Ziel dieses Praktikums ist es, die Binomial Heap Datenstruktur in Form einer Java Klasse zu implementieren. Um die Datenstruktur allgemein zu halten, kommt das Generics Konzept zum Einsatz. Somit ist es zum Beispiel möglich, folgende Instanz zu definieren:

```
BinomialHeap<Integer,String> heap = new BinomialHeap<Integer,String>
```

Die Instanz `heap` speichert einen `Integer` Schlüssel zusammen mit einem `String` Datensatz.

Die Knoten der im Binomial Heap gespeicherten Binomialbäume werden durch die `Node` Datenstruktur abgebildet. Im Unterschied zur Vorlesung enthält die Datenstruktur ein Attribut `entry` vom Typ `HeapEntry`. Ein solcher Datensatz zeigt wiederum zurück auf den Knoten. Auf diese Weise kann man direkt auf ein im Binomial Heap gespeichertes Datum zugreifen (bzw. auf den Knoten, der dieses Datum beinhaltet). Auf diese Weise kann man den Schlüssel eines Datums verkleinern oder das Datum aus dem Heap entfernen.

Die Rumpfklassen befinden sich in einem ZIP Archiv auf dem Vorlesungsserver. Das Archiv enthält folgende Klassen:

| Name | Paket | Beschreibung |
|---|---|---|
| `HeapEntry` | `binheap` | Datenstruktur zum direkten Zugriff auf die Elemente im Heap |
| `Node` | `binheap` | Knoten in einem Binomialbaum |
| `BinomialHeap` | `binheap` | Binomial Heap Datenstruktur |

**Aufgabe 1.** Installieren Sie die Rumpfklassen und machen Sie sich mit deren Aufbau vertraut.

a) Welche Attribute und Methoden enthält die Klasse `Node`?

b) Welche Attribute und Methoden enthält die Klasse `HeapEntry`?

c) Welche Attribute und Methoden enthält die Klasse `BinomialHeap`?

**Aufgabe 2.** Die Methoden zum Einfügen von Datensätzen in den Binomial Heap werden im Rahmen dieser Aufgabe implementiert.

a) Implementieren Sie die Methode `link()` zum Verknüpfen von zwei Binomialbäumen $B_k$ zu einem Binomialbaum $B_{k+1}$.

b) Implementieren Sie die Methode `merge()` zum Zusammenfügen zweier Binomial Heaps zu einer verketteten Liste von Binomialbäumen, die nach aufsteigendem Wurzelgrad sortiert ist.

c) Implementieren Sie die Methode `union()` zur Vereinigung zweier Binomial Heaps.

d) Implementieren Sie die Methode `insert()` zum Einfügen eines Heap Entry's in den Binomial Heap.

e) Implementieren Sie die Methode `minimum()` zur Suche eines Elements mit minimalem Schlüssel.

*Hinweis:* Implementieren Sie zunächst alle Methoden. Der Test der Implementierung erfolgt in Aufgabe 2.

**Aufgabe 3.** Nach der Implementierung steht nun der Test derselbigen an. Legen Sie hierzu folgende Binomial Heap Instanz an:

```
BinomialHeap<Integer,String> heap = new BinomialHeap<Integer,String>
```

In diesem Heap werden Paare bestehend aus einer Zahl und einem String gespeichert. Die Zahl dient als Schlüssel.

a) Fügen Sie Elemente mit dem Schlüssel $i$ und Datum `Ëlement` $i$`"` in den Heap ein, wobei $i$ die Werte $20, 19, \ldots, 2, 1$ in genau dieser Reihenfolge durchläuft.

b) Benutzen Sie die Methode `dumpHeap()`, um den Heap auf der Konsole auszugeben. Vergleichen Sie die Ausgabe mit dem Dump am Ende dieses Dokuments. Bei Problemen leistet der Debugger nützliche Dienste.

**Aufgabe 4.** Nachdem das Einfügen in den Heap funktioniert, kümmern wir uns nun um das Extrahieren des Minimums sowie um das Löschen von Elementen.

a) Implementieren Sie die Methode `invertChildren()`, die die Kindliste eines Knotens invertiert und dabei die `parent`-Zeiger auf `null` setzt.

b) Implementieren Sie die Methode `extractMinimum()` zum Extrahieren eines Knotens mit minimalem Schlüssel.

c) Testen Sie die Methode `extractMinimum()`, indem Sie mit den Daten aus Aufgabe 2 die zehn kleinsten Elemente entfernen.

d) Implementieren Sie die Methode `decreaseKey()` zum Verkleinern des Schlüssels eines im Heap gespeicherten Elements.

e) Testen Sie die Methode `decreaseKey()`, indem Sie in dem Heap von Aufgabe 2 dem Datensatz $\langle 10, \text{Element } 10 \rangle$ den Schlüssel 0 zuweisen.

f) Implementieren Sie die Methode `moveToRoot()`, mit der der Inhalt eines Knotens an die Wurzel des entsprechenden Binomialbaums verschoben wird.

g) Implementieren Sie die Methode `delete()` zum Löschen eines Elements aus dem Heap. Setzen Sie hierzu die Methode `moveToRoot()` und `invertChildren()` ein.

h) Testen Sie die Methode `delete()`, indem Sie im Heap von Aufgabe 2 das Element mit dem Schlüssel 13 löschen.

**Aufgabe 5.** Die letzte Aufgabe besteht aus einem abschließenden Test.

a) Fügen Sie in einen leeren Heap die folgenden Daten ein:

| *Schlüssel* | *Datum* |
|---|---|
| 7 | `erfolgreich` |
| 15 | `haben` |
| 5 | `das` |
| 2 | `Glueckwunsch` |
| 8 | `absolviert` |
| 12 | `Praktikum` |
| 1 | `Herzlichen` |
| 3 | `Sie` |

b) Verkleinern Sie den Schlüssel des Wertes `haben` auf 4.

c) Verkleinern Sie den Schlüssel des Wertes `Praktikum` auf 6.

d) Geben Sie alle Elemente in aufsteigender Reihenfolge aus, indem Sie sie nacheinander aus dem Heap extrahieren.

Dump zu Aufgabe 2:

```
============================================================
= Einfügen von: <20,Element 20>
============================================================

Heap Dump
=========
Tree with root 20 (d:0)
  Node 20 (d:0): leaf


============================================================
= Einfügen von: <19,Element 19>
============================================================

Heap Dump
=========
Tree with root 19 (d:1)
  Node 19 (d:1): children --> 20 (d:0)
  - Node 20 (d:0): leaf


============================================================
= Einfügen von: <18,Element 18>
============================================================

Heap Dump
=========
Tree with root 18 (d:0)
  Node 18 (d:0): leaf
Tree with root 19 (d:1)
  Node 19 (d:1): children --> 20 (d:0)
  - Node 20 (d:0): leaf


============================================================
= Einfügen von: <17,Element 17>
============================================================

Heap Dump
=========
Tree with root 17 (d:2)
```

```
  Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  - Node 19 (d:1): children --> 20 (d:0)
  -- Node 20 (d:0): leaf
  - Node 18 (d:0): leaf


============================================================
= Einfügen von: <16,Element 16>
============================================================

Heap Dump
=========
Tree with root 16 (d:0)
  Node 16 (d:0): leaf
Tree with root 17 (d:2)
  Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  - Node 19 (d:1): children --> 20 (d:0)
  -- Node 20 (d:0): leaf
  - Node 18 (d:0): leaf


============================================================
= Einfügen von: <15,Element 15>
============================================================

Heap Dump
=========
Tree with root 15 (d:1)
  Node 15 (d:1): children --> 16 (d:0)
  - Node 16 (d:0): leaf
Tree with root 17 (d:2)
  Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  - Node 19 (d:1): children --> 20 (d:0)
  -- Node 20 (d:0): leaf
  - Node 18 (d:0): leaf


============================================================
= Einfügen von: <14,Element 14>
============================================================

Heap Dump
=========
```

```
Tree with root 14 (d:0)
  Node 14 (d:0): leaf
Tree with root 15 (d:1)
  Node 15 (d:1): children --> 16 (d:0)
  - Node 16 (d:0): leaf
Tree with root 17 (d:2)
  Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  - Node 19 (d:1): children --> 20 (d:0)
  -- Node 20 (d:0): leaf
  - Node 18 (d:0): leaf


============================================================
= Einfügen von: <13,Element 13>
============================================================

Heap Dump
=========
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
  - Node 15 (d:1): children --> 16 (d:0)
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <12,Element 12>
============================================================

Heap Dump
=========
Tree with root 12 (d:0)
  Node 12 (d:0): leaf
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
```

```
  - Node 15 (d:1): children --> 16 (d:0)
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <11,Element 11>
============================================================

Heap Dump
=========
Tree with root 11 (d:1)
  Node 11 (d:1): children --> 12 (d:0)
  - Node 12 (d:0): leaf
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
  - Node 15 (d:1): children --> 16 (d:0)
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <10,Element 10>
============================================================

Heap Dump
=========
Tree with root 10 (d:0)
  Node 10 (d:0): leaf
Tree with root 11 (d:1)
  Node 11 (d:1): children --> 12 (d:0)
  - Node 12 (d:0): leaf
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
  - Node 15 (d:1): children --> 16 (d:0)
```

```
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <9,Element 9>
============================================================

Heap Dump
=========
Tree with root 9 (d:2)
  Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  - Node 11 (d:1): children --> 12 (d:0)
  -- Node 12 (d:0): leaf
  - Node 10 (d:0): leaf
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
  - Node 15 (d:1): children --> 16 (d:0)
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <8,Element 8>
============================================================

Heap Dump
=========
Tree with root 8 (d:0)
  Node 8 (d:0): leaf
Tree with root 9 (d:2)
  Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  - Node 11 (d:1): children --> 12 (d:0)
  -- Node 12 (d:0): leaf
  - Node 10 (d:0): leaf
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
```

```
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
  - Node 15 (d:1): children --> 16 (d:0)
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <7,Element 7>
============================================================

Heap Dump
=========
Tree with root 7 (d:1)
  Node 7 (d:1): children --> 8 (d:0)
  - Node 8 (d:0): leaf
Tree with root 9 (d:2)
  Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  - Node 11 (d:1): children --> 12 (d:0)
  -- Node 12 (d:0): leaf
  - Node 10 (d:0): leaf
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
  - Node 15 (d:1): children --> 16 (d:0)
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <6,Element 6>
============================================================

Heap Dump
=========
Tree with root 6 (d:0)
  Node 6 (d:0): leaf
Tree with root 7 (d:1)
  Node 7 (d:1): children --> 8 (d:0)
  - Node 8 (d:0): leaf
```

```
Tree with root 9 (d:2)
  Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  - Node 11 (d:1): children --> 12 (d:0)
  -- Node 12 (d:0): leaf
  - Node 10 (d:0): leaf
Tree with root 13 (d:3)
  Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  - Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  -- Node 19 (d:1): children --> 20 (d:0)
  --- Node 20 (d:0): leaf
  -- Node 18 (d:0): leaf
  - Node 15 (d:1): children --> 16 (d:0)
  -- Node 16 (d:0): leaf
  - Node 14 (d:0): leaf


============================================================
= Einfügen von: <5,Element 5>
============================================================

Heap Dump
=========
Tree with root 5 (d:4)
  Node 5 (d:4): children --> 13 (d:3) --> 9 (d:2) --> 7 (d:1) --> 6 (d:0)
  - Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  -- Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  --- Node 19 (d:1): children --> 20 (d:0)
  ---- Node 20 (d:0): leaf
  --- Node 18 (d:0): leaf
  -- Node 15 (d:1): children --> 16 (d:0)
  --- Node 16 (d:0): leaf
  -- Node 14 (d:0): leaf
  - Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  -- Node 11 (d:1): children --> 12 (d:0)
  --- Node 12 (d:0): leaf
  -- Node 10 (d:0): leaf
  - Node 7 (d:1): children --> 8 (d:0)
  -- Node 8 (d:0): leaf
  - Node 6 (d:0): leaf


============================================================
= Einfügen von: <4,Element 4>
```

```
============================================================

Heap Dump
=========
Tree with root 4 (d:0)
  Node 4 (d:0): leaf
Tree with root 5 (d:4)
  Node 5 (d:4): children --> 13 (d:3) --> 9 (d:2) --> 7 (d:1) --> 6 (d:0)
  - Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  -- Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  --- Node 19 (d:1): children --> 20 (d:0)
  ---- Node 20 (d:0): leaf
  --- Node 18 (d:0): leaf
  -- Node 15 (d:1): children --> 16 (d:0)
  --- Node 16 (d:0): leaf
  -- Node 14 (d:0): leaf
  - Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  -- Node 11 (d:1): children --> 12 (d:0)
  --- Node 12 (d:0): leaf
  -- Node 10 (d:0): leaf
  - Node 7 (d:1): children --> 8 (d:0)
  -- Node 8 (d:0): leaf
  - Node 6 (d:0): leaf


============================================================
= Einfügen von: <3,Element 3>
============================================================

Heap Dump
=========
Tree with root 3 (d:1)
  Node 3 (d:1): children --> 4 (d:0)
  - Node 4 (d:0): leaf
Tree with root 5 (d:4)
  Node 5 (d:4): children --> 13 (d:3) --> 9 (d:2) --> 7 (d:1) --> 6 (d:0)
  - Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  -- Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  --- Node 19 (d:1): children --> 20 (d:0)
  ---- Node 20 (d:0): leaf
  --- Node 18 (d:0): leaf
  -- Node 15 (d:1): children --> 16 (d:0)
  --- Node 16 (d:0): leaf
```

```
  -- Node 14 (d:0): leaf
  - Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  -- Node 11 (d:1): children --> 12 (d:0)
  --- Node 12 (d:0): leaf
  -- Node 10 (d:0): leaf
  - Node 7 (d:1): children --> 8 (d:0)
  -- Node 8 (d:0): leaf
  - Node 6 (d:0): leaf


============================================================
= Einfügen von: <2,Element 2>
============================================================

Heap Dump
=========
Tree with root 2 (d:0)
  Node 2 (d:0): leaf
Tree with root 3 (d:1)
  Node 3 (d:1): children --> 4 (d:0)
  - Node 4 (d:0): leaf
Tree with root 5 (d:4)
  Node 5 (d:4): children --> 13 (d:3) --> 9 (d:2) --> 7 (d:1) --> 6 (d:0)
  - Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  -- Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  --- Node 19 (d:1): children --> 20 (d:0)
  ---- Node 20 (d:0): leaf
  --- Node 18 (d:0): leaf
  -- Node 15 (d:1): children --> 16 (d:0)
  --- Node 16 (d:0): leaf
  -- Node 14 (d:0): leaf
  - Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  -- Node 11 (d:1): children --> 12 (d:0)
  --- Node 12 (d:0): leaf
  -- Node 10 (d:0): leaf
  - Node 7 (d:1): children --> 8 (d:0)
  -- Node 8 (d:0): leaf
  - Node 6 (d:0): leaf


============================================================
= Einfügen von: <1,Element 1>
============================================================
```

```
Heap Dump
=========
Tree with root 1 (d:2)
  Node 1 (d:2): children --> 3 (d:1) --> 2 (d:0)
  - Node 3 (d:1): children --> 4 (d:0)
  -- Node 4 (d:0): leaf
  - Node 2 (d:0): leaf
Tree with root 5 (d:4)
  Node 5 (d:4): children --> 13 (d:3) --> 9 (d:2) --> 7 (d:1) --> 6 (d:0)
  - Node 13 (d:3): children --> 17 (d:2) --> 15 (d:1) --> 14 (d:0)
  -- Node 17 (d:2): children --> 19 (d:1) --> 18 (d:0)
  --- Node 19 (d:1): children --> 20 (d:0)
  ---- Node 20 (d:0): leaf
  --- Node 18 (d:0): leaf
  -- Node 15 (d:1): children --> 16 (d:0)
  --- Node 16 (d:0): leaf
  -- Node 14 (d:0): leaf
  - Node 9 (d:2): children --> 11 (d:1) --> 10 (d:0)
  -- Node 11 (d:1): children --> 12 (d:0)
  --- Node 12 (d:0): leaf
  -- Node 10 (d:0): leaf
  - Node 7 (d:1): children --> 8 (d:0)
  -- Node 8 (d:0): leaf
  - Node 6 (d:0): leaf
```