

```

1 class Node {
2   int x, y;
3   Node(int x, int y) { //constructor method
4     this.x = x;  this.y = y;
5   }
6 }

```

*Depth-first-Variante (mit Stack):*

```

7 void floodFill(ImageProcessor ip, int x, int y, int label) {
8   Stack s = new Stack();
9   s.push(new Node(x,y));
10  while (!s.isEmpty()){
11    Node n = (Node) s.pop();
12    if ((n.x>=0) && (n.x<width) && (n.y>=0) && (n.y<height)
13        && ip.getPixel(n.x,n.y)==1) {
14      ip.putPixel(n.x,n.y,label);
15      s.push(new Node(n.x+1,n.y));
16      s.push(new Node(n.x,n.y+1));
17      s.push(new Node(n.x,n.y-1));
18      s.push(new Node(n.x-1,n.y));
19    }
20  }
21 }

```

*Breadth-first-Variante (mit Queue):*

```

22 void floodFill(ImageProcessor ip, int x, int y, int label) {
23   LinkedList q = new LinkedList(); // Queue
24   q.addFirst(new Node(x,y));
25   while (!q.isEmpty()) {
26     Node n = (Node) q.removeLast();
27     if ((n.x>=0) && (n.x<width) && (n.y>=0) && (n.y<height)
28         && ip.getPixel(n.x,n.y)==1) {
29       ip.putPixel(n.x,n.y,label);
30       q.addFirst(new Node(n.x+1,n.y));
31       q.addFirst(new Node(n.x,n.y+1));
32       q.addFirst(new Node(n.x,n.y-1));
33       q.addFirst(new Node(n.x-1,n.y));
34     }
35   }
36 }

```

**Programm 11.1.** *Flood Filling* (Java-Implementierung). Die *Depth-first*-Variante verwendet als Datenstruktur die Java-Klasse `Stack` mit den Methoden `push()`, `pop()` und `isEmpty()`.