252-0027-00: Einführung in die Programmierung I

Simulation Programmierprüfung

Dies ist kein offizielles Eprog Dokument und es wurde nicht vom EProg-Team auf Korrektheit überprüft. Es dient dem Zweck in der Übungsstunde eine 60-minütige Programmierprüfung zu simulieren. Aufgaben 1 und 2a sollten lösbar sein in dieser Zeit. Aufgabe 2b ist schwieriger.

Aufgabe 1: Grösster rechter Wert

Der grösste rechte Wert eines Elements in einem Array, ist der grösste Wert, der sich rechts von diesem Element im Array befindet.
Implementieren sie die Methode `Task1.rightDistance(int[] input)`. Die Methode soll für jeden Wert im Array den Abstand zu seinem grössten rechten Wert zurückgeben. Falls der grösste rechte Wert mehrfach vorkommt, dann sollte der kürzeste aller Abstände zurückgegeben werden. Das allerletzte Element im Array hat keinen grössten rechten Wert, deshalb sollten Sie da immer `-1` zurückgeben. All diese Werte sollten in einem int[] zurückgegeben werden. Sie können davon ausgehen, dass input nicht `null` ist und mindestens `1` Element enthält.

Als Beispiel: für die Eingabe `[1, 1, 1, 5, 4]` sollte die Methode `[3, 2, 1, 1, -1]` zurückgeben.

Tests finden Sie in der Datei "Task1Test.java".

Aufgabe 2: Multiplan-Kamera

Eine Multiplan-Kamera bzw. Mehrfachebenen-Kamera ist ein Kamerasetup, welches in der ersten Hälfte des 20. Jahrunderts zur Aufnahme von Trickfilmen eingesetzt wurde. Dabei wird auf den Boden ein Hintergrund gelegt, darauf ein Gerüst mit mehreren Ebenen gestellt und die Kamera wird so befestigt, dass sie von einer beliebigen Ebene von oben nach unten ein Bild aufnimmt. In die verschiedenen Ebenen können Glasplatten mit Bildern oder Zeichnungen gelegt werden. Diese Bilder überdecken dadurch den Hintergrund und die Platten darunter während die transparenten Glasflächen rund um die Bilder und Zeichnungen die Ebenen darunter durchscheinen lassen.

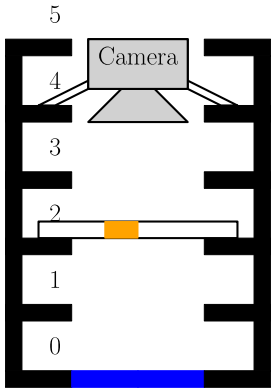

Figure 1: eine Multiplan-Kamera mit nur 6 Ebenen von der Seite. Der Hintergrund ist blau. Auf Ebene 2 befindet sich eine Glasplatte mit ein paar orangen Pixeln. Das Kamerbild - geschossen aus Ebene 4 - ist also blau mit einem orangen Fleck.

In dieser Aufgabe arbeiten Sie mit drei Klassen:

- **Pixel**: Jeder Pixel hat einen rot-, grün- und blau-Wert mit Zahlen zwischen 0 und 255. Rot wird zum Beispiel durch einen `Pixel(255,0,0)` dargestellt. Diese Klasse dürfen Sie NICHT verändern.
- **ImageFrame**: Das ImageFrame repräsentiert eine 2D-Matrix von Pixeln. Die Pixel werden im Pixel-Array `frame` gespeichert und Sie dürfen davon ausgehen, dass jedes Element im Array ein unabhängiges Pixel-Objekt ist, dass nur genau an dieser Stelle in genau diesem ImageFrame vorkommt. Sie müssen auch sicherstellen, dass diese Bedingung immer gilt. Sie dürfen diese Klasse bearbeiten (neue Felder/Methoden hinzufügen, bestehende Methoden/Konstruktoren bearbeiten), aber verändern Sie NICHT die Signatur, Namen oder Typen von bestehenden Konstruktoren, Methoden oder Feldern.
- **MultiplaneCamera**: Diese Klasse repräsentiert das Multiplan-Kamera-Setup mit 11 Ebenen (nummeriert mit Zahlen von 0 bis 10). In jede Ebene können teilweise bemalte Glasplatten reingelegt werden. In unserem Setup können Glasplatten nicht entfernt werden. Falls eine Ebene leer ist, so kann die Kamera stattdessen temporär da angebracht werden, um ein Bild aufzunehmen.

Aufgabe a)
Implementieren Sie die Methoden `fillRectangle` und `drawDiagonal` der Klasse `ImageFrame`. Die Dimensionen eines ImageFrames ist gegeben durch das Pixel-Array `frame`. Sie dürfen davon ausgehen, dass es sich dabei immer um eine rechteckige Matrix handelt, welche mit nicht-null Pixeln gefüllt ist.

- `fillRectangle(int x0, int y0, int x1, int y1, int r, int g, int b)`:
  Diese Methode soll das Rechteck von (x0,y0) bis und mit (x1,y1) mit der Farbe (r,g,b) einfärben.
- `drawDiagonal(int x0, int y0, int r, int g, int b)`:
  Diese Methode soll eine Diagonale mit Startpunkt (x0,y0) und Farbe (r,g,b) zeichnen. Das heisst die Pixel an Position (x0,y0), (x0+1, y0+1), (x0+2, y0+2), ... sollen eingefärbt werden, so dass die Diagonale bis an den Rand des Pixel-Arrays reicht.

Sie dürfen davon ausgehen, dass die Methode nur mit x- und y-Werten aufgerufen wird, welche innerhalb der Matrix liegen und r,g und b Werte von 0 bis 255 haben.

Aufgabe b) (Challenge)
Implementieren sie die Methoden `insertPlane` und `getPhoto` der Klasse `MultiplaneCamera`.

- `insertPlane(int height, int x0, int y0, int x1, int y1, int r, int g, int b)`:
  Diese Methode soll eine Glasplatte mit einem Rechteck oder einer Diagonale auf Ebene `height` in die Multiplan-Kamera einfügen. Der Rest der Glasplatte ist jeweils vollständig transparent. Falls es sich um eine Diagonale handelt, so ist $x1 = y1 = -1$. Sie dürfen davon ausgehen, dass die x- und y-Werte innerhalb der Matrix liegen (ausser im zuvor genannten Fall) und r,g und b Werte von 0 bis 255 haben. Die Rechtecke und Diagonalen funktionieren sonst analog zu Teilaufgabe a. Die Höhe `height` kann Werte von 0 bis 10 annehmen. Die neue Platte soll nur eingefügt werden, wenn sich nicht bereits eine Platte auf der spezifizierten Höhe befindet. Anderenfalls soll der Methodenaufruf einfach "ignoriert" werden - also keinen Effekt haben.
- `ImageFrame getPhoto(int height)`:
  Die Kamera wird temporär auf Höhe `height` platziert und soll eine Aufnahme nach unten machen. Rechtecke und Diagonalen auf den platzierten Ebenen können dabei die Figuren auf den darunterliegenden Ebenen und den Hintergrund (teilweise) überdecken - ein Pixel an Position (x,y) auf Höhe h überdeckt einen Pixel an derselben Position auf Höhe h' falls $h' < h$. Ebenen ohne Glasplatten sind komplett transparent. Generieren Sie diese Bildaufnahme und geben Sie sie als ImageFrame zurück. Sollte sich auf Höhe `height` bereits eine Glasplatte befinden, geben Sie stattdessen `null` zurück. Sie dürfen annehmen, dass `height` nur Werte von 0 bis und mit 10 annimmt. Die Glasplatten und die Photos haben jeweils die Dimensionen des Hintergrundes.

Hinweis: Bei dieser Teilaufgabe müssen Sie selber einen geeigneten Weg finden, die eingefügten Glasplatten irgendwo zu speichern, sodass Sie in der Methode `getPhoto` dann die entsprechenden Aufnahmen zurückgeben können.

Tests finden Sie in der Datei "Task2Test.java".