



Der Raspberry Pi und Spieleentwicklung für Kinder



Wo läuft eigentlich überall Java?





Wo läuft eigentlich überall Java?

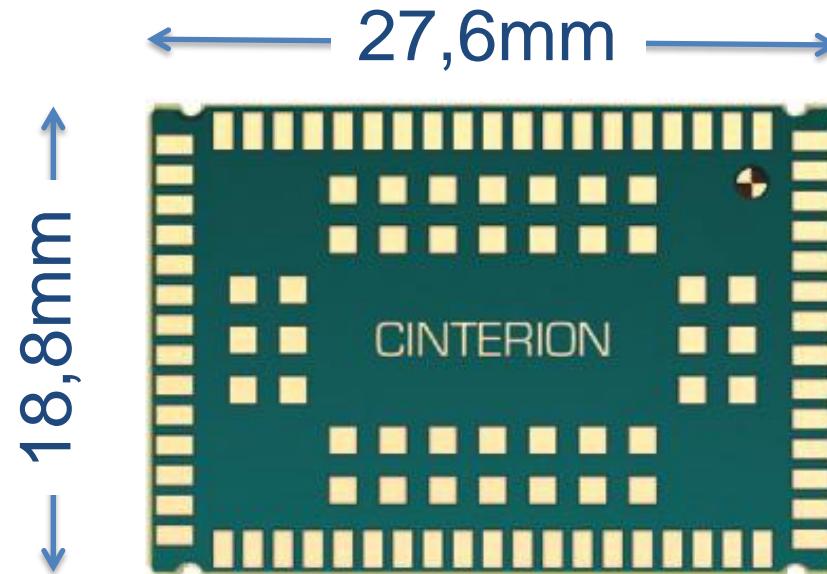


Java und 3G in einem kleinen Packet

- Cinterion EHS5



Das ist wirklich klein...







Es gibt Java zum Nachtisch

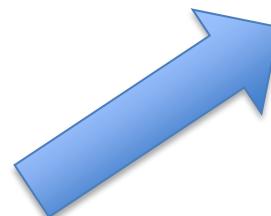
Raspberry Pi (das ist ein englisches Wortspiel:
es klingt wie **Himbeerkuchen**)





Der Raspberry Pi ist nicht teuer

Ca. 40€



RASPBERRY PI B :: Raspberry Pi Typ B, 512 MB

Artikel-Nr.: RASPBERRY PI B

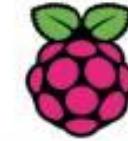
38,95 €

inkl. gesetzl. MwSt. zzgl. Versandkosten

ab Lager

Stück: **in den Warenkorb**

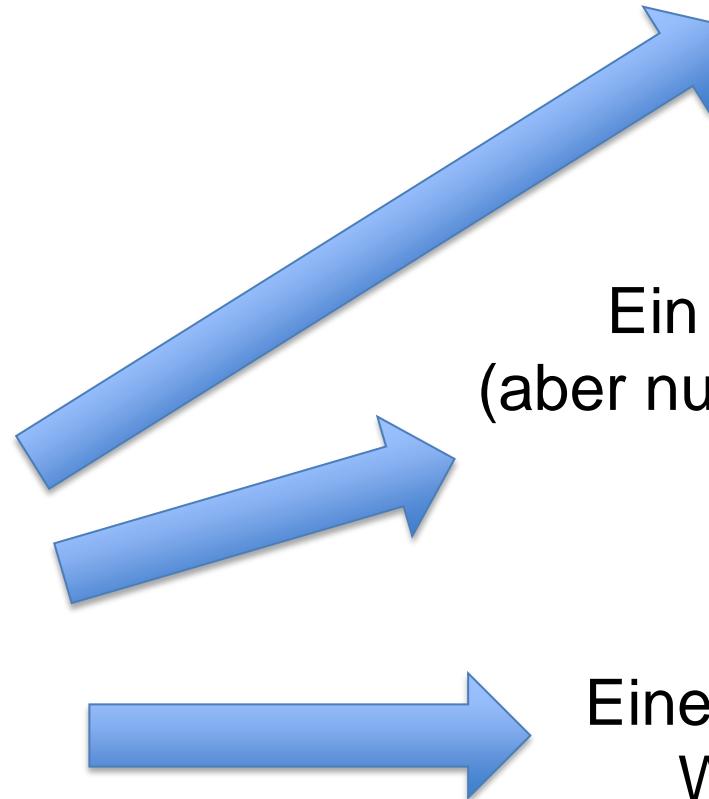
Warengruppe: 0



RaspberryPi

Der Raspberry Pi ist nicht teuer

Ca. 40€



Eine (große)
Torte

Ein Fahrrad
(aber nur ein Reifen)

Eine Packung
Windeln

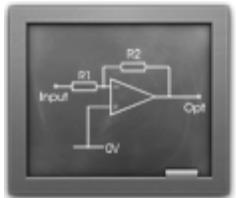


Chalkboard Electronics

Touchscreen

- Als 10" oder als 7" Variante
- Anschlüsse: HDMI/USB
- Getestet mit Java 8
- 10% exklusiver Rabatt:

G1FOU796Z083



Chalkboard
Electronics



So richtest Du Deinen Raspberry Pi ein

Schritt 1: Installiere Linux

Schritt 2: Kopiere dir Java 8 für ARM EA

Schritt 3: Lass JVM Programme und Anwendungen
auf deinem Pi laufen



<http://steveonjava.com/javafx-on-raspberry-pi-3-easy-steps/>



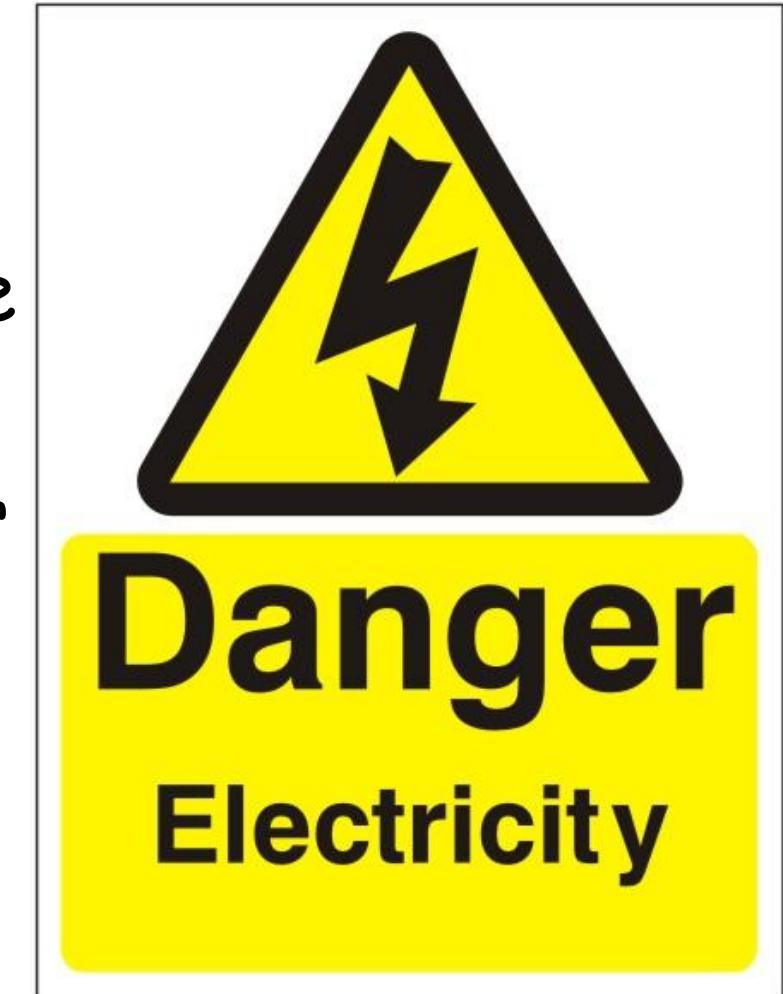
Was kommt mit Eurer Laborausstattung mit

1. Touch Screen
2. SD Karte
3. Tastatur
4. Eine gelbe Schachtel
 - Stromadapter
 - LVDS Kabel/Board
 - Raspberry Pi Model B
 - Mini-USB Kabel (Stromanschluss)
 - Micro-USB Kabel (Tastaturanschluss)

Hebt bitte alle Verpackungen für später auf!

Sicherheit beim Umgang mit Elektronik!

- Stecker ziehen bevor es los geht!
- Entladet Euch von statischer Elektrizität, indem ihr eine metallene Oberfläche anfasst
- Fasst keine freiliegende Drähte oder metallenen Teile an
- Die SD-Karte niemals einstecken oder herausziehen während der Pi an eine Stromquelle angeschlossen ist





So schließt ihr den Pi an (Teil A)

Wichtig: Verbindet alle Teile korrekt bevor Ihr den Pi mit Strom versorgt

1. Steckt die SD Karte in den Pi

- Ihr seht dann deren Unterseite wenn ihr auf den Pi schaut

2. Steckt das HDMI Kabel in den HDMI-Anschluss Eures Pi

So schließt ihr den Pi an (Teil B)

3. Verbindet den Stromanschluss des Pi mit dem HDMI Board

- Nehmt dazu das Micro-USB Kabel (das ist das kurze Kabel)

4. Steckt das LSVD-Kabel in die Rückseite des Monitors

- Die Seite mit den Goldkontakte muss nach oben zeigen
- Und seid vorsichtig: Die Kontakte sind empfindlich und leicht zerbrechlich

So schließt ihr den Pi an (Teil B)

5. Verbindet das Ende des USB-Kabels mit einem der USB-Anschlüsse des Pi

- So sorgt ihr dafür, dass der Monitor auf Berührung reagiert (es ist schließlich ein Touch Screen)

6. Schließt nun noch die Tastatur an

- Dazu nehmt ihr das Mini-USB Kabel (dies ist das lange Kabel)

Prüft nochmal alles sorgfältig und schliesst Euren Pi an die Stromversorgung an



Läuft alles?



Hacking Time!

- Ihr solltet jetzt eine Reihe von leuchtenden LEDs sehen (dies besagt, das der Pi startet)
 - Das Starten (auch Booten genannt) dauert ungefähr 30 Sekunden
- Dann sollte der LCD-Monitor aufleuchten
 - Wenn der Bildschirm zu dunkel ist kann das daran liegen, dass der Lichtsensor des LCD-Monitor verdeckt ist
- Ihr solltet einen Linux Startbildschirm mit viel Text sehen



Anmeldung

Am Anmeldebildschirm gebt zuerst den Benutzernamen ein:

- **pi**

Und dann das Passwort:

- **raspberry**



So startet Ihr Eure erste Anwendung

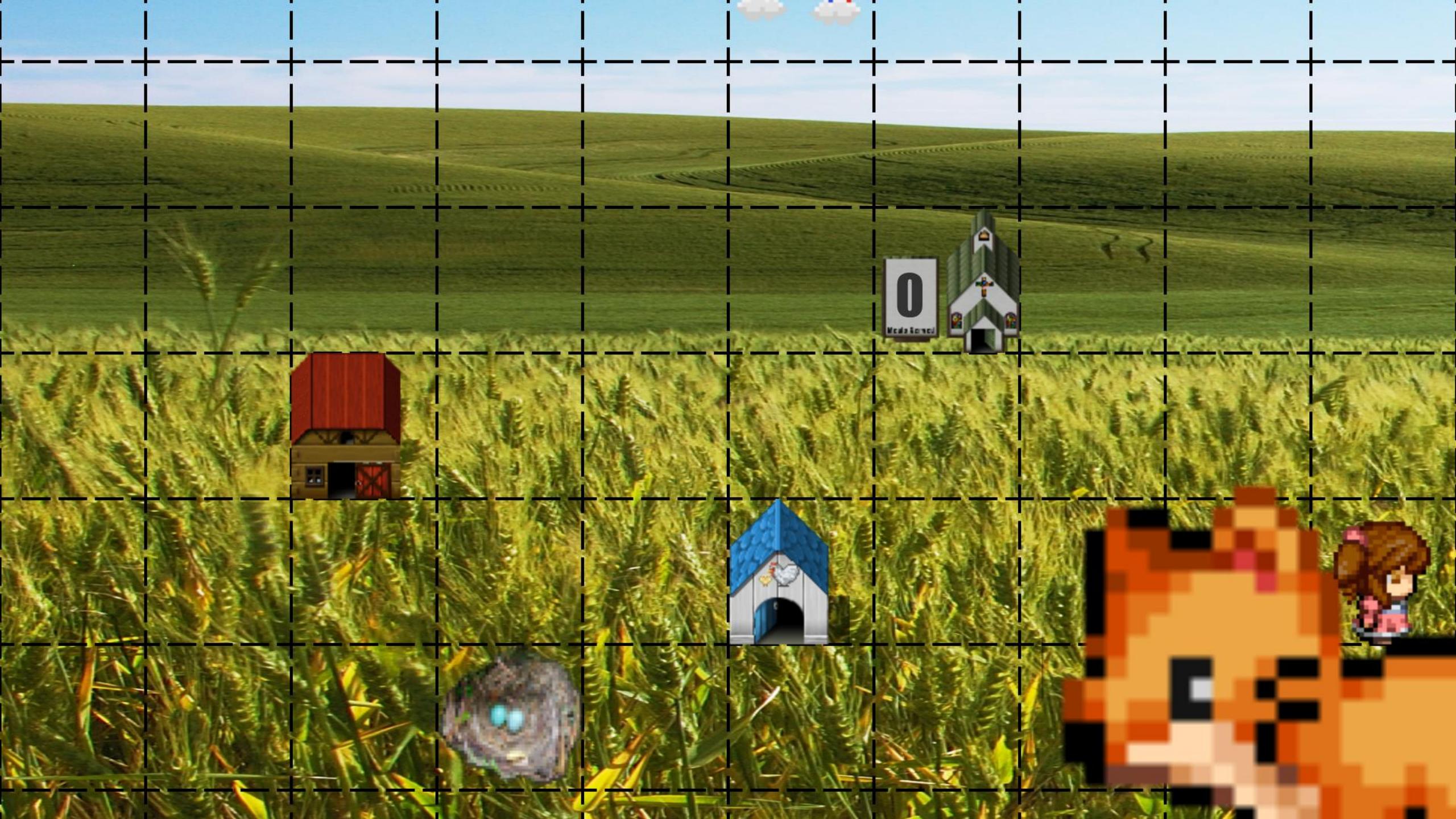
Wechselt in das Verzeichnis unseres Projektes

- `cd MaryHadALittleLambda`

Und lasst die Ameise (englisch ant) für Euch das Skript abarbeiten

- `ant`







Den Code hucken

Startet den Texteditor *nano*:

- nano src/sample/MapObject.java

So speichert ihr eure Änderungen:

- Control-O Enter

Um nano wieder zu beenden, drückt:

- ## • Control-X

Um das Programm zu übersetzen und ausführen, schreibt:

- ant

www.devoxx4kids.org/deutschland

```
GNU nano 2.1.2-svn      File: ./Download/SVN/nano/src/nano.c

/* Disable mouse support. */
void disable_mouse_support(void)
{
    mousemask(0, NULL);■
    mouseinterval(oldinterval);
}

/* Enable mouse support. */
void enable_mouse_support(void)
{
    mousemask(ALL_MOUSE_EVENTS, NULL);
    oldinterval = mouseinterval(50);
}

/* Initialize mouse support. Enable it if the USE_MOUSE flag is set,
 * and disable it otherwise. */
void mouse_init(void)
{
    if (ISSET(USE_MOUSE))
```

Mary Had a Little Lambda

Mary Had a Little Lamb
Whose fleece was white as snow
And everywhere that Mary went
Lambda was sure to go!

Mary Had a Little Lamb ist ein englischer Kinderreim aus dem 19ten Jahrhundert von Sarah Josepha Hale. Er erzählt von einem kleinen Mädchen mit einem Lamm...



<https://github.com/steveonjava/MaryHadALittleLambda>

Datenströme generieren



Aus einer Sammlung (englisch **collection**):

- `anyCollection.stream();`

Aus einer **bekannten** Menge von Objekten:

- `Stream.of("bananen", "orangen", "aepfel");`

Aus Zahlbereichen:

- `IntStream.range(0, 50)`



Oder **iterativ** (d.h. schrittweise):

- Stream.iterate(Color.RED,
 c -> Color.hsb(c.getHue() + .1,
 c.getSaturation(),
 c.getBrightness())
);

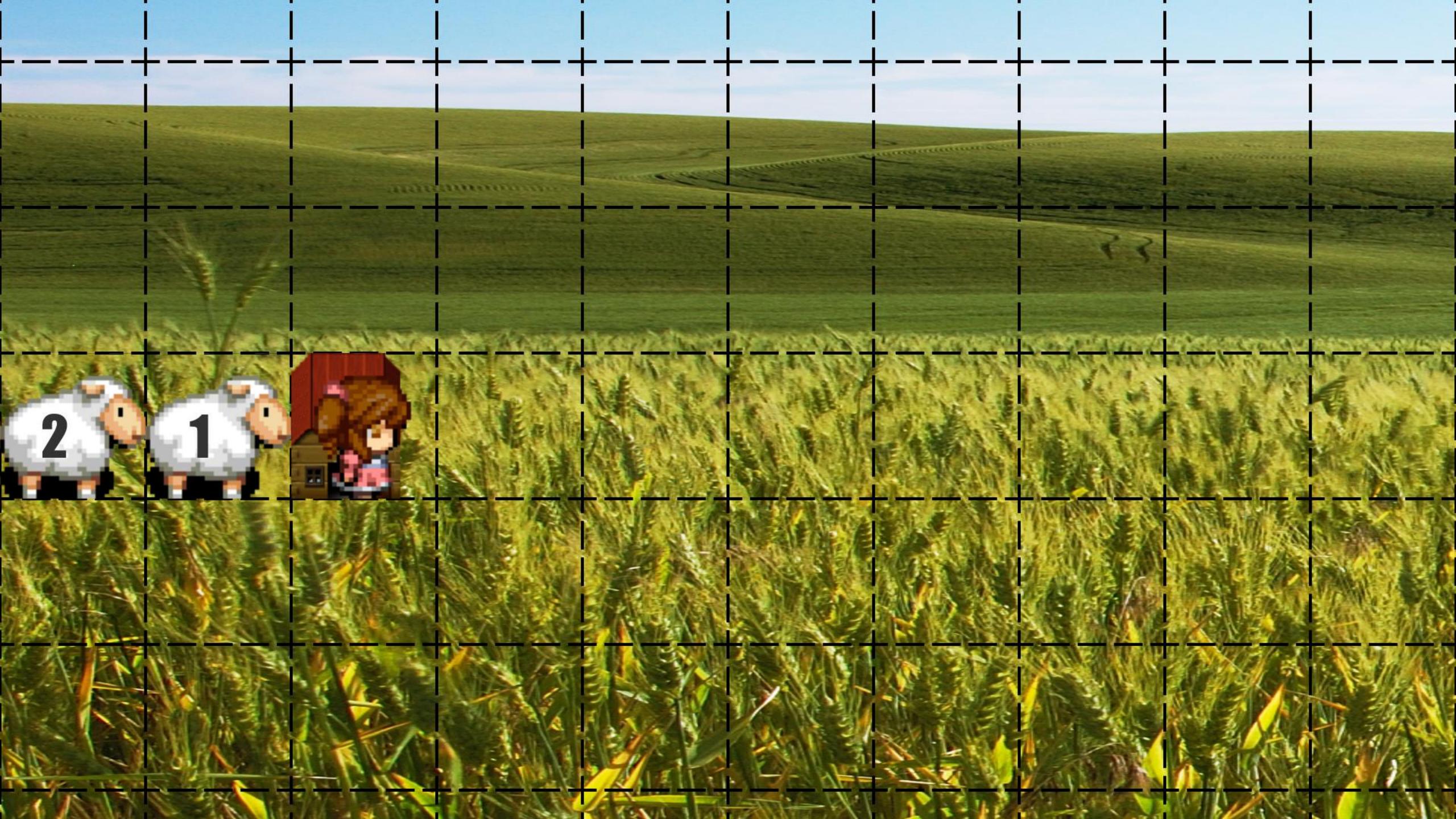


Lasst uns ein paar Tiere erzeugen!

```
SpriteView tail = s.getAnimals().isEmpty() ?  
    s : s.getAnimals().get(s.getAnimals().size() - 1);
```



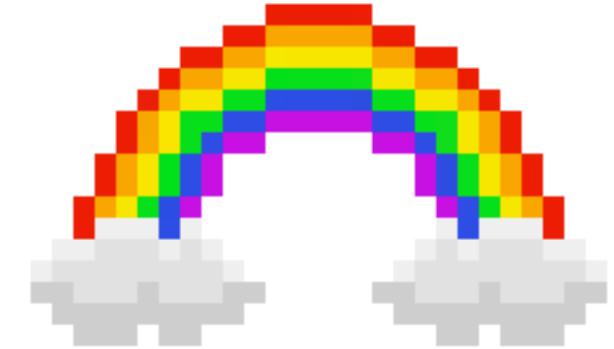
```
Stream.iterate(tail, SpriteView.Lamb::new)  
    .substream(1, 8)  
    .forEach(s.getAnimals()::add);
```



2

1

Datenströme filtern



Prädikatausdrücke (*Ja oder Nein Fragen*):

- ```
public interface Predicate<T> {
 public boolean test(T t);
}
```

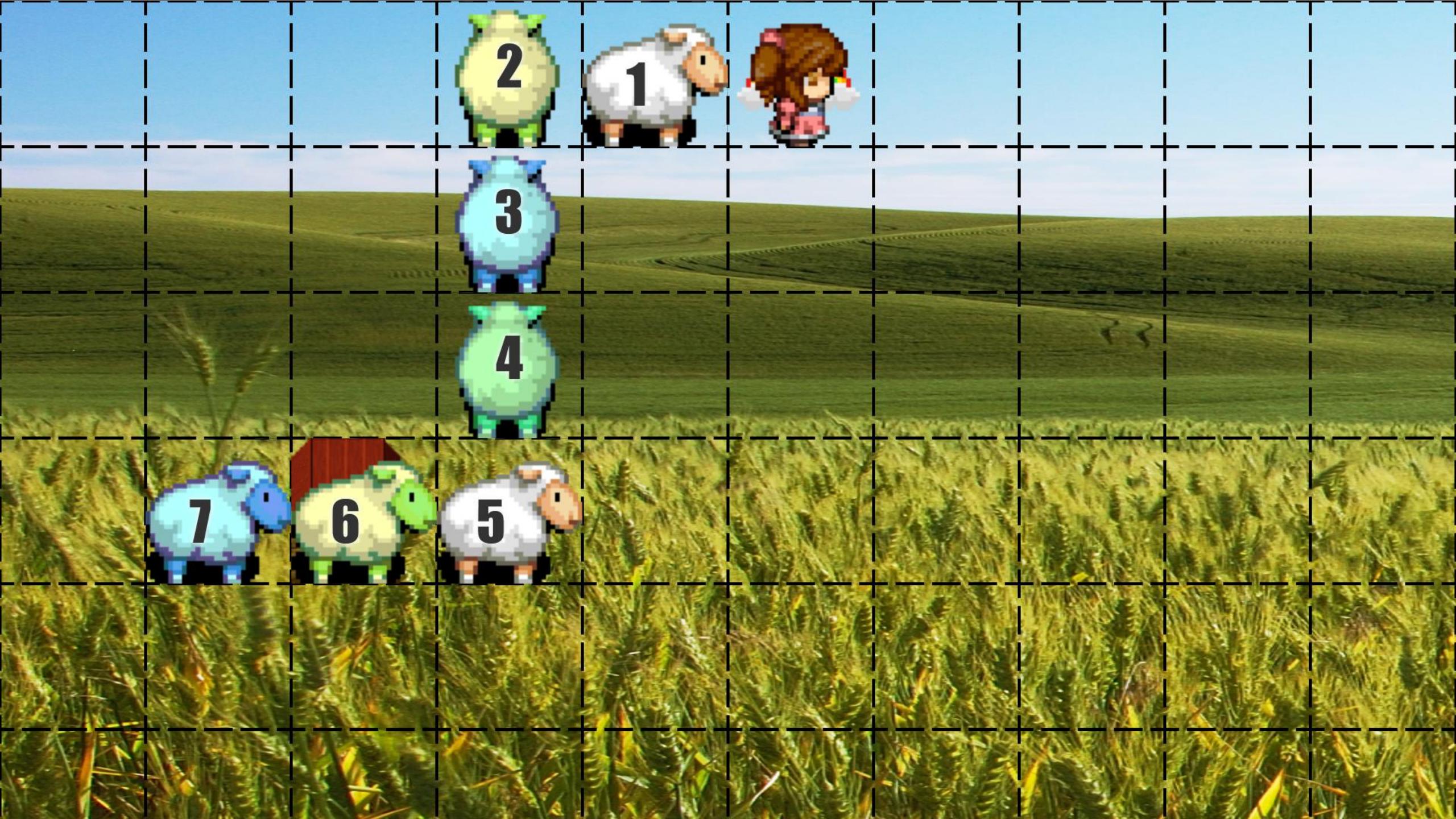
Und so schließen wir Minderjährige aus:

- ```
erwachsene = teilnehmer.filter(a -> a.getAge() >= 18)
```

Bunte Lämmer!



```
s.getAnimals().stream()  
    .filter(a -> a.getNumber() % 4 == 2)  
    .forEach(a -> a.setColor(Color.YELLOW));  
  
s.getAnimals().stream()  
    .filter(a -> a.getNumber() % 4 == 3)  
    .forEach(a -> a.setColor(Color.CYAN));  
  
s.getAnimals().stream()  
    .filter(a -> a.getNumber() % 4 == 0)  
    .forEach(a -> a.setColor(Color.GREEN));
```



2

1



3

4

7

6

5

Sammlungen filtern

`Collection.removeIf`

- Entfernt alle Elemente, auf die ein Prädikat zutrifft



`List.replaceAll`

- Filtert und ersetzt mit einem einstelligen Operator

`ObservableCollection.filtered`

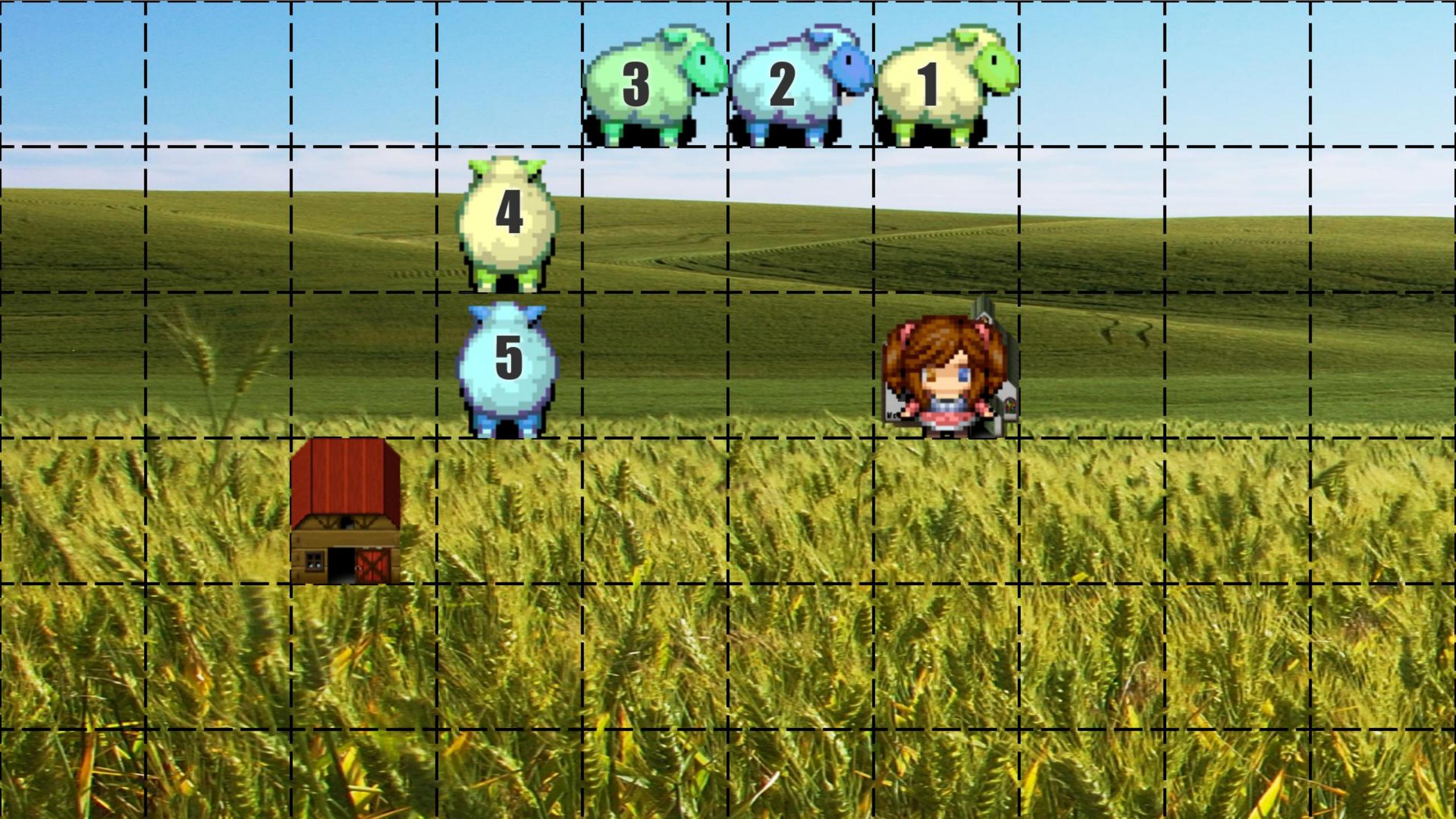
- Gibt eine nach einem Prädikat gefilterte Liste zurück (die auch ein Observable ist)



Wählerische Esser

```
Predicate<SpriteView> pure =  
    a -> a.getColor() == null;  
  
mealsServed.set(mealsServed.get() +  
    s.getAnimals().filtered(pure).size()  
);  
  
s.getAnimals().removeIf(pure);
```





Datenströme abbilden



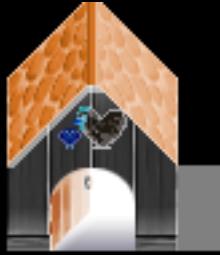
Wendet eine Abbildungsvorschrift auf jedes Element eines Datenstroms an:

- Function<? super T, ? extends R>

Ergebnis: Eine Liste mit der gleichen Anzahl von Elementen, die jetzt aber einen anderen Typ haben kann.



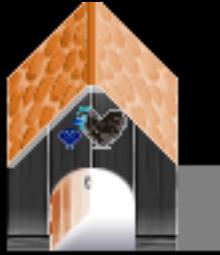
Einzelabbildung (Single Map)



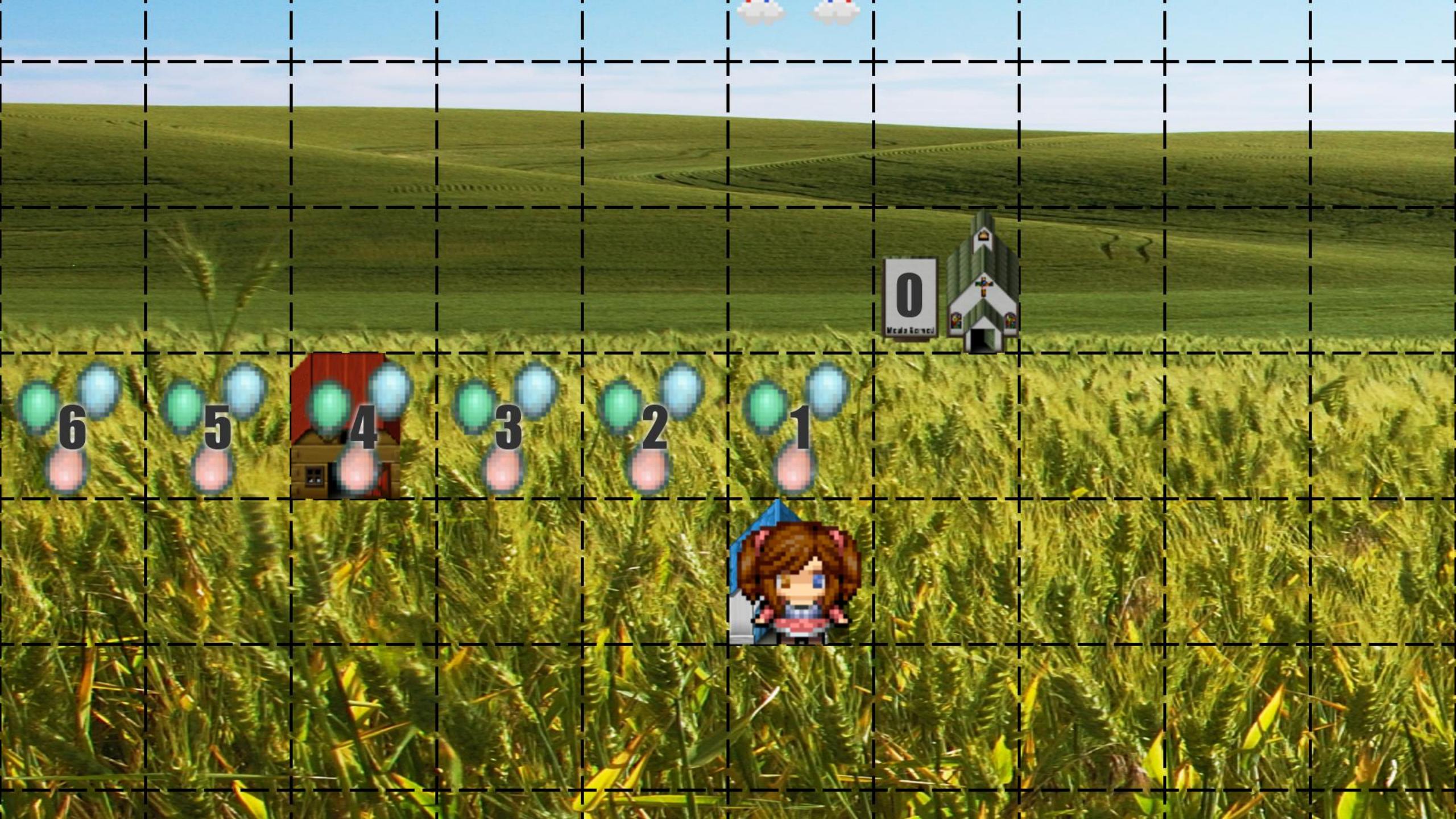
```
s.getAnimals().setAll(s.getAnimals()
    .stream()
    .map(sv -> new Eggs(sv.getFollowing()))
    .collect(Collectors.toList())
);
```



Oder eine Doppelabbildung (Double Map)!



```
s.getAnimals().setAll(s.getAnimals()
    .stream()
    .map(SpriteView::getFollowing)
    .map(Eggs::new)
    .collect(Collectors.toList()))
);
```



6

5

4

3

2

1

0
Meds Scored

Flache Abbildungen



Wendet eine **eins-zu-viele** Funktion auf jedes Element an, die aus jedem Element mehrere Elemente machen kann:

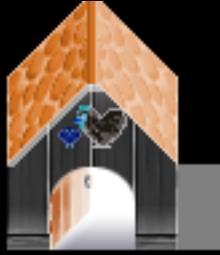
- Function<? super T, ? extends Stream<? extends R>>

Das Ergebnis wird dann zu einem einzigen Datenstrom reduziert.

Ergebnis: Die Liste kann länger werden und einen anderen Typ bekommen.



Eier ausbrüten



```
s.getAnimals().setAll(s.getAnimals()
    .stream()
    .flatMap(SpriteView.Eggs::hatch)
    .collect(Collectors.toList()))
);
```



20

19

16

13

10

7

4

18

2

3

8

9

0

Medals Scored

Reduzieren



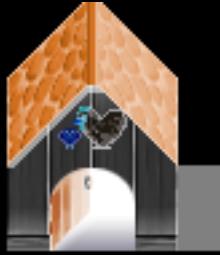
Reduziert eine Liste auf einen einzigen Wert:

- Identity: T
- Accumulator: `BinaryOperator<T>`

Ergebnis: Eine Liste des selben Typs, aber mit nur noch einem Element.

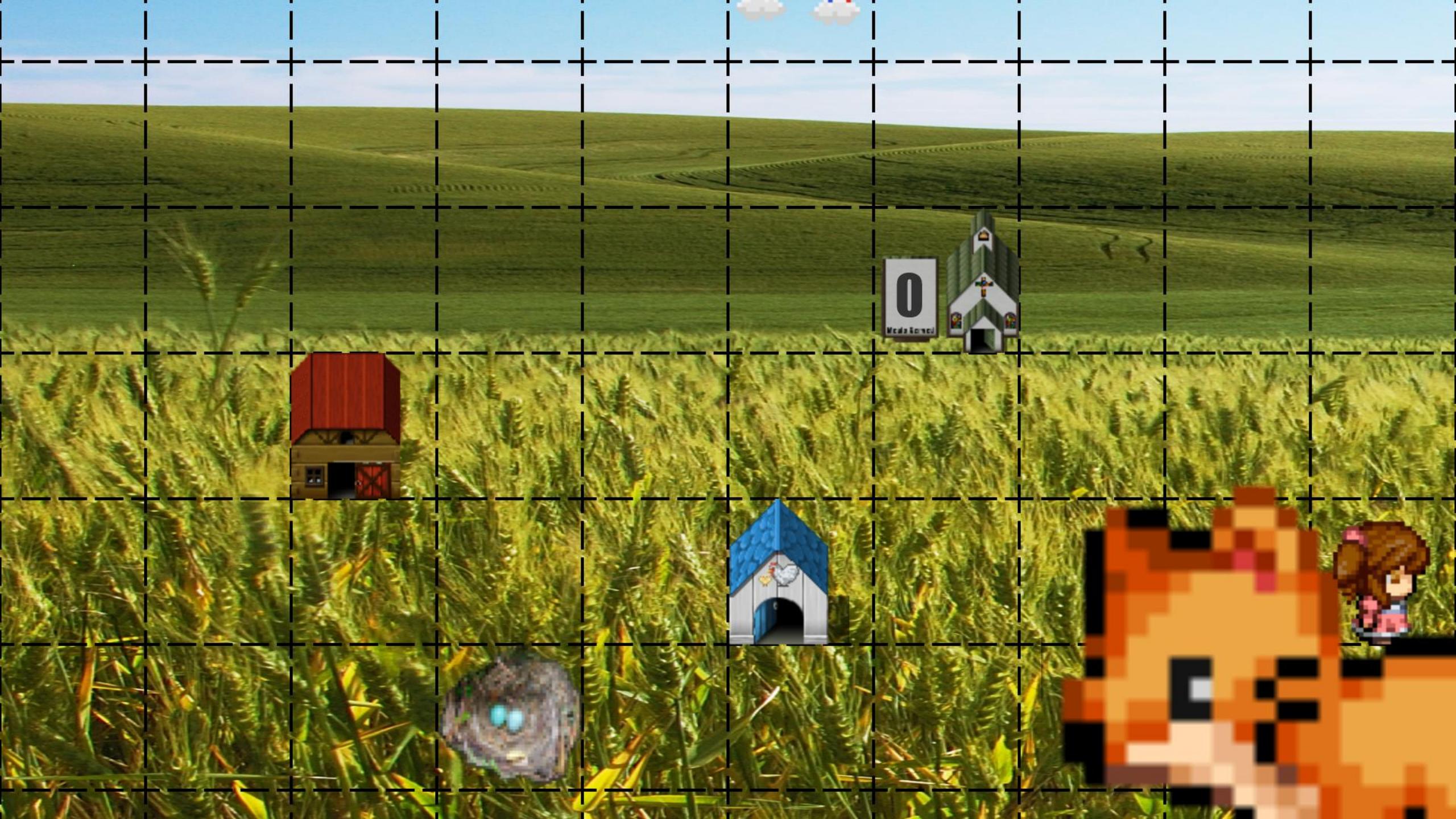


Und der (kleine) Fuchs hat sie alle gefressen!



```
Double mealSize = shepherd.getAnimals()  
    .stream()  
    .map(SpriteView::getScaleX)  
    .reduce(0.0, Double::sum);
```

```
setScaleX(getScaleX() + mealSize * .2);  
setScaleY(getScaleY() + mealSize * .2);  
shepherd.getAnimals().clear();
```



Das Mary Had a Little Lambda Projekt

- Open-source Projekt, das die Eigenschaften der **Lambda-Ausdrücke** demonstriert
- Anschauliche Repräsentation von Strömen (**streams**), Filtern (**filters**) und Abbildungen (**maps**)

<https://github.com/steveonjava/MaryHadALittleLambda>





Designed by
Stephen Chin (@steveonjava)
<http://steveonjava.com/>

ORACLE®

www.devoxx4kids.org/deutschland

NightHacking Tour



Real Geeks
Live Hacking
nighthacking.com