verwenden Sie für die Umrechnung vom RGB- in den YCbCr-Farbraum folgende Vorschrift:

$$
\begin{aligned}
Y' &= &&+ (0.299\cdot &&R'_D) + (0.587\cdot &&G'_D) + (0.114\cdot &&B'_D)\\
C_B &= 128 - &&(0.168736\cdot &&R'_D) - (0.331264\cdot &&G'_D) + (0.5\cdot &&B'_D)\\
C_R &= 128 + &&(0.5\cdot &&R'_D) - (0.418688\cdot &&G'_D) - (0.081312\cdot &&B'_D)
\end{aligned}
$$

4P

AUFGABE 1.3 — 8P

BILDSKALIERUNG

1. Pixelwiederholung (Nächster Nachbar)

Das Originalbild soll auf die Hälfte der Bildbreite bzw. -höhe herunterskaliert werden und das Ergebnis anschließend wieder auf die ursprüngliche Größe vergrößert werden. Implementieren Sie dazu eine Funktion **`resize`**, die zunächst einfach durch Weglassen (bzw. Unterabtastung von Pixeln) eine Verkleinerung erzielt, und anschließend umgekehrt durch Pixelwiederholung (Einfügen von Pixeln) eine Vergrößerung auf die vorherige Größe vornimmt.

(Vergrößerter Bildausschnitt der Ausgabe)

3P

2. Skalierung durch Interpolation

Auch hier soll das Originalbild auf die Hälfte der Bildbreite bzw. -höhe herunterskaliert werden und das Ergebnis anschließend wieder auf die ursprüngliche Größe gebracht werden – hierbei jedoch mit Hilfe einer Interpolation (bilinear). Implementieren Sie analog zu oben eine Funktion **`resample`**, die statt mit Pixelwiederholung mit Interpolation (hier i.W. Mittelwertbildung) beim Verkleinern und anschließenden Vergrößern arbeitet.

(Vergrößerter Bildausschnitt der Ausgabe)

5P

AUFGABE 1.4 — 8P

FARBREDUKTION MITTELS FARBRASTERUNG

In dieser Aufgabe soll die Farbtiefe des Beispielbildes auf 3 bit reduziert werden, sodass lediglich 8 Farben (neben Schwarz und Weiß die Grundfarben Rot, Grün, Blau und die Mischfarben Cyan, Magenta, Gelb) verwendet werden:

1. Ordered Dithering

Implementieren Sie eine Funktion **`orderedDithering`**, die das Bild nach dem Ordered-Dithering-Prinzip auf die vorgegebene Farbpalette reduziert.

Verwenden Sie dazu die im Quellcode bereitgestellte 8x8-Bayer-Matrix.

Folgender Pseudocode als Hilfestellung:

```
for each y
    for each x
        pixel[x][y] += bayer_8x8[x mod 8][y mod 8]
        pixel[x][y] = get_closest_palette_color(pixel[x][y])
```

3P

2. Dithering mit dem Floyd-Steinberg-Algorithmus

Implementieren Sie den Floyd-Steinberg-Algorithmus als Funktion **`floydSteinbergDithering`**.

Der Floyd-Steinberg-Algorithmus arbeitet mit folgender Matrix zur Verteilung der durch die Quantisierung entstehenden Fehler (Abweichung vom Originalwert) auf benachbarte Pixel:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & P & \frac{7}{16} \\ \frac{3}{16} & \frac{5}{16} & \frac{1}{16} \end{bmatrix}$$

5P

ZUSATZAUFGABE

Überlegen bzw. recherchieren Sie ein Vorgehen zur Erstellung einer idealeren Farbpalette als Alternative zu der in 1.4 vorgegebenen. Implementieren Sie eine Funktion, die aus dem Originalbild 8 geeignete Farben berechnet und aus diesen eine Farbpalette bildet. Diese soll mit einem der beiden Dithering-Verfahren verwendet werden, um ein farbreduziertes Bild zu generieren.	4P