Introduction to functional programming with Java

Falk Sippach

@sippsack

http://blog.oio.de

JAVA FORUM NORD

BASEL I BERN I BRUGG I BUKAREST I DÜSSELDORF I FRANKFURT A.M. I FREIBURG I.BR. I GENF
HAMBURG I KOPENHAGEN I LAUSANNE I MANNHEIM I MÜNCHEN I STUTTGART I WIEN I ZÜRICH

part of trivadis group

Falk Sippach

Trainer, Berater, Entwickler

Co-Organisator

Commiter DukeCon

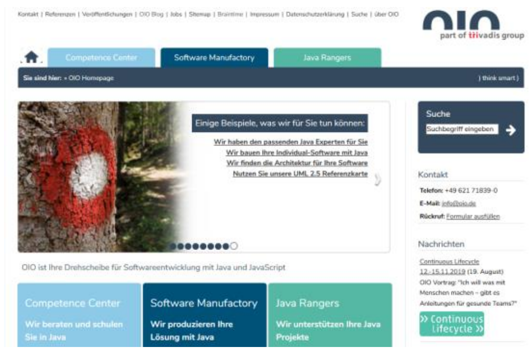
@sippsack

http://blog.oio.de

BASEL I BERN I BRUGG I BUKAREST I DÜSSELDORF I FRANKFURT A.M. I FREIBURG I.BR. I GENF
HAMBURG I KOPENHAGEN I LAUSANNE I MANNHEIM I MÜNCHEN I STUTTGART I WIEN I ZÜRICH

part of trivadis group

OIO – Part of Trivadis


Kontakt | Referenzen | Veröffentlichungen | OIO Blog | Jobs | Sitemap | Braintime | Impressum | Datenschutzerklärung | Suche | über OIO
part of trivadis group
Competence Center
Software Manufactory
Java Rangers
Sie sind hier: » OIO Homepage
{ think smart }
Suche
Suchbegriff eingeben
Kontakt
Nachrichten
» Continuous Lifecycle »

Einige Beispiele, was wir für Sie tun können:
Wir haben den passenden Java Experten für Sie
Wir bauen Ihre Individual-Software mit Java
Wir finden die Architektur für Ihre Software
Nutzen Sie unsere UML 2.5 Referenzkarte

OIO ist Ihre Drehscheibe für Softwareentwicklung mit Java und JavaScript

Competence Center – Wir beraten und schulen Sie in Java
Software Manufactory – Wir produzieren Ihre Lösung mit Java
Java Rangers – Wir unterstützen Ihre Java Projekte

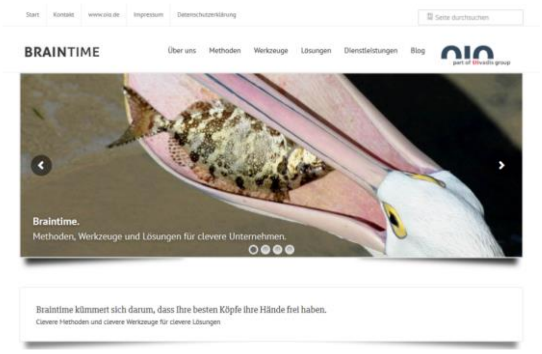
Telefon: +49 621 71839-0
E-Mail: info@oio.de
Rückruf: Formular ausfüllen

Continuous Lifecycle 12.-15.11.2019 (19. August)
OIO Vortrag: "Ich will was mit Menschen machen – gibt es Anleitungen für gesunde Teams?"

http://www.oio.de

OIO ist die Drehscheibe der Trivadis-Gruppe
für Softwareentwicklung mit Java und JavaScript

Schulung, Beratung und Programmierung



Braintime.
Methoden, Werkzeuge und Lösungen für clevere Unternehmen.

Braintime kümmert sich darum, dass Ihre besten Köpfe ihre Hände frei haben.
Clevere Methoden und clevere Werkzeuge für clevere Lösungen

http://www.braintime.de

OIO Braintime
Ihr Atlassian Platinum Solution Partner

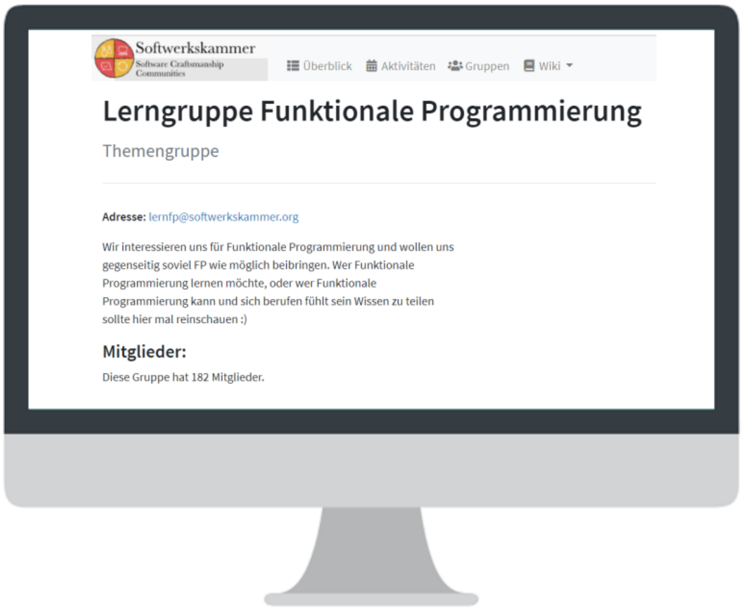
Methoden, Werkzeuge und Lösungen

Abstract

Funktionale Programmierung ist im Moment in aller Munde. Seit Version 8 und Lambdas/Streams stehen auch Java-Anwendern diverse Werkzeuge zur Verfügung. Daher wird es Zeit, sich mit den grundlegenden Konzepten der funktionalen Programmierung auseinanderzusetzen.

Nach diesem Vortrag wirst Du verstehen, was eine pure Funktion ist und warum referentielle Transparenz bzw. Seiteneffektfreiheit wichtige Konzepte sind. Wir schauen zudem auf Value Types und wie funktionale Datenstrukturen aufgebaut sind und wie man dank Bedarfsauswertung auch mit sehr großen Datenmengen effizient umgehen kann. Weiterhin besprechen wir die Elemente der Wiederverwendung wie Funktionskomposition, Currying, partielle Funktionsaufrufe und Funktionen höherer Ordnung. Abschließend werfen wir noch ein Blick auf die Destrukturierung von Datenstrukturen mittels Pattern Matching, das Kapseln von Seiteneffekten und wie man in seiner Softwarearchitektur einen funktionalen Kern umsetzt.

trivadis



Softwerkskammer
Software Craftsmanship Communities

Überblick Aktivitäten Gruppen Wiki

Lerngruppe Funktionale Programmierung

Themengruppe

Adresse: lernfp@softwerkskammer.org

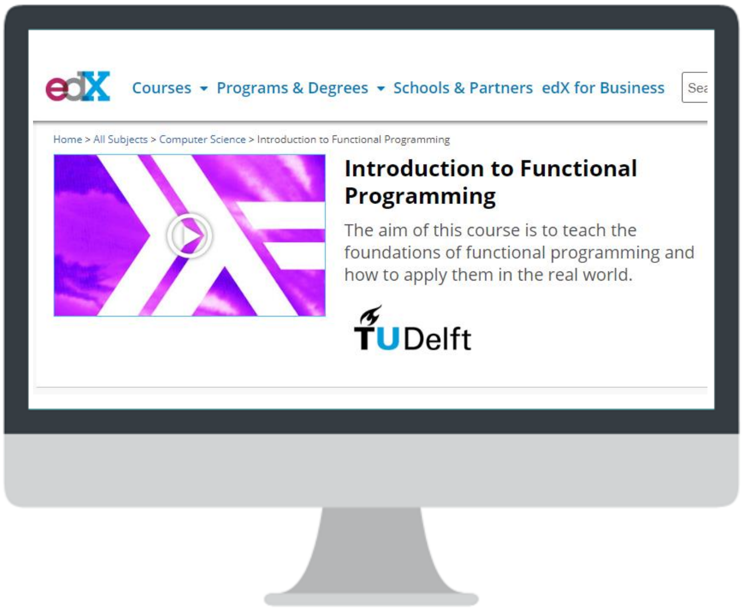
Wir interessieren uns für Funktionale Programmierung und wollen uns gegenseitig soviel FP wie möglich beibringen. Wer Funktionale Programmierung lernen möchte, oder wer Funktionale Programmierung kann und sich berufen fühlt sein Wissen zu teilen sollte hier mal reinschauen :)

Mitglieder:

Diese Gruppe hat 182 Mitglieder.

5

trivadis



edX Courses Programs & Degrees Schools & Partners edX for Business Sea

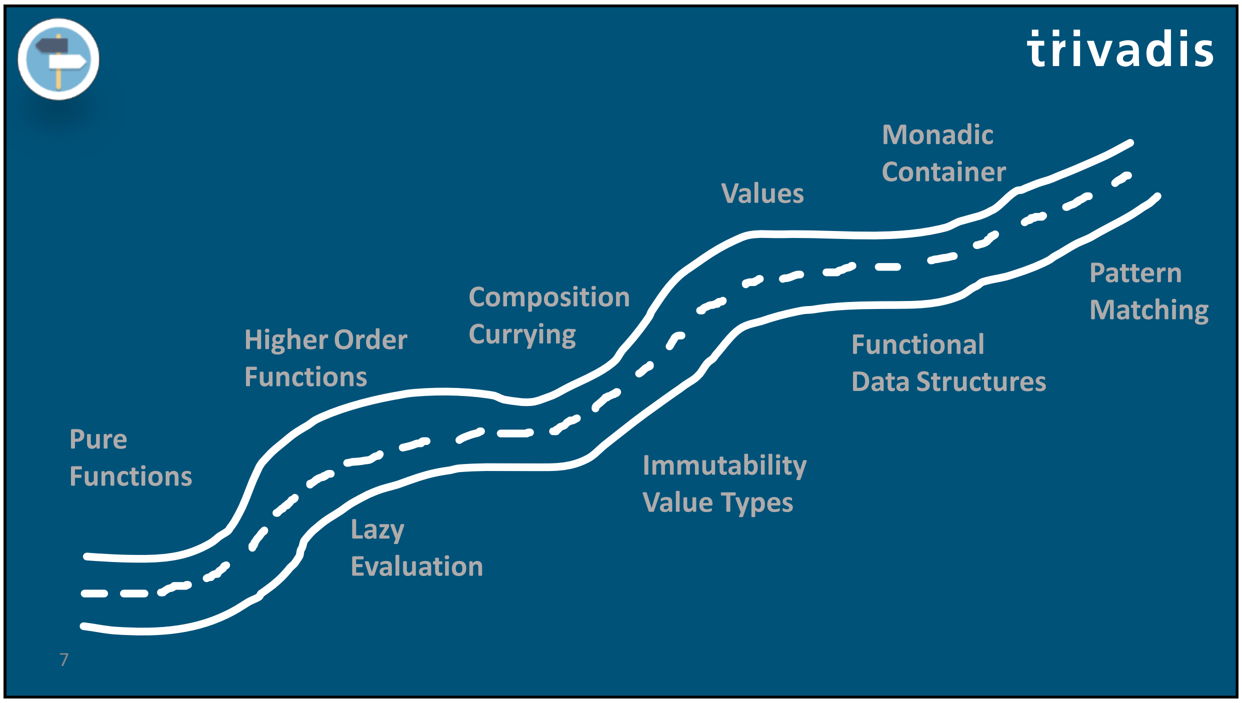
Home > All Subjects > Computer Science > Introduction to Functional Programming

Introduction to Functional Programming

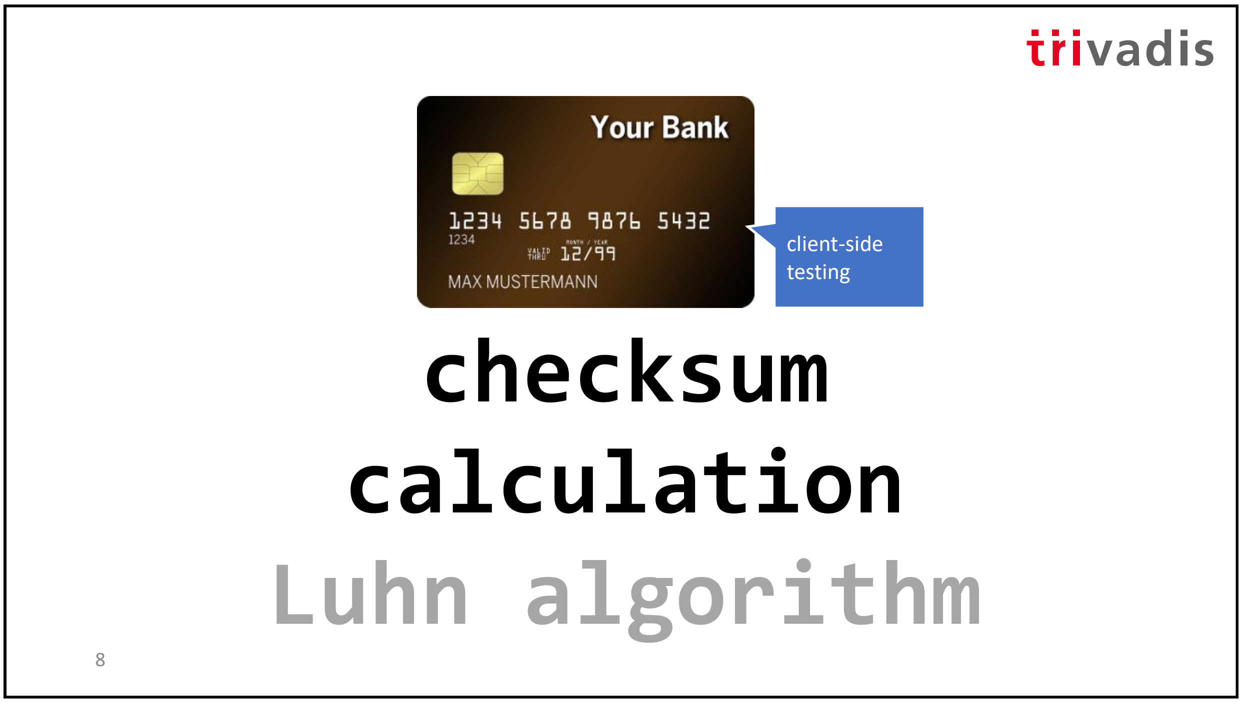
The aim of this course is to teach the foundations of functional programming and how to apply them in the real world.

TUDelft

6



7

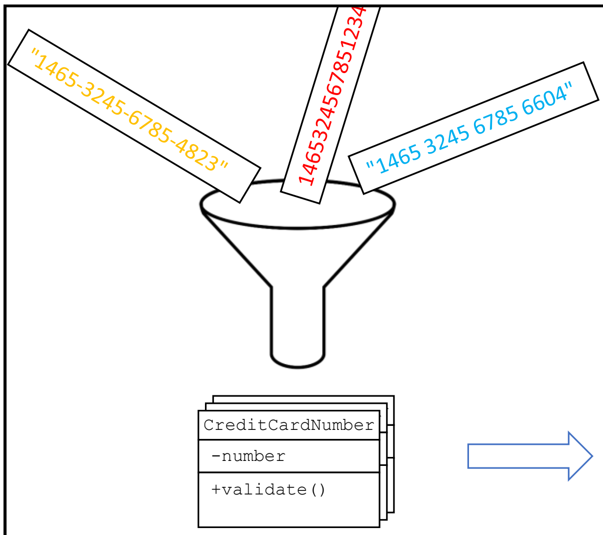

trivadis
Your Bank
1234 5678 9876 5432
1234
VALID THRU 12/99
MAX MUSTERMANN
client-side testing

checksum calculation

Luhn algorithm

8

trivadis

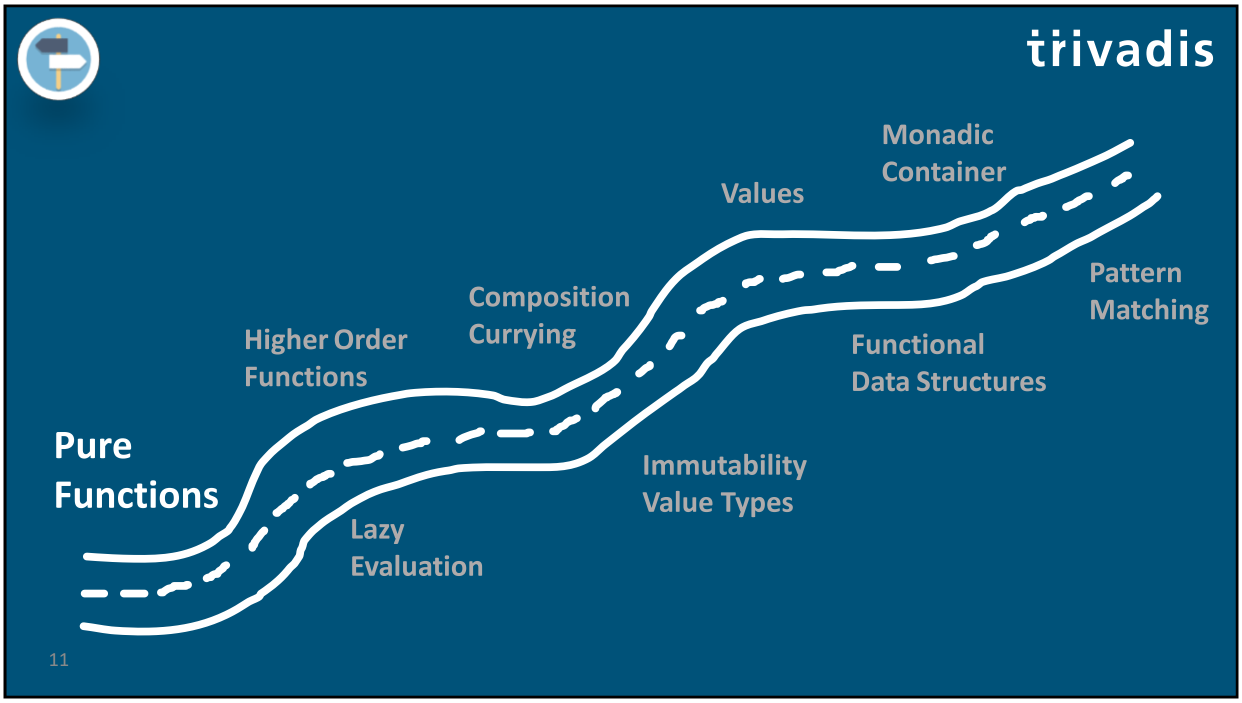


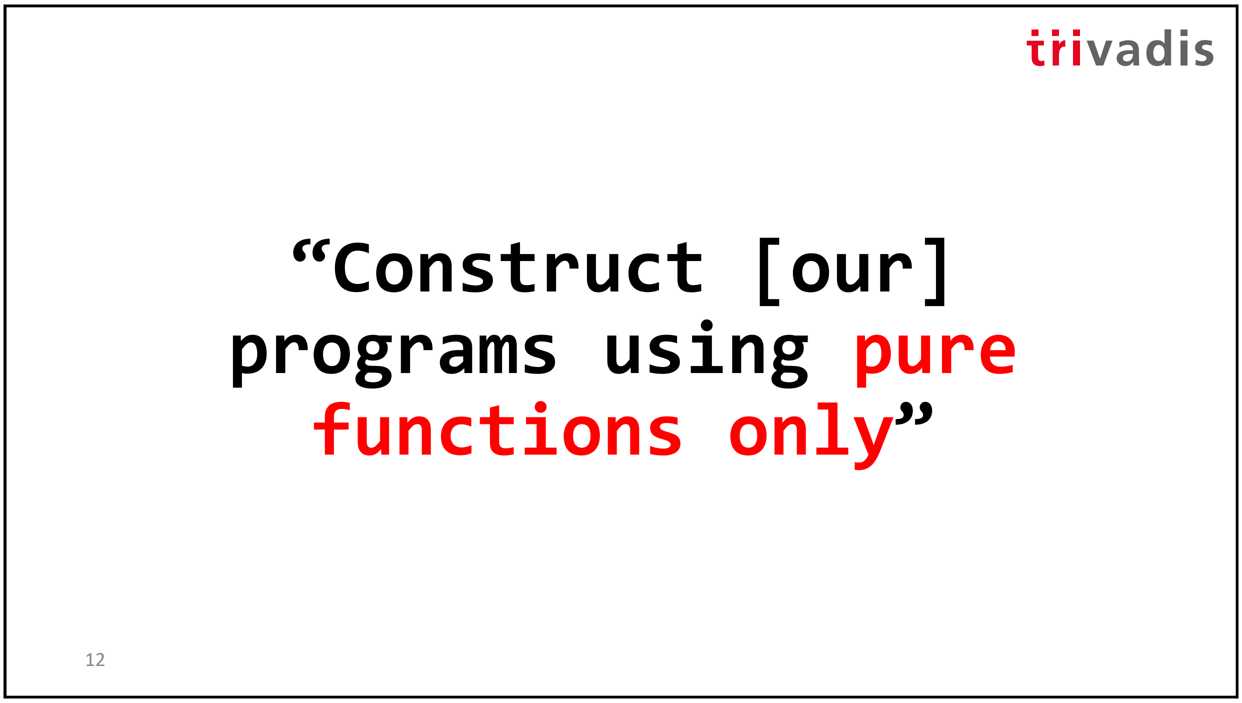
Nummer	gültig
1465324567854823	✓
1465324567851234	☐
1465324567856604	✓

9

trivadis

Java 8 + VAVR.io




trivadis
12

“Construct [our] programs using pure functions only”

Luhn-Algorithmus

trivadis

```
static Function1<Long, Seq<Integer>> toDigits = number ->
    CharSeq.of(Long.toString(number)).map(c -> c - '0');

static Function1<Seq<Integer>, Seq<Integer>> reverse = Seq::reverse;

static Function1<Seq<Integer>, Seq<Integer>> double2nd =
    digits -> digits.zipWithIndex().map(t -> t._1 * (t._2 % 2 + 1));

static Function1<Seq<Integer>, Integer> sumDigits = digits ->
    digits.map(i -> i.longValue()).flatMap(toDigits).sum().intValue();

static Function1<Integer, Boolean> divisibleBy10 = number ->
    number % 10 == 0;
```

13

trivadis

“Pure functions have no side effects”

14

trivadis

Modifying a variable
Throwing an exception
Printing to the console
Writing to a file

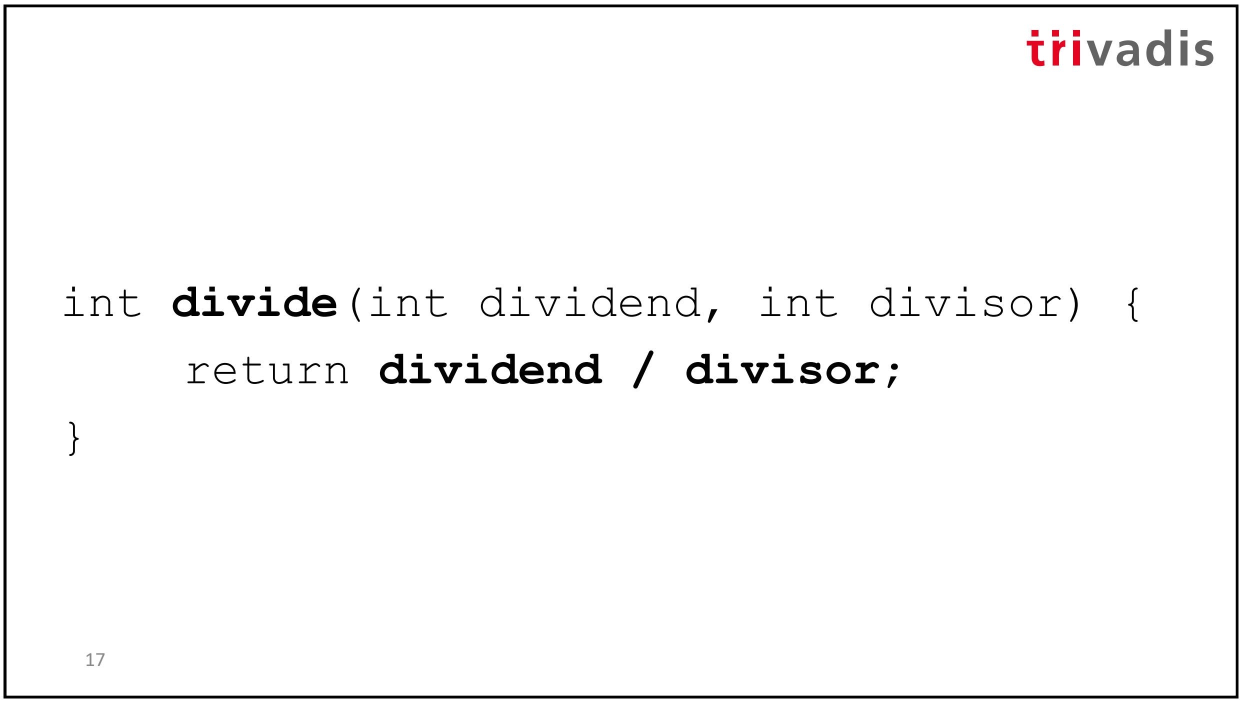
Side effects

15

trivadis

“A function has a side effect if it does something other than simply return a result”

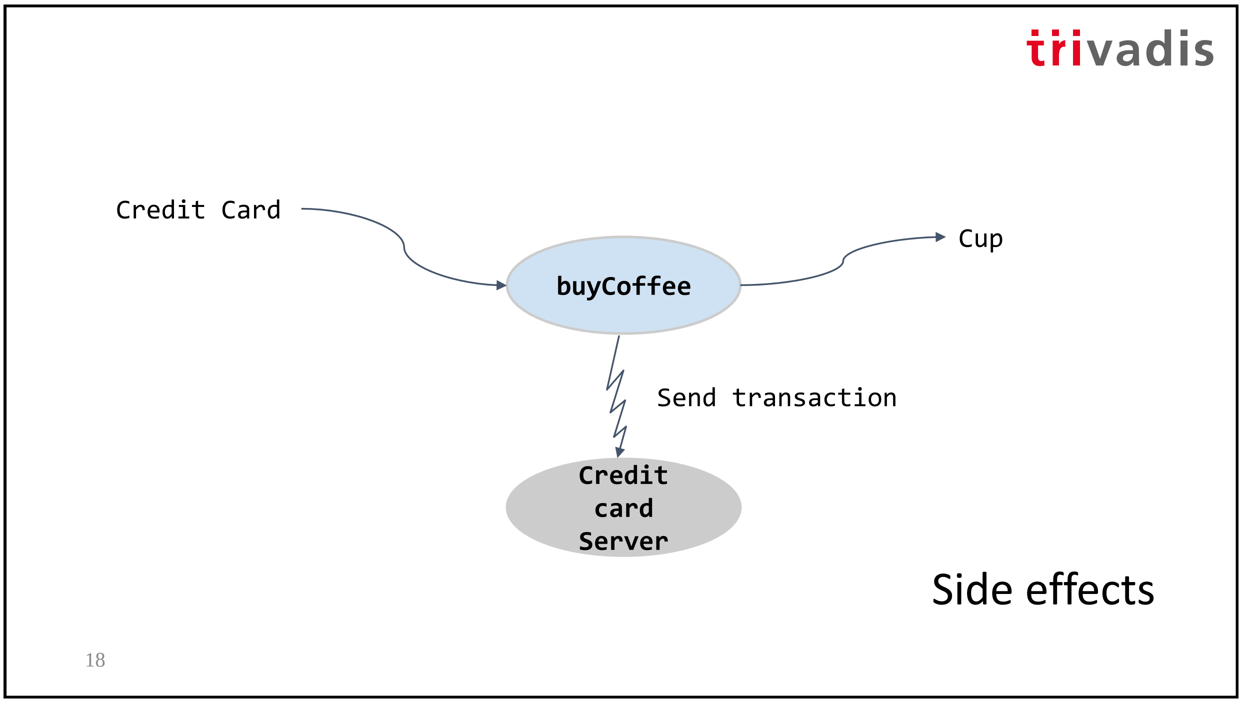
16



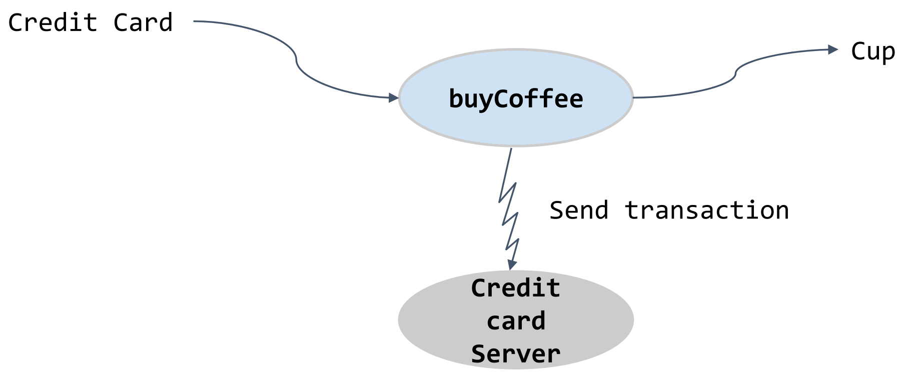
trivadis

```
int divide(int dividend, int divisor) {
    return dividend / divisor;
}
```

17

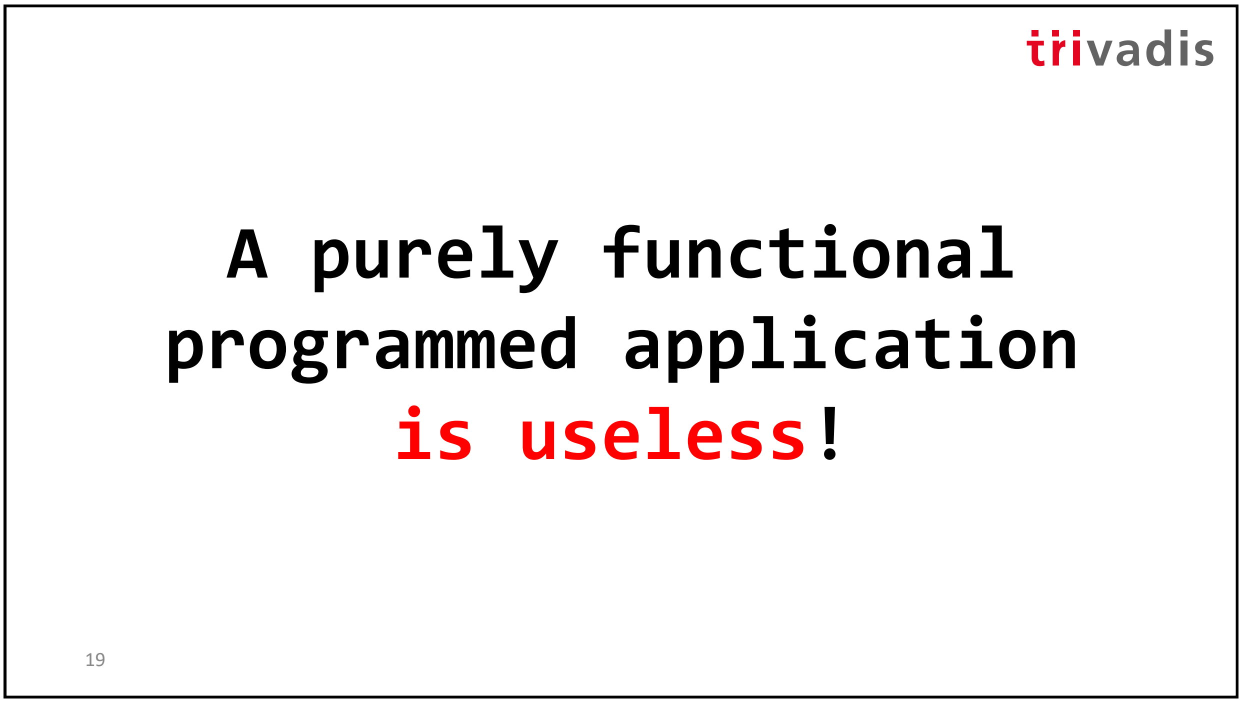


trivadis



Side effects

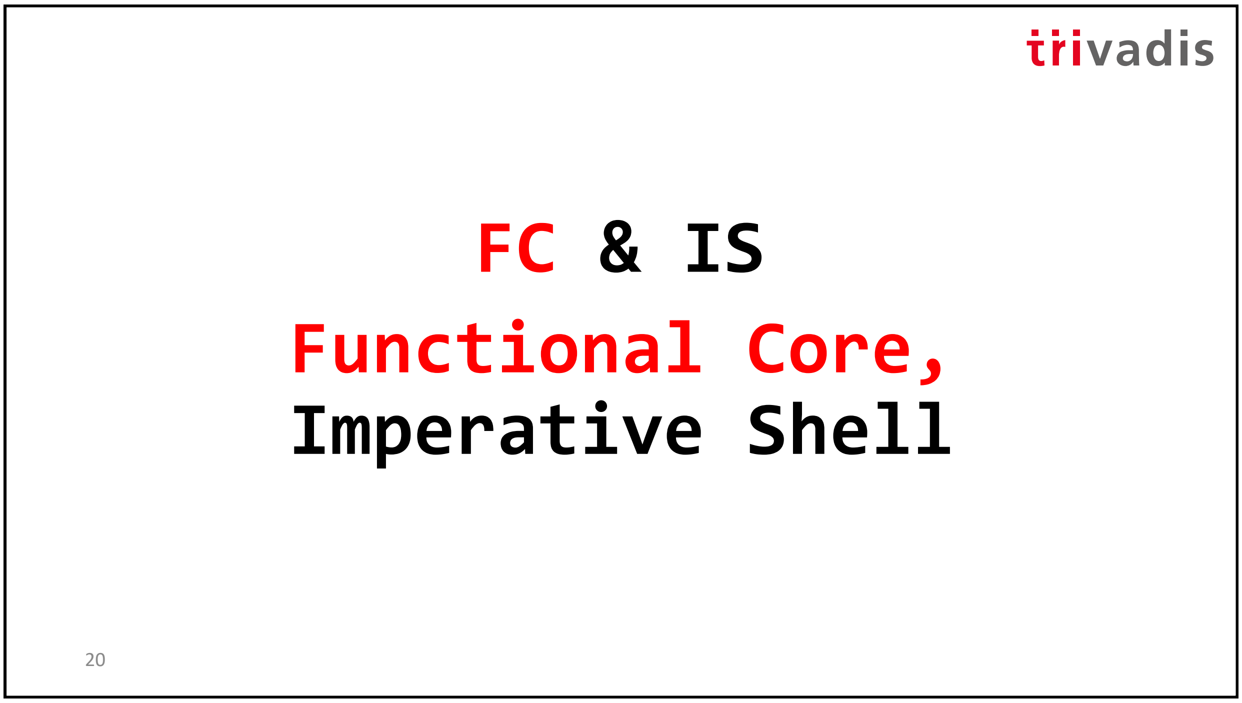
18



trivadis

A purely functional programmed application is useless!

19



trivadis

FC & IS

Functional Core, Imperative Shell

20

trivadis

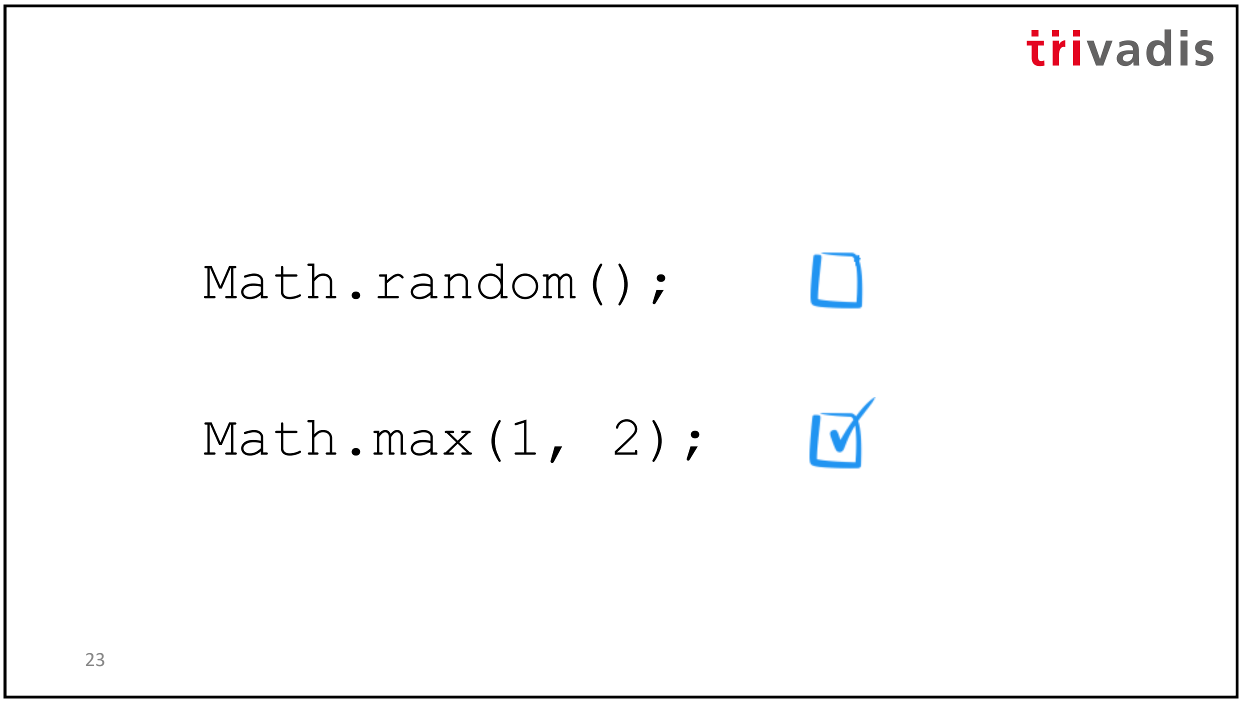
“An expression is referential transparent, if it can be replaced by its result without changing the meaning of the program”

21

trivadis

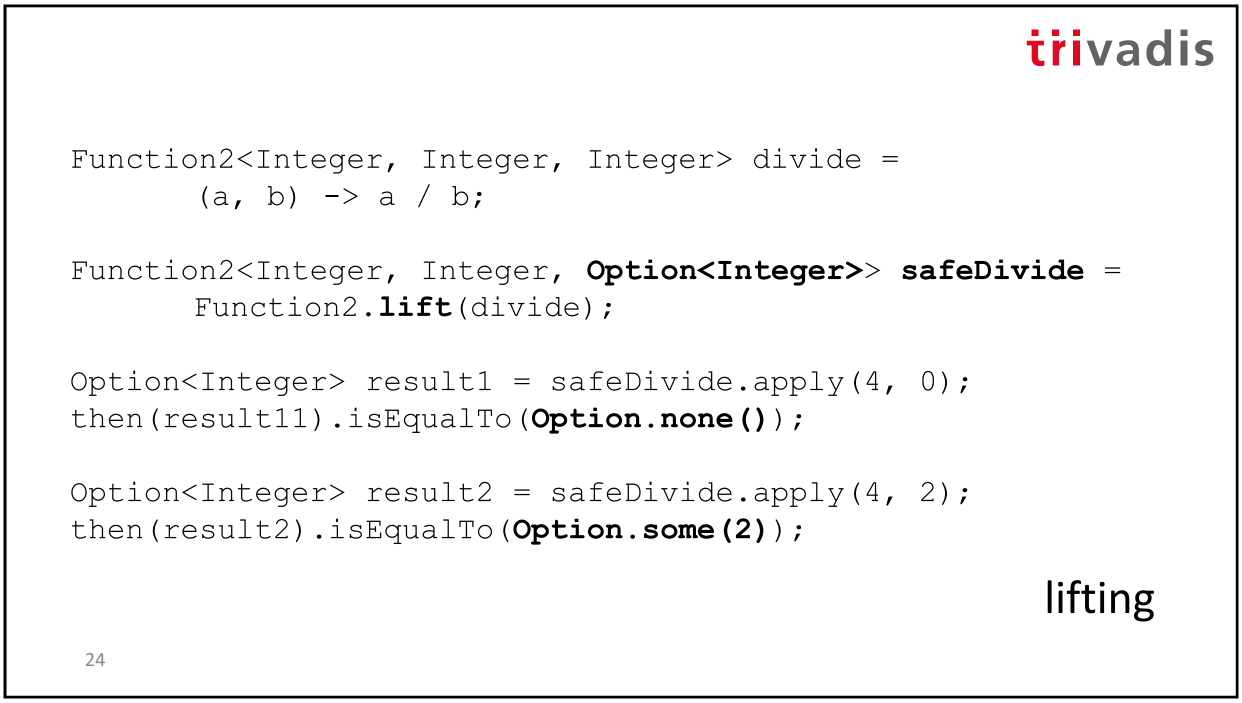
“A function **is pure** if **calling** it with **referential transparent arguments** is also **referential transparent**”

22



```
Math.random();    ☐

Math.max(1, 2);   ☑
```

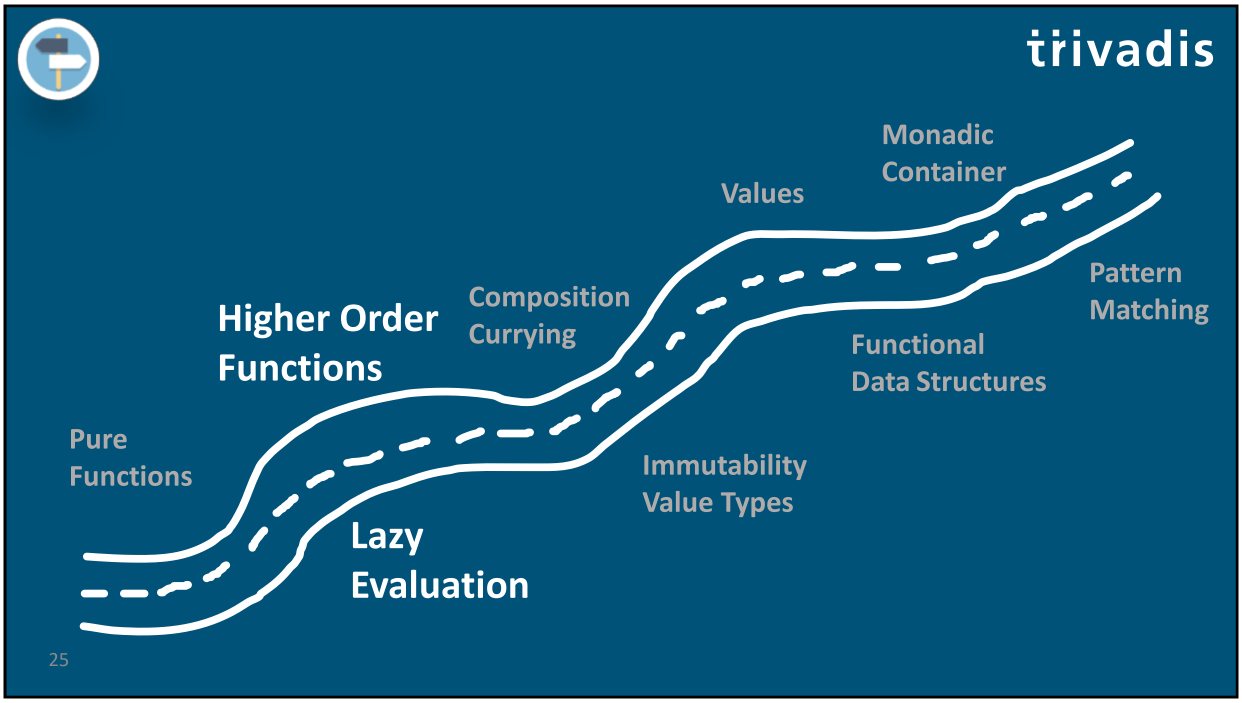
23

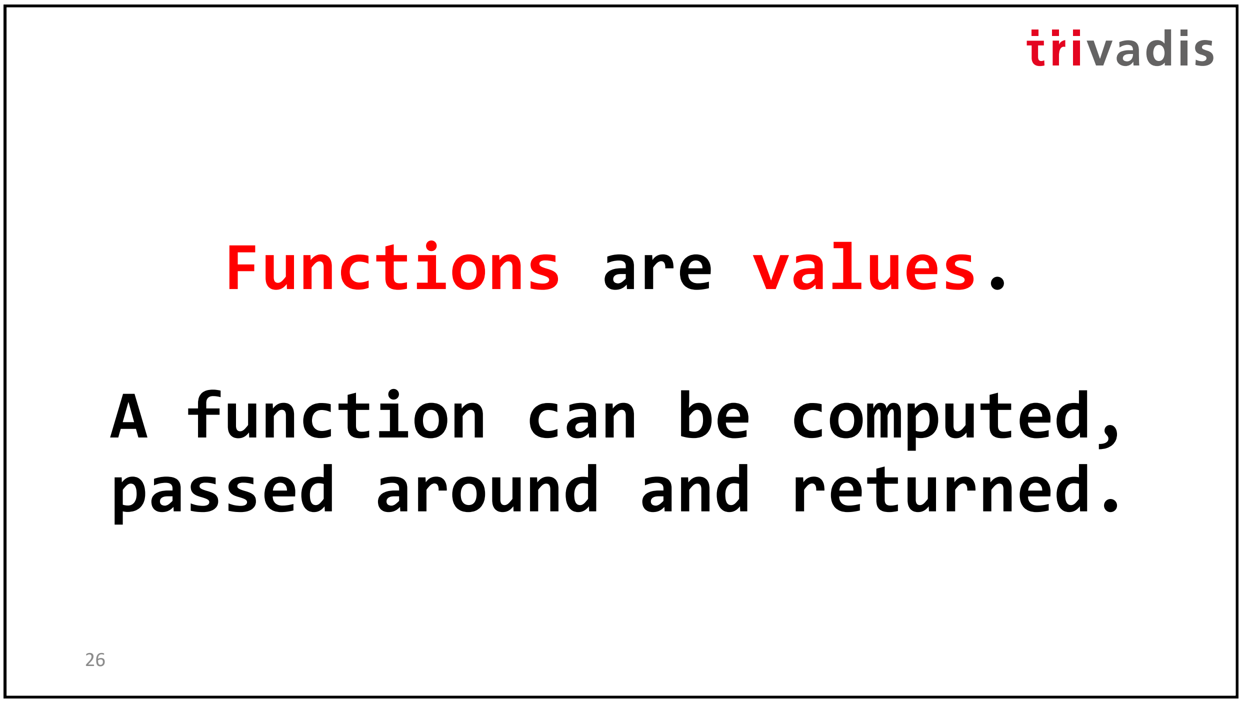


```
Function2<Integer, Integer, Integer> divide =
       (a, b) -> a / b;

Function2<Integer, Integer, Option<Integer>> safeDivide =
       Function2.lift(divide);

Option<Integer> result1 = safeDivide.apply(4, 0);
then(result11).isEqualTo(Option.none());

Option<Integer> result2 = safeDivide.apply(4, 2);
then(result2).isEqualTo(Option.some(2));
```

lifting

24




trivadis

Functions are values.

A function can be computed, passed around and returned.

26

trivadis

“A higher order function is a function that takes a function as an argument and/or returns a function.”

trivadis

```
creditCardNumber
    .chars() // convert to int
    .map(in -> in - '0') // multiply by 1, 2 alternating
    .map(n -> n * (i[0] = i[0] == 1 ? 2 : 1)) // sum of digits
    .map(n -> n > 9 ? n - 9 : n)
    .sum() % 10 == 0;
```

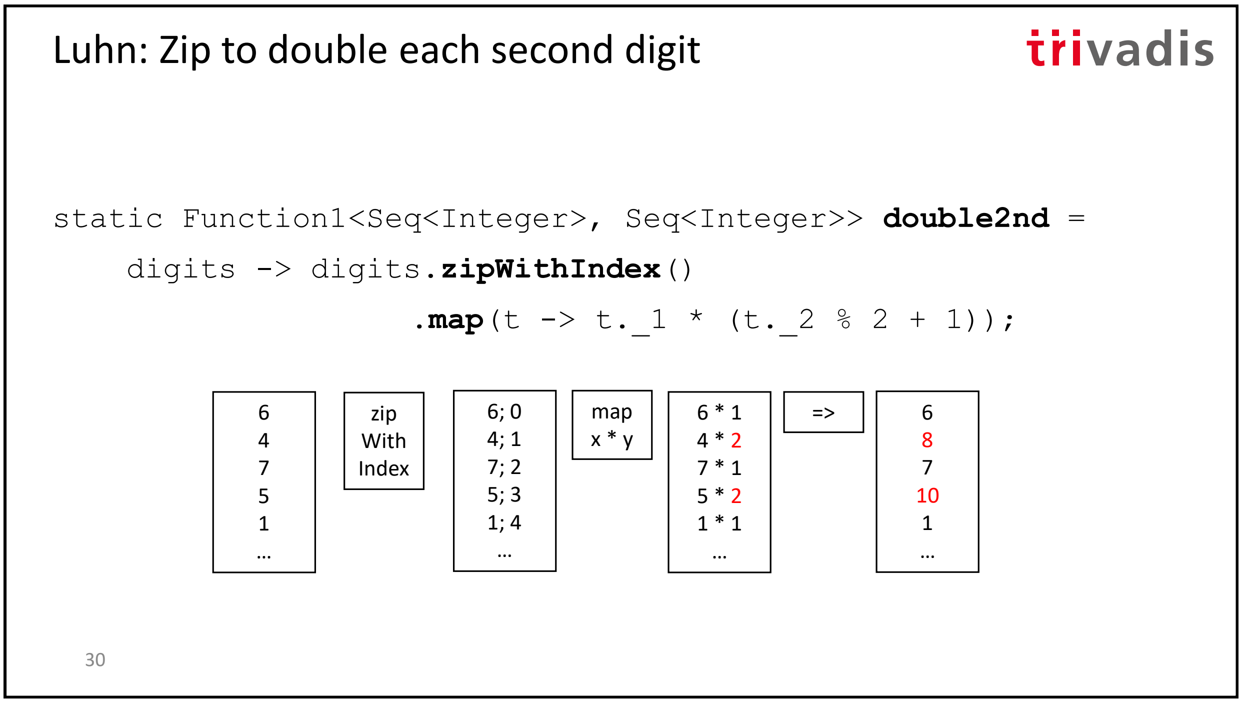
higher order functions

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
…

```
final Stream<String> fizzes = Stream.of("", "", "Fizz").cycle();
final Stream<String> buzzes = Stream.of("", "", "", "", "Buzz").cycle();
final Stream<String> fizzBuzzes = fizzes.zipWith(buzzes, (t1, t2) -> t1 + t2);
final Stream<String> result = fizzBuzzes
        .zipWith(Stream.from(1), (_1, _2) -> _1.isEmpty() ? _2.toString() : _1);

result.take(20).forEach(System.out::println);
```

Beispiel von: https://www.sitepoint.com/functional-fizzbuzz-with-vavr

https://pxhere.com/de/photo/487335

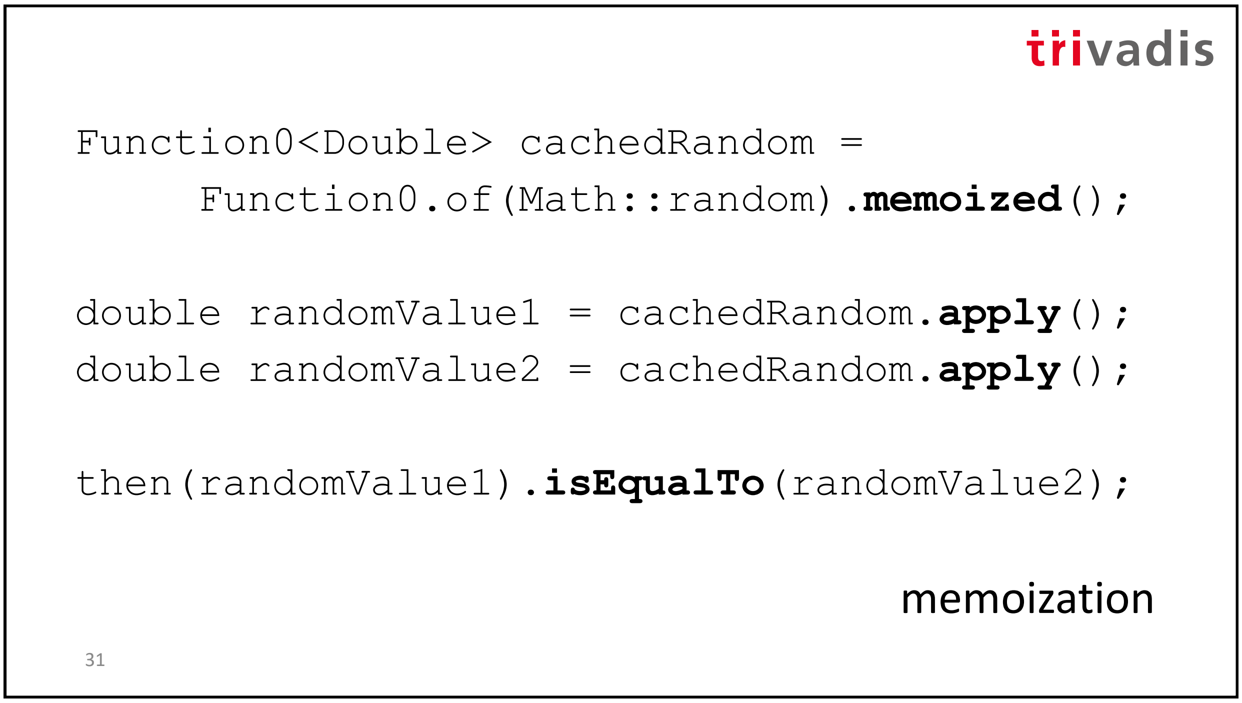


Luhn: Zip to double each second digit

trivadis

```
static Function1<Seq<Integer>, Seq<Integer>> double2nd =
    digits -> digits.zipWithIndex()
                    .map(t -> t._1 * (t._2 % 2 + 1));
```

30

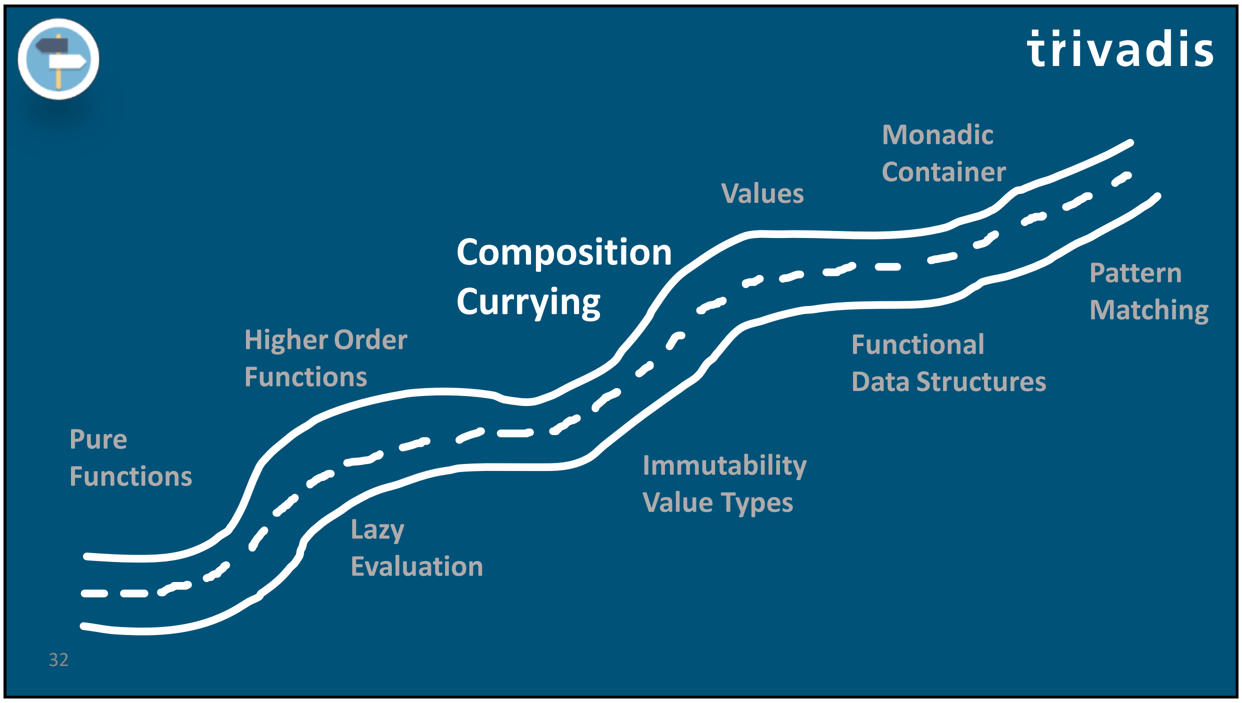


trivadis

```
Function0<Double> cachedRandom =
      Function0.of(Math::random).memoized();

double randomValue1 = cachedRandom.apply();
double randomValue2 = cachedRandom.apply();

then(randomValue1).isEqualTo(randomValue2);
```

memoization

31



trivadis

"... function composition is an act or mechanism to combine simple functions to build more complicated ones."

33

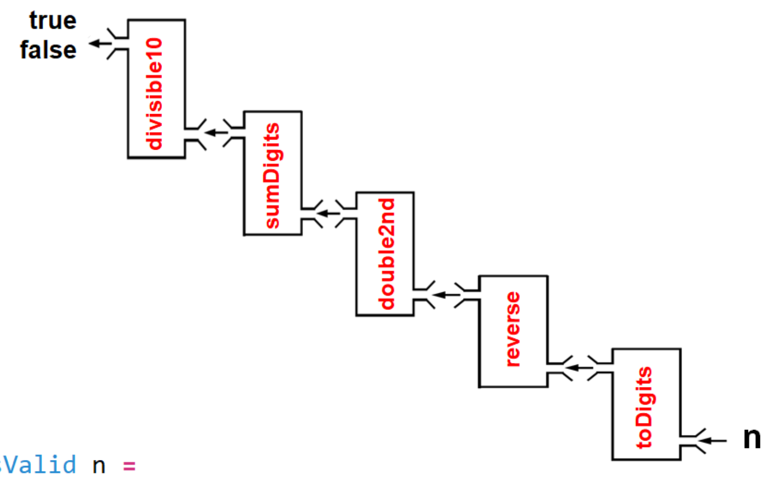
trivadis

```
static Function1<Long, Boolean> isValid =
        toDigits.andThen(reverse)
                .andThen(double2nd)
                .andThen(sumDigits)
                .andThen(divisibleBy10);
```

function composition

34

trivadis



```
isValid n =
divisibleBy10(sumDigits(double2nd(reverse(toDigits(n)))))
```

35

Luhn Algorithmus in funktional

trivadis

```
isValid n =
        divisibleBy10 (
          sumDigits (
            double2nd (
              reverse (
                toDigits(n)
              )
            )
          )
        )
```

- Validierungsfunktion
- Teilbar durch 10?
- Aufsummieren
- jede 2. verdoppeln
- Ziffern umdrehen
- Aufsplitten in Ziffern

36

Luhn: Individual steps

trivadis

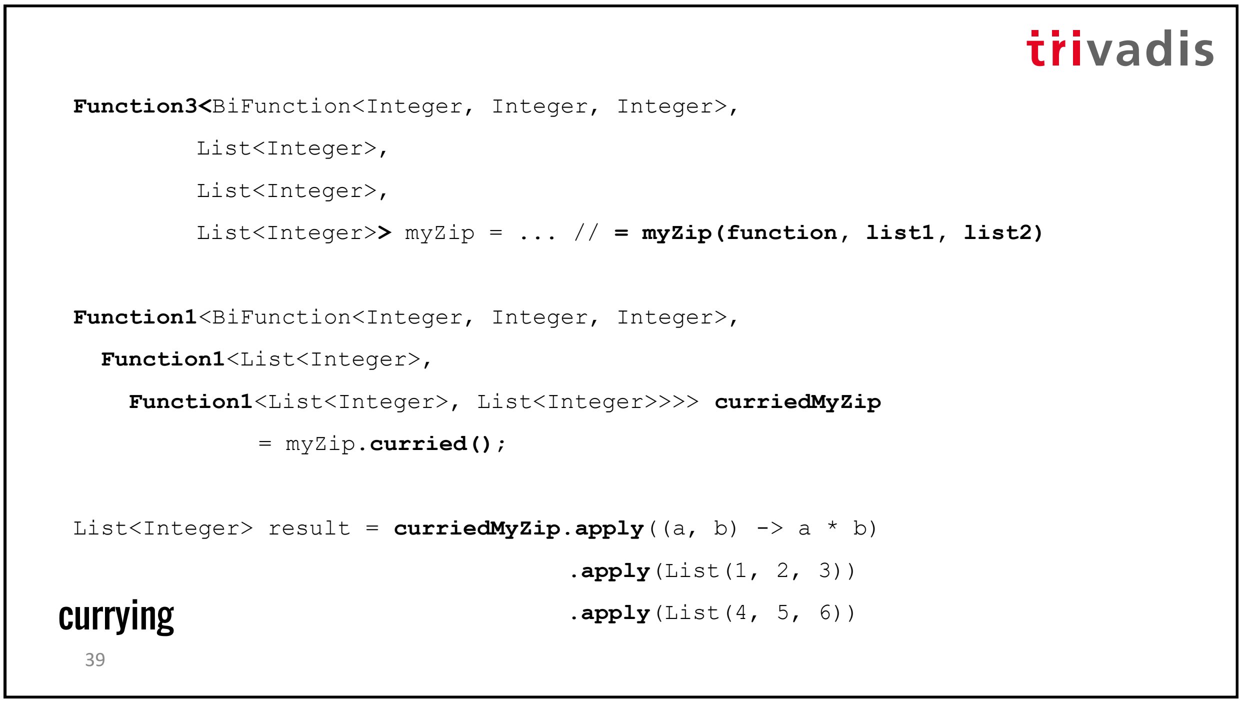
```
static Function1<Long, Seq<Integer>> toDigits = number ->
    CharSeq.of(Long.toString(number)).map(c -> c - '0');

static Function1<Seq<Integer>, Seq<Integer>> reverse = Seq::reverse;

static Function1<Seq<Integer>, Seq<Integer>> double2nd =
    digits -> digits.zipWithIndex().map(t -> t._1 * (t._2 % 2 + 1));

static Function1<Seq<Integer>, Integer> sumDigits = digits ->
    digits.map(i -> i.longValue()).flatMap(toDigits).sum().intValue();

static Function1<Integer, Boolean> divisibleBy10 = number ->
    number % 10 == 0;
```

37

trivadis

"... currying is the technique of translating the evaluation of a function that takes multiple arguments into evaluating a sequence of functions, each with a single argument."

38

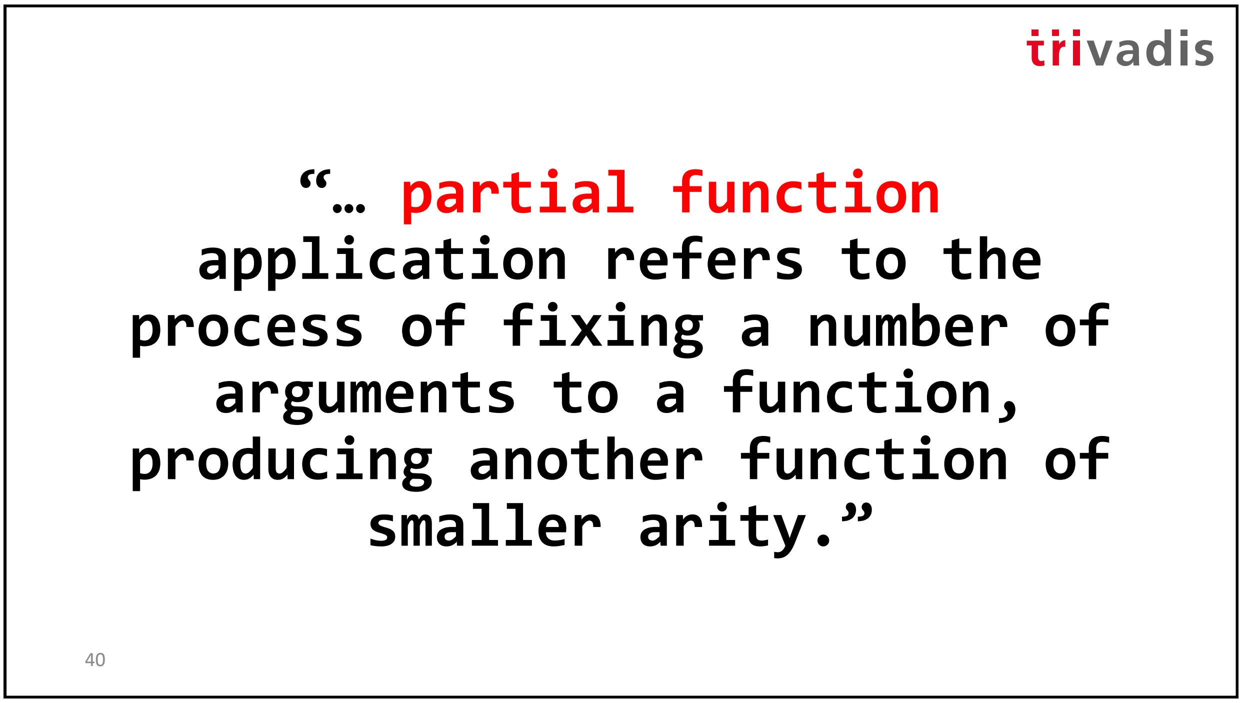


trivadis

```
Function3<BiFunction<Integer, Integer, Integer>,
         List<Integer>,
         List<Integer>,
         List<Integer>> myZip = ... // = myZip(function, list1, list2)

Function1<BiFunction<Integer, Integer, Integer>,
  Function1<List<Integer>,
    Function1<List<Integer>, List<Integer>>>> curriedMyZip
              = myZip.curried();

List<Integer> result = curriedMyZip.apply((a, b) -> a * b)
                                   .apply(List(1, 2, 3))
                                   .apply(List(4, 5, 6))
```

currying

39



trivadis

“… partial function application refers to the process of fixing a number of arguments to a function, producing another function of smaller arity.”

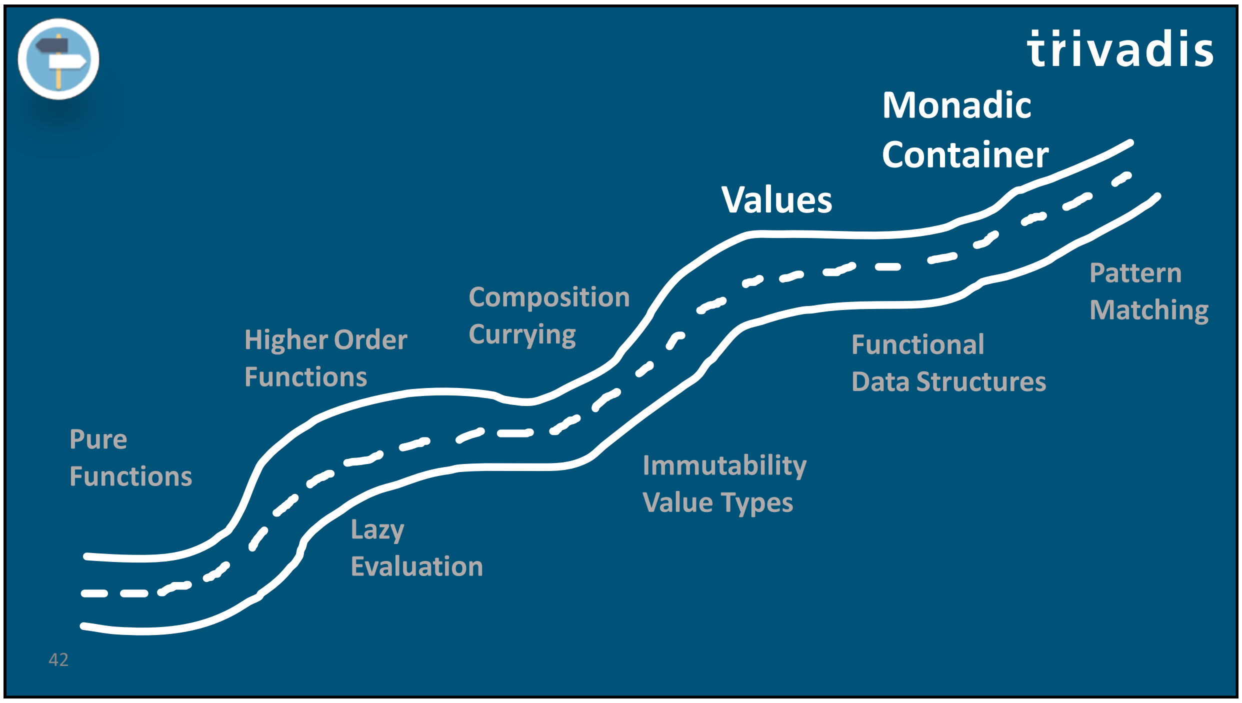
40

trivadis

```
Function3<BiFunction<Integer, Integer, Integer>,
          List<Integer>,
          List<Integer>,
          List<Integer>> myZip = ... // = myZip(function, list1, list2)

Function1<List<Integer>, List<Integer>> zipped =
     myZip.curried()
               .apply((a, b) -> a * b)
               .apply(List(1, 2, 3));

zipped.apply(List(4, 5, 6))
```

partial function application

41



trivadis

```
Tuple2<String, Integer> java8 = Tuple.of("Java", 8);

Tuple2<String, Integer> vavr1 = java8.map(
        s -> s.substring(2) + "vr",
        i -> i / 8);

String vavr = vavr1._1;
int one = vavr1._2;
```

43

trivadis

Try (Sucess, Failure)
Either (Left, Right)
Option (Some, None)
Validation (Valid, NotValid)

44

trivadis

```java
static CreditCardNumber from(String s) {
    return new CreditCardNumber(Long.parseLong(s));
}
```

```java
static Try<CreditCardNumber> fromWithTry(String s) {
    return Try.of(() -> Long.parseLong(s))
            .map(n -> new CreditCardNumber(n));
}
```

```java
System.out.println(CreditCardNumber.fromWithTry( s: "abc").getOrNull());
System.out.println(CreditCardNumber.fromWithTry( s: "abc").getCause().getClass().getName());
System.out.println(CreditCardNumber.fromWithTry( s: "123").get());
```

Try

45

trivadis

```java
static Either<String, CreditCardNumber> fromWithEither(String s) {
    try {
        return Either.right(new CreditCardNumber(Long.parseLong(s)));
    } catch (NumberFormatException e) {
        return Either.left(String.format("wrong credit card number format: %s", s));
    }
}
```

```java
System.out.println(CreditCardNumber.fromWithEither( s: "abc").isLeft());
System.out.println(CreditCardNumber.fromWithEither( s: "123").isRight());
System.out.println(CreditCardNumber.fromWithEither( s: "123").left().getOrElse( other: "no error"));
System.out.println(CreditCardNumber.fromWithEither( s: "abc").left().get());
```

Either

46

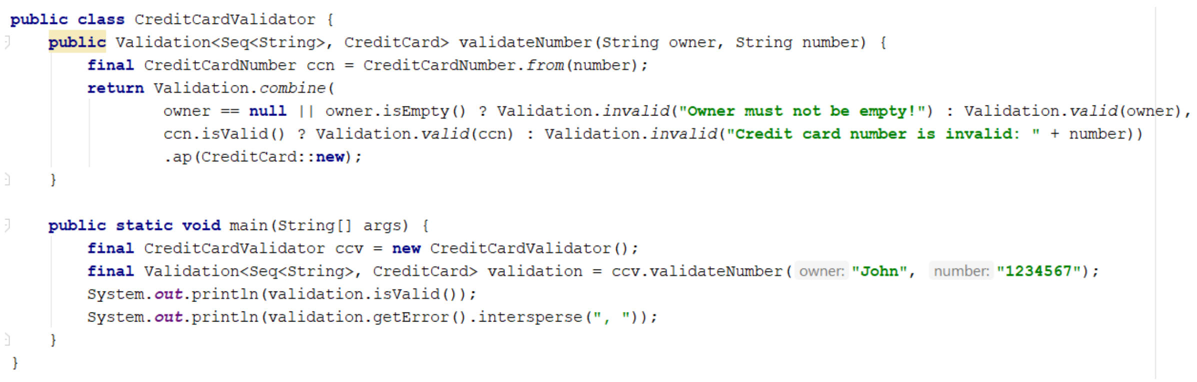
trivadis

```
String helloWorld = Option.of("Hello")
        .map(value -> value + " Falk")
        .peek(value -> LOG.debug("Value: {}", value))
        .getOrElse(() -> "Hello World");
```

Option

47

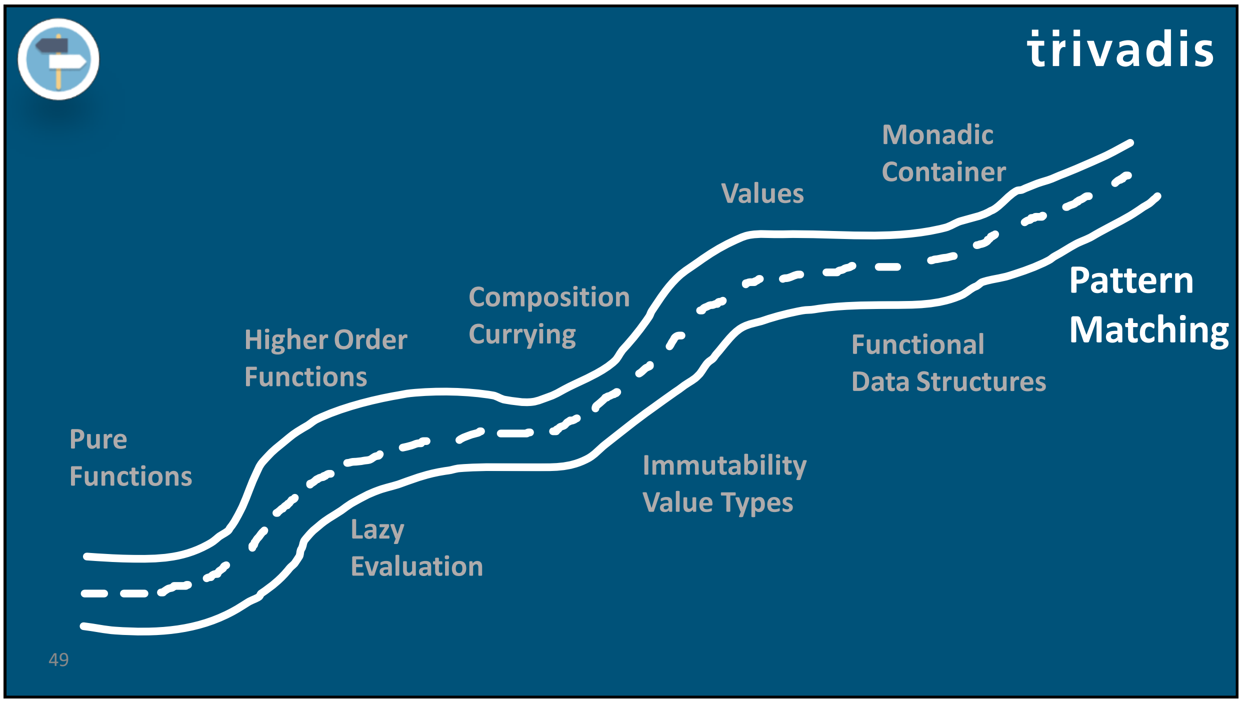
trivadis



```
public class CreditCardValidator {
    public Validation<Seq<String>, CreditCard> validateNumber(String owner, String number) {
        final CreditCardNumber ccn = CreditCardNumber.from(number);
        return Validation.combine(
                owner == null || owner.isEmpty() ? Validation.invalid("Owner must not be empty!") : Validation.valid(owner),
                ccn.isValid() ? Validation.valid(ccn) : Validation.invalid("Credit card number is invalid: " + number))
                .ap(CreditCard::new);
    }

    public static void main(String[] args) {
        final CreditCardValidator ccv = new CreditCardValidator();
        final Validation<Seq<String>, CreditCard> validation = ccv.validateNumber( owner: "John",  number: "1234567");
        System.out.println(validation.isValid());
        System.out.println(validation.getError().intersperse(", "));
    }
}
```

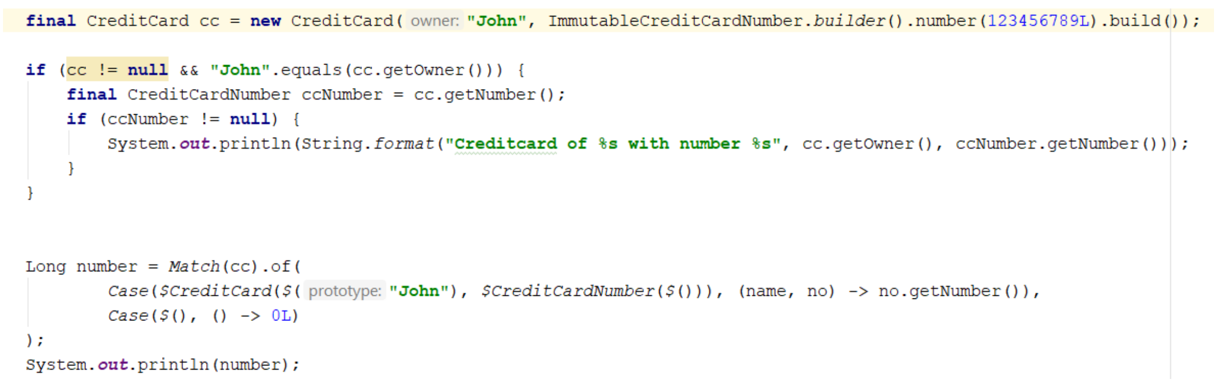
Validation

48

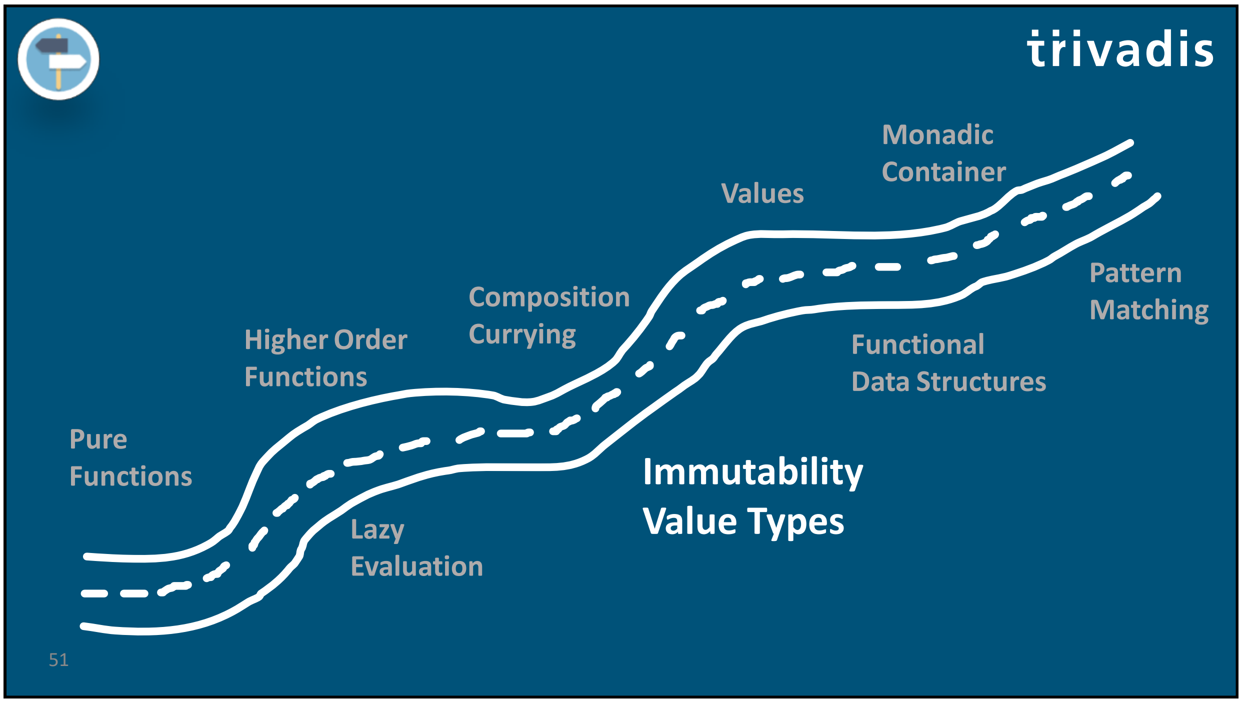


Pattern Matching: Destrukturieren von Objekten

trivadis


```
final CreditCard cc = new CreditCard( owner: "John", ImmutableCreditCardNumber.builder().number(123456789L).build());

if (cc != null && "John".equals(cc.getOwner())) {
    final CreditCardNumber ccNumber = cc.getNumber();
    if (ccNumber != null) {
        System.out.println(String.format("Creditcard of %s with number %s", cc.getOwner(), ccNumber.getNumber()));
    }
}


Long number = Match(cc).of(
        Case($CreditCard($( prototype: "John"), $CreditCardNumber($())), (name, no) -> no.getNumber()),
        Case($(), () -> 0L)
);
System.out.println(number);
```

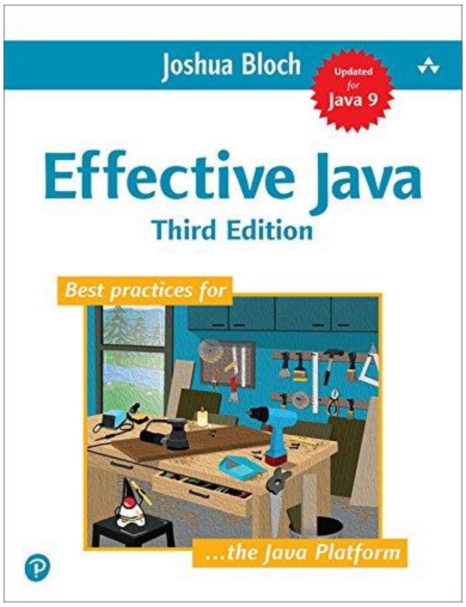
50



trivadis

Immutability

trivadis



trivadis

Only one state
Thread Safety
Instances can be shared
Inner states can be used together

Benefits

trivadis

Project Lombok

Immutables stars 1,662

VAVR.io

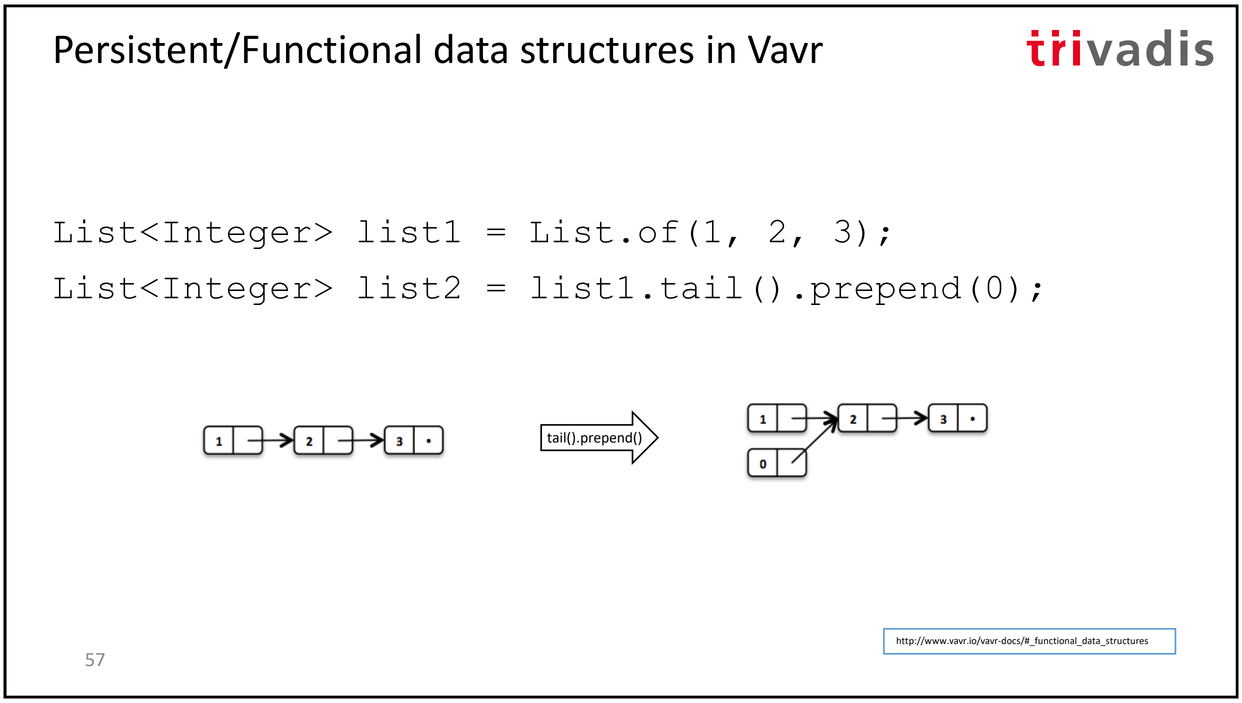
Tools, libraries

Java Collections are mutable!

trivadis

```
interface Collection<E> {
    void clear();
}

List<String> list = Collections
                       .unmodifiableList(otherList);

list.add("Boom");
```

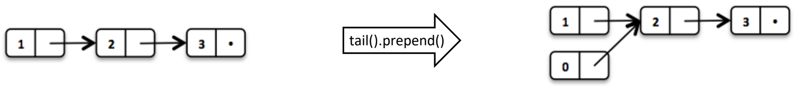
56



Persistent/Functional data structures in Vavr

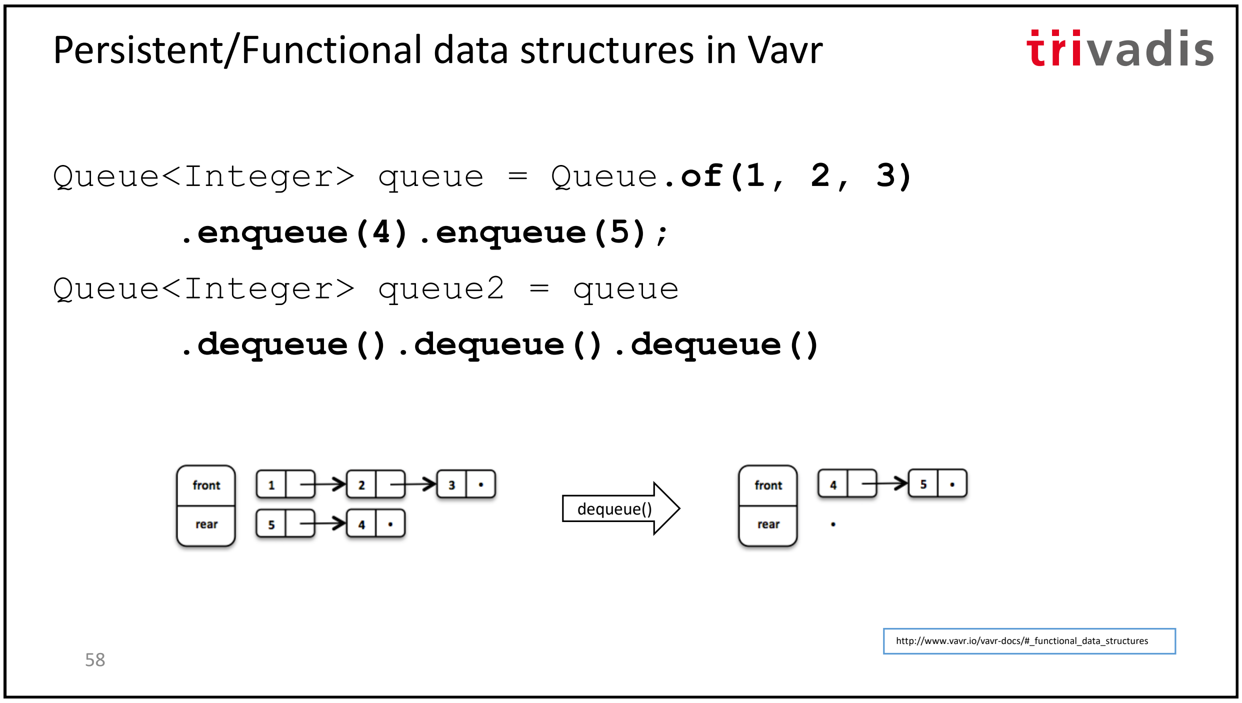
trivadis

```
List<Integer> list1 = List.of(1, 2, 3);
List<Integer> list2 = list1.tail().prepend(0);
```



http://www.vavr.io/vavr-docs/#_functional_data_structures

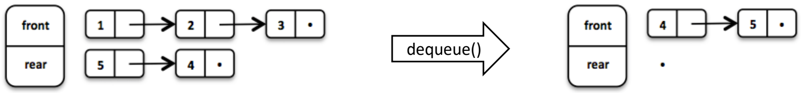
57



Persistent/Functional data structures in Vavr

trivadis

```
Queue<Integer> queue = Queue.of(1, 2, 3)
      .enqueue(4).enqueue(5);
Queue<Integer> queue2 = queue
      .dequeue().dequeue().dequeue()
```



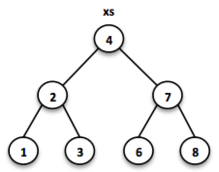
http://www.vavr.io/vavr-docs/#_functional_data_structures

58

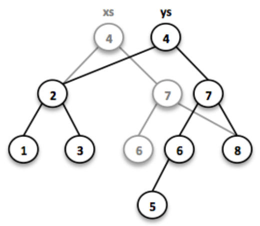
Persistent/Functional data structures in Vavr

trivadis

```
// = TreeSet(1, 2, 3, 4, 6, 7, 8)
SortedSet<Integer> xs =
      TreeSet.of(6, 1, 3, 2, 4, 7, 8);
```



```
// = TreeSet(1, 2, 3, 4, 5, 6, 7, 8)
SortedSet<Integer> ys = xs.add(5);
```



http://www.vavr.io/vavr-docs/#_functional_data_structures

59

trivadis

```
List<User> result = users.stream()
  .filter(user -> {
    try {
        return user.validate();
    } catch (Exception ex) {
        return false;
    }
  })
  .map(user -> user.name)
  .collect(Collectors.toList());
```

```
List<User> result = List.ofAll(users)
     .filter(user ->
        Try.of(user::validateAddress)
              .getOrElse(false)
     )
    .map(user -> user.name);

java.util.List<User> result2 =
    result.toJavaList();
```

java.util.List

Java 8

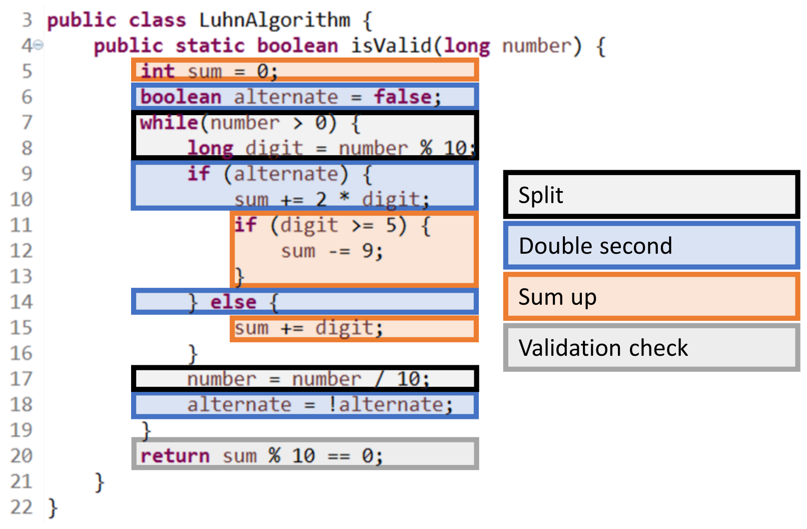
VAVR.io

60

trivadis

FAZIT

trivadis



```
public class LuhnAlgorithm {
    public static boolean isValid(long number) {
        int sum = 0;
        boolean alternate = false;
        while(number > 0) {
            long digit = number % 10;
            if (alternate) {
                sum += 2 * digit;
                if (digit >= 5) {
                    sum -= 9;
                }
            } else {
                sum += digit;
            }
            number = number / 10;
            alternate = !alternate;
        }
        return sum % 10 == 0;
    }
}
```

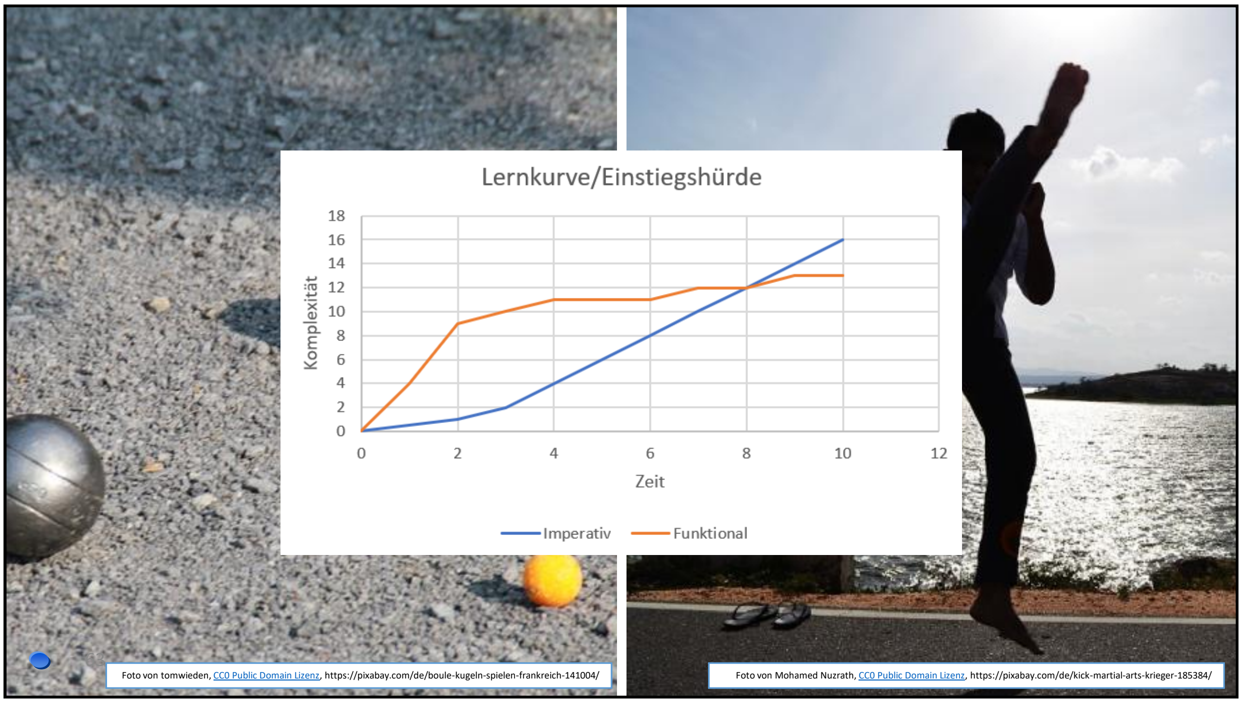
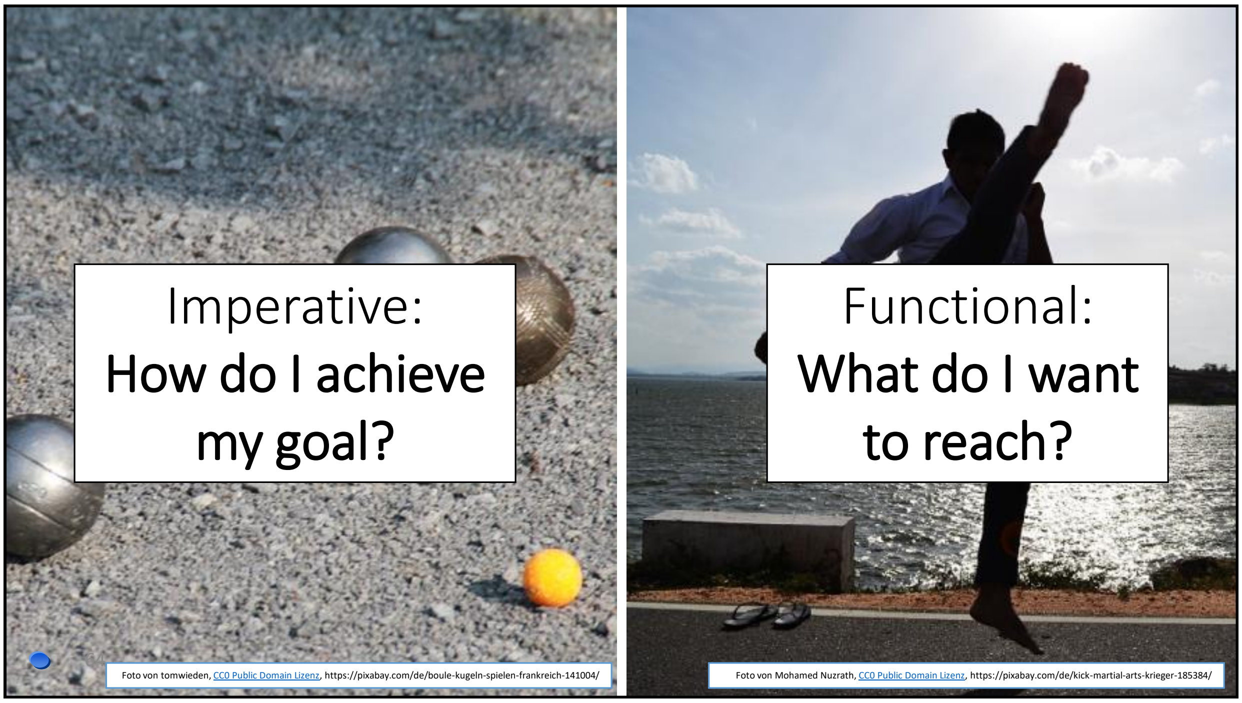


Foto von tomwieden, CC0 Public Domain Lizenz, https://pixabay.com/de/boule-kugeln-spielen-frankreich-141004/

Foto von Mohamed Nuzrath, CC0 Public Domain Lizenz, https://pixabay.com/de/kick-martial-arts-krieger-185384/

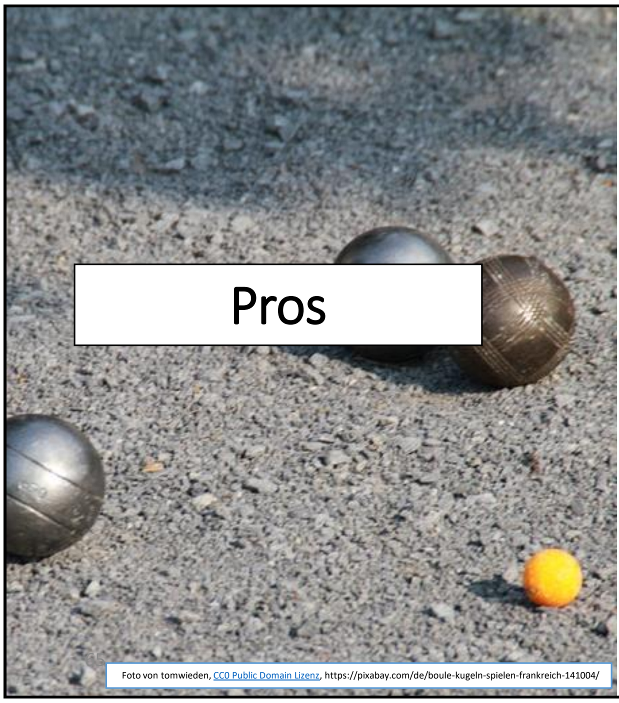


Imperative:
How do I achieve my goal?

Functional:
What do I want to reach?

Foto von tomwieden, CC0 Public Domain Lizenz, https://pixabay.com/de/boule-kugeln-spielen-frankreich-141004/

Foto von Mohamed Nuzrath, CC0 Public Domain Lizenz, https://pixabay.com/de/kick-martial-arts-krieger-185384/



Pros

trivadis

- just works if compiles
- easy to understand, easy to conclude
- side-effect-free
- easy test/debugging
- easy to parallelize
- can be modularized and easily reassembled
- high code quality

Foto von tomwieden, CC0 Public Domain Lizenz, https://pixabay.com/de/boule-kugeln-spielen-frankreich-141004/

trivadis

66

Foto von Hans Braxmeier, CC0 Public Domain Lizenz, https://pixabay.com/de/teller-suppenteller-essteller-1365805/

https://pixabay.com/de/photos/eisberg-antarktis-polaren-blau-eis-404966/

trivadis

Java 8 + Immutables stars 1,662

VAVR.IO

Project Lombok

68

trivadis

Immutable data types
+
pure functions

key for better Java

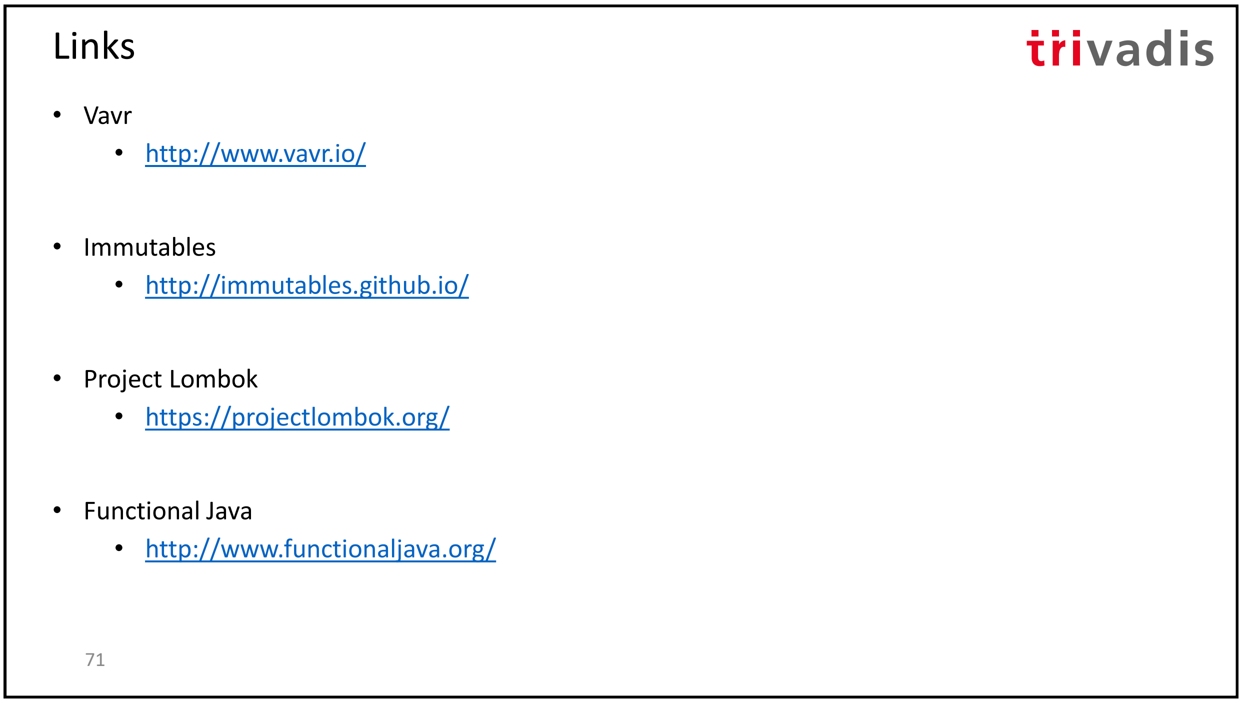
69

Links

trivadis

- Code-Beispiele
  - https://github.com/sippsack/jvm-functional-language-battle
- Learn You a Haskell for Great Good!
  - http://learnyouahaskell.com/chapters
- LYAH (Learn You a Haskell) adaptions for Frege
  - https://github.com/Frege/frege/wiki/LYAH-adaptions-for-Frege
- Onlinekurs TU Delft (FP 101):
  - https://courses.edx.org/courses/DelftX/FP101x/3T2014/info

70



Links

trivadis

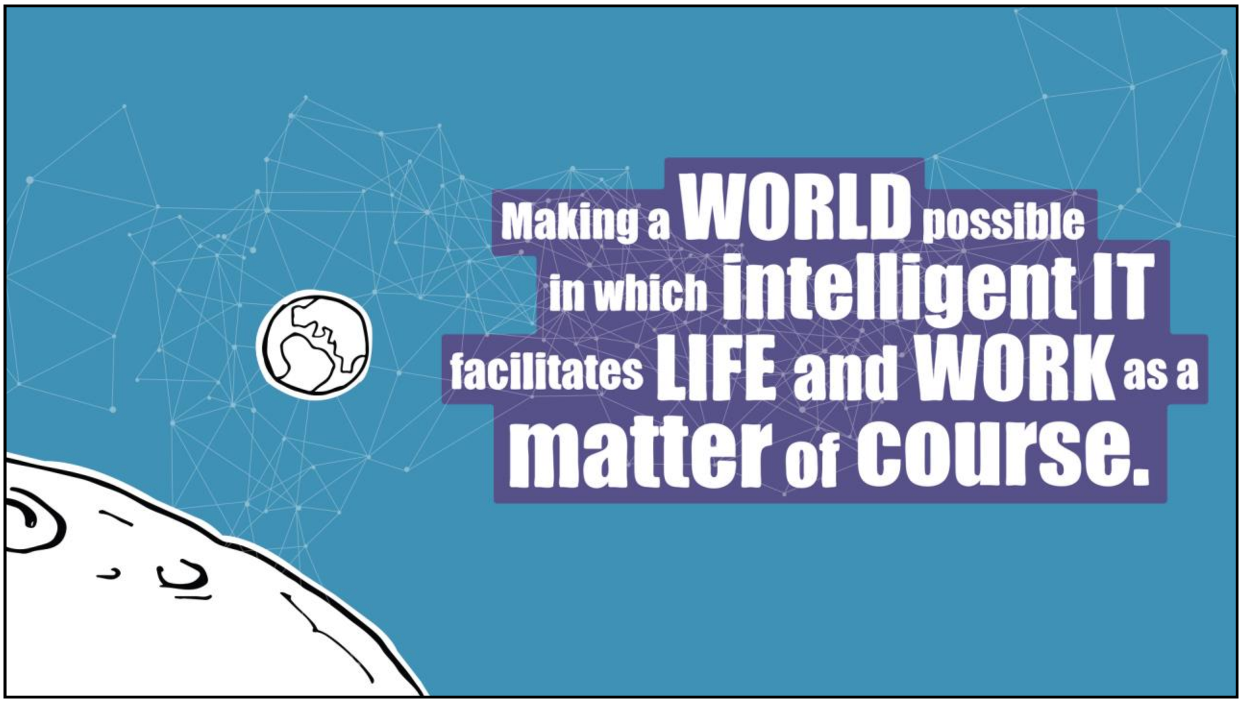
- Vavr
  - http://www.vavr.io/
- Immutables
  - http://immutables.github.io/
- Project Lombok
  - https://projectlombok.org/
- Functional Java
  - http://www.functionaljava.org/

71

trivadis

Thank you

Questions?



Making a WORLD possible in which intelligent IT facilitates LIFE and WORK as a matter of course.