TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Computer Vision Group
Einführung in die Informatik I für Ingenieure (MSE)
Prof. Dr. Daniel Cremers, Mohamed Souiai

WS 2012
Übungsblatt 7
07.12.2012

Aufgabe 7.1 (Ü) Parkhaus

Ziel dieser Aufgabe ist es, ein Parkhaus zu simulieren.

a) Implementieren Sie die Klasse `Auto` mit den privaten Attributen `marke` und `kennzeichen`, sowie den beiden Methoden `String getMarke()` und `String getKennzeichen()` und einem sinnvollen Konstruktor.
b) Erweitern Sie die Klasse `Auto` um ein privates Attribut `int fahrgestellnummer` und eine öffentliche Methode `int getFahrgestellnummer()`. Die Fahrgestellnummer soll bei jedem neu erzeugten `Auto`-Objekt um eins erhöht werden. **Hinweis:** Dies funktioniert ähnlich wie bei der Klasse `Count` in der Vorlesung.
c) Implementieren Sie zusätzlich die Methode `String toString()`, die eine String Darstellung des Auto Objektes zurückgibt z.B. **Marke: Opel, Kennzeichen: M4**.
d) Implementieren Sie die Klasse `Parkhaus`, die Parkhäuser mit n Parkplätzen repräsentiert. Dabei soll die Größe n des Parkhauses im Konstruktor übergeben werden.
e) Fügen Sie der Klasse `Parkhaus` eine Methode `int parken(Auto a)` hinzu, bei der das Auto `a` den ersten freien Parkplatz des Parkhauses belegt. Die Methode soll dabei die Nummer des belegten Parkplatzes zurückliefern. Wenn das Parkhaus voll ist, soll `-1` zurückgegeben werden.
f) Erweitern Sie die Klasse `Parkhaus` um die Methode `int suchen(String k)`, die mit Hilfe des Kennzeichens ein Auto im Parkhaus sucht. Ist das Auto mit dem Kennzeichen `k` im Parkhaus, soll die Methode die Nummer des Parkplatzes ausgeben, ansonsten `-1`.
g) Erweitern Sie die Klasse `Parkhaus` um die Methode `Auto wegfahren(String k)`, das den vom Auto mit dem Kennzeichen `k` belegten Parkplatz wieder freigibt und das Auto zurückgibt. Falls das Auto mit Kennzeichen `k` nicht im Parkhaus ist, soll `null` zurückgegeben werden.
h) Implementieren Sie zuletzt die Methode `String toString()`, die einen String zurückgibt zur geeigneten Ausgabe eines Parkplatz Objektes in Form einer Auflistung aller Parkplätze mit den dort geparkten Autos.

Lösungsvorschlag 7.1

Die Klasse Auto:

```
public class Auto {

    private String marke, kennzeichen;
    private int fahrgestellnummer;
    // Klassen-Attribut zaehlt die Anzahl aller erstellten Autos.
    private static int anzahl = 0;

    public Auto(String m, String k) {
```

```
        this.marke = m;
        this.kennzeichen = k;
        anzahl++;
        this.fahrgestellnummer = anzahl;
    }

    public String getKennzeichen() {
        return kennzeichen;
    }

    public String getMarke() {
        return marke;
    }

    public int getFahrgestellnummer() {
        return fahrgestellnummer;
    }

    public String toString() {
        return "Auto " + fahrgestellnummer + ": (Marke: " + marke + ",
            Kennzeichen: " + kennzeichen + ")";
    }
}
```

Die Klasse Parkhaus:

```
public class Parkhaus {

    // Array, das Autos enthaelt
    private Auto[] plaetze;

    // Konstruktor
    public Parkhaus(int groesse) {
        this.plaetze = new Auto[groesse];
    }

    // Auto a parkt auf erstem freien Platz
    public int parken(Auto a) {
        int platz;
        for (platz = 0; platz < plaetze.length; platz++) {
            if (plaetze[platz] == null) { // Platz frei
                plaetze[platz] = a;
                return platz;
            }
        }
        return -1; // kein Platz frei
    }

    // Suche Parkplatz von Auto mit Kennzeichen k
    public int suchen(String k) {
        for (int platz = 0; platz < plaetze.length; platz++) {
            if (plaetze[platz] != null // Parkplatz belegt
                    && plaetze[platz].getKennzeichen().equals(k)) { //
                        richtiges Kennzeichen
                return platz;
            }
        }
        return -1;
```

```
    }

    // Auto mit Kennzeichen k gibt Parkplatz frei
    public Auto wegfahren(String k) {
        int platz = suchen(k);
        if (platz == -1) {
            return null;
        } else {
            Auto auto = plaetze[platz];
            plaetze[platz] = null;
            return auto;
        }
    }

    public String toString() {
        String ret = "Parkhaus:\n";
        for (int platz = 0; platz < plaetze.length; platz++) {
            ret += "[" + platz + ": ";
            if (plaetze[platz] != null) { // Parkplatz belegt
                ret += plaetze[platz];
            } else {
                ret += "_________";
            }
            ret += "]\n";
        }
        return ret;
    }
}
```

Aufgabe 7.2 (H) Komplexe Zahlen

Komplexe Zahlen sind Zahlen der Form :

$$a + b \cdot \mathrm{i} \tag{1}$$

Dabei wird a als Realteil und b als Imaginärteil von $a + b \cdot \mathrm{i}$ bezeichnet.

Implementieren Sie die Klasse `Complex` zur Repräsentation komplexer Zahlen. Definieren Sie neben einem geeigneten Konstruktor für die Klasse auch Objekt-Methoden für folgende Operationen:

a) Realteil ausgeben: `public double real()`
b) Imaginärlteil ausgeben: `public double imag()`
c) Addition mit einer übergebenen komplexen Zahl: `Complex add(Complex r)` . Für die Addition zweier komplexer Zahlen $a + b \cdot \mathrm{i}$ und $c + d \cdot \mathrm{i}$ gilt:

$$(a + b\,\mathrm{i}) + (c + d\,\mathrm{i}) = (a + c) + (b + d)\,\mathrm{i}. \tag{2}$$

d) Subtraktion einer übergebenen komplexen Zahl: `Complex sub(Complex r)`. Analog gilt für die Subtraktion:

$$(a + b\,\mathrm{i}) - (c + d\,\mathrm{i}) = (a - c) + (b - d) \tag{3}$$

e) Multiplikation mit einer übergebenen komplexen Zahl: `Complex mul(Complex r)`. Für die Multiplikation gilt entsprechend

$$(a + b\,\mathrm{i}) \cdot (c + d\,\mathrm{i}) = (ac - bd) + (ad + bc) \cdot \mathrm{i}. \tag{4}$$

f) Multiplikation mit einer übergebenen reellen Zahl: `Complex mul(double r)`
g) Vergleich zwischen 2 komplexen Zahlen: `bool isEqualTo(Complex r)`
h) Eine Methode: `public String toString()`, die die komplexe Zahl als String in der form $a + bi$ zurückgibt.

Lösungsvorschlag 7.2

```
class Complex {

    private double u;
    private double v;

    Complex() {
        u = 0;
        v = 0;
    }

    Complex(double x, double y) {
        u = x;
        v = y;
    }
    // Realteil

    public double real() {
        return u;
    }
    // Imagiaerteil

    public double imag() {
        return v;
    }

    // Addition zweier komplexer Zahlen
    public Complex add(Complex z) {
        return new Complex(u + z.u, v + z.v);
    }
    // Subtraktion zweier komplexer Zahlen

    public Complex sub(Complex z) {
        return new Complex(u - z.u, v - z.v);
    }
    // Multiplikation zweier komplexer Zahlen

    public Complex mult(Complex z) {
        return new Complex(u * z.u - v * z.v, u * z.v + v * z.u);
    }
    // Multiplikation einer komplexen Zahl mit einer " double"-Zahl

    public Complex mult(double x) {
        return new Complex(u * x, v * x);
    }

    public boolean isEqualTo(Complex z) {
        return (z.u == u && z.v == v);
    }
```

```
    public String toString() {
        if (v >= 0) {
            return (u + " + " + v + "i");
        } else {
            return (u + " - " + -v + "i");
        }
    }
}

public class ComplexTest {

    public static void main(String[] args) {

        Complex u, v, z;
        u = new Complex(1, 2);
        System.out.println("u: " + u);
        v = new Complex(3, -4.5);
        System.out.println("v: " + v);
        // Addiere u und v
        z = v.add(u);
        System.out.println("v+u: " + z);
        // Subtraktion u - v
        z = u.sub(v);
        System.out.println("u-v:" + z);
        // Multiplikation u * v
        z = u.mult(v);
        System.out.println("u*v" + z);
        // Multiplikation v * u
        z = v.mult(u);
        System.out.println("v*u" + z);
        // Multiplikation mit
        double x = 5.1;
        z = v.mult(x);
        System.out.println("v*x:" + z);
        if(z.isEqualTo(v))
        System.out.println(z+" is gleich "+v);
    }
}
```