



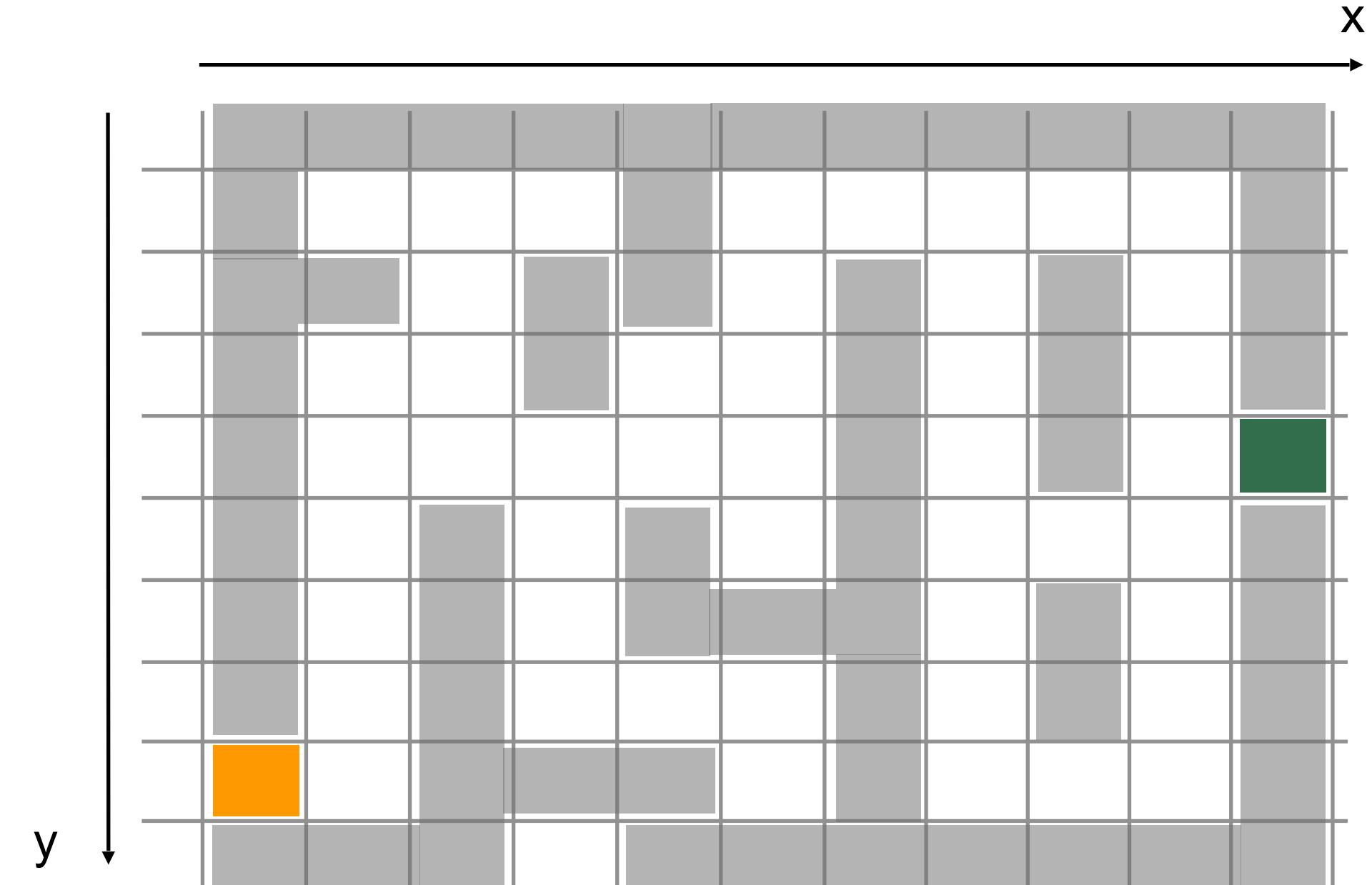
Stegreifprojekt 2 / DAS LABYRINTH

Programmieren I – Prof. Dr. Jörg Berdux

Das Labyrinth / Aufgabe

BB-8 hat sich auf den Weg gemacht, um seinen Freund R2-D2 zu suchen. Dummerweise hat er sich verlaufen und sucht nun nach dem Ausgang.

Sie als Medieninformatikerin bzw. Medieninformatiker können ihm natürlich helfen, indem Sie ihm beibringen durch ein Labyrinth zu laufen, so dass er immer den Ausgang findet.



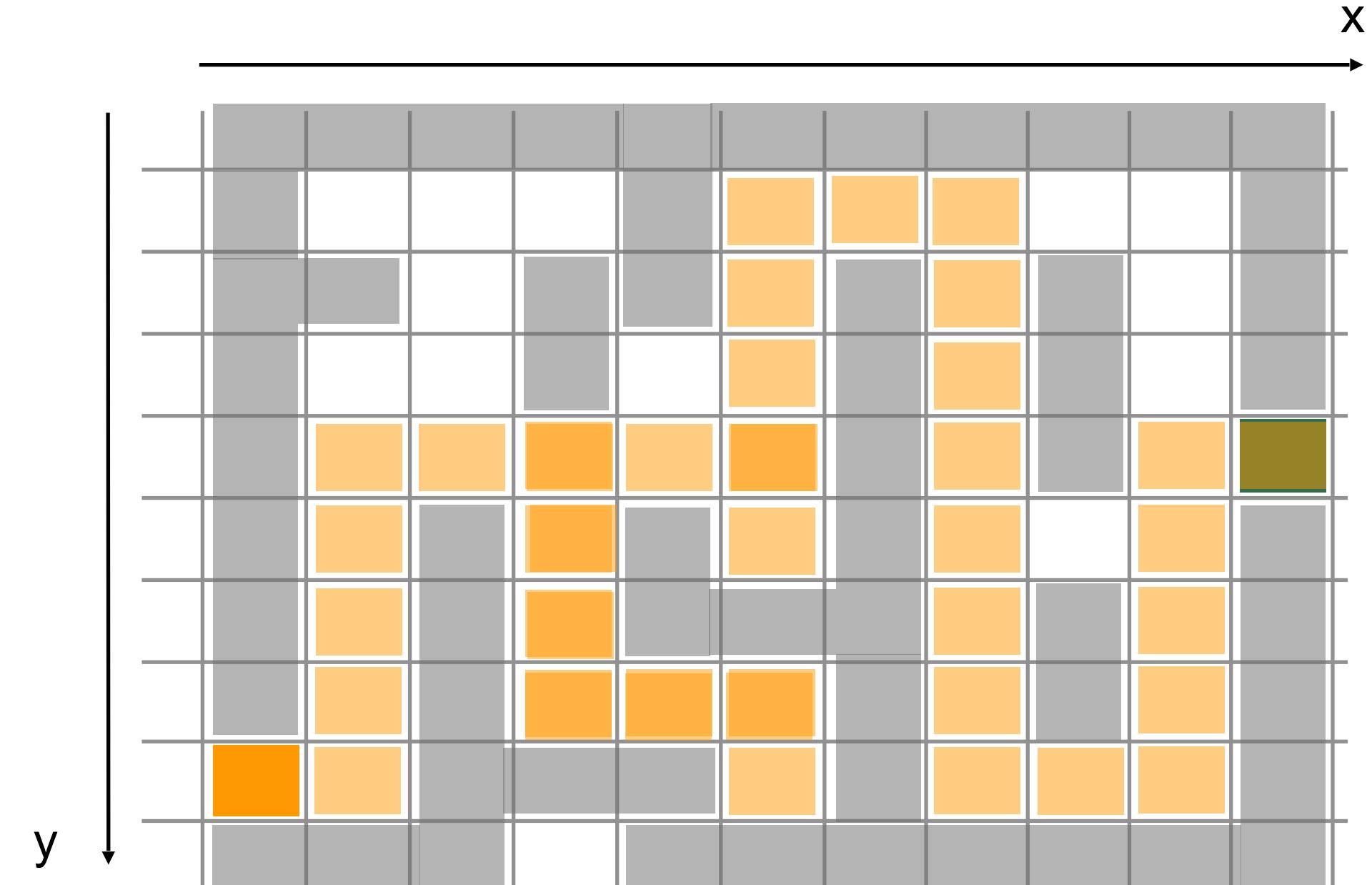
Schreiben Sie ein Java-Programm, das ein zweidimensionales Labyrinth verwaltet und BB-8 den Weg durch das Labyrinth suchen lässt.

- Verwenden Sie für die Lösung ein zweidimensionales Array mit geeignetem Datentyp, in dem die Wände des Labyrinths, die Startposition von BB-8 und der Ausgang platziert werden. Zusätzlich hat BB-8 noch eine weitere Karte, in der er sich immer merken kann, wo er bereits war und was er an dieser Stelle gemacht hat.
- Um den Weg zu finden, verwendet BB-8 die von Ihnen programmierte Strategie der Rechten-Hand-Regel.
- BB-8 soll nach dieser Strategie Schritt für Schritt gehen, so dass man seine Suche nachvollziehen kann. Sobald BB-8 das Ziel erreicht hat, soll sein Weg in die Freiheit dargestellt werden.



Rechte-Hand-Regel

- Die Rechte-Hand-Regel ist die wohl bekannteste Regel, um aus einem Labyrinth zu kommen. Hierbei versucht man sich immer mit der rechten Hand an der Wand entlang zu tasten. D.h. sobald man rechts abbiegen kann, macht man dies auch.
- Soweit die Mauern zusammenhängen und die Außenwände geschlossen sind, kommt man mit diesem Prinzip an den Ausgang.



- Um BB-8 dieses Vorgehen zu ermöglichen, muss er natürlich immer eine Blickrichtung besitzen, so dass er auch immer weiß, ob er rechts eine Wand hat oder ob er seine Richtung ändern muss, um der Wand zu folgen.
- Am Anfang soll BB-8 immer eine Position und Ausrichtung innerhalb des Labyrinths besitzen.
- Selbstverständlich hat das Labyrinth einen Ausgang, zu dem es auch einen oder mehrere Wege gibt, so dass BB-8 auch eine Chance hat den Ausgang zu finden.
- BB-8 zählt die Schritte, die er gebraucht hat mit, um hinterher von seinem Erfolg – den Weg in x Schritten gefunden zu haben – erzählen zu können.
- Zusätzlich merkt er sich alle Stellen, an denen er vorbeigekommen ist, in einer eigenen Karte. Am Ausgang angekommen, kann er dann gleich von seinem zurückgelegten Weg berichten.



Labyrinth

Wer Spaß daran hat, kann natürlich auch noch ein immer wieder zufällig erzeugtes Labyrinth anbieten. Sie müssen dabei nicht sicherstellen, dass man den Ausgang erreichen kann.

Auswahl von Labyrinthen

- › BB-8 soll als Spezialist für Labyrinthe natürlich durch unterschiedliche Labyrinthe laufen können. Daher sollen in dem Programm neben dem vorgegebenen Labyrinth zwei weitere frei durch Sie zu definierende Labyrinthe definiert werden, die unterschiedlich groß sein sollen.
- › Der Nutzer hat am Anfang des Programms die Möglichkeit, eines der angebotenen Labyrinthe auszuwählen.

Kodierung / Labyrinth

- › Alle Labyrinthe sind durch eine einheitliche Kodierung beschrieben, so dass beliebige Labyrinthe definiert werden können. Neben den Wänden mit '.' werden die Startposition von BB-8 mit 'B' und der Ausgang des Labyrinths mit 'A' als Zeichen kodiert.
- › Zu Beginn des Programms muss somit zunächst die Position von BB-8 aus der Labyrinth-Definition ermittelt werden, damit er dann Schritt für Schritt weitergehen kann. Auch den Ausgang kann man entsprechend aus der Karte lesen oder bei jedem Schritt überprüfen, ob man den Ausgang gefunden hat.

Kodierung / Laufweg

- › Um den Laufweg von BB-8 auch noch am Ende nachvollziehen zu können, soll BB-8 seinen Weg Schritt für Schritt in einer zweiten Karte (Array) markieren. Hierbei soll seine jeweilige Laufrichtung, die seiner aktuellen Blickrichtung entspricht, festgehalten werden.
Die Blickrichtung von BB-8 soll durch die Zeichen ^, >, v, < dargestellt werden.



Anzeige

- Das Labyrinth soll nach jedem Schritt von BB-8 angezeigt werden, so dass man den Weg von BB-8 verfolgen kann. Die Blickrichtung von BB-8 soll durch die Zeichen ^, >, v, < dargestellt werden.
 - Sobald BB-8 den Ausgang gefunden hat, hat er natürlich viel über seine Abenteuer zu erzählen und berichtet von seinem Weg durch das Labyrinth, indem er seine protokolierte Karte vorzeigt.
 - Beispiel:

BB-8 auf der Suche ...

A scatter plot showing data points as black dots. The points form several distinct clusters: a top row of 11 points, a middle row of 5 points, a bottom row of 5 points, and a central column of 11 points. A large letter 'V' is positioned in the bottom-left corner.

Schritt 1:

A 10x10 grid of black dots representing a sparse matrix. The matrix has several zero entries, indicated by empty squares. A vector v is shown as a column of dots on the left, and a vector u is shown as a row of dots at the bottom. An arrow labeled $>$ points from the bottom right towards the center of the matrix.

Schritt 2:

Schritt 3

Schritt 4

Schritt 5

Schritt

Schritt

Schritt



Schritt 81:

Ausgang in 81 Schritten
gefunden.

So hab ich den Ausgang
gefunden



Hinweis / Verzögerung

- Um die Suche des Ausgangs etwas zu animieren, können Sie in Ihrer Schleife eine kleine Verzögerung einbauen. Wenn Sie diese Verzögerung um n Millisekunden am Ende der Schleife platzieren, gibt es jeweils eine entsprechende Verzögerung bevor der nächste Schritt von BB-8 berechnet und angezeigt wird.

```
...  
try {  
    Thread.sleep(250);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}  
...  
...
```

- Alternativ können Sie selbstverständlich auch auf eine Benutzereingabe warten, um den nächsten Schritt zu berechnen.



Gruppen – Praktikum A

Gruppe 1

- Patrick Haltof
- Fabio Bertels
- Finn Schindel

Gruppe 2

- Victoria Thee
- Timothy Doukhin
- Beate Arnold

Gruppe 3

- Leon Kamp
- Ana-Maria Adanaia
- Ramtin Hajizadeh Ganji



Gruppe 4

- Tim Reinicke
- Paul Trippe
- Aron-Merlin Schlegel

Gruppe 5

- Hiep Ngoc Pham
- Raphael Kunrath
- Alena Teliatnikova

Gruppe 6

- Marigona Sejdiu
- Khanh Linh Truong

Gruppen – Praktikum B

Gruppe 1

- Alexander Hüge
- Rosalie Kunz
- Kevin Emunds

Gruppe 2

- Gabriel Stetter
- Elias Demirci
- Gurjender Singh Hanson

Gruppe 3

- Lulia McCoy
- Marieke Schmitz

Gruppe 4

- Emily Jung
- Moritz Hebestreit
- Houssam Sakah

Gruppe 5

- Yannick Trabandt
- Daniel Wenz

Gruppe 6

- Thimo Scholl
- Johannes Weber
- Martin Lahdo



Gruppen – Praktikum C

Gruppe 1

- Alexander Müller
- Erina Daraz
- Felix Ruf

Gruppe 2

- Borna Juliae
- Oliver Beer
- Niklas Vogel

Gruppe 3

- Marvin Wernli
- Pascal Zöphel
- Johanna Schad

Gruppe 4

- Felix Wüst
- Max Muthler

Gruppe 5

- Marie Bohnert
- Neele Wolf
- Tatiana Vallo

Gruppe 6

- Nico Hunsicker
- Tom Gouthier
- Fabian Rehbogen





Gruppe 1

- ▶ Timon Noll
- ▶ Arthur Fieguth
- ▶ Elisabeth Burkhardt

Gruppe 2

- ▶ Julia Honisch
- ▶ Niklas Lämmermann
- ▶ Astrid Klemmer

Gruppe 3

- ▶ Atta Farsimadan
- ▶ Robin Leber
- ▶ Robin Marxen

Gruppe 4

- ▶ Victor Perez Ramirez
- ▶ Philipp Zechner
- ▶ Julian Stäger

Gruppe 5

- ▶ Anastasiia Evdash
- ▶ Larisa Bertinchamp

Gruppe 6

- ▶ Jasmin Günther
- ▶ Fabiane Hofmann
- ▶ Oliver Jochum

Gruppe 7

- ▶ Jonas Goebels
- ▶ Jens Hornung

Abgabe

Die Abgabe erfolgt über das
Portal <https://read.mi.hs-rm.de>

Abgegeben werden muss:

- Die Quelltext-Datei (source code file)



Kriterien für Punktevergabe

Das Programm muss

- korrekt funktionieren
- den Verlauf der Wegsuche Schritt für Schritt ausgeben
- den gesamten gelaufenen Weg am Ende nochmals separat ausgeben
- zweidimensionale Arrays sinnvoll einsetzen
- sinnvoll in Methoden strukturiert sein
 - Einhaltung der Konventionen
 - Sinnvolle Methoden
- (inhaltlich) wohl strukturiert sein
- mit Javadoc dokumentiert sein

Punktevergabe

Es werden bis zu 4 Punkte für die Abgabe vergeben.

