Allgemeines Programmierpraktikum **SS 2008**
Dr. Henrik Brosenne Georg-August Universität Göttingen
Institut für Informatik

Übung 6

Testat nach Mittwoch, 11.06., 14.00 Uhr in der Gruppenübung.

Vorbereitung

Java

Informieren Sie sich in der *Java 2 Platform Standard Edition 5.0 API Specification* `http://java.sun.com/j2se/1.5.0/docs/api/` über die Klasse `java.lang.Integer` und die typisierte Schnittstelle `java.lang.Comparable<T>`.

Aufgabe 1 – 100 Punkte

Typisierte Klassen

1. Ändern Sie die Klasse `CompRational` von Übung 5, Aufgabe 2.1 so, dass die typisierte Schnittstelle `java.lang.Comparable<T>` implementiert wird.

2. Programmieren Sie eine **typsisierte** Klasse `RedBlackTree`, einen Rot-Schwarz-Baum für Referenzen auf Objekte von Klassen die `java.lang.Comparable<T>` implementieren.

   Programmieren Sie mindestens folgende Methoden.

   - `insert` fügt ein Element in den Baum ein. Enthält der Baum schon ein äquivalentes Element (`compareTo == 0`), wird eine Exception ausgelöst.
   - `contains` gibt Auskunft, ob ein äquivalentes Element in Baum enthalten ist oder nicht.
   - `getMaximum` liefert das maximale Element im Baum zurück.
   - `getMinimum` liefert das minimale Element im Baum zurück.

3. Schreiben Sie ein ausführbare Klasse, die rationale Zahlen auf der Kommandozeile übergeben bekommt und die größte und kleinste der Zahlen ausgibt.

   Die Kommandozeile enthält eine gerade Anzahl von ganzen Zahlen. Zwei aufeinanderfolgende Zahlen werden als Zähler und Nenner einer rationalen Zahl (`CompRational`) interpretiert.

   Speichern Sie die übergebenen Zahlen in einem Rot-Schwarz-Baum für `CompRational` und lassen Sie das größte und kleinste Element ausgeben.

4. Schreiben Sie ein ausführbare Klasse, die ganze Zahlen (`java.lang.Integer`) auf der Kommandozeile übergeben bekommt und die größte und kleinste der Zahlen ausgibt.

   Speichern Sie die übergebenen Zahlen in einem Rot-Schwarz-Baum für `Integer` und lassen Sie das größte und kleinste Element ausgeben.

Allgemeine Anforderungen

Kommentieren Sie Ihren Programmtext **ausführlich**. Spärliche und/oder schlechte Kommentierung führt zu Punktabzug.

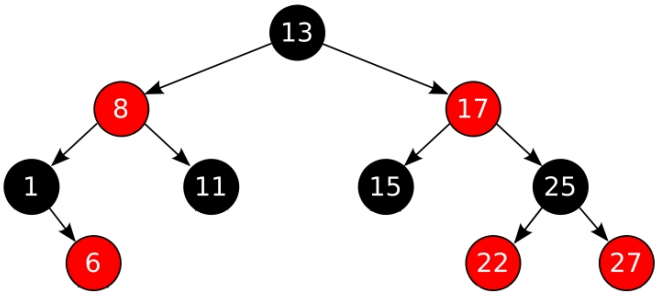
Rot-Schwarz-Bäume

Ein Rot-Schwarz-Baum ist ein binärer Suchbaum, in dem jeder Knoten als Zusatzinformation entweder rot oder schwarz eingefärbt ist. Neben den Bedingungen für einen binären Suchbäume, muss ein Rot-Schwarz-Bäume noch folgende Forderungen erfüllen.

- Die Wurzel des Baums ist schwarz.
- Jeder rote Knoten ist entweder ein Blatt oder hat zwei schwarze Kinder.
- Alle Pfade von einem gegebenen Knoten zu jedem von diesem Knoten erreichbaren Blattknoten, enthält die gleiche Anzahl schwarzer Knoten (Schwarztiefe).

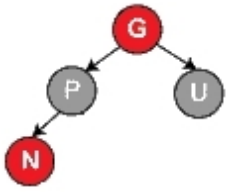
Beispiel

Ein Rot-Schwarz-Baum mit Schwarztiefe 2.



Das Einfügen in den Rot-Schwarz-Baum funktioniert wie das Einfügen in einen binären Suchbaum, der neue Knoten ist rot, damit bleibt die Schwarztiefe des Baumes erhalten. Nach dem Einfügen könnte jedoch eine der beiden andere Eigenschaften des Rot-Schwarz-Baumes verletzt sein, dann muss man den Baum reparieren.

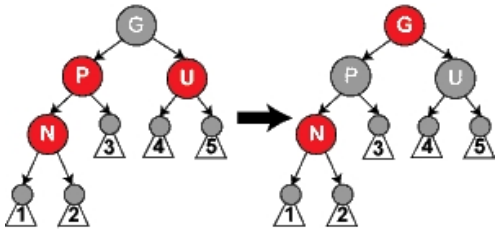
Wir betrachten den neuen Knoten (*new, N*), den Vater (*parent, P*), den Onkel (*uncle, U*) und den Großvater (*grandparent, G*).



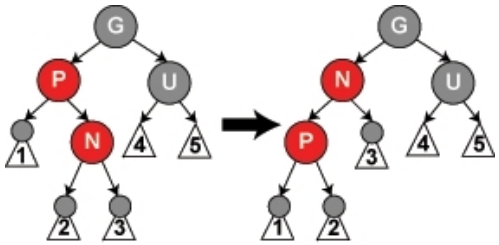
Fall 1
Der Vater des neuen Knotens ist schwarz. Es ist nichts zu tun, alle Eigenschaften des Rot-Schwarz-Baumes gelten.

Fall 2
Der neue Knoten ist die Wurzel. Färbe den neuen Knoten schwarz.

Fall 3
Vater und Onkel des neuen Knotens sind rot. Färbe Vater und Onkel schwarz. Starte die Reparatur erneut mit dem Großvater als neuem Knoten.



Fall 4
Der neue Knoten hat einen schwarzen oder keinen Onkel und ist das rechte Kind seines roten Vaters. Durch eine Linksrotation um den Vater wird die Rolle des neuen Knotens und seines Vaters vertauscht.



Fall 5
Der neue Knoten hat einen schwarzen oder keinen Onkel und ist das linke Kind seines roten Vaters. Durch eine Rechtsrotation um den Großvater wird der Großvater zum Kind des Vaters. Die Farben des Vaters und des ehemaligen Großvaters werden vertauscht.