Retro Gaming With Lambdas

Stephen Chin (@steveonjava)
Java Technology Ambassador
JavaOne Content Chair

JDK 8 Feature Overview

Innovation

- Lambda aka Closures
- Language Interop
- Nashorn

Core Libraries

- Parallel operations for core collections APIs
- Improvements in functionality
- Improved type inference

General Goodness

- JVM enhancements
- No PermGen limitations
- Performance Improvements

Java for Everyone

- Profiles for constrained devices
- JSR 310 - Date & Time APIs
- Non-Gregorian calendars
- Unicode 6.2
- ResourceBundle.
- BCP47 locale matching
- Globalization & Accessibility

Tools

- Compiler control & logging
- JSR 308 - Annotations on Java Type
- Native app bundling
- App Store Bundling tools

Client

- Deployment enhancements
- JavaFX 8
- Java SE Embedded support
- Enhanced HTML5 support
- 3D shapes and attributes
- Printing

Security

- Limited doPrivilege
- NSA Suite B algorithm support
- SNI Server Side support
- DSA updated to FIPS186-3
- AEAD JSSE CipherSuites

Coordinated Platform Upgrade with Lambda

- Language
  - Lambda Expressions (closures)
  - Interface Evolution with default methods
- Libraries
  - Bulk data operations on Collections
  - More library support for parallelism
- JVM
  - Default methods
  - Enhancements to invokedynamic

Download JDK 8 Early Access Release

Java.net The Source for Java Technology Collaboration

JDK 8
- Downloads
- Feedback Forum
- OpenJDK
- Planet JDK

JDK™ 8 Early Access Releases

April 04, 2013

8 Build b84

Summary of changes in JDK 8 build b84

Previous Early Access Releases
Feedback forum
Report Bugs

Please note:

This list offers files for different platforms - please be sure to select the proper file(s) for your platform. Carefully review the files listed below to select the ones you want, then click the link(s) to download.

License Agreement:

You must accept the Pre-Production Software Evaluation Agreement for Java SE to download this software.

○ Accept License Agreement | ○ Decline License Agreement

Documentation

- JDK Docs (44.81 MB zip | HTML)
- JavaFX Docs (9.81 MB zip)

Platforms	JRE	JDK

IDE Support for Lambdas

```
for (Shape s : shapes) {
   if (s.getColor() == BLUE)
      s.setColor(RED);
}
```

```
shapes.forEach(s -> {
   if (s.getColor() == BLUE)
      s.setColor(RED);
});
```

Extension Methods

```
interface Collection<T> {
  default void forEach(Block<T> action) {
    Objects.requireNonNull(action);
    for (T t : this)
      action.apply(t);
  }

  // Rest of Collection methods…
}
```

Lambda Syntax

- `(int a, int b) -> { return a + b; }`
- `(a, b) -> a + b`
- `a -> a * a`
- `() -> a`
- `() -> { System.out.println("done"); }`

Method References

- `Objects::toString` `obj -> Objects.toString(obj)`
- `Object::toString` `obj -> obj.toString()`
- `obj::toString` `() -> obj.toString()`
- `Object::new` `() -> new Object()`

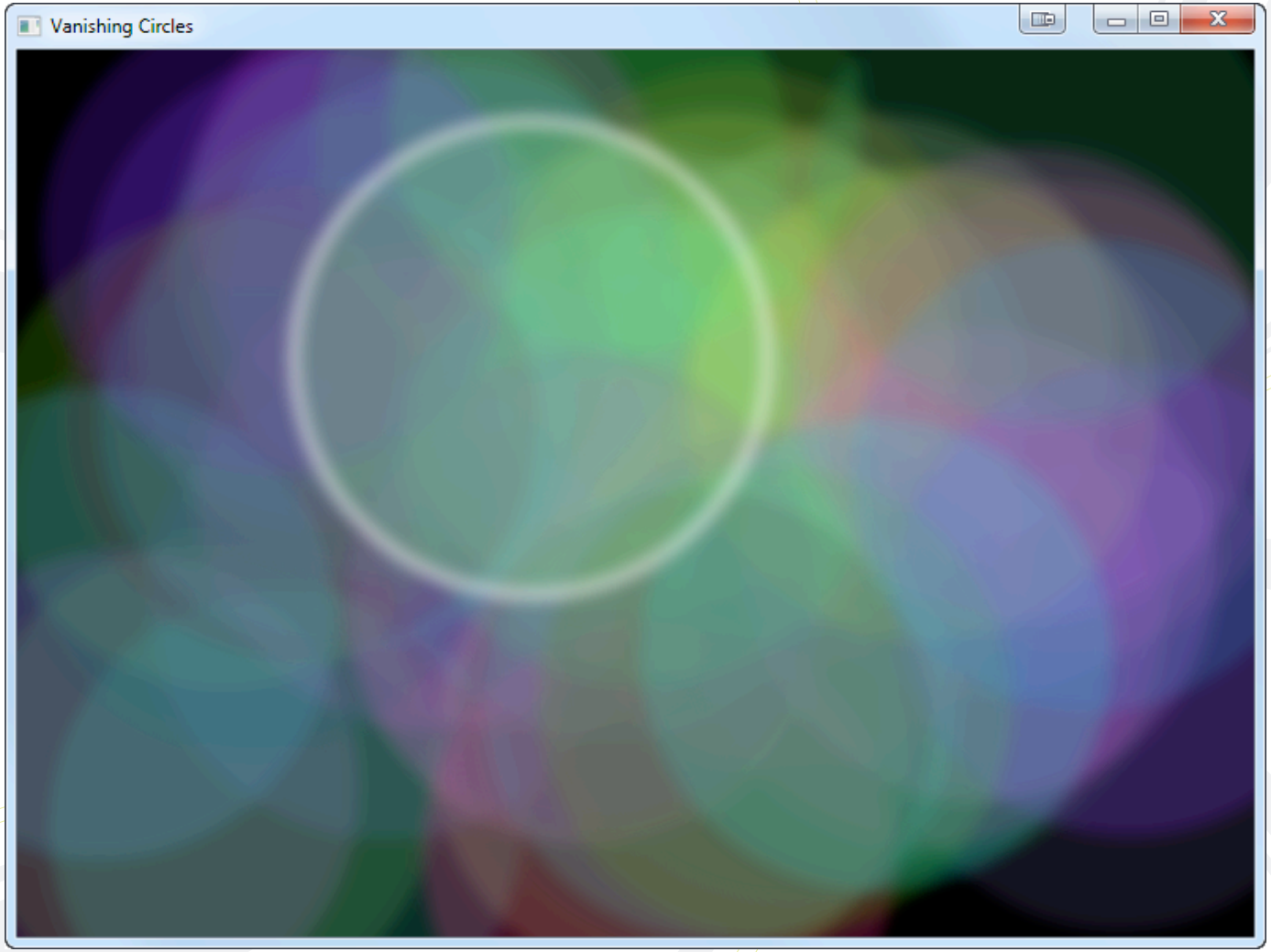
Functional Interfaces

```
@FunctionalInterface

@FunctionalInterface
interface Sum {
  int add(int a, int b);
}

sum = (a, b) -> a + b;
sum = Integer::sum;
```

Vanishing Circles



Application Skeleton

```
public class VanishingCircles extends Application {
  public static void main(String[] args) {
    Application.launch(args);
  }
  @Override
  public void start(Stage primaryStage) {
    primaryStage.setTitle("Vanishing Circles");
    Group root = new Group();
    Scene scene = new Scene(root, 800, 600, Color.BLACK);
    [create the circles…]
    root.getChildren().addAll(circles);
    primaryStage.setScene(scene);
    primaryStage.show();
    [begin the animation…]
  }
}
```

Create the Circles

```
List<Circle> circles = new ArrayList<Circle>();
for (int i = 0; i < 50; i++) {
  final Circle circle = new Circle(150);
  circle.setCenterX(Math.random() * 800);
  circle.setCenterY(Math.random() * 600);
  circle.setFill(new Color(Math.random(), Math.random(),
                           Math.random(), .2));
  circle.setEffect(new BoxBlur(10, 10, 3));
  circle.setStroke(Color.WHITE);
  [setup binding…]
  [setup event listeners…]
  circles.add(circle);
}
```

Create the Circles

```
List<Circle> circles = Streams.generate(() -> {
  final Circle circle = new Circle(150);
  circle.setCenterX(Math.random() * 800);
  circle.setCenterY(Math.random() * 600);
  circle.setFill(new Color(Math.random(), Math.random(),
                           Math.random(), .2));
  circle.setEffect(new BoxBlur(10, 10, 3));
  circle.setStroke(Color.WHITE);
  [setup binding…]
  [setup event listeners…]
  return circle;
}).limit(50).collect(Collectors.toList());
```

Setup Binding

```
circle.strokeWidthProperty().bind(Bindings
   .when(circle.hoverProperty())
   .then(4)
   .otherwise(0)
);
```

Setup Event Listeners

```
circle.addEventHandler(MouseEvent.MOUSE_CLICKED,
                       new EventHandler<MouseEvent>() {
  public void handle(MouseEvent t) {
    KeyValue collapse = new KeyValue(circle.radiusProperty(), 0);
    new Timeline(new KeyFrame(Duration.seconds(3),
                              collapse)).play();
  }
});
```

Lambda-ized Event Listeners

```
circle.addEventHandler(MouseEvent.MOUSE_CLICKED, e -> {
  KeyValue collapse = new KeyValue(circle.radiusProperty(), 0);
  new Timeline(new KeyFrame(Duration.seconds(3),
                            collapse)).play();
});
```

Begin the Animation

```
Timeline moveCircles = new Timeline();
for (Circle circle : circles) {
  KeyValue moveX = new KeyValue(circle.centerXProperty(),
                                Math.random() * 800);
  KeyValue moveY = new KeyValue(circle.centerYProperty(),
                                Math.random() * 600);
  moveCircles.getKeyFrames().add(new KeyFrame(Duration.seconds(40),
                                              moveX, moveY));
}
moveCircles.play();
```

Lambda-ized Begin the Animation

```
Timeline moveCircles = new Timeline();
circles.forEach(circle -> {
  KeyValue moveX = new KeyValue(circle.centerXProperty(),
                                Math.random() * 800);
  KeyValue moveY = new KeyValue(circle.centerYProperty(),
                                Math.random() * 600);
  moveCircles.getKeyFrames().add(new KeyFrame(Duration.seconds(40),
                                              moveX, moveY));
});
moveCircles.play();
```

Mary Had a Little Lambda

Mary had a little lambda
Whose fleece was white as snow
And everywhere that Mary went
Lambda was sure to go!

https://github.com/steveonjava/MaryHadALittleLambda

Generating Streams

From a collection:

```
anyCollection.stream();
```

Known set of objects:

```
Stream.of("bananas", "oranges", "apples");
```

Numeric range:

```
IntStream.range(0, 50)
```

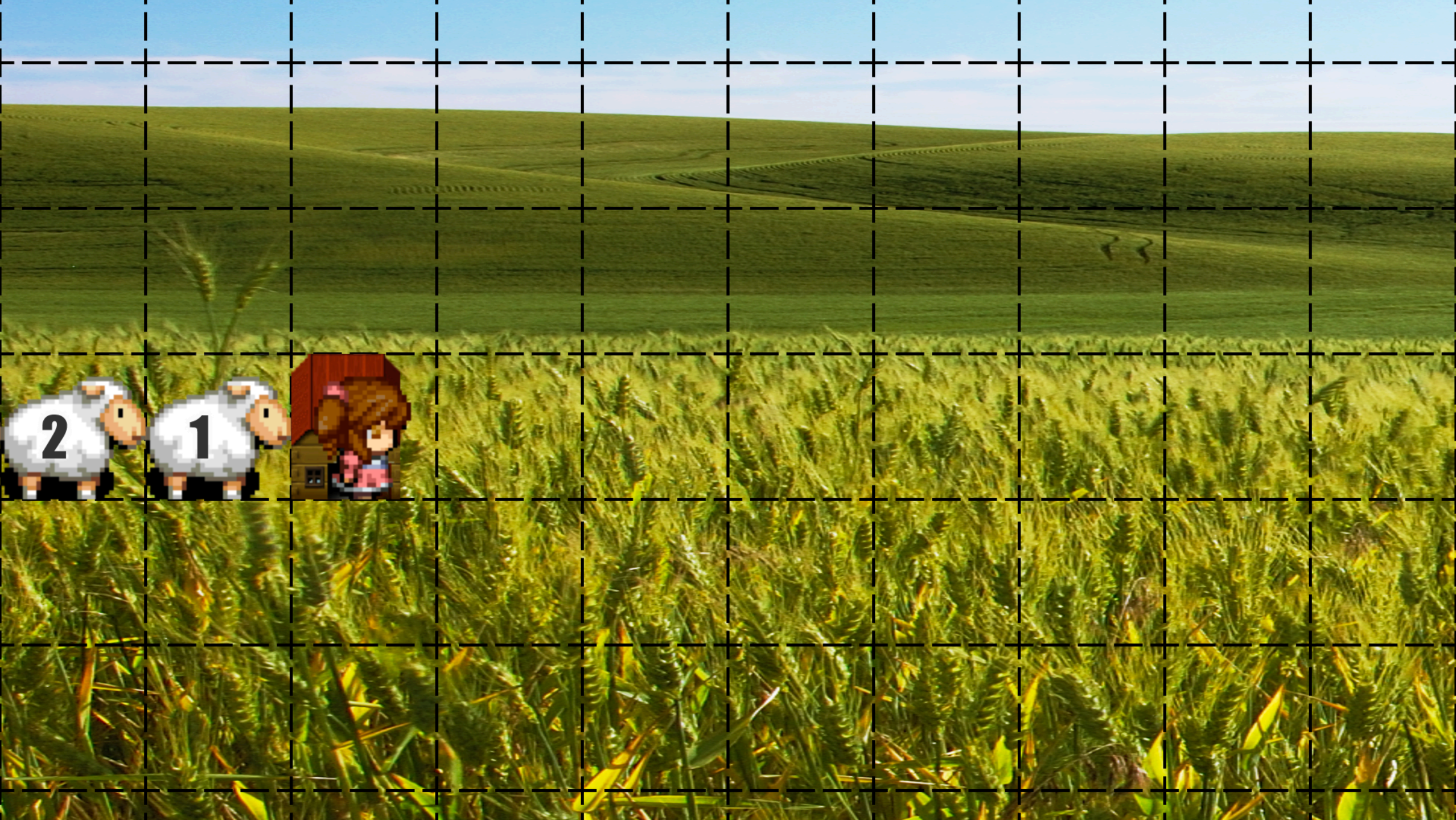
Iteratively:

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

Let's Create Some Barn Animals!

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

Stream.iterate(tail, SpriteView.Lamb::new)
    .skip(1).limit(7)
    .forEach(s.getAnimals()::add);
```


2
1

Filtering Streams

Predicate Expression

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

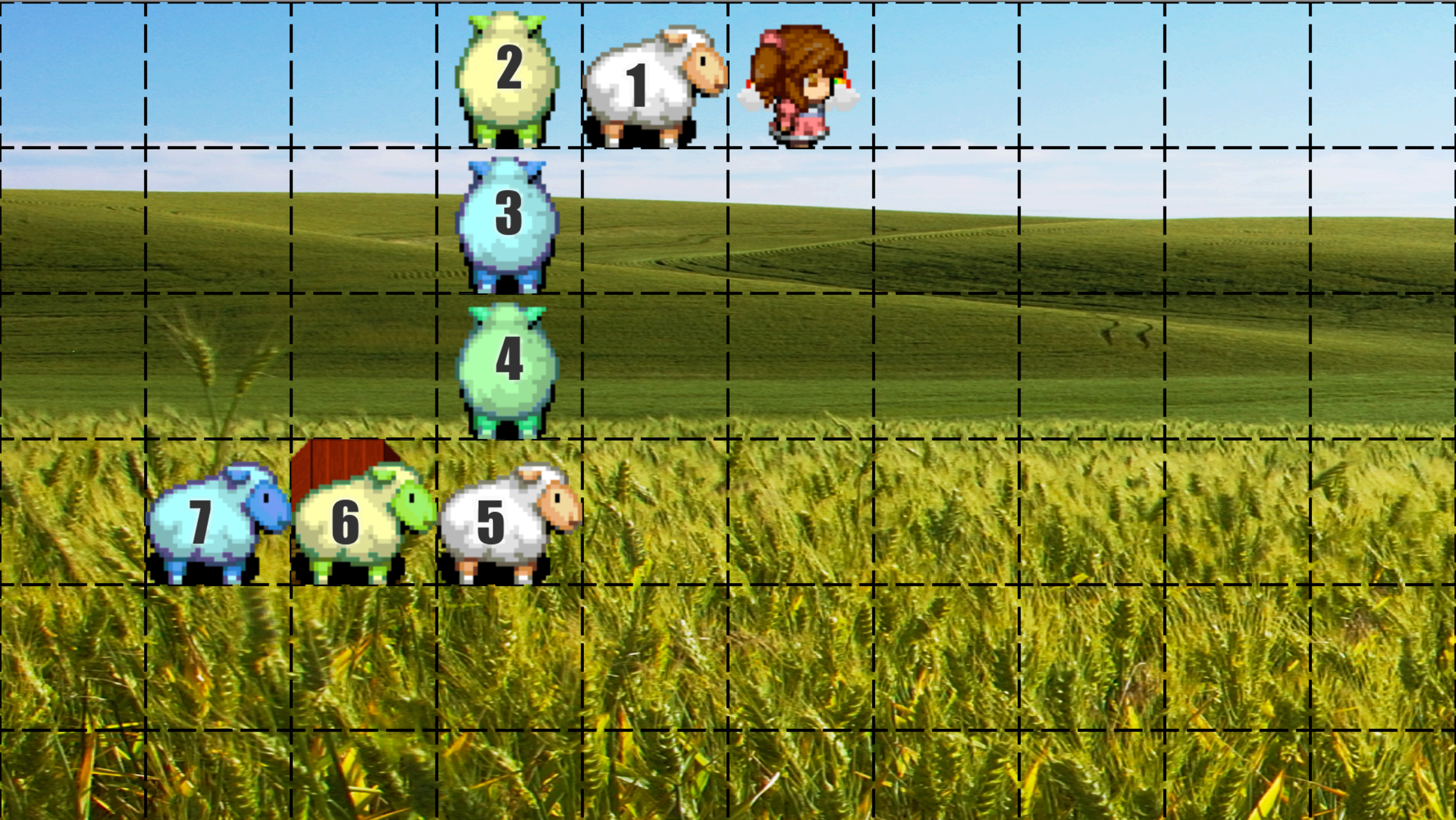
Filter out minors

```
> hackers = attendees.filter(a -> a.getAge() >= 1.8)
```


TWINLIFE
ORACLE

Rainbow-colored Lambs!

```
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

Filtering Collections

Collection.removeIf

- Removes all elements that match the predicate

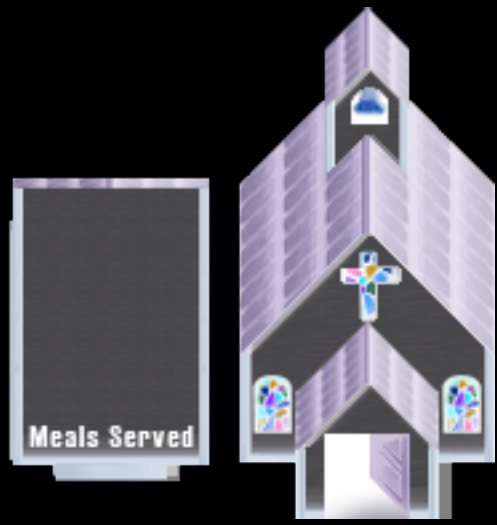
List.replaceAll

- In-place filtering and replacement using an unary operator

ObservableCollection.filtered

- Returns a list filtered by a predicate this is also Observable

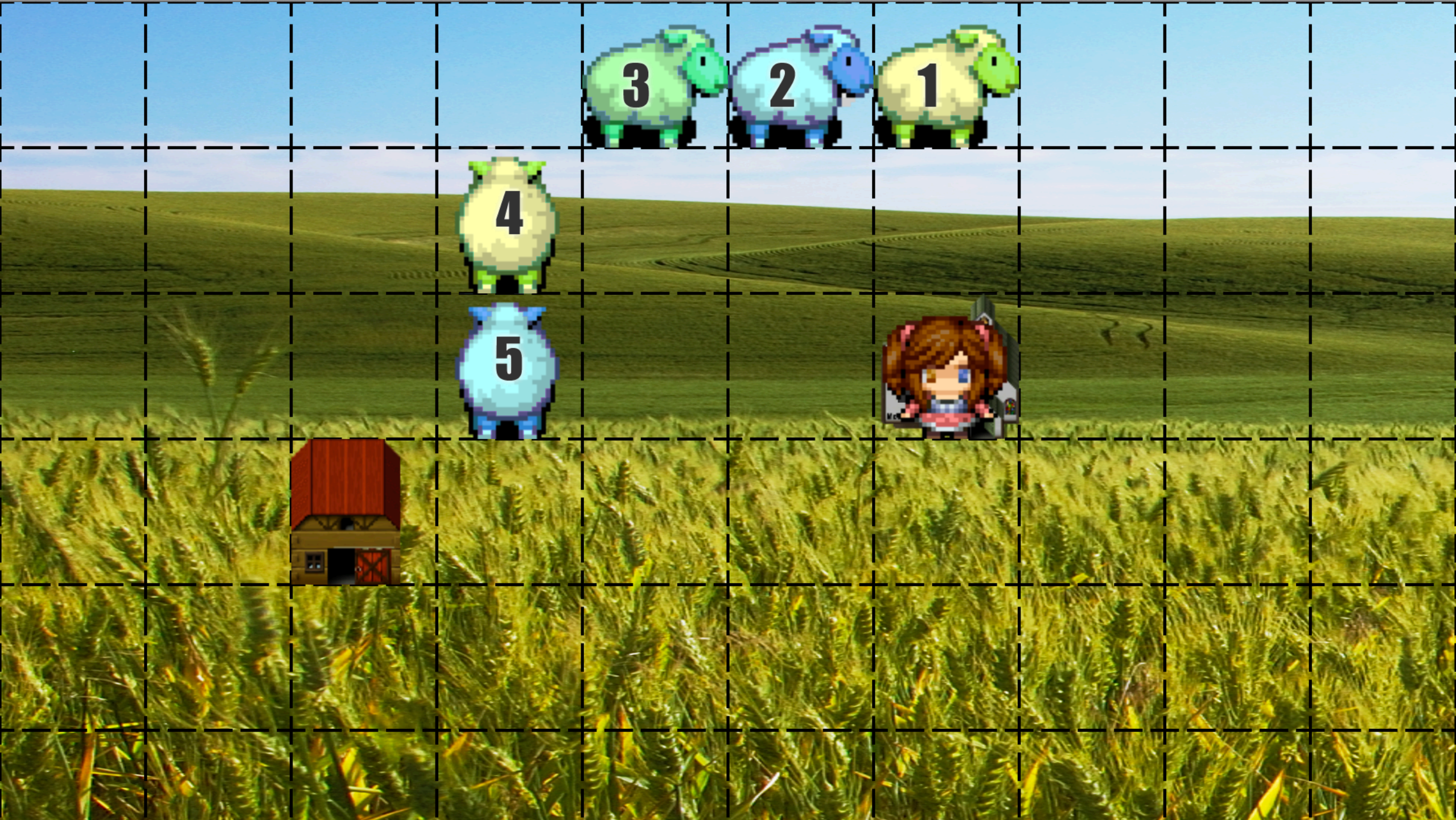
Picky Eaters…




```
Predicate<SpriteView> pure =
  a -> a.getColor() == null;

mealsServed.set(mealsServed.get() +
  s.getAnimals().filtered(pure).size()
);

s.getAnimals().removeIf(pure);
```


3
2
1
4
5

Mapping Streams

Applies a Map Function to each element:

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

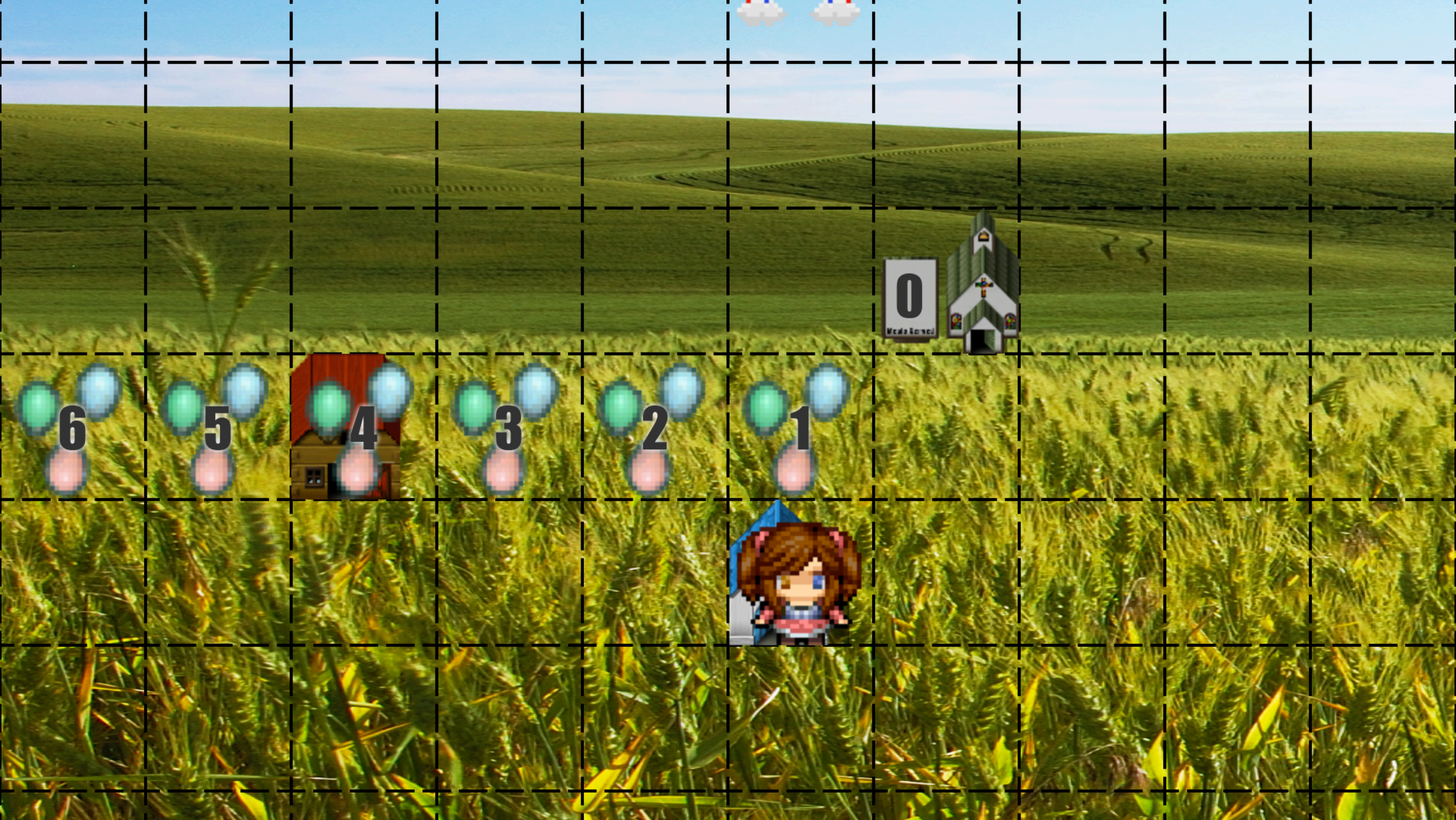
Result: List is the same size, but may be a different type.

Single Map

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

Or a Double Map!

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


6
5
4
3
2
1
0

Flat Map

Applies a One-to-Many Map Function to each element:

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

And then flattens the result into a single stream.

Result: The list may get longer and the type may be different.

Hatching Eggs

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


9
8
0
20
19
16
13
10
7
4
1
2
3

Reduce

Reduces a list to a single element given:

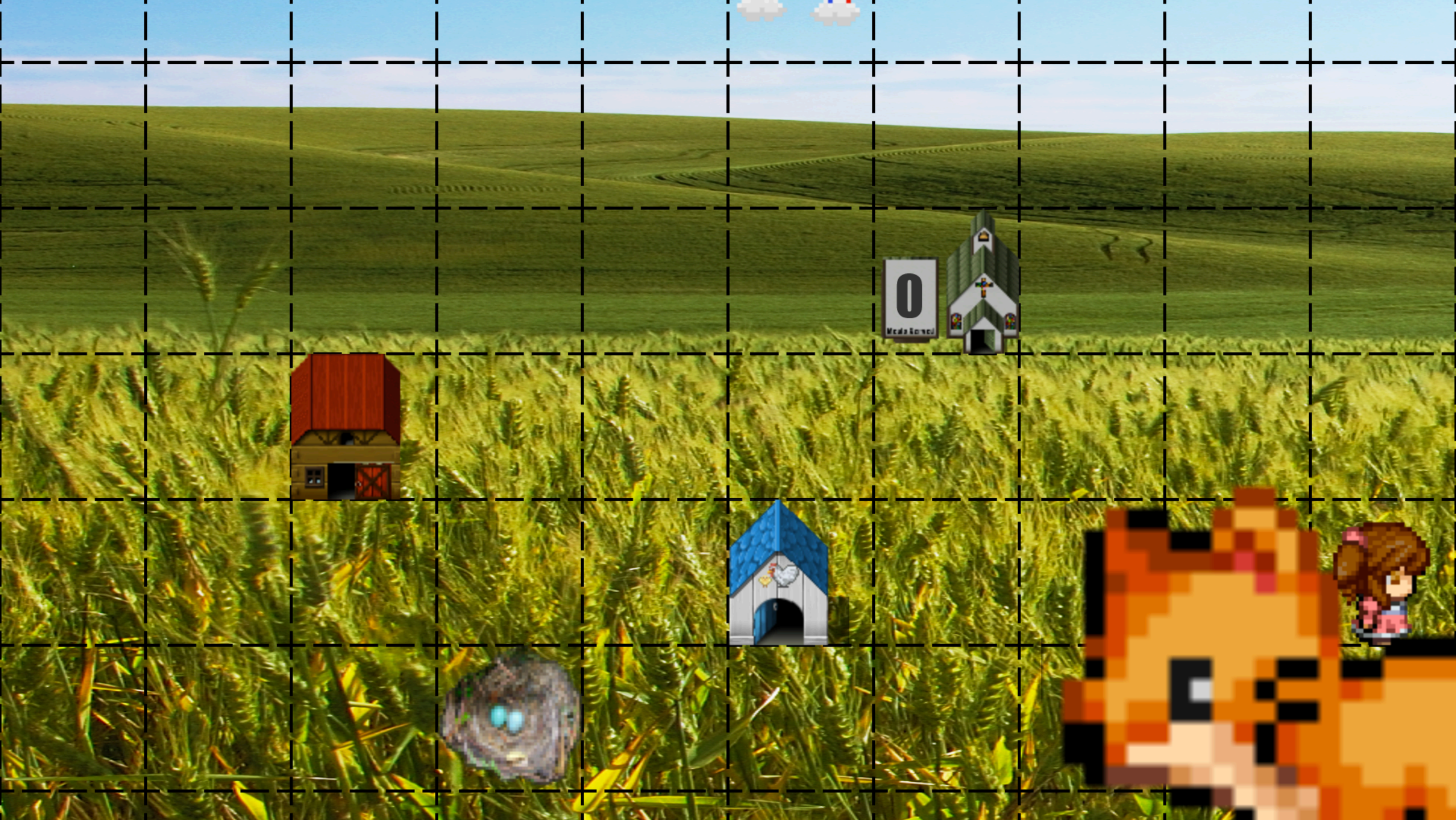
- Identity: T
- Accumulator: BinaryOperator<T>

Result: List of the same type, but only 1 element left.

And the (formerly little) Fox ate them all!

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

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


0

Parallel Streams

Apply parallel() to any stream to optimize for multicore

- Collection.parallelStream() is a convenient shortcut
- Use sequential() to get back a normal stream
- Sometimes unordered() can improve performance

Sequential and parallel streams should give the same results as long as:

- No Interference – You don't modify the data source
- Stateless Behavior – No side effects

Parallel walking…

```
animals.parallelStream()
    .reduce(location.get(), (loc, sprt) -> {
        sprt.moveTo(loc);
        return sprt.location.get();
    }, (loc1, loc2) -> loc1);
```


3
2
1
0
WUT?

Mary Had a Little Lambda Project

- Open-source project to demonstrate lambda features
- Visual representation of streams, filters, and maps

https://github.com/steveonjava/MaryHadALittleLambda

Stephen Chin
tweet: @steveonjava
blog: http://steveonjava.com

NightHacking Tour

Real Geeks
Live Hacking

nighthacking.com

Graphics Image Credits

- Mary image by Terra-chan: http://www.rpgmakervx.net/index.php?showtopic=29404
- Lamb image by Mack: http://www.rpgmakervx.net/index.php?showtopic=15704
- Chicken, Church, Barn, and Coop image by LovelyBlue: http://l0velyblue.deviantart.com/art/Chicken-203764427
- Fox image by PinedaVX: http://www.rpgmakervx.net/index.php?showtopic=9422
- Nest image derived from Lokilech's Amselnest: http://commons.wikimedia.org/wiki/File:Amselnest_lokilech.jpg
- Background image by Victor Szalvay: http://www.flickr.com/photos/55502991@N00/172603855

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.