34. Bundeswettbewerb Informatik – Junioraufgabe 2

Team „ByteSector“

Lösungsidee

Die Aufgabe besteht darin, zu prüfen, ob es möglich ist, mit den Einschränkungen, die Kassiopeia in ihrer Bewegung hat, alle weißen Felder zu betreten, ohne dabei ein schwarzes überqueren zu müssen. Da sich Kassiopeia nur horizontal und vertikal in direkt an ihre aktuelle Position angrenzende Felder bewegen darf, bedeutet dies, dass sie nicht in Bereiche vordringen kann, die durch schwarze Felder komplett von Kassiopeias Startposition abgeschnitten sind oder deren Verbindung lediglich über eine Diagonale besteht. Somit kann sie nur dann alle Felder betreten, wenn sich alle weißen Felder in einem zusammenhängenden Bereich befinden.

Um dies zu überprüfen wird eine Methode angewandt, die an den Farbeimer aus Bildbearbeitungsprogrammen erinnert: Es werden einfach alle direkt erreichbaren Felder ausgehend von Kassiopeias Startposition markiert und anschließend geprüft, ob noch nicht-markierte Felder existieren. Ist dem so, liegen diese in einem abgeschlossenen Bereich und sind unerreichbar.

Umsetzung

Die Umsetzung erfolgt als ein in Java geschriebenes Programm, welches als Jar-Datei exportiert wurde und zusammen mit einer dazugehörigen Batch-Datei dieser Dokumentation beiliegt. Getestet wurde das Programm unter Windows 7 Home Premium SP1 64-Bit mit JDK 1.8.0_65 64-Bit.

Die folgenden Funktionen sind besitzen den Zugriffsmodifikator *protected*, da sie in Aufgabe 1 wiederverwendet werden.

Der erste Aufgabe des Programmes besteht darin, die Textdatei, welche die Daten über Quadratien beinhaltet, auszulesen. Dazu dienen die folgenden Funktionen *getMapFile* und *parseMapfile*. Erstere fragt den Benutzer nach dem Pfad zur Datei, während die zweite diese liest und die darin enthaltenen Informationen über die Beschaffenheit von Feldern und Kassiopeias Startposition in entsprechenden Variablen speichert.

```
// Die Karte wird in Form eines zweidimensionalen Boolean-Arrays gespeichert.
// Schwarze Felder werden mit 'true', weiße mit 'false' wiedergegeben.
boolean[][] map;
// Kassiopeias Startposition wird in zwei Integern für ihre Abszisse (X) und Ordinate (Y) aufgeteilt.
int kassX, kassY;

/** Erfragt die einzulesende Datei **/
protected File getMapfile(){
	// Das Objekt zum Anzeigen einer Dateiabfrage wird erstellt.
	JFileChooser fileChooser = new JFileChooser();
	// Als Standardordner wird das Benutzerverzeichnis festgelegt.
	// Aus diesem kann aber selbstverständlich frei navigiert werden.
	fileChooser.setCurrentDirectory(new File(System.getProperty("user.home")));
	// Schließlich wird der Dialog geöffnet...
	int result = fileChooser.showOpenDialog(null);
	// und die Eingabe - sofern vorhanden - weitergegeben.
	if (result == JFileChooser.APPROVE_OPTION)
		return(fileChooser.getSelectedFile());
```

```
		else
			// Sollte keine Eingabe getätigt worden sein, wird das Programm beendet.
			System.exit(0);

		// Rein formale Wertrückgabe, wird nie erreicht.
		return null;
	}
	/** Liest die eingegebene Datei aus und speichert die Informationen in die entsprechenden Variablen **/
	protected void parseMapfile(File mapfile){
		// Die Beispieldatei wird zeilenweise eingelesen und die Zeilen in einer Liste aus Strings zwischengespeichert.
		List<String> lines = null;
		try {
			lines = Files.readAllLines(mapfile.toPath());
		} catch (IOException e) {
			e.printStackTrace();
			System.exit(0);
		}
		// Es wird davon ausgegangen, dass gültige Beispieldateien eingegeben werden, weshalb außer des erforderlichen
		// Abfangens einer IO-Ausnahme keine Sicherung eingebaut wird.

		// Die Breite und Höhe werden ausgelesen, indem die erste Zeile am Leerzeichen
		// geteilt und die beiden Zahlen konvertiert werden.
		int width = Integer.parseInt(lines.get(0).split(" ")[1]),
				height = Integer.parseInt(lines.get(0).split(" ")[0]);
			// Der Karten-Array wird mit den nun bekannten Werten für Breite und Höhe initialisiert.
		map = new boolean[width][height];

		// Es wird durch alle Zeichen ab der zweiten Zeile iteriert.
		for (int i = 0;i < width; i++)
			for (int j = 0; j < height; j++){
				// Es wird beim Auslesen der Zeichen der Index der Zeile um eins korrigiert, um die erste Zeile zu überspringen
				char curChar = lines.get(j+1).charAt(i);

				// Überprüfen, ob es sich bei dem aktuellen Zeichen um Kassiopeias Position handelt und diese ggf. notieren
				if (curChar == 'K'){
					kassX = i;
					kassY = j;
				}
				// Schwarze Felder, gekennzeichnet durch eine Raute, werden als Boolean
				// mit Wert 'true' gespeichert, weiße Felder mit 'false'.
				map[i][j] = (lines.get(j+1).charAt(i) == '#');
			}


	}
```

Nach dem Import der Daten über Quadratien werden alle direkt erreichbaren weißen Felder von Kassiopeias Position aus schwarz gefärbt. Dies wird rekursiv durch eine Funktion gelöst, die prüft, ob ein gegebenes Feld weiß ist und bei erfüllter Bedingung das schwarz färbt und die Funktion für alle anliegenden vier Felder aufruft.

```
/** Färbt alle aus gegebener Position direkt erreichbaren Felder schwarz. **/
protected void floodFill(int x, int y){
	// Falls aktuelles Feld weiß ist...
	if (!map[x][y]){
		// ... schwarz färben und Füll-Kommando an direkt angrenzende Felder weitergeben.
		map[x][y] = true;
		floodFill(x + 1, y);
		floodFill(x - 1, y);
		floodFill(x, y + 1);
		floodFill(x, y - 1);
	}
}
```

Die obige Funktion *floodFill* wird erstmals mit den Koordinaten von Kassiopeia von der Funktion *checkAccess* aufgerufen, welche anschließend überprüft, ob noch weiße Felder vorhanden sind und das Endergebnis zurückliefert.

```
/** Überprüft, ob Kassiopeia alle Felder erreichen kann **/
	protected boolean checkAccess(){
	// Alle Felder, die Kassiopeia erreichen kann, sind auch für den Floodfill erreichbar.
	floodFill(kassX, kassY);

	// Sind nach dem Floodfill noch Felder weiß, sind diese nicht durch einfache horizontale und vertikale Bewegungen
	// erreichbar und es ist Kassiopeia unmöglich, alle Felder zu betreten.
	boolean whiteLeft = false;
	loopX: // Äußere Schleife wird benannt, um sie später abbrechen zu können.
	for (int i = 0;i < map.length; i++)
		for (int j = 0; j < map[0].length; j++)
			if (!map[i][j]){
				whiteLeft = true;
				break loopX; // Sobald ein weißes Feld gefunden wurde, kann die Suche abgebrochen werden.
			}

	return !whiteLeft;
}
```

Letztendlich wird das von *checkAccess* gelieferte Ergebnis innerhalb des Konstruktors der Klasse durch einen MessageDialog ausgegeben:

```
String result;
if (checkAccess())
	result = "Kassiopeia kann alle Felder erreichen.";
else
	result = "Kassiopeia kann NICHT alle Felder erreichen.";

JOptionPane.showMessageDialog(null, result, "Ergebnis", JoptionPane.INFORMATION_MESSAGE);
```

BEISPIELE

Beispiel-Welt	Ausgabe
kassiopeia0	Kassiopeia kann alle Felder erreichen.
kassiopeia1	Kassiopeia kann NICHT alle Felder erreichen.
kassiopeia2	Kassiopeia kann alle Felder erreichen.
kassiopeia3	Kassiopeia kann alle Felder erreichen.
kassiopeia4	Kassiopeia kann alle Felder erreichen.
kassiopeia5	Kassiopeia kann alle Felder erreichen.
kassiopeia6	Kassiopeia kann alle Felder erreichen.
kassiopeia7	Kassiopeia kann alle Felder erreichen.