Junioraufgabe: Baulwürfe

Team-ID: 00357

Team: SRZ Info 4

Bearbeiter/-innen dieser Aufgabe:
Vincent Eichhorn

Inhaltsverzeichnis

Lösungsidee

Die Lösungsidee bestand darin die Baulwurf-Karte in ein zweidimensionales Array mit booleschen Werten zu zerlegen. Dann sollte die Karte iterativ mit einem Schema eines Baulwurf-Baus im gleichen Datenformat vergleichen werden. Wenn eine Übereinstimmung gefunden wurde soll eine Zählvariable ums eins erhöht werden.

Umsetzung

Das Programm sollte prozedural aufgebaut werden. Erster Schritt war es in der Main-Methode die Eingabe in ein zweidimensionales Array mit booleschen werden zu übertragen. Der Wert „false“ steht hierbei für ein leeres Feld und der Wert „true“ für ein „X“ in der Karte. Über die ersten beiden Zeilen in den Kartentextdokumenten kann dabei die Größe des Arrays bestimmt werden. Mit der Mehtode `newMap(int widht, int height)` kann hierbei ein leeres Karten-Array erzeugt werden. Jedes Feld in der Karte besitz nun den Wert „false“. Über eine Iteration mit den Werten der Größe der Karte kann nun Zeile für Zeile und Zeichen für Zeichen die Eingabe in Text-Form in das zweidimensionale boolesche Array übertragen werden. Mit der Methode `countBaulwürfe(boolean[][] map)` soll nun die Anzahl der Baulwurf-Baue berechnet werden. Diese Methode geht dabei iterativ über jedes Feld in der Karte und vergleicht den nach rechts unten liegenden 3x4-Teil mit einem Schema der Baulwürfe. Wenn der Vergleich erfolgreich ist, dann wird eine Zählvariable ums eins erhöht. Wenn die Karte nicht mehr die Abmessungen des Schemas entspricht ist der Vergleich ebenfalls fehlerhaft. Die Methode `compare(boolean[][] part, boolean[][] schema)` vergleicht hierbei die Teilkarte und das Schema. Die Teilkarte kann mit der Methode `getPart(boolean[][] map, int x, int y)` erstrellt werden.

Beispiele

```
24
6
XXX              X
X X   X    XXX       X
X X        X XXXX
XXX        X XX X      X
     X     XXXX X
              XXX
```

Diese Karte erbringt nach dem Programmaufruf eine Baulwurf-Anzahl von 3. Auf der Karte sind auch 3 Baue zu finden. Da angegeben ist, dass sich die Baulwurf-Baue niemals überlappen muss dieser Spezialfall nicht betrachtet werden. Die einzige Möglichkeit ist dass das zwei Baue genau nebeneinander liegen, jedoch kann das Programm damit umgehen.

Quellcode

```java
/**
 * Zählt die Baulwürfe auf einer gegebenen Karte
 * @param map - Karte aus 2d boolean array
 * @return Anzahl der Baulwürfe als int
 */
public static int countBaulwürfe(boolean[][] map) {
    int n = 0;
    //Iterativ auf Bauwurf-Struktur prüfen
    for(int i = 0; i < map.length; i++) {
        for(int j = 0; j < map[0].length; j++) {
            if(compare(getPart(map, j, i), schema)) {
                n++;
            }
        }
    }
    return n;
}

/**
 * Vergleicht eine Karte mti einem gegeben Vergleichsschema
 * @param part - Karte
 * @param schema - Vergleichsschema
 * @return Übereinstimmung true
 */
public static boolean compare(boolean[][] part, boolean[][] schema) {
    //Wenn Größe von part und schema nicht passt: abbrechen return false
    if(part.length != schema.length || part[0].length != schema[0].length) return
false;

    //Vergeleichen von part und schema
    for(int i = 0; i < part.length; i++) {
        for(int j = 0; j < part[0].length; j++) {
            //Wenn Unstimmigkeit gefunden: abbrechen return false
            if(part[i][j] != schema[i][j]) return false;
        }
    }

    //part und schema stimmen überein
    return true;
}
```

```java
/**
 * Schneidet einen 3x4 oder kleineren Teil aus einer Karte an angegebener Punktkoordinate raus.
 * @param map - Urpsungskarte
 * @param x - Spaltenindex des oberen linken Feldes
 * @param y - Reihenindex des oberen linken Feldes
 * @return Karte mit den Abmessungen 3x4 oder kleiner
 */
public static boolean[][] getPart(boolean[][] map, int x, int y) {
    //Endkoordinate für Reihe berechnen
    int endY = y + 4;
    if(endY >= map.length ) endY = map.length;
    //Alle Reihen in berech y:endY heraustrennen
    boolean[][] part = Arrays.copyOfRange(map, y, endY);
    //Endkoordinate für Spalte berechnen
    int endX = x + 3;
    if(endX >= map[0].length) endX = map[0].length;
    //Jeder Reihe nicht genutze Spalten wegschneiden
    for(int i = 0; i < part.length; i++) {
        part[i] = Arrays.copyOfRange(part[i], x, endX);
    }
    return part;
}
```