Übungsblatt 3

(15. November 2010)

Aufgabe 1 In der Vorlesung wurden als Datenstruktur zur Klassifizierung von Farbpunkten eines Bildes KD-Bäume eingeführt. KD-Bäume sind binäre Suchbäume für mehrdimensionale Daten. In unserem Fall für Farbobjekte mit drei Farbdimensionen. Auf jeder Ebene des Baumes wird auf eine andere der drei Farbdaten im RGB Farbraum unterschieden.

Schreiben Sie eine Klasse, die KD-Bäume realisiert und die folgende Schnittstelle implementiert. Der innere Aufzählungstyp gibt dabei an, welche Farbe beim aktuellen Baumknoten benutzt wurde, um den Größer-Vergleich für diese Ebene durchzuführen.

Die Typvariable `A` steht für den Typ der Kategorien, in die Farbpunkte eingeordnet werden. Wenn also nur eine Entscheidung zwischen Vorder und Hintergrund zu treffen ist, kann diese vom Typ `Boolean` instanziiert werden.

Die Methode `nextNabour` gibt den ähnlichsten Farbpunkt und sein Kategorie zurück.

Die Methode `categorize` soll eine Farbe einer der angelernten Kategorien zuordnen. Hier wird eine Liste der Kategorien der k-nächsten Nachbarn zurück gegeben.

Die Methode `insert` für den Aufbau des Baumes vorgesehen.

Schließlich soll die Methode `newInstance` es ermöglichen, neue leere Baumobjekte zu erzeugen.

Ein kleines graphisches Testprogramm, um mit Ihrer Implementierung eine einfache Bilderkennung durchzuführen, wird zur Abgabe geliefert.

KD.java

```
1  import java.awt.Color;
2  import java.util.List;
3
4  public interface KD<A> {
5    enum ColorCut {red, green, blue;
6
7      ColorCut next(){return values()[(this.ordinal()+1)%3];};
8
9      int getColorValue(Color c){
10       switch (this) {
11         case red: return c.getRed();
12         case blue: return c.getBlue();
13         default: return c.getGreen();
14       }
```

```
15      }
16    }
17
18    KD<A> newInstance();
19
20    KD<A> getParent();
21    void setParent(KD<A> p);
22
23    void setLeft(KD<A> p);
24    KD<A>  getLeft();
25
26    void setRight(KD<A> p);
27    KD<A>  getRight();
28
29    void setColorCut(ColorCut cc);
30    ColorCut getColorCut();
31
32    Pair<Color, A> getElement();
33    void setElement(Pair<Color, A> element);
34
35    Pair<Color,A> nextNabour(Color c);
36
37    List<A> categorize(Color c);
38
39    void insert(Pair<Color,A> p);
40  }
```

Hier noch die kleine Hilfsklasse `Pair`.

Pair.java

```
1  public class Pair<A, B> {
2    A fst;
3    B snd;
4    public Pair(A fst, B snd) {
5      this.fst = fst;
6      this.snd = snd;
7    }
8    @Override
9    public String toString() {
10     return "("+fst+", "+snd+")";
11   }
12 }
```