Andreas Podelski
Fachbereich 14 – Informatik
Universität des Saarlandes,
Max-Planck-Institut für Informatik

Programmiersprachen (WS 1996/97)

23. Januar 1997

Übungsblatt 12

Abdelwaheb Ayari
Hubert Baumeister

Aufgabe 1: (3 Punkte)
Implementiere die Suchverfahren Breitensuche, Tiefensuche und Bestensuche in Java (vergleiche Übungsblatt 8). Ein `Searcher` versteht die Nachrichten `startProblem`, die die Ausgangskonfiguration setzt und `nextSolution`, die die nächste Lösung berechnet und ausgibt. Die Nachricht `currentProblem` liefert die Konfiguration, die gerade aktuell getestet wird.
Problemkonfigurationen verstehen die Nachrichten `isSolution`, die `true` liefert, falls sie eine Lösung darstellen und `false` sonst. Die Nachricht `next` liefert die Nachfolgekonfigurationen einer Konfiguration. Ein `Searcher` soll Beobachter zulassen und sie benachrichtigen, wenn eine neue Konfiguration untersucht wird oder wenn eine Lösung gefunden wurde.
Schreibe ein Testprogramm, das die ersten 10 Lösungen des Beispielproblems vom 8. Übungsblatt, mittels Tiefen-, Breiten- und Bestensuche, und die dabei besuchten Konfigurationen ausgibt.

Aufgabe 2: (3 Punkte)
Implementiere Klassen, die Konfigurationen des "Wege des Springers" Problems repräsentieren, damit sie mit dem Suchverfahren aus Aufgabe 1 gelöst werden können.
Gegeben sei ein $n \times n$ Schachbrett. Von einem beliebigen Startpunkt aus soll sich ein Springer so über das Brett bewegen, daß er auf jedem Feld des Schachbretts einmal gewesen ist und kein Feld zweimal besucht hat. Ein Springer kann sich von seinem Ausgangspunkt zwei Felder in einer Richtung und ein Feld in der Richtung senkrecht zur ersten Richtung bewegen.

1. Implementiere die Methode `toString`, die eine Lösungskonfiguration des Problems als String zurückgibt.
2. Gib eine Lösung für $n = 8$ und für $n = 11$ an.

Hinweis: Man verwendet Bestensuche, wobei eine Konfiguration besser ist als die andere, wenn der Weg länger ist, oder wenn beide Wege gleich sind, die Anzahl möglicher Nachfolgekonfigurationen kleiner ist.

Aufgabe 3: (4 Punkte)
Implementiere eine Methode `paint(Graphics g)`, die eine Konfiguration für das "Wege des Springers" Problem zeichnet (s.a. Beispiel) und eine Methode `Dimension size()`, die die Größe der Zeichnung zurückgibt.

Implementiere ein Applet für das Suchen von Lösungen des Springer Problems. Mit Hilfe der Benutzerschnittstelle ist es möglich n zwischen 4 und 15 zu wählen. Wird auf das Schachbrett mit der Maus geklickt, wird die nächste Lösung berechnet und angezeigt. Die dabei von dem Suchverfahren getesteten Konfigurationen sollen nacheinander ohne Pause angezeigt werden.
Hinweis: Auf den Webseiten zur Vorlesung gibt es den Source Code für das MVC Beispiel aus der Vorlesung.
Hier ist eine Beispielausgabe für eine Konfiguration des Springer Problems, die eine Sackgasse darstellt:

	30	9	32	15			48
8	21	6	23	4	47		
29	10	31	14	33	16	49	
20	7	22	5	24	3	46	
11	28	13	34	17	50	43	52
36	19	38	25	40	53	2	45
27	12	35	18		44	51	42
	37	26	39	54	41		1