University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Programmierung für Mathematik HS11

Übung 6

1 Aufgabe: Vererbung

1.1 Lernziele

1. Vererbung verstehen und korrekt implementieren können.
2. Methoden überschreiben können.

1.2 Aufgabenstellung

Passen Sie Ihre Klassen von der Übung 5, Aufgabe 3 so an bzw. ergänzen Sie neue Klassen, dass folgende Problemstellung gelöst wird. Versuchen sie dabei Codeduplizität zu vermeiden, indem Sie Vererbung nutzen.

Eine Universität verwendet Ihr Programm um die Studenten zu speichern. Jetzt will die Universität neuerdings auch das Personal speichern, diese haben natürlich auch einen Namen und ein Geburtsjahr und zusätzlich noch einen Lohn. Die Universität soll die Möglichkeit haben eine Liste mit diesen Angaben des Personals zu drucken. Ausserdem sollen von der Sekretärin wichtige Mitteilungen versendet werden können, welche von dem Personal und den Studenten gleich empfangen werden sollen (wenn die Mitteilung erhalten wurde, soll jeweils von jeder Person der Univerität eine kurze Meldung ausgegeben werden, dass die Nachricht angekommen ist). Um einen besseren Überblick über die Finanzen der Universität zu haben, soll es die Möglichkeit geben, die Summe aller Löhne der Mitarbeiter zu bekommen.

Testen Sie ihr neues Programm mit einem `TestDriver`.

2 Aufgabe: Numbers and digits

2.1 Lernziele

1. Das Wissen über Referenzen vertiefen.
2. Die Verwendung von `while`-Schleifen vertiefen.

2.2 Aufgabenstellung

Wir wollen mittels Java einzelne Ziffern und schliesslich ganze Zahlen, bestehend aus einzelnen Ziffern, implementieren. Wir nutzen dabei aus, dass eine Zahl immer aus einer Reihe oder Kette von Ziffern besteht. Mittels Referenzen wollen wir dabei die Kette dynamisch in der Länge wachsen lassen. Schliesslich wollen das Ergebnis nutzen, um besonders grosse natürliche Zahlen miteinander zu addieren.

a) Die Ziffern

Schreiben Sie eine Klasse `Digit` mit einer Instanzvariable `value` vom Typ `int`, welche nur über einen Konstruktor gesetzt werden kann. Da die Variable `value` nur für eine einzelne Ziffer gedacht ist, soll nur jeweils die letzte Ziffer der übergebenen Zahl gespeichert werden. Schreiben Sie ausserdem eine getter-Methode.

Als nächstes ergänzen wir die Klasse um eine Instanzvariable `leftdigit`, welche später jeweils auf die nächste Ziffer zur Linken verweisen soll. Schreiben sie passende getter- und setter-Methoden für `leftdigit`. (Achten Sie darauf, dass die getter- und setter-Methoden mit Referenzen arbeiten.)

b) Die Zahlen

Als nächstes erstellen wir eine Klasse `Number`. Sie soll eine Instanzvariable `firstDigit` vom Typ `Digit` besitzen und auf das erste Element unserer Ziffernkette zeigen.

Ergänzen Sie die Klasse um eine Methode `addDigit` mit einem Parameter `value` vom Typ `int`. Diese Methode soll bei Aufruf jeweils eine neue Instanz der Klasse `Digit` mit dem als Paramter übergebenen Wert `value` erzeugen. Ist `firstElement` gleich `null`, so wird diese Variable mit dem neuen Objekt initialisiert. Ist dies nicht der Fall, so muss nun über eine while-Schleife das letzte Element der Ziffernkette ermittelt werden und danach das Objekt angefügt werden.

Als letztes brauchen wir noch eine Methode `printNumber`, welche die ganze Ziffernkette als eine lange Zahl auf der Konsole ausgibt. Die Ziffer in der Varialbe `firstElement` soll dabei ganz rechts stehen.

c) Addition mit Zahlen

Als letztes wollen wir versuchen, mit unseren Ziffernketten eine Addition durchzuführen. Wir schreiben dazu eine Methode `addNumber`, welche als Parameter ein Objekt `number` vom Typ `Number` erwartet und als Ergebnis eine neue Zahl, ebenfalls vom Typ `Number` zurück gibt.

Als erstes erstellen wir in der Methode ein neues Objekt `result` vom Typ `Number`, in welcher nachher unser Ergebnis stehen soll.

Danach soll die Methode überprüfen, ob sowohl die eigene Instanzvariable `firstElement` als auch die übergebene Variable `number` und dessen `firstElement` `null` sind. Ist dies der Fall, so soll eine Fehlermeldung ausgegeben werden, andernfalls soll die Addition wie folgt durchgeführt werden:

1. Initialisieren Sie zwei interne Variablen `digit1` und `digit2` vom Typ `Digit` mit `firstElement` und `Number.firstElement`.
2. Fügen Sie der Variable `result` mittels der Methode `addOneDigit` eine neue Ziffer hinzu, welche das Ergebnis aus der Addition von `digit1` und `digit2` sein soll.
3. Initialisieren Sie eine neue Variable `carry` mit dem Übertrag der Addition.
4. Schreiben Sie als nächstes eine `while`-Schleife, welche bei jedem Durchlauf jeweils eine Ziffer nach links geht und `result` mittels `addOneDigit`die Summe von `digit1` und `digit2` plus den Übetrag aus der vorherigen Addition hinzufügt. Achten Sie dabei darauf, dass eine Ziffernkette länger sein kann als die andere.
5. Vergessen Sie nicht, den Übertrag `carry` bei der Addition zu berücksichtigen.

d) TestDriver

Versuchen Sie, mit einem TestDriver folgende Berechnung druchzuführen:

$$999999999999999999999999999999999999999999999999999999 + 1$$

Hinweis: Die Zahl besteht aus 54 mal der Ziffer 9.

3 Aufgabe: Lineare Gleichungssysteme

3.1 Lernziele

1. Gauss-Eliminationsverfahren implementieren können.
2. Selbstständig Ideen entwickeln um ein mathematisches Problem mit Java lösen zu können.

3.2 Aufgabenstellung

Schreiben Sie ein Programm welches das folgende lineare Gleichungssystem (in den unbekannten $x_i, i \in \{1, ..., n\}$) lösen soll.

$$\begin{aligned}
\alpha_{11} \cdot x_1 + \alpha_{12} \cdot x_2 + ... + \alpha_{1n} \cdot x_n &= \beta_1 \\
\alpha_{21} \cdot x_1 + \alpha_{22} \cdot x_2 + ... + \alpha_{2n} \cdot x_n &= \beta_2 \\
&\vdots \\
\alpha_{n1} \cdot x_1 + \alpha_{n2} \cdot x_2 + ... + \alpha_{nn} \cdot x_n &= \beta_n
\end{aligned}$$

oder in Matrizenschreibweise:

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix}$$

Es darf angenommen werden, dass das Gleichungssystem genau eine Lösung besitzt. Verwenden Sie das Gauss-Eliminationsverfahren um den Lösungsvektor zu bekommen.

Falls Sie dazu Fragen haben, können Sie diese im OLAT-Forum stellen.