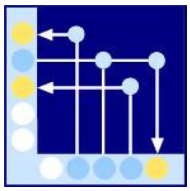
Hochschule Aalen
Fakultät Elektronik und Informatik
Studiengang Informatik



Programmieren 2

Vorlesung im Wintersemester 2014/2015
Prof. Dr. habil. Christian Heinlein

6. Übungsblatt (18. November 2014)

Aufgabe 6: Unterklassen

Implementieren Sie zu der in der Vorlesung vorgestellten Klasse `Account` (vgl. Skript, §4.4) eine Unterklasse `ChargedAccount` zur Repräsentation gebührenpflichtiger Konten!

In einer zusätzlichen privaten Objektvariablen `count` soll die Anzahl der bisher auf diesem Konto ausgeführten und noch nicht in Rechnung gestellten Buchungen gespeichert werden.

Die geerbten Methoden `deposit` und `withdraw` sollen so überschrieben werden, dass bei jeder solchen Buchung die Variable `count` um eins erhöht wird. Zur Ausführung der eigentlichen Buchung soll die entsprechende Methode der Oberklasse `Account` aufgerufen werden.

Eine zusätzliche öffentliche Objektmethode `charge` soll für jede auf dem Konto ausgeführte Buchung einen bestimmten Betrag, der in einer gleichnamigen öffentlichen Klassenvariablen mit Anfangswert 10 Cent gespeichert wird, als Gebühr vom aktuellen Kontostand abziehen und anschließend den Buchungszähler `count` auf null zurücksetzen.

Analog zur Klasse `Account`, sollen öffentliche Konstruktoren angeboten werden, die als Parameter den Kontoinhaber und optional den anfänglichen Kontostand erhalten und jeweils den entsprechenden Konstruktor der Oberklasse aufrufen.

Schreiben Sie zusätzlich eine Testklasse mit einer `main`-Methode, die ein gebührenpflichtiges und ein limitiertes Konto erzeugt, beide in jeweils einer `Account`-Variablen speichert und auf ihnen einige Buchungen ausführt! Anschließend soll für das gebührenpflichtige Konto die Methode `charge` aufgerufen werden (hierfür muss das Konto per Typecast in `ChargedAccount` umgewandelt werden) und der Kontostand beider Konten ausgegeben werden.

Überlegen Sie sorgfältig, welche Methoden bei einer Überweisung von einem gebührenpflichtigen auf ein limitiertes Konto bzw. umgekehrt direkt und indirekt ausgeführt werden, und zeichnen Sie entsprechende Aufrufbäume! (Eventuell sind auch Ausgabeanweisungen zur Kontrolle in den betroffenen Methoden hilfreich.)

Der Code für gewöhnliche und limitierte Konten steht auf der Vorlesungs-Webseite zur Verfügung und kann unverändert übernommen werden.

Lösung

```
// Gebührenpflichtiges Konto.
class ChargedAccount extends Account {
    // Buchungszähler.
    private int count = 0;

    // Gebühr pro Buchung in Cent.
    public static int charge = 10;

    // Neues gebührenpflichtiges Konto mit Inhaber h
    // und ggf. Anfangsbetrag b initialisieren.
    public ChargedAccount (String h) {
        super(h);
    }
    public ChargedAccount (String h, int b) {
        super(h, b);
    }

    // Überschreibungen von deposit und withdraw,
    // die den Buchungszähler erhöhen.
    public void deposit (int amount) {
        super.deposit(amount);
        count++;
    }
    public void withdraw (int amount) {
        super.withdraw(amount);
        count++;
    }

    // Gebühren für ausgeführte Buchungen abbuchen
    // und Buchungszähler zurücksetzen.
    public void charge () {
        super.withdraw(count * charge);
        count = 0;
    }
}

// Test.
class ChargedAccountTest {
    public static void main (String [] args) {
        // Ein gebührenpflichtiges und ein limitiertes Konto erzeugen
        // und beide in Account-Variablen speichern.
        Account ca = new ChargedAccount("Mustermann", 5000);
        Account la = new LimitedAccount("Musterfrau", 3000, 1000);

        // Einige Buchungen ausführen.
        ca.deposit(5000);
        la.deposit(3000);
        ca.withdraw(5000);
        la.withdraw(3000);
        ca.transfer(5000, la);
        la.transfer(3000, ca);
```

```
        // Kontoführungsgebühren abbuchen.
        // (Damit der Compiler den Aufruf der Methode charge erlaubt,
        // muss das Aufrufobjekt den *statischen* Typ ChargedAccount
        // besitzen. Daher ist ein Typecast erforderlich.)
        ((ChargedAccount)ca).charge();

        // Beide Kontostände ausgeben.
        System.out.println("Konto ca: " + ca.balance()); // 2960
        System.out.println("Konto la: " + la.balance()); // 5000
    }
}
```

Aufrufbäume der beiden `transfer`-Aufrufe