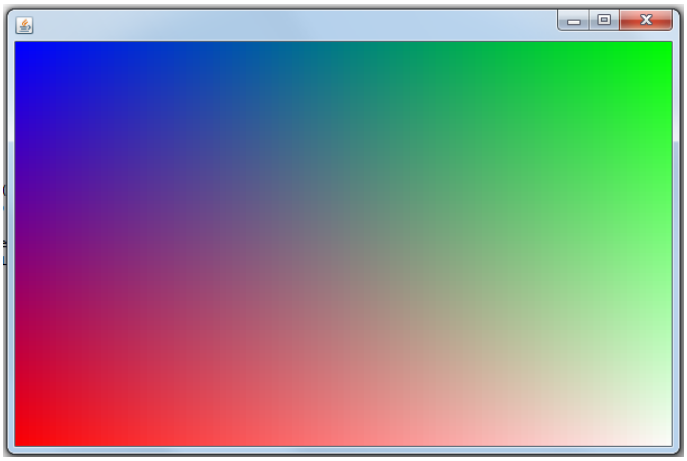
Übung 1

Aufgabe 1:

Schreiben Sie ein Java Programm, dass ein Fenster öffnet, vier unterschiedliche Farben für die vier Ecken des Fensters definiert und einen Farbverlauf über die vier Farben durchführt.



Aufgabe 2:

Erweitern Sie Ihr Programm aus Aufgabe 1, so dass die Mausposition im Fenster eine fünfte Farbe definiert und der Farbverlauf von den Ecken zur Maus verläuft. Der Farbverlauf soll neu berechnet werden, wenn die Maus eine neue Position erhält (Tipp: MausMotionEvent).

Übung 2

Aufgabe 1:

Schreiben Sie ein Java Programm, dass ein Bild einliest und ein Histogramm („welche Farben werden wie oft verwendet“) über die verwendeten Farben erstellt. Das Histogramm soll die komplette Farbe und **nicht** die Farbanteile darstellen, d.h. es soll die Information, dass z.B. die Farbe 0xfff20017 123mal in dem Bild vorkommt. Es soll **nicht** berechnet werden, dass der Blauanteil 0x17 423mal in irgendwelchen Farben in dem Bild vorkommen.

Aufgabe 2:

Verwenden Sie die obige Information, um Ihr eingelesenes Bild nur mit 50% der ursprünglichen Farben darzustellen. Bilden Sie dazu die eingesparten Farben auf ähnliche, noch vorhandene ab. Zeichnen Sie das Bild in einem Fenster. Sparen Sie die 50% von Farben ein, die am wenigsten gebraucht werden. Die Ähnlichkeit von Farben können Sie über die Differenzen zwischen dem Rot-, Grün- und Blauanteil bestimmen. Je kleiner die Summe der absoluten Differenzen ist, desto ähnlicher sind sich die Farben.

Aufgabe 3:

Schreiben Sie ein Programm, das eine **beliebige** Anzahl von Bildern einliest und dann von einem Bild zu nächsten überblendet. Das letzte Bild wird in das erste Bild überblendet.

Übung 3

Aufgabe 1:

Schreiben Sie ein Java Programm, das ein Bild einliest und für unterschiedliche Teile des Bildes unterschiedliche Transformationen durchführt, so dass das Ergebnisbild inhomogen erscheint. Verwenden Sie dazu die Matrizenrechnung. Hierzu ist eine Klasse zu erstellen, die Matrizen und Vektoren darstellen kann und miteinander mittels der Matrizenmultiplikation verknüpfen kann.

Beispiel: unten links wird eine Translation durchgeführt, während oben rechts eine Rotation, oben links eine Skalierung und rechts eine x-Scherung stattfindet.

Aufgabe 2:

Schreiben Sie ein Java Programm, dass ein Bild einliest und mittels der Matrizenrechnung eine Transformation des Bildes berechnet, bei dem in zwei Bildpunkten Strudel wie in der Vorlesung entstehen.

Übung 4

Aufgabe 1:

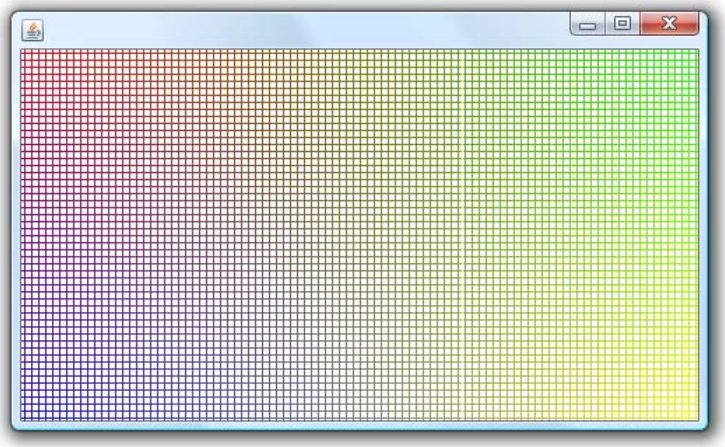
Implementieren Sie in Java eine Klassenmethode

```
drawLine(Graphics g,
         int x1,int y1,int x2,int y2,
         Color cStart,Color cEnd)
```

die im Graphikkontext g eine Linie von `(x1,y1)` nach `(x2,y2)` zeichnet und dabei einen Farbverlauf von der Farbe `cStart` zur Farbe `cEnd` annimmt. Wichtig: die Linie soll von einem beliebigen Startpunkt zu einem beliebigen Endpunkt gezeichnet werden können. Testen Sie, ob senkrechte und waagerechte Linien, auch von links nach rechts und von rechts nach links, von oben nach unten und auch von unten nach oben gezeichnet werden können. Die Klassenmethode soll das Verfahren von Bresenham verwenden.

Aufgabe 2:

Verwenden Sie Ihre obige Methode, um ein Bild mit einer Gitterstruktur zu erzeugen, das einen Farbverlauf aufzeigt. Dabei haben die vier Eckpunkte unterschiedliche Farben, d.h. für die Farben, die nicht in den Ecken starten bzw. enden müssen noch erst die Start- und Endfarben gemäß des Farbverlaufs berechnet werden. Beispiel für die Farben rot (oben links), grün (oben rechts), blau (unten links) und gelb (unten rechts).



Aufgabe 3:

Implementieren Sie in Java eine Klassenmethode

```
drawEdge(Graphics g,
         int x1,int y1,int w1,int h1,
         int x2,int y2,int w2,int h2)
```

die vom Rechteck (`x1,y1,w1,h1`) zum Rechteck (`x2,y2,w2,h2`) im Grafikkontext `g` einen Pfeil zeichnet (wird für die Aufgaben 12, 13 und 14 benötigt).

Übung 5

Aufgabe 1:

Implementieren Sie in Java eine Klassenmethode

```
drawCircle(Graphics g,int x0,int y0,int r,
           Color cNorth, Color cEast,
           Color cSouth, Color cWest)
```

die im Graphikkontext g einen Kreis um den Punkt (x0,y0) mit dem Radius r zeichnet. Dabei soll der Kreis einen Farbverlauf aufzeigen. Die vier übergebenen Farben definieren die Farben oben und unten, links und rechts. Die Klassenmethode soll das Verfahren von Bresenham verwenden.

Aufgabe 2:

Implementieren Sie in Java eine Klassenmethode

```
fillCircle(Graphics g,int x0,int y0,int r)
```

die im Graphikkontext g einen ausgefüllten Kreis um den Punkt (x0,y0) mit dem Radius r zeichnet. Die Klassenmethode soll das Verfahren von Bresenham verwenden.

Aufgabe 3:

Implementieren Sie in Java eine Klassenmethode

```
fillCircle(Graphics g,int x0,int y0,int r,
           Color cEast, Color cWest)
```

die im Graphikkontext g einen ausgefüllten Kreis um den Punkt (x0,y0) mit dem Radius r zeichnet. Dabei soll es einen Farbverlauf von der Farbe "cEast" (rechts) zu der Farbe "cWest" (links) geben. Die Klassenmethode soll das Verfahren von Bresenham verwenden.

Übung 6

Aufgabe 1:

Optimieren Sie Ihre Implementierung zu Übung 2 Aufgabe 2, indem Sie für das Finden von ähnlichen Farben die aus der Vorlesung vorgestellte mehrdimensionale Approximation verwenden. Verwenden Sie für das Sortieren den `Quicksort` oder binäre Bäume.

Übung 7

Aufgabe 1:

Implementieren Sie den Top-Down 2-3-4-Baum (mit Optimierung, sprich 4er Knoten werden beim Abstieg zuerst aufgetrennt). Verwenden Sie dazu eine Knotenklasse, die sich bis zu drei Schlüssel und bis zu vier Nachfahren merken kann.

Wenden Sie Ihre Implementierung auf die folgende Reihenfolge von Schlüsseln an und lassen Sie sich die entstehende Baumstruktur ausgeben.

13 –7 34 3 5 12 9 -200 45 14 -1 15 -6 18 -9 44

Aufgabe 2:

Implementieren Sie für den Top-Down 2-3-4-Baum eine Ausgabemethode für uDrawGraph zur Visualisierung des Baums.

Übung 8

Aufgabe 1:

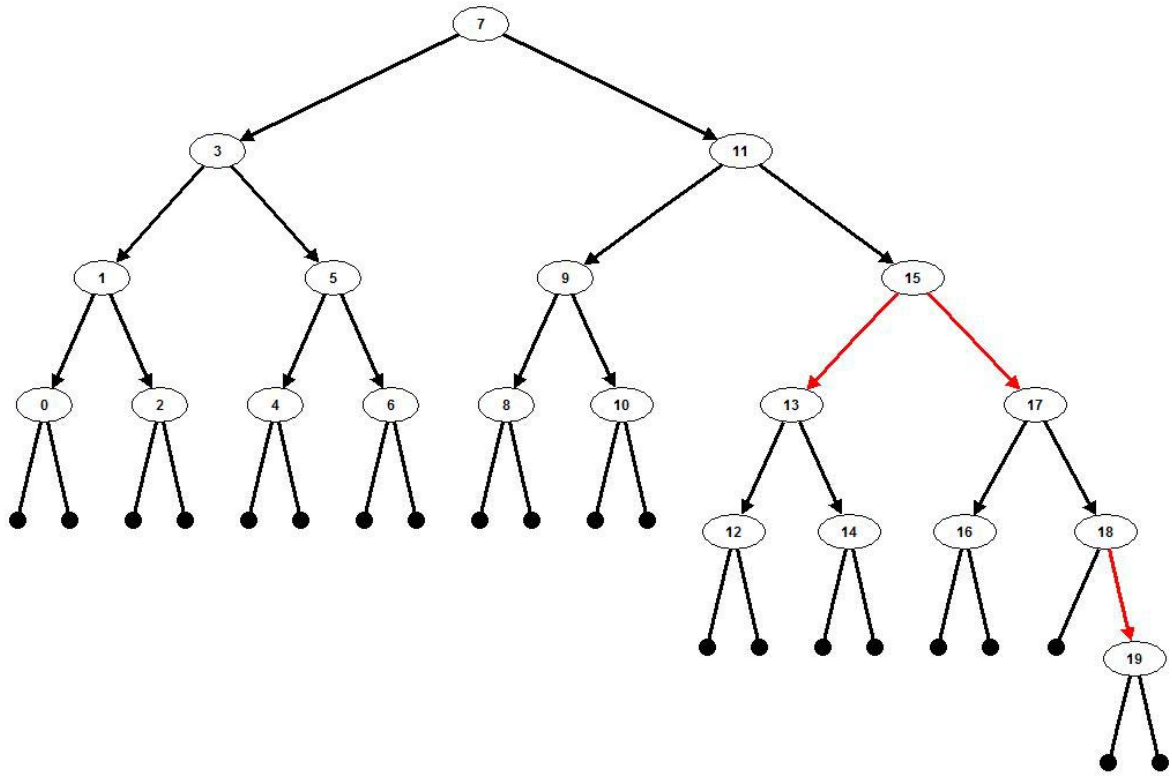
Implementieren Sie einen Rot-Schwarz Baum ohne Verwendung der `NodeHandler` Klasse und ohne Rückwärtsverkettung der Knoten zu ihren Väterknoten. Testen Sie Ihre Implementierung, indem Sie sehr viele Einträge einfügen und den Baum an Ende ausdrucken, so dass die Baumstruktur zu erkennen ist.

Aufgabe 2:

Implementieren Sie für den Rot-Schwarz Baum eine Ausgabemethode für uDrawGraph (siehe Aufgabe Übung 8) zur Visualisierung des Baums.

Aufgabe 3:

Implementieren Sie für den Rot-Schwarz Baum eine Methode, die die Tiefe des Baums ermittelt. Dabei sollen aber 3er und 4er Knoten zusammengefasst werden, d.h. rote Kanten werden nicht gezählt. Der nachfolgende Baum hätte damit eine Tiefe von 3.



Übung 9

Aufgabe 1:

Implementieren Sie für Ihren Rot-Schwarz Baum aus Übung 10 die `remove` Methode ohne Verwendung der `NodeHandler` Klasse.

Wenden Sie Ihre Implementierung auf die folgende Reihenfolge von Schlüsseln an und lassen Sie sich die entstehende Baumstruktur ausgeben.

13 –7 34 3 5 12 9 -200 45 14 -1 15 -6 18 -9 44

Danach löschen Sie die Schlüssel

13 -7 45 12 -1

und lassen sich nach jedem Löschen immer wieder die Baumstruktur ausgeben.

Übung 10

Aufgabe 1:

Implementieren Sie Patrica Trees ohne Verwendung der `NodeHandler` Klasse, in denen Sie zu Strings beliebige Werte (Generics verwenden) abspeichern können.

Aufgabe 2:

Implementieren Sie für Patricia Trees eine Ausgabemethode für uDrawGraph zur Visualisierung.

Übung 11

Aufgabe 1:

Implementieren Sie die RoBDDs und fügen Sie noch Funktionen für Konjunktion, Disjunktion, Negation, Implikation und Äquivalenz hinzu. Variablen sollen nicht mehr durch Zahlen sondern durch Namen repräsentiert werden. Dazu sollten Sie eine weitere Hashtabelle einführen, die die Namen auf die Variablenintegerwerte abbildet.

Aufgabe 2:

Implementieren Sie für RoBDDs eine Ausgabemethode für uDrawGraph zur Visualisierung.

Übung 12

Aufgabe 1:

Implementieren Sie zu einem gewichteten Graphen auf Basis eine Adjazenzlistenimplementierung(!!!!) die Berechnung eines minimalen Spannbaums. Verwenden Sie dazu eine Prioritätenheap. Zeigen Sie das Ergebnis grafisch an.

Testen (Laufzeitmessung) Sie Ihre Datenstruktur mit einem Graphen, der mindestens 1000 Knoten und 5000 Kanten hat.

Aufgabe 2:

Implementieren Sie zu einem ungewichteten Graphen auf Basis eine Adjazenzlistenimplementierung(!!!!) die Berechnung der maximalen Zusammenhangskomponenten. Zeichnen Sie den Graphen und färben Sie die unterschiedlichen Zusammenhangskomponenten unterschiedlich farbig ein.

Aufgabe 3:

Schreiben Sie eine Methode, die zwei Knoten k_1 und k_2 übergeben bekommt und `true` zurückgibt, wenn es einen Weg von k_1 zu k_2 gibt. Wenn es einen solchen Weg gibt, soll er farbig in dem Graphen markiert werden.

Übung 13

Aufgabe 1:

Implementieren Sie das Verfahren von Huffman zur Datenkomprimierung.

Testen Sie Ihr Programm, indem Sie beliebige Texte eingeben, diese komprimieren lassen und die Komprimierung anschließend wieder dekomprimieren.