

Übungsblatt 12

Ausgabe: 17.01.2001

Aufgabe 41 (Klassendefinition): 40 Punkte

Implementieren Sie in Java eine Klasse `Roman`, die den Umgang mit römischen Zahlen (mit Werten zwischen 1 und 3999) ermöglicht. Im (für diese Aufgabe definierten) römischen Zahlensystem steht das Symbol I für 1, V für 5, X für 10, L für 50, C für 100, D für 500 und M für 1000. Symbole ergeben hintereinander geschrieben die römische Zahl. Symbole mit größeren Werten stehen dabei normalerweise vor Symbolen mit niedrigeren Werten. Der Wert einer römischen Zahl wird in diesem Fall berechnet durch die Summe der Werte der einzelnen Symbole. Falls ein Symbol mit niedrigerem Wert vor einem Symbol mit höherem Wert erscheint (es darf übrigens jeweils höchstens **ein** niedrigeres Symbol **einem** höheren Symbol vorangestellt werden), errechnet sich der Wert dieses Teils der römischen Zahl als die Differenz des höheren und des niedrigeren Wertes. Die Symbole I, X, C und M dürfen bis zu dreimal hintereinander stehen; die Symbole V, L und D kommen immer nur einzeln vor. Die Umrechnung von Dezimalzahlen in römische Zahlen ist übrigens nicht eineindeutig. So lässt sich z.B. die Dezimalzahl 1990 römisch darstellen als MCMXC bzw. MXM. Beispiele:

```
3999 = MMMCMXCIX
  48 = XLVIII
  764 = DCCLXIV
1234 = MCCXXXIV
  581 = DLXXXI
```

Implementieren Sie folgende Methoden:

- Konvertieren eines String-Objektes, das eine römische Zahl repräsentiert, in einen int-Wert. Sie können davon ausgehen, dass das String-Objekt eine gültige römische Zahl repräsentiert. (8 Punkte)
- Konvertieren eines int-Wertes in einen String, der eine römische Zahl repräsentiert (8 Punkte)
- Initialisieren einer römischen Zahl mit einem String, der eine römische Zahl repräsentiert (Konstruktor) (3 Punkte)
- Initialisieren einer römischen Zahl mit einem int-Wert (Konstruktor) (3 Punkte)
- Initialisieren einer römischen mit einer bereits existierenden römischen Zahl, d.h. anschließend sind die beiden römischen Zahlen wertgleich (siehe Teilaufgabe (h)) (Copy-Konstruktor) (3 Punkte)
- Clonieren einer römischen Zahl, d.h. initialisieren einer neuen römischen Zahl mit einer bereits existierenden römischen Zahl, d.h. anschließend sind die beiden römischen Zahlen wertgleich (siehe Teilaufgabe (h)) (Überschreiben der von der Klasse `Object` geerbten Methode `clone`) (3 Punkte)

- g) Umwandeln einer römischen Zahl in ein String-Objekt (bestehend aus den oben genannten Symbolen) (Überschreiben der von der Klasse `Object` geerbten Methode `toString`) (3 Punkte)
- h) Überprüfen auf Wertegleichheit zweier römischer Zahlen (Überschreiben der von der Klasse `Object` geerbten Methode `equals`) (3 Punkte)
- i) Addition zweier römischer Zahlen (3 Punkte)

Schreiben Sie ein kleines Testprogramm (3 Punkte).

Aufgabe 42 (Vererbung): **20 Punkte**

Gegeben sei folgender Source-Code:

```
class Grossvater {
    int x = 3;
    int y = -4;
}

class Vater extends Grossvater {
    float x = 4.5F;
    int z;
    public Vater(int z) {
        this.z = z;
    }
}

class Sohn extends Vater {
    long a;
    double x = -18.5;
    public Sohn(long a) {
        super(5);
        this.a = a;
    }
}
```

Leiten Sie von der Klasse `Sohn` eine Klasse `Enkel` ab, die eine Methode besitzt, in der die Werte aller Attribute, die ein Objekte der Klasse `Enkel` besitzt, addiert werden und die die Summe auf den Bildschirm ausgibt. Schreiben Sie ein kleines Testprogramm!

Aufgabe 43 (Polymorphie): **20 Punkte**

Gegeben sei folgender Source-Code:

```
interface Spieler {
    public int naechsteGeradeZahl(int aktuelleZahl);
}

public class Zahlenspiel {
    int zahl;
    Spieler a, b;
    public Zahlenspiel(Spieler a, Spieler b) {
        this.a = a;
        this.b = b;
        this.zahl = 0;
    }
}
```

```

public void spielen() {
    Spieler akt = a;
    while (true) {
        int erg;
        if ((erg=akt.naechsteGeradeZahl(this.zahl)) != this.zahl+2){
            System.out.println("Falsch; verloren!"); break;
        }
        this.zahl = erg;
        System.out.println(this.zahl);
        if (akt == a) akt = b;
        else          akt = a;
    }
}

public static void main(String[] args) {
    Spieler a = ...
    Spieler b = ...
    Zahlenspiel spiel = new Zahlenspiel(a, b);
    spiel.spielen();
}
}

```

Hier sollen zwei Spieler jeweils abwechselnd die nächste gerade Zahl berechnen. Implementieren Sie das Interface `Spieler` zweimal; zum ersten für einen menschlichen Spieler, der die Zahl über die Tastatur eingibt, und zum zweiten für ein Programm, das die nächste gerade Zahl berechnet. Ersetzen Sie die Pünktchen in der `main`-Funktion, so dass ein Mensch gegen ein Programm spielen kann.

Aufgabe 44 (PAROB-Spiel): 20 Punkte

Diesmal tun wir etwas für unseren Computergegner. Unser Programm bekommt beigebracht, wie sich „gute“ und „schlechte“ Stellungen unterscheiden.

Erweitern Sie dazu diejenige Ihrer Klassen, die ein PAROB-Spielbrett implementiert, um eine Methode `public int bewerteStellung()`. Diese Methode soll einen Stellungswert liefern, welche die Stellung durch bloßes „Hinsehen“ bewertet. Der Wert soll positiv sein, wenn Spieler Weiß besser steht (je höher desto besser steht Spieler Weiß), und negativ, wenn Spieler Schwarz besser steht (je niedriger desto besser steht Spieler Schwarz), und 0 für eine ausgeglichene Stellung. **Hier ist Ihre Phantasie gefordert!** Im Test gegen andere Programme (nächster Übungszettel) wird sich dann zeigen, wie gut Ihre Bewertung wirklich ist.

Achtung: Bewerten Sie wirklich nur aktuelle Stellungen! In Ihre Bewertungsfunktion soll auf keinen Fall die Analyse nachfolgender Züge mit eingehen! Unter der Annahme, dass die Stellung für einen Spieler umso besser ist, je mehr Figuren er noch auf dem Spielbrett besitzt, würde eine sehr einfache Bewertungsfunktion bspw. die Differenz der sich noch auf dem Spielbrett befindlichen weißen und schwarzen Spielfiguren zurückliefern.