

```

package src;

import java.util.Collections;

/**
 * Stellt eine virtuelle Meilenverwaltung dar.
 * Mit dieser koennen Meilenkonten erstellt werden,
 * Meilen hinzugefuegt, transferiert und abgehoben werden.
 * Es gibt zwei Arten von Meilenkonten. Dies sind PinkAccount
 * und PlatiniumAccount.
 * Diese unterscheiden sich in der Anzahl Zusatzmeilen
 * welche es für geflogene Meilen gibt und in der Moeglichkeit,
 * das Meilenkonto zu ueberziehen.
 * Die MeilenVerwaltung (MilageAdmin) stellt eine Uebersicht
 * ueber alle Konten und den Gesamtbetrag der Meilen,
 * welche gesamthaft in den Meilenkonten vorhanden sind.
 */
public class MilageAdmin extends Hashtable implements Observer {
    /**
     * Enthaeht zu jeder Zeit die Summe aller Meilen,
     * welche gesamthaft in den Meilenkonten vorhanden sind.
     * Dieser Gesamtbetrag wird bei Veraenderungen auf einem
     * Konto nachgefuehrt.
     */
    private float currentMilageBalance;

    /**
     * Enthaeht die aktiven Meilenkonten.
     */
    private Hashtable accounts;

    /**
     * Enthaeht die naechste freie Nummer
     * welche einem Meilenkonto gegeben werden kann.
     */
    private int nextMilageAccountNo;

    /**
     * Konstruktor
     * PRE: true
     * POST: true
     */
    public MilageAdmin() {
        currentMilageBalance = 0.0f;
        nextMilageAccountNo = 1;
        accounts = new Hashtable();
    }

    /**
     * Fuegt ein neues Meilenkonto hinzu.
     * Das Meilenkonto darf noch nicht existieren
     * innerhalb der Meilenverwaltung.
     * PRE:
     * POST:
     */
    void addMilageAccount(MilageAccount a) {
        currentMilageBalance += a.balance;
        a.addObserver(this);
        accounts.put(new Integer(a.getAccountNo()), a);
    }

    /**
     * Gibt die Summe der (Meilen)-Saldi aller registrierten Konti zurueck.
     *
     * PRE: true
     * POST: result >= 0
     */
}

```

```

public float getBankBalance() {
    return currentMilageBalance;
}

/**
 * Entfernt ein Meilenkonto.
 * Der Meilensaldo der Meilenbank wird dabei nachgefuehrt.
 * PRE:
 * POST:
 */
public void removeMilageAccount(MilageAccount a) {
    accounts.remove(new Integer(a.getAccountNo()));
    currentMilageBalance -= a.balance;
}

/**
 * Entfernt ein Meilenkonto, welches durch die Nummer accountNo
 * identifiziert wird.
 * Der Meilensaldo der Meilenverwaltung wird dabei nachgefuehrt.
 * PRE:
 * POST:
 */
public void removeMilageAccount(int accountNo) {
    removeMilageAccount(getMilageAccount(accountNo));
}

/**
 * Transferiert den Betrag amount von Meilenkonto srcAccount zu Meilenkonto
 * destAccount.
 * PRE:
 * POST:
 */
public void transferMiles(int srcAccount, int destAccount, float amount) {
    MilageAccount a, b;
    a = getMilageAccount(srcAccount);
    b = getMilageAccount(destAccount);
    a.withdraw(amount);
    b.deposit(amount);
}

/**
 * Fuegt Anzahl Meilen dem Meilenkonto mit der Nummer accountNo hinzu.
 * Der Meilensaldo der Verwaltung wird dabei nachgefuehrt.
 * PRE:
 * POST:
 */
public void depositMiles(int accountNo, float amount) {
    getMilageAccount(accountNo).deposit(amount);
}

/**
 * Hebt vom Meilenkonto mit der Kontonummer accountNo eine
 * bestimmte Menge (amount) Meilen ab
 * Der Meilensaldo der Meilenverwaltung wird dabei nachgefuehrt.
 * PRE:
 * POST:
 */
public void withdrawMiles(int accountNo, float amount) {
    getMilageAccount(accountNo).withdraw(amount);
}

/**
 * Gibt ein bestimmtes Meilenkonto-Objekt mit der Nummer accountNo zurueck.
 * PRE: accountExists(accountNo)
 * POST: result != null
 */
MilageAccount getMilageAccount(int accountNo) {

```

```

        return (MilageAccount) accounts.get(new Integer(accountNo));
    }

    /**
     * Gibt die Anzahl Meilen aus, welche auf einem
     * bestimmten Meilenkonto sind
     * PRE: accountExists(accountNo)
     * POST: result == getAccount(accountNo).balance
     */
    public float getBalance(int accountNo) {
        return ((MilageAccount) accounts.get(new Integer(accountNo))).balance;
    }

    /**
     * Gibt true zurueck, falls ein Meilenkonto mit der
     * gegebenen Nummer accountNo existiert. Sonst false.
     *
     * PRE: true
     * POST: if (exists x : Account in
     *         accounts | x.getAccountNo() == accountNo) then
     *         result==true
     *         else
     *         result==false
     */
    public boolean accountExists(int accountNo) {
        return (accounts.get(new Integer(accountNo)) != null);
    }

    /**
     * Gibt true zurueck, falls das Meilenkonto bereits
     * der Bank zugeordnet ist, ansonsten false.
     * PRE: a != null
     * POST: if (exists x : Account in accounts | x == a) then
     *         result==true
     *         else
     *         result==false
     */
    boolean accountExists(MilageAccount a) {
        return (accountExists(a.getAccountNo()));
    }

    /**
     * Rueckrufschnittstelle eines Kontobeobachters. Konten,
     * deren Anzahl Meilen geaendert werden,
     * melden den alten Saldo ueber diese Methode.
     * Anschliessend wird mittels dem
     * neuen und dem alten Saldo der neue Gesamtsaldo der Bank berechnet.
     *
     * PRE: o != null and arg != null and
     *       o instanceof Account and arg instanceof Float
     * POST: currentAccountBalance =
     *         currentAccountBalance@PRE -
     *         ((Float)arg).floatValue() +
     *         ((Account)o).balance
     */
    public void update(Observable o, Object arg) {
        Float f = (Float) arg;
        currentMilageBalance -= f.floatValue();
        currentMilageBalance += ((MilageAccount) o).balance;
    }

    /**
     * Gibt eine Uebersicht ueber alle Konten aus.
     *
     * PRE: true
     * POST: true

```

```

    */
    public void summarizeBank() {
        Enumeration e = Collections.enumeration(accounts.values());
        while (e.hasMoreElements()) {
            System.out.println(e.nextElement());
        }
        System.out.println("Total amount of Miles:" +
            this.currentMilageBalance);
        System.out.println("*****");
        System.out.println();
    }

    /**
     * Erzeugt ein neues PlatiniumKonto aus einem PinkKonto
     * und fuegt dieses der Menge von Meilenkonten
     * hinzu. Die Meilenkontonummer wird dem Aufrufer zurueckgegeben.
     * PlatiniumKontos können nur als eine Art
     * "Upgrade" von einem Pinkkonto entstehen.
     * Dabei wird der PinkAccount gelöscht und dem neuen Platinium-
     * Account der Saldo des PinkAccount gegeben.
     * Die AccountNummer bleibt die gleiche und wird dem Aufrufer zurückgegeben.
     *
     * PRE:
     * POST:
     */
    public int upgradeAccount(PinkAccount pa) {
        float amount=pa.balance;
        PlatiniumAccount p = new PlatiniumAccount(pa.getAccountNo());
        removeMilageAccount(pa);
        addMilageAccount(p);
        p.deposit(amount);
        return p.getAccountNo();
    }

    /**
     * Erzeugt ein neues PinkAccount und fuegt dieses der Menge von Meilenkonten
    hinzu.
     * Die Meilenkontonummer wird dem Aufrufer zurueckgegeben.
     *
     * PRE: true
     * POST: result == nextAccountNo@PRE and
     *       accountExists(result)
     */
    public int getNewPinkAccount() {
        PinkAccount c = newPinkAccount();
        addMilageAccount(c);
        return c.getAccountNo();
    }

    /**
     * Erzeugt ein neues PinkKonto aus einem PlatiniumKonto
     * und fuegt dieses der Menge von Meilenkonten
     * hinzu. Die Meilenkontonummer wird dem Aufrufer zurueckgegeben.
     * Dabei wird der PlatiniumAccount gelöscht
     * und dem neuen PinkAccount der Saldo des PinkAccount gegeben.
     * Die AccountNummer bleibt die gleiche und wird dem Aufrufer zurückgegeben.
     *
     * PRE:
     * POST:
     */
    public int downgradeAccount(PlatiniumAccount pa) {
        float amount=pa.balance;
        PinkAccount p = new PinkAccount(pa.getAccountNo());
        removeMilageAccount(pa);
        addMilageAccount(p);
        p.deposit(amount);
        return p.getAccountNo();
    }
}

```

```
/**
 * Hilfsmethode zum Erzeugen von PinkAccounts
 * PRE: true
 * POST: result != null
 */
private PinkAccount newPinkAccount() {
    PinkAccount p = new PinkAccount(nextMilageAccountNo);
    nextMilageAccountNo++;
    return p;
}
}
```