Praktische Informatik 2 SoSe 98

Prof. Bernd Krieg-Brückner
Axel Lankenau
Shi Hui
Christoph Lüth

4. Übungsblatt

Ausgabe: 15.06.98

Abgabe: 29.06.98

---

1 *Eine Menge zu tun...* *Wichtung: 25%*

Definieren Sie eine Schnittstelle `IntSet`, die folgende Methoden für die Manipulation einer Menge ganzer Zahlen enthält:

`addElem()`	Fügt ein Element in die Menge ein
`removeElem()`	Löscht das entsprechende Element
`isElem()`	Prüft, ob entsprechendes Element enthalten
`isEmpty()`	Prüft, ob die Menge leer ist
`toString()`	Erzeugt einen String, der die Elemente der Menge enthält

Wählen Sie geeignete Argumente und Rückgabetypen und begründen Sie Ihre Entscheidung! Beachten Sie, daß *Menge* hier im Sinne des mathematischen Begriffs gemeint ist.

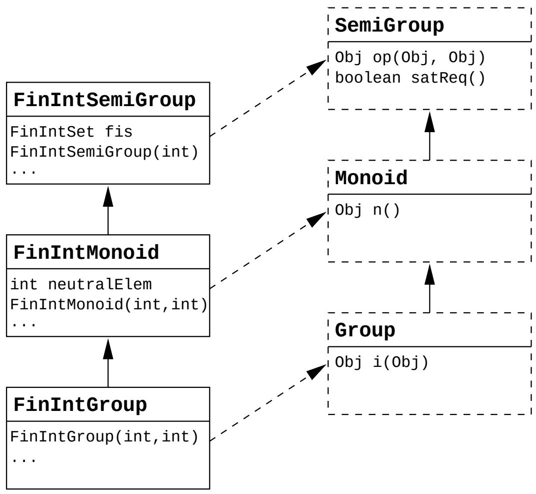
2 *Listige Menge...* *Wichtung: 30%*

Implementieren Sie eine Klasse `MyIntSet`, die die Schnittstelle `IntSet` aus Aufgabe 1 bereitstellt. Dabei sollen diejenigen Gruppen, deren Matrikelnummernsumme *gerade* ist, diese Menge als *Liste* (siehe Übungsblatt 3 oder Tutorium), und die anderen Gruppen als *Vektor* (siehe `java.util.Vector`) repräsentieren.

3 *Von der Menge zur Gruppe...* *Wichtung: 45%*

Die Klassen– und Schnittstellenhierarchie in nebenstehender Abbildung soll implementiert werden. Die Schnittstellen (dargestellt als gestrichelte Kästen, `Obj` steht für `Object`) definieren die Operationen, die jeweils eine Halbgruppe, ein Monoid und eine Gruppe ausmachen. Durch die Vererbung enthalten alle Schnittstellen die Operation `satReq()`, mit der in den implementierten Klassen dann überprüft werden kann, ob das entsprechende Objekt die Bedingungen einer Halbgruppe, eines Monoids bzw. einer Gruppe erfüllt.

Die Klassen `FinIntSemiGroup`, `FinIntMonoid` und `FinIntGroup` implementieren jeweils ihre Schnittstellen und repräsentieren Halbgruppen, Monoide und Gruppen auf endlichen Mengen natürlicher Zahlen. Neben den in der Abbildung dargestellten Datenfeldern und Methoden sind weitere sinnvolle Methoden zu implementieren.



Wie in der Abbildung angedeutet, besitzt die Klasse `FinIntSemiGroup` als Datenfeld ein Objekt vom Typ `FinIntSet`. Die Klasse `FinIntSet` implementiert eine Schnittstelle `Set`, die dieselben Methoden wie die aus Aufgabe 1 bekannte Schnittstelle definiert, allerdings auf Objekten operiert. Ein Objekt vom Typ `FinIntSet` ist eine endliche Menge natürlicher Zahlen. Sowohl die Klasse `FinIntSet` als auch die neue Schnittstelle `Set` sind zu implementieren.

Es ist zu beachten, daß die Methode `Object op(Object,Object)` in der Klasse `FinIntSemiGroup` definiert werden muß. Der Einfachheit halber soll als Verknüpfung $op(a, b) = (a \cdot b)\%max$ mit $a, b \in \mathbb{N}_{max}$ gewählt werden, da dies für Teilmengen von $\mathbb{N}_{max} = \{n \in \mathbb{N}_0 \mid n < max\}$ eine assoziative Verknüpfung ist. **!**

Bedingungen (zu prüfen in der `satReq()`–Methode):
Eine nicht–leere Menge M wird *Halbgruppe* (Klasse `FinIntSemiGroup`) genannt, wenn auf ihr eine assoziative Verknüpfung $op : M \times M \to M$ definiert ist. Wie erwähnt, wird als diese Verknüpfung hier die Modulo–Multiplikation `op` gewählt (s.o.), wobei die Abgeschlossenheit zu überprüfen ist. **!**

Eine Halbgruppe M wird *Monoid* (Klasse `FinIntMonoid`) genannt, wenn es eine nullstellige Operation $n :\to M$ gibt, so daß gilt: $\forall a \in M : op(a, n()) = op(n(), a) = a$

Das neutrale Element soll im Konstruktor der Klasse `FinIntMonoid` bestimmt werden können — für die hier gewählte Verknüpfung ist es 1. Es ist über die `public`—Methode `n()` zugreifbar.

Ein Monoid M wird *Gruppe* (Klasse `FinIntGroup`) genannt, wenn es eine einstellige Operation $i : M \to M$ gibt, so daß gilt: $\forall x \in M : op(x, i(x)) = op(i(x), x) = n()$

Dabei liefert $n()$ das neutrale Element, $i(x)$ das inverse Element zu x.

Die entworfenen Klassen sollen wie folgt benutzt werden können: Beim Anlegen eines Objekts (z.B. eines Monoids) sollen die nötigen Parameter (z.B. max) übergeben werden können. Dem Objekt sollen dann Elemente aus der Menge $\mathbb{N}_{max} = \{n \in \mathbb{N}_0 \mid n < max\}$ hinzugefügt werden können. Ein Entfernen der Elemente ist ebenfalls vorzusehen. Die Überprüfungsmethode `satReq()` soll für *jedes* Objekt aufrufbar sein und überprüfen, ob die entsprechenden Eigenschaften erfüllt werden (z.B. Existenz einer inversen Operation bei Gruppen).

Bei der Implementierung der abgeleiteten Klassen ist genau zu überlegen, ob direkt auf geerbte Methoden zurückgegriffen werden kann, ob geerbte Methoden überschrieben werden müssen oder ob zusätzliche Methoden einbezogen werden müssen. Dokumentation!

Sie können Ihre Klasse testen, indem Sie prüfen, ob die folgende Testklasse eine vergleichbare Ausgabe produziert, wie unten angegeben. **!**

```
public class Algebra {
  public static void main (String args[]) {
    FinIntSemiGroup s = new FinIntSemiGroup(6); FinIntMonoid    m = new FinIntMonoid(6,1);
    FinIntGroup     g = new FinIntGroup(6,1);
    s.addElem(new Integer(3));m.addElem(new Integer(3));g.addElem(new Integer(3));
    System.out.println(s.toString() + "...Halbgruppe: " + s.satReq());
    // dito für m,g
    s.addElem(new Integer(1));m.addElem(new Integer(1));g.addElem(new Integer(1));
    // Ausgabe wie oben für s,m,g
    s.addElem(new Integer(5));m.addElem(new Integer(5));g.addElem(new Integer(5));
    s.removeElem(new Integer(3));m.removeElem(new Integer(3));g.removeElem(new Integer(3));
    // Ausgabe wie oben für s,m,g
  }
}
```

Sollte folgende Ausgabe produzieren: **!**

```
3     ...Halbgruppe: true      3     ...Monoid: false      3     ...Gruppe: false
3 1   ...Halbgruppe: true      3 1   ...Monoid: true       3 1   ...Gruppe: false
1 5   ...Halbgruppe: true      1 5   ...Monoid: true       1 5   ...Gruppe: true
```

Die Menge $\{3, 1\}$ ist also unter den genannten Bedingungen zwar ein Monoid aber keine Gruppe. **!** Letzteres liegt daran, daß es mit der gewählten Verknüpfung `op` für $max = 6$ kein Inverses zum Element 3 gibt. Die Menge $\{1, 5\}$ erfüllt dagegen in diesem Zusammenhang die Gruppenbedingungen, da hier die 5 invers zu sich selbst ist, da $op(5, 5) = 1$ gilt.