Prof. Dr. Franz Schweiggert   Abteilung Angewandte Informationsverarbeitung   27. Oktober 2004
Dr. Johannes Mayer   Blatt 2

Software Engineering Praxis (WS 2004/2005)

Abgabetermin: 5. November 2004

1 Ich breche, duch brichst, ... Brüche! (5 Punkte)

Implementieren Sie eine eigene Klasse zum Arbeiten mit *gekürzten* Brüchen. Diese Klasse soll Operationen zum Addieren, Subtrahieren, Multiplizieren und Dividieren unterstützen. Außerdem dürfen natürlich auch eine toString()-Methode, sowie eine value()-Methode, die den Wert als double liefert, nicht fehlen.

Beachten Sie bei Ihrer Entwicklung das Prinzip der Kapselung. Mit „Fehlern" brauchen Sie sich (momentan) nicht herumzuschlagen. :-)

2 Frauensache äh Damenprobleme (5 Punkte)

Sie kennen doch sicher das 8-Damen-Problem: Dabei soll man 8 Damen auf einem Schachbrett so positionieren, dass sich keine zwei Damen bedrohen.

Folgende Klasse *Main* löst das Problem für ein Brett der Größe 4 (Parameter `size` von check()).

```
import java.util.Vector;

public class Main {
    public static Figur erzeugeFigur(int x, int y) {
        return new Dame(x, y);
    }

    // versuche anzahl Figuren auf einem size x size Brett zu platzieren
    // (Figuren vom Typ, wie sie erzeugeFigur() anlegt)
    public static boolean positioniere(int size, int anzahl, Vector figuren) {
        if (anzahl == 0)
            return true;
        else {
            // probiere alle Brettpositionen durch
            for (int x = 1; x <= size; x++)
                for_y: for (int y = 1; y <= size; y++) {
                    Figur f = erzeugeFigur(x, y);
                    // teste, ob f mit allen vorher gesetzten Figuren
                    // kombiniert werden kann
                    for (int i = 0; i < figuren.size(); i++) {
                        Figur f1 = (Figur) figuren.get(i);
                        // f1 und f knnen nicht kombiniert werden,
                        // falls beide auf der selben Position sind,
                        // oder eine die andere schlagen wrde
                        if (f1.illegal(f))
                            continue for_y; // nchster Durchlauf von for y=...
                    }
                    // Figur aufnehmen in die Liste der gesetzten Figuren
                    figuren.add(f);
                    // ... und nun rekursiv die restlichen anzahl-1
                    // Figuren auf dem Brett positionieren
                    if (positioniere(size, anzahl-1, figuren)) {
                        // lief alles gut, dann Figur ausgeben und Erfolg
                        // melden
                        System.out.println(""+f);
                        return true;
                    }
                    // andernfalls Figur wieder entfernen und mit
                    // nchster Mglichkeit weiter probieren
                    figuren.remove(figuren.size()-1);
                }
            return false; // keine Lsung mit Ausgangssituation (figuren)
        }
    }

    public static void main(String[] args) {
        if (!positioniere(4, 4, new Vector()))
            System.out.println("Nicht mglich!");
    }
}
```

Wenn es die geeigneten Klassen *Figur* und *Dame* gibt, dann macht die main-Methode von Main folgende Ausgabe:

```
thales$ java Main
Dame(4,3)
Dame(3,1)
Dame(2,4)
Dame(1,2)
thales$
```

(a) (3 Punkte)
Schreiben Sie nun geeignete Klassen *Figur* und *Dame*, so dass obiges Programm funktioniert.

(b) (2 Punkte)
Schreiben Sie nun noch Klassen *Koenig*, *Turm*, *Springer* und *Laeufer*, so dass man auch das 4-König-Problem, das 4-Turm-Problem, etc. mit der Klasse Main lösen kann, indem man nur die Methode erzeugeFigur() abändert.

Hinweis: Die Klasse *Figur* sollte die Position speichern (und diese beim Konstruktor entgegen nehmen). Außerdem sollte Sie Methoden toString(), schlaegt() und illegal() haben – schlaegt() kann man verwenden, um illegal() zu implementieren.

Die Klasse *Dame* muss dann nur mit der Position den Oberklassen-Konstruktor aufrufen und die Methoden schlaegt() und toString() sinnvoll überschreiben.

Den Betrag einer `int` kann man mit der Funktion `Math.abs(.)` berechnen.

Viel Erfolg!