

Wintersemester 2014/2015
Übungen zur Vorlesung
Objektorientierte Softwareentwicklung (BA-INF-024)
Musterlösung zur Online-Aufgabe 2

Aufgabe 1 (*Damenproblem: 20 Punkte - (2+4+7+7) Punkte*)

Schon im Jahre 1850 wurde u. a. von C. F. GAUSS die folgende Aufgabe betrachtet:

Man finde eine Stellung für acht Damen auf einem Schachbrett, so dass keine zwei Damen sich gegenseitig schlagen können.

Eine Dame kann im Schachspiel beliebig viele Spielfelder horizontal, vertikal oder diagonal laufen. Die Damen sind also so zu platzieren, dass jede Zeile, jede Spalte und jede Diagonale des Schachbretts höchstens eine Dame enthält.

In dieser Aufgabe soll nun der Rahmen für eine mögliche Lösung des n -Damenproblems geschaffen werden, d. h. auf einem $n \times n$ Felder großen Schachbrett sollen n Damen so platziert werden, dass sie sich nicht gegenseitig schlagen können. Vorgegeben seien die folgenden Teile der JAVA-Klasse `Damenproblem`:

```
public class Damenproblem {
    private int n;
    private int[] [] feld;

    public Damenproblem (int d) {
        ...
    }
    public void ausgabe() {
        ...
    }
    public boolean korrektPlatziert() {
        ...
    }
    static public void main (String[] arg) {
        Damenproblem d0k = new Damenproblem (4);
        d0k.feld[1][0] = 1;
        d0k.feld[3][1] = 1;
        d0k.feld[0][2] = 1;
        d0k.feld[2][3] = 1;
        if (d0k.korrektPlatziert()) {
            System.out.println ("d0k ist eine moegliche Loesung!");
        }
        else {
            System.out.println ("d0k ist keine erlaubte Loesung!");
        }
        d0k.ausgabe();
    }
}
```

Das 2-dimensionale Array `feld` speichert eine mögliche Platzierung der Damen. Der Wert 0 soll bedeuten, dass das Feld leer ist, der Wert 1 soll bedeuten, dass das Feld mit einer Dame besetzt ist.

a) 3 Punkte

Implementieren Sie den Konstruktor, der für eine übergebene Größe d das Spielfeld initialisiert (d. h. jedes Feld mit dem Wert 0 belegt und die Dimension in der Variablen n speichert).

Musterlösung:

```

public Damenproblem (int d) {
    feld = new int[d][d];
    n = d;
    anzahlLoesungen=0;
}

```

b) Implementieren Sie die Methode **ausgabe**, die mit Hilfe der Befehle `System.out.print` und `System.out.println` das Spielfeld auf der Konsole ausgibt. Die in `main` aufgebaute Stellung soll z.B. wie folgt ausgegeben werden:

```

0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0

```

Musterlösung:

```

public void ausgabe() {
    for (int row=0;row<n;row++) {
        for (int col=0;col<n-1;col++) {
            System.out.print(feld[row][col] + " ");
        }
        System.out.println(feld[row][n-1]);
    }
    System.out.println();
}

```

c) Implementieren Sie die Methode **korrektPlatziert**, die `true` zurückliefert, wenn auf dem $(n \times n)$ -Spielfeld n Damen so platziert sind, dass sie sich nicht gegenseitig schlagen können, sonst `false`.

Musterlösung:

```

public boolean istBedroht(int row, int col) {
    // Betrachtung der Bedrohungen "von links" genügt
    for (int i=0;i<col;i++) {
        // Bedrohung durch Dame in derselben Reihe
        if (feld[row][i]==1) return true;
        // Bedrohung durch Dame auf "negativer" Diagonalen
        if ((row-i-1)>=0 && feld[row-i-1][col-i-1]==1) return true;
        // Bedrohung durch Dame auf "positiver" Diagonalen
        if ((row+i+1)<n && feld[row+i+1][col-i-1]==1) return true;
    }
    for (int i=0;i<n;i++) {
        // Bedrohung durch Dame in derselben Spalte
        if ((i!=row)&&(feld[i][col]==1)) return true;
    }
    return false;
}

```

```

public boolean korrektPlatziert() {
    int queens=0;
    for(int col = 0; col < n; col++){
        for(int row = 0; row < n; row++){
            if(feld[row][col] == 1){
                queens++;
                if(istBedroht(row,col)) return false;
            }
        }
    }
    if (queens!=n) return false;
    return true;
}

```

d) Schreiben Sie eine Methode `platziere`, welche auf dem $n \times n$ -Felder grossen Spielfeld n Damen so aufstellt, dass sie sich nicht gegenseitig schlagen können, und die alle möglichen Lösungen mit Hilfe der Ausgabemethode auf der Konsole ausgibt.

Prinzipiell können Sie die Methode `platziere` beliebig implementieren. Hier aber noch ein paar Tipps, wie eine solche Lösung aussehen könnte:

- Die übliche Lösung benutzt die Signatur `void platziere(int i)`. Mit dieser Methode wird in der i -ten Spalte des Spielfelds eine Dame platziert. Anschließend wird *rekursiv* `platziere(i+1)` aufgerufen.
- Innerhalb einer Spalte stehen n Positionen (bzw. Zeilen) für die Platzierung einer Dame zur Verfügung. Trivialerweise kann man hier stets in der ersten Zeile anfangen und dann der Reihe nach die anderen Positionen besetzen.
- Die geschickte Kombination der beiden Ansätze führt zu einer Lösung des Problems.
- Geben Sie während der Programmentwicklung nach jeder Platzierung einer Dame das Spielfeld aus, damit Sie die Funktionsweise Ihres Algorithmus am Bildschirm überprüfen können.
- Wann kann der Algorithmus testen, ob die Damen auf dem Spielfeld korrekt platziert sind?
- Wann soll der Algorithmus eine Lösung ausgeben?

Musterlösung:

```
// in main-Funktion aufzurufen mit platziere(0)
```

```
void platziere(int col) {  
    if (col==n) {  
        anzahlLoesungen++;  
        ausgabe();  
    }  
    else {  
        for (int row=0;row<n;row++) {  
            if (!istBedroht(row,col)) {  
                feld[row][col]=1;  
                platziere(col+1);  
            }  
            feld[row][col]=0;  
        }  
    }  
}
```

```
...
```

```
// Beispiel für main-Funktion zum Testen
```

```
static public void main (String[] arg) {  
    Damenproblem d0k = new Damenproblem(8);  
    d0k.platziere(0);  
    System.out.print("Es gibt " + d0k.anzahlLoesungen + " Lösungen für das ");  
    System.out.println(d0k.n + "-Damenproblem.");  
}
```