Constantin Lazari, Marco Wettstein 22. Oktober 2013

1. Nochmal Komplexität

   (a) Vergleichen Sie das Wachstum untenstehender Programme und geben Sie diese in der

   Landau-Notation an.

```
while x < n do
  x := x + 1
  n := n - 1
end

while x < n do
        n := n / 2
        x := 2 * x
end
```

Lösung:

Beide sind Element von $\mathcal{O}(n)$

2. Listen

   Implementieren Sie den Datentyp einer einfach verketteten Liste (mit integer Datenfeldern). Die Listen sollen folgende Funktionalitäten aufweisen:

   - Das erste Element der Liste auslesen.
   - Das letzte Element der Liste auslesen.
   - Ein Objekt am Anfang der Liste hinzufügen.
   - Ein Objekt am Schluss der Liste hinzufügen.
   - Anzahl Elemente der Liste zurückgeben.
   - Mit einer anderen Liste vergleichen.
   - Abfragen ob ein bestimmtes Objekt in der Liste vorkommt.

Lösung:

```
public class IntegerList {

        private int value;
        private IntegerList tail;

        public IntegerList(int value) {
                this.value = value;
        }

        public IntegerList(int value, IntegerList tail) {
                this.value = value;
                this.tail = tail;
```

```
        }

        public void addToTail(int value) {
                if (this.tail == null)
                        this.tail = new IntegerList(value);
                else
                        this.tail.addToTail(value);
        }

        public int removeHead() {
                if (this.tail == null)
                        throw new NullPointerException("cannot
                           remove my own head");
                int headValue = this.getHeadValue();

                this.value = this.tail.getHeadValue();
                this.tail = this.tail.tail;
                return headValue;
        }

        public void addToHead(int value) {
                this.tail = new IntegerList(this.value, this.tail
                   );
                this.value = value;
        }

        public int getHeadValue() {
                return value;
        }

        public int getTailValue() {
                if (this.tail == null)
                        return getHeadValue();
                else
                        return this.tail.getTailValue();
        }

        public int getSize() {
                if (this.tail == null)
                        return 1;
                else
                        return 1 + this.tail.getSize();
        }

        @Override
        public boolean equals(Object obj) {
                if (obj == null)
                        return false;
                if (!(obj instanceof IntegerList))
                        return false;
                IntegerList otherList = (IntegerList) obj;

                if (this.getHeadValue() != otherList.getHeadValue
                   ())
                        return false;
```

```
                if (this.tail != null && otherList.tail != null)
                        return tail.equals(otherList.tail);

                return this.tail == null && otherList.tail ==
                    null;

        }

        public boolean contains(int value) {
                if (this.value == value)
                        return true;

                if (this.tail != null)
                        return this.tail.contains(value);
                else
                        return false;

        }

        @Override
        public String toString() {

                return this.value
                                + (this.tail != null ? "," + this
                                    .tail.toString() : "");

        }

        public IntegerList clone() {
                IntegerList clone = new IntegerList(getHeadValue
                    ());
                if (tail != null)
                        clone.tail = tail.clone();
                return clone;
        }

}
```

[language=Java]

3. Menge

   Benutzen Sie Ihre Implementation von Listen aus der ersten Aufgabe und implementieren Sie den Datentyp einer Menge mit folgenden Funktionalitäten:

   - Abfrage ob ein bestimmtes Element zur Menge gehört.
   - Die Menge als String von der Form fx1, x2, . . .g zurückgeben.
   - Ein Element hinzufügen.
   - Mit einer anderen Menge vereinigen.
   - mit einer anderen Menge schneiden.
   - Anzahl Elemente der Menge abfragen. Beachten Sie, dass mehrfach vorkommende Elemente nur einmal gezählt werden sollen.
   - Mit einer anderen Menge vergleichen. Beachten Sie, dass beim Vergleich von Mengen die Reihenfolge und Wiederholungen keine Rolle spielen.

Lösung:

```
public class IntegerSet {

        private IntegerList data;

        public IntegerSet() {

        }

        public void addElement(int value) {
                if (data == null)
                        data = new IntegerList(value);
                else if (!data.contains(value))
                        data.addToTail(value);

        }

        public boolean hasElement(int value) {
                if (data == null)
                        return false;
                else
                        return data.contains(value);
        }

        @Override
        public String toString() {
                if (data == null)
                        return "{}";
                else
                        return "{" + data.toString() + "}";
        }

        public int getSize() {

                if (data == null)
```

```
                return 0;
        else
                return data.getSize();
    }

    public int removeElement() {
        if (data == null)
                throw new NullPointerException(
                                "cannot remove element
                                   from empty set");
        if (data.getSize() > 1)
                return data.removeHead();
        else {
                int value = data.getHeadValue();
                data = null;
                return value;
        }
    }

    public void union(IntegerSet otherSet) {

        if (otherSet != null) {

                otherSet = (IntegerSet) otherSet.clone();

                while (otherSet.getSize() > 0)
                        addElement(otherSet.removeElement
                           ());

        }
    }

    public void interSec(IntegerSet otherSet) {
        if (otherSet != null) {
                otherSet = (IntegerSet) otherSet.clone();

                IntegerList intersection = null;
                while (otherSet.getSize() > 0) {
                        int value = otherSet.
                           removeElement();
                        if (hasElement(value)) {
                                if (intersection == null)
                                        intersection =
                                           new
                                           IntegerList(
                                           value);
                                else
                                        intersection.
                                           addToTail(
                                           value);
                        }
                }
                data = intersection;
        } else {
                // make empty set
```

```
                        data = null;
                }
        }

        @Override
        public boolean equals(Object obj) {
                if (obj == null)
                        return false;
                if (!(obj instanceof IntegerSet))
                        return false;
                IntegerSet otherSet = (IntegerSet) obj;

                if (getSize() != otherSet.getSize())
                        return false;

                otherSet = (IntegerSet) otherSet.clone();

                otherSet.union(this);
                System.out.println(otherSet.getSize());

                return otherSet.getSize() == getSize();
        }

        public IntegerSet clone() {
                IntegerSet clone = new IntegerSet();
                if (data != null)
                        clone.data = data.clone();

                return clone;
        }

}
```

[language=Java]