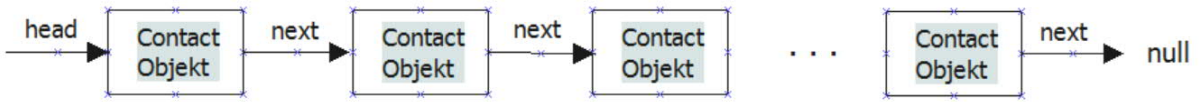
Übung 05: Vernetzung von Objekten, statische Attribute und Methoden

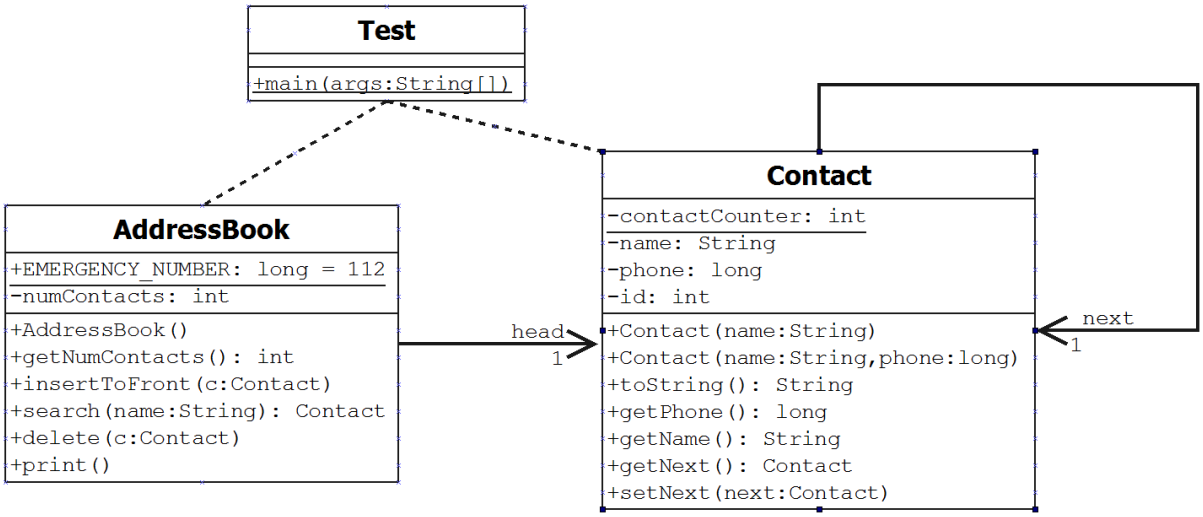
Auf diesem Übungsblatt entwickeln Sie ein einfaches Adressbuch, das in einer ***einfach verketteten Liste*** Adresskontakte verwaltet. Das Programm besteht aus 3 Klassen:

- Die Klasse `Contact` definiert einen Namen, eine Telefonnummer und ***eine interne ID***. Die ID stellt eine fortlaufende, aufsteigende Nummerierung aller bislang erzeugten Objekte dar.
- Die Klasse `Addressbook` stellt Methoden zum Hinzufügen, zum Suchen und zum Entfernen von Kontakten bereit. In der Klasse gibt es nur einen Verweis auf den ersten Kontakt der Liste (`head`).
- Die Klasse `Test` testet die Funktionalität des Adressbuchs und ist bereits vorgegeben.

Ein Adressbuch kennt den Anfang der Liste (`head`). Ferner existiert innerhalb jeder Instanz von `Contact` im Attribut `next` ein Verweis auf den nächsten Kontakt der Liste. Um das Adressbuch zu durchsuchen, muss man also bei `head` beginnen und sich durch die Liste der Kontakte „hangeln“ bis man auf einen `null`-Verweis trifft. Die folgende Abbildung illustriert das Prinzip.



Das folgende UML Klassendiagramm zeigt die Beziehungen zwischen Objekten der Klassen `AddressBook` und `Contact`. Im konkreten Fall bestehen auch ***Assoziationen zwischen Objekten der gleichen Klasse***, nämlich der Klasse `Contact`. Statische Attribute und Methoden sind unterstrichen.



Aufgabe 1

Implementieren Sie die Klasse `Contact` gemäß dem UML Diagramm! Beachten Sie:

- Überlegen Sie, wie man die Assoziation „`next`“ zwischen 2 Kontakten umsetzen kann.
- Erzeugen Sie Getter- und Setter-Methoden in IntelliJ automatisch.
- Es gibt 2 Konstruktoren. Falls die Telefonnummer nicht bekannt ist, so wird die Telefonnummer einfach auf 0 gesetzt.
- `toString()`: Liefert einen String zurück, der das Objekt beschreibt. Der String soll den Namen und die Telefonnummer enthalten.

(umblättern)

Aufgabe 2

Implementieren Sie die Klasse `Addressbook` gemäß dem UML Diagramm. Beachten Sie:

- Das Attribut `numContacts` speichert wie viele Elemente aktuell in der Liste sind.
- `insertToFront(.)` fügt einen Kontakt `c` am Beginn der Liste ein.
- `search(.)` sucht nach einem Kontakt mit einem bestimmten Namen und gibt `null` zurück, falls kein solcher Kontakt in der Liste enthalten ist. *Hinweis:* Verwenden Sie die Methode `equals(.)` der Klasse `String` um 2 Strings auf Gleichheit zu vergleichen.
- `delete(.)` löscht einen Kontakt aus der Liste.
- Die Methode `print()` gibt auf der Konsole alle Kontakte des Adressbuchs (= alle Elemente der Kontaktliste) aus.

Aufgabe 3

a) Versuchen Sie zu verstehen, was innerhalb von `Test.java` gemacht wird.
b) Testen Sie Ihre Implementierung mit der bereits vorgegebenen Klasse `Test`.
c) Wie viele Instanzen / Objekte der Klasse `Contact` werden insgesamt erzeugt?
d) Sind die Klassen `Contact` und `Addressbook` *immutable*?