prepare for…

From mess to masterpiece

refactoring done right

@SvenRuppert

@SvenRuppert

has been coding java since 1996

@SvenRuppert

has been coding java since 1996

Projects in the field of:

- Automobile-industry
- Energy
- Finance / Leasing
- Space- Satellit-
- Government / UN / World-bank

Where?

- Europe
- Asia - from India up to Malaysia

Our journey today

what is the situation.....

@SvenRuppert

At the time I started - 3 yrs ago

@SvenRuppert

At the time I started - 3 yrs ago

Codebase is > 13 years old

At the time I started - 3 yrs ago

@SvenRuppert

Codebase is > 13 years old

no test coverage

At the time I started - 3 yrs ago

@SvenRuppert

Codebase is > 13 years old

no test coverage

Code Lords

At the time I started - 3 yrs ago

@SvenRuppert

Codebase is > 13 years old

no test coverage

Code Lords

nearly 15% are retired soon

At the time I started - 3 yrs ago

@SvenRuppert

Codebase is > 13 years old

no test coverage

Code Lords

nearly 15% are retired soon

over 50% are since 15 yrs in the company

At the time I started - 3 yrs ago

Codebase is > 13 years old

no test coverage

Code Lords

nearly 15% are retired soon

over 50% are since 15 yrs in the company

hiring only students

At the time I started - 3 yrs ago

Codebase is > 13 years old

no test coverage

Code Lords

nearly 15% are retired soon

over 50% are since 15 yrs in the company

hiring only students

the developers learned Java at this project

At the time I started - 3 yrs ago

Codebase is > 13 years old

no test coverage

Code Lords

nearly 15% are retired soon

over 50% are since 15 yrs in the company

hiring only students

the developers learned Java at this project

the first feeling was like ……

At the time I started - 3 yrs ago

@SvenRuppert



@SvenRuppert

What is part of a refactoring ?

What is part of a refactoring ?

the team

@SvenRuppert

What is part of a refactoring ?

the team

the source code

@SvenRuppert

What is part of a refactoring ?

the team

the source code

the processes

What is part of a refactoring ?

the team

the source code

the processes

the project management

What is part of a refactoring ?

the team

the source code

the project management

the processes

the knowledge

What is part of a refactoring ?

the team

the source code

the project management

the processes

budget and time line

the knowledge

What is part of a refactoring ?

the team

the source code

the processes

the project management

budget and time line

the knowledge

infrastructure

What is part of a refactoring ?

the team
the project management
the knowledge
the source code
infrastructure

team transformation

the processes
budget and time line

C-level transformation

What is part of a refactoring ?

the team
the project management
the knowledge
the source code
infrastructure



team transformation

prepare for

Refactoring - the first steps

without this you should not start

@SvenRuppert

first steps - Start Small

@SvenRuppert



first steps - Think Bigger

@SvenRuppert



first steps - Build a Team

@SvenRuppert



first steps - build trust

@SvenRuppert



first steps - build trust

@SvenRuppert



prepare for

Refactoring - the way of working

start with individual responsibility

@SvenRuppert

The way of working

we started step by step

work from everywhere at your time

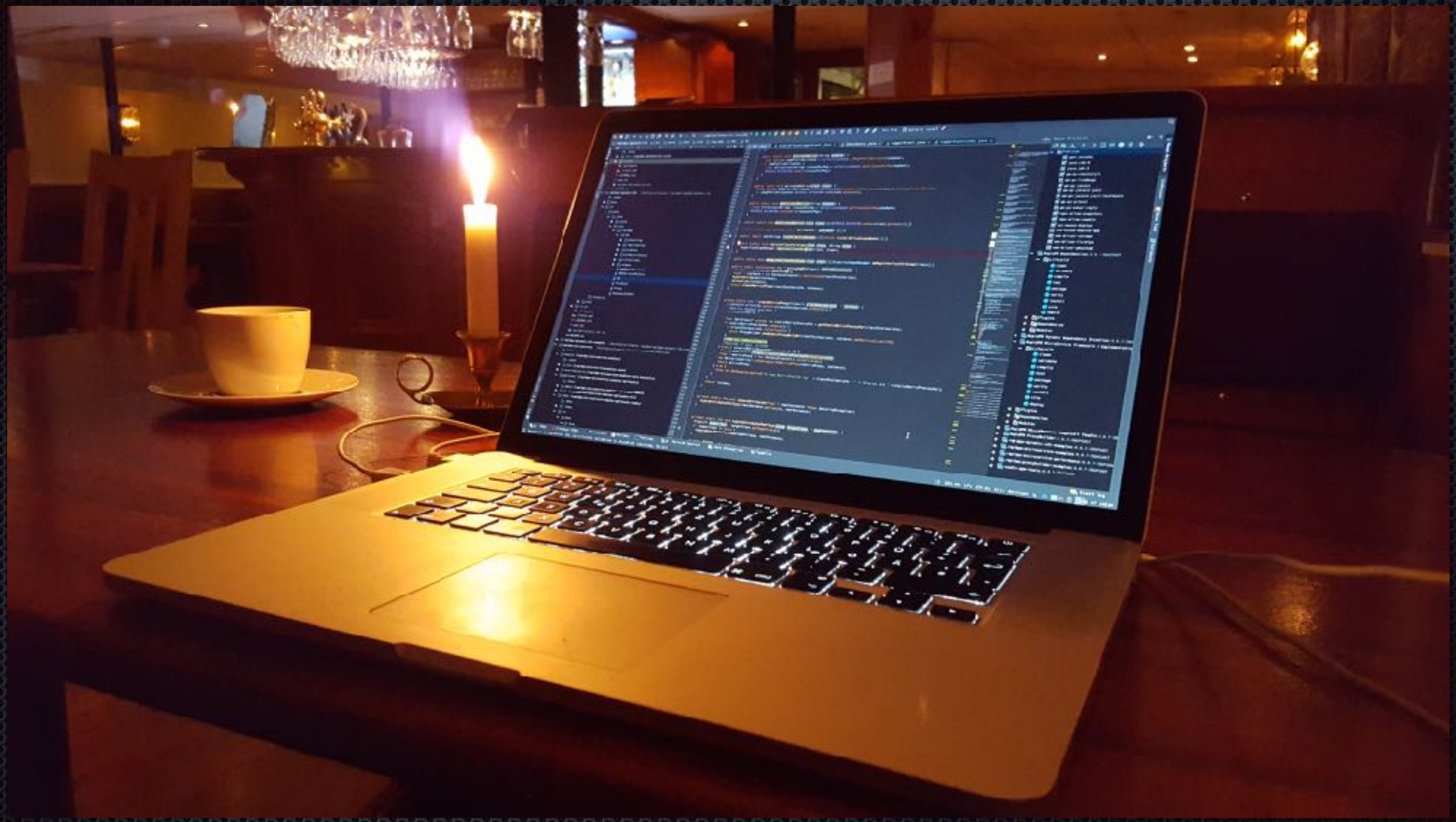
20% Time have time to play or try

Closed Source versus Open Source

start learning how to practice it ?

work from everywhere

@SvenRuppert



work from everywhere

@SvenRuppert

work from everywhere

change your environment

work from everywhere

change your environment

use Messenger like Slack

work from everywhere

@SvenRuppert

change your environment

use Messenger like Slack

avoid mails

work from everywhere

change your environment

use Messenger like Slack

avoid mails

remote meetings with zoom

work from everywhere

change your environment

use Messenger like Slack

avoid mails

remote meetings with zoom

remote Pair-Programming

work from everywhere

@SvenRuppert

change your environment

use Messenger like Slack

avoid mails

remote meetings with zoom

remote Pair-Programming

work async.

work from everywhere

@SvenRuppert

change your environment

use Messenger like Slack avoid mails

remote meetings with zoom

remote Pair-Programming

work async.

work at your time that is the best for you

work from everywhere

work at your time that is the best for you

work from everywhere

@SvenRuppert

work at your time that is the best for you



work from everywhere

work at your time that is the best for you

my „normal“ day…



work from everywhere

@SvenRuppert

work at your time that is the best for you



my „normal“ day…

05AM to 07AM working

work from everywhere

@SvenRuppert

work at your time that is the best for you



my „normal“ day…

05AM to 07AM working
07AM to 09AM breakfast with the family

work from everywhere

@SvenRuppert

work at your time that is the best for you



my „normal“ day…

05AM to 07AM working
07AM to 09AM breakfast with the family
09AM to 12AM working

work from everywhere

@SvenRuppert

work at your time that is the best for you



my „normal“ day…

05AM to 07AM	working
07AM to 09AM	breakfast with the family
09AM to 12AM	working
12AM to 07PM	playing with my son

work from everywhere

work at your time that is the best for you



my „normal“ day…

05AM to 07AM	working
07AM to 09AM	breakfast with the family
09AM to 12AM	working
12AM to 07PM	playing with my son
07PM to 09PM	playing with my wife

work from everywhere

work at your time that is the best for you



my „normal“ day…

05AM to 07AM	working
07AM to 09AM	breakfast with the family
09AM to 12AM	working
12AM to 07PM	playing with my son
07PM to 09PM	playing with my wife
09PM to 11PM	working

work from everywhere

@SvenRuppert

work at your time that is the best for you



my „normal“ day…

05AM to 07AM	working
07AM to 09AM	breakfast with the family
09AM to 12AM	working
12AM to 07PM	playing with my son
07PM to 09PM	playing with my wife
09PM to 11PM	working

and yes… I need only 6h sleep ;-)

work from everywhere

work at your time that is the best for you

my „normal“ day…

05AM to 07AM working
09AM to 12AM working
09PM to 11PM working

work from everywhere

work at your time that is the best for you

my „normal“ day…

05AM to 07AM	working	working with Asia
09AM to 12AM	working	
09PM to 11PM	working	

work from everywhere

work at your time that is the best for you

my „normal“ day…

05AM to 07AM	working	working with Asia
09AM to 12AM	working	working with Europe
09PM to 11PM	working	

work from everywhere

work at your time that is the best for you

my „normal“ day…

05AM to 07AM	working	working with Asia
09AM to 12AM	working	working with Europe
09PM to 11PM	working	working with US

The way of working

@SvenRuppert

The way of working

20% Time

The way of working

20% Time why you should do it?

The way of working

@SvenRuppert

20% Time why you should do it?



The way of working

20% Time why you should do it?

The way of working

20% Time why you should do it?



The way of working

20% Time

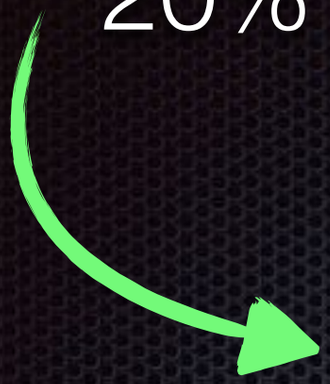
The way of working

20% Time how to make it useful ?

The way of working

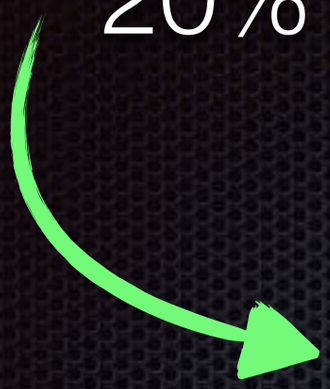
@SvenRuppert

20% Time how to make it useful ?



The way of working

20% Time how to make it useful ?



for the developer

The way of working

20% Time how to make it useful ?

for the developer

The way of working

20% Time how to make it useful ?

- for the developer
- for the company

The way of working

20% Time how to make it useful ?

- for the developer
- for the company

example : POC with the team in the Mountains

prepare for

Refactoring - Divide your code

what is the part you are earning money with

@SvenRuppert

Closed- versus Open Source

Closed- versus Open Source

Analyzing the existing Code

Closed- versus Open Source

Analyzing the existing Code

in our case : a lot of infrastructure....

Closed- versus Open Source

Analyzing the existing Code

in our case : a lot of infrastructure....

..with this we are not earning money

Closed- versus Open Source

Analyzing the existing Code

in our case : a lot of infrastructure....

..with this we are not earning money

we divided the

Closed- versus Open Source

Analyzing the existing Code

in our case : a lot of infrastructure....

..with this we are not earning money

we divided the

code base

Closed- versus Open Source

Analyzing the existing Code

in our case : a lot of infrastructure....

..with this we are not earning money

we divided the

code base

dev. environment

Closed- versus Open Source

we divided the

code base

dev. environment

Closed- versus Open Source

we divided the

code base

dev. environment

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

could write articles about it

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

could write articles about it

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

could write articles about it

will lead to better documentation

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

could write articles about it

will lead to better documentation

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

out of company rules

could write articles about it

will lead to better documentation

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

could write articles about it

out of company rules

free choice of tools

will lead to better documentation

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

could write articles about it

out of company rules

free choice of tools

will lead to better documentation

Closed- versus Open Source

we divided the

code base

dev. environment

discuss at conferences

could write articles about it

out of company rules

free choice of tools

will lead to better documentation

higher motivation

prepare for

Refactoring - the reactions

don´t forget the team you are working with

@SvenRuppert

The way of working

@SvenRuppert

The way of working

Start Learning again

The way of working

Start Learning again
for example:

The way of working

Start Learning again

for example:

focus on a new paradigm

The way of working

Start Learning again

for example:

focus on a new paradigm functional

The way of working

Start Learning again

for example:

focus on a new paradigm functional reactive

The way of working

Start Learning again

for example:

focus on a new paradigm functional reactive

add a new Language

The way of working

Start Learning again

for example:

focus on a new paradigm functional reactive

add a new Language

but not everyone want to learn

The way of working

Start Learning again

for example:

focus on a new paradigm functional reactive

add a new Language

but not everyone want to learn

or better..... some are learning faster

The way of working

Start Learning again

for example:

focus on a new paradigm functional reactive

add a new Language

but not everyone want to learn

or better…. some are learning faster

this sometimes leads to …..

reactions….

The way of working

Start Learning again

this sometimes leads to ..... reactions....

The way of working

Start Learning again

this sometimes leads to ..... reactions....



The way of working

Start Learning again

this sometimes leads to ..... reactions....

running away



The way of working

Start Learning again

this sometimes leads to ..... reactions....

The way of working

Start Learning again

this sometimes leads to ..... reactions....



The way of working

Start Learning again

this sometimes leads to ..... reactions....

expecting something



The way of working

Start Learning again

this sometimes leads to ..... reactions....

The way of working

@SvenRuppert

Start Learning again

this sometimes leads to ..... reactions....

feeling the near end

The way of working

Start Learning again

this sometimes leads to ..... reactions....

feeling the near end



The way of working

Start Learning again

this sometimes leads to ..... reactions....

The way of working

Start Learning again

this sometimes leads to ..... reactions....

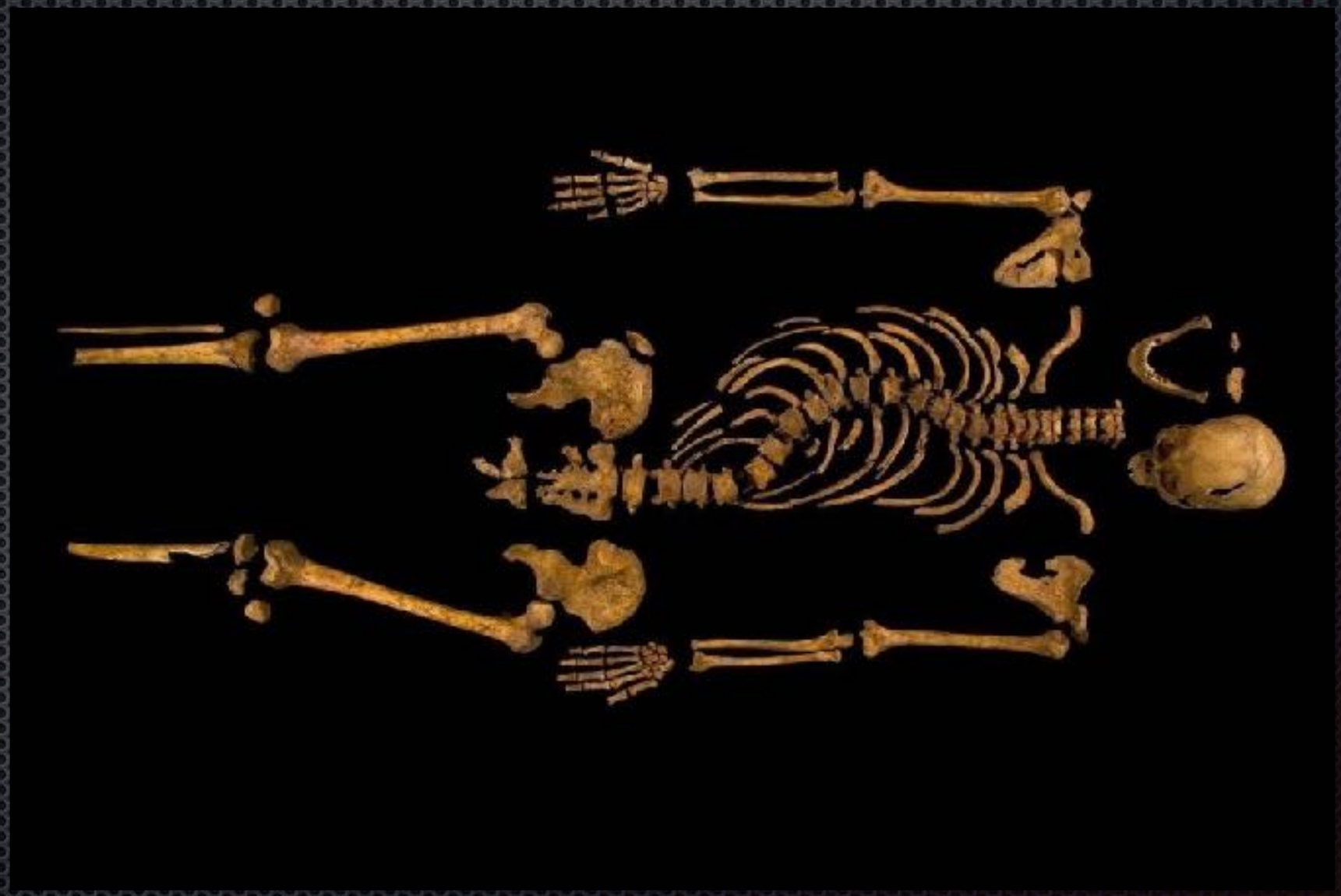
or only
feeling to old

The way of working

Start Learning again

this sometimes leads to ..... reactions....

or only
feeling to old



The way of working

Start Learning again

this sometimes leads to ..... reactions....

The way of working

Start Learning again

this sometimes leads to ..... reactions....

if you don´t want
to loose them

The way of working

Start Learning again

this sometimes leads to ..... reactions....

if you don´t want
to loose them

you have to deal with this

The way of working

Start Learning again

this sometimes leads to ..... reactions....

if you don´t want
to loose them

you have to deal with this

make them
happy again

The way of working

Start Learning again

this sometimes leads to ….. reactions….

if you don´t want
to loose them

you have to deal with this

make them
happy again



The way of working

@SvenRuppert

Start Learning again

this sometimes leads to ….. reactions….

if you don´t want
to loose them

you have to deal with this

make them
happy again

How we solved this?



prepare for

Refactoring - Projectmanagement

one size fit´s all ?

@SvenRuppert

The way of working

@SvenRuppert

The way of working

we are a product company

The way of working

we are a product company

with a lot of LTS contracts

The way of working

we are a product company

with a lot of LTS contracts

we created three streams

The way of working

we are a product company

with a lot of LTS contracts

we created three streams

Consultants - play and throw away

The way of working

we are a product company

with a lot of LTS contracts

we created three streams

Consultants - play and throw away

Core Developers - collect and clean

The way of working

we are a product company

with a lot of LTS contracts

we created three streams

Consultants - play and throw away

Core Developers - collect and clean

LTS Developers - keep alive

The way of working

we are a product company

with a lot of LTS contracts

we created three streams

Consultants - play and throw away

Core Developers - collect and clean

LTS Developers - keep alive

Why ?

The way of working

@SvenRuppert

Consultants - play and throw away

Core Developer - collect and clean

LTS Developer - keep alive

Why ?

The way of working

Consultants - play and throw away

Core Developer - collect and clean

LTS Developer - keep alive

Why ?

some want to learn and experiment

The way of working

Consultants - play and throw away

Core Developer - collect and clean

LTS Developer - keep alive

Why ?

some want to learn and experiment

some need more stabil environments

The way of working

Consultants - play and throw away

Core Developer - collect and clean

LTS Developer - keep alive

Why ?

some want to learn and experiment

some need more stabil environments

some did not want to change something

The way of working

@SvenRuppert

Why ?

Consultants - play and throw away
some want to learn and experiment

Core Developer - collect and clean
some need more stabil environments

LTS Developer - keep alive
some did not want to change something

The way of working

Why ?

Consultants - play and throw away
some want to learn and experiment
amount of hours paid by the customer

Core Developer - collect and clean
some need more stabil environments

LTS Developer - keep alive
some did not want to change something

The way of working

Why ?

Consultants - play and throw away
some want to learn and experiment
amount of hours paid by the customer

Core Developer - collect and clean
some need more stabil environments
rated on hours the systems are stabil

LTS Developer - keep alive
some did not want to change something

The way of working

Why ?

Consultants - play and throw away
some want to learn and experiment
amount of hours paid by the customer

Core Developer - collect and clean
some need more stabil environments
rated on hours the systems are stabil

LTS Developer - keep alive
some did not want to change something
fixed income - bonus based on Change Request

The way of working

Consultants - play and throw away

LTS Developer - keep alive

Core Developer - collect and clean

The way of working

Consultants - play and throw away

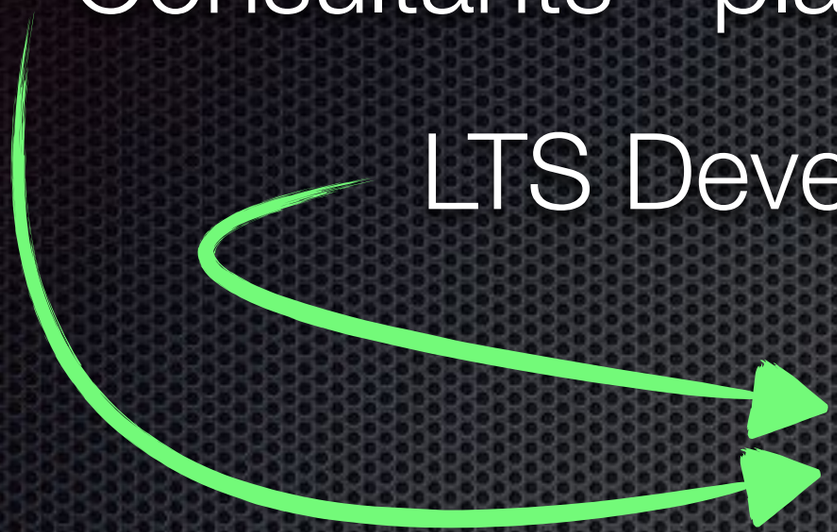
LTS Developer - keep alive

Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive



Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive

customer driven

Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive

customer driven

Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive

customer driven

SCRUM

Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive

customer driven

SCRUM

Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive

customer driven

SCRUM

roadmap driven

Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive

customer driven

SCRUM

roadmap driven

Core Developer - collect and clean

The way of working

Consultants - play and throw away

LTS Developer - keep alive

customer driven

SCRUM

roadmap driven

Core Developer - collect and clean

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers

Consultants

Consultants

Core - Developers

Core - Developers

Kanban

The way of working

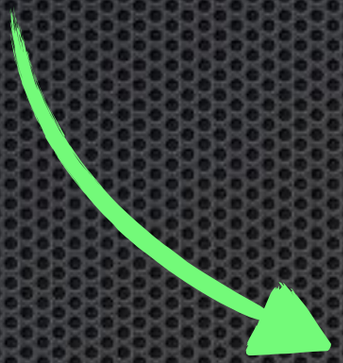
@SvenRuppert

SCRUM

LTS - Developers

Consultants

Consultants



Core - Developers

Core - Developers

Kanban

The way of working

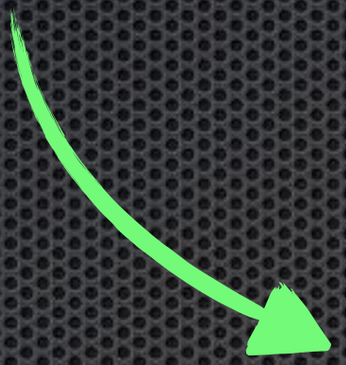
@SvenRuppert

SCRUM

LTS - Developers

Consultants

Consultants



Fire

Core - Developers

Core - Developers

Kanban

The way of working

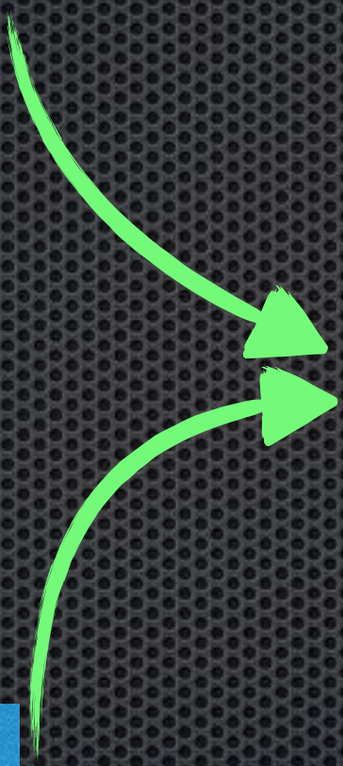
@SvenRuppert

SCRUM

LTS - Developers

Consultants

Consultants



Fire

Core - Developers

Core - Developers

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers

Consultants

Consultants



Fire

Core - Developers

Core - Developers

Kanban

The way of working

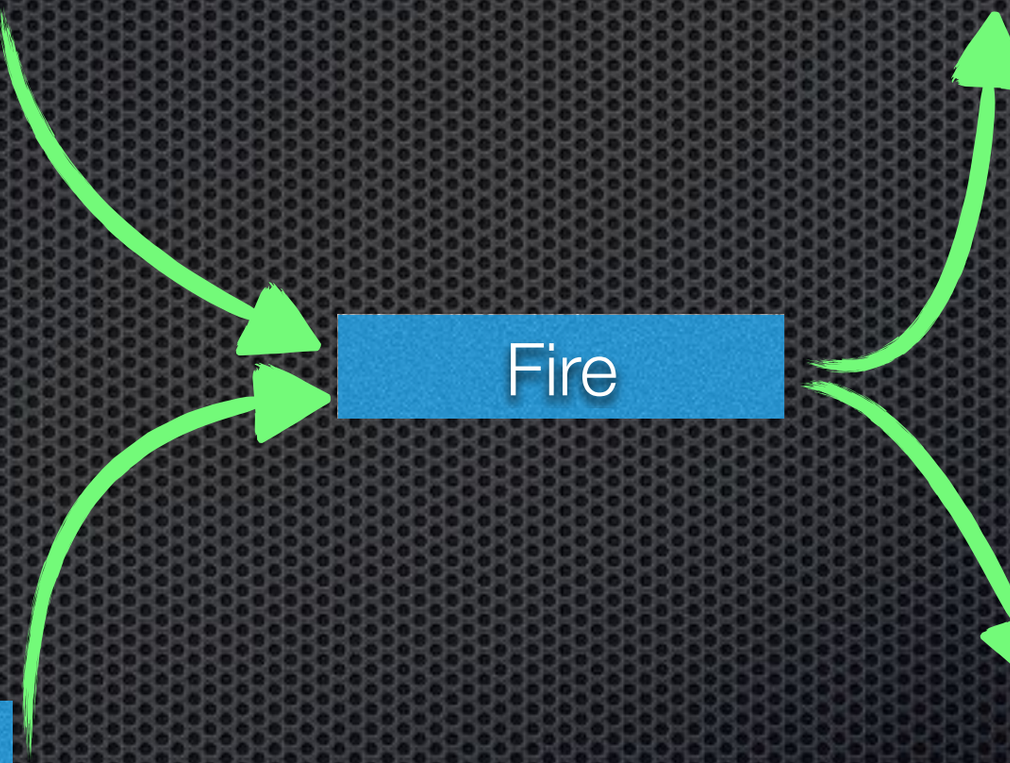
@SvenRuppert

SCRUM

LTS - Developers

Consultants

Consultants



Core - Developers

Core - Developers

Kanban

The way of working

@SvenRuppert

SCRUM

Kanban

The way of working

SCRUM

LTS - Developers	LTS - Developers	LTS - Developers

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants



Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants

time based planning

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants

time based planning

Core - Developers | Core - Developers | Core - Developers

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants

time based planning

Core - Developers | Core - Developers | Core - Developers

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants

time based planning

version based

Core - Developers | Core - Developers | Core - Developers

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants



time based planning **fixed size time slots**



version based

Core - Developers | Core - Developers | Core - Developers

Kanban

The way of working

@SvenRuppert

SCRUM

LTS - Developers | LTS - Developers | LTS - Developers

Consultants | Consultants | Consultants | Consultants

time based planning **fixed size time slots**

version based **dynamic size time slots**

Core - Developers | Core - Developers | Core - Developers

Kanban

prepare for

Refactoring - Start learning again

learn how to learn together with a team again

@SvenRuppert

Knowledge Sharing

@SvenRuppert

- LTS - Developers
- Consultants
- Core - Developers

Knowledge Sharing

@SvenRuppert

- LTS - Developers
- Consultants
- Core - Developers

build a source of knowledge

Knowledge Sharing

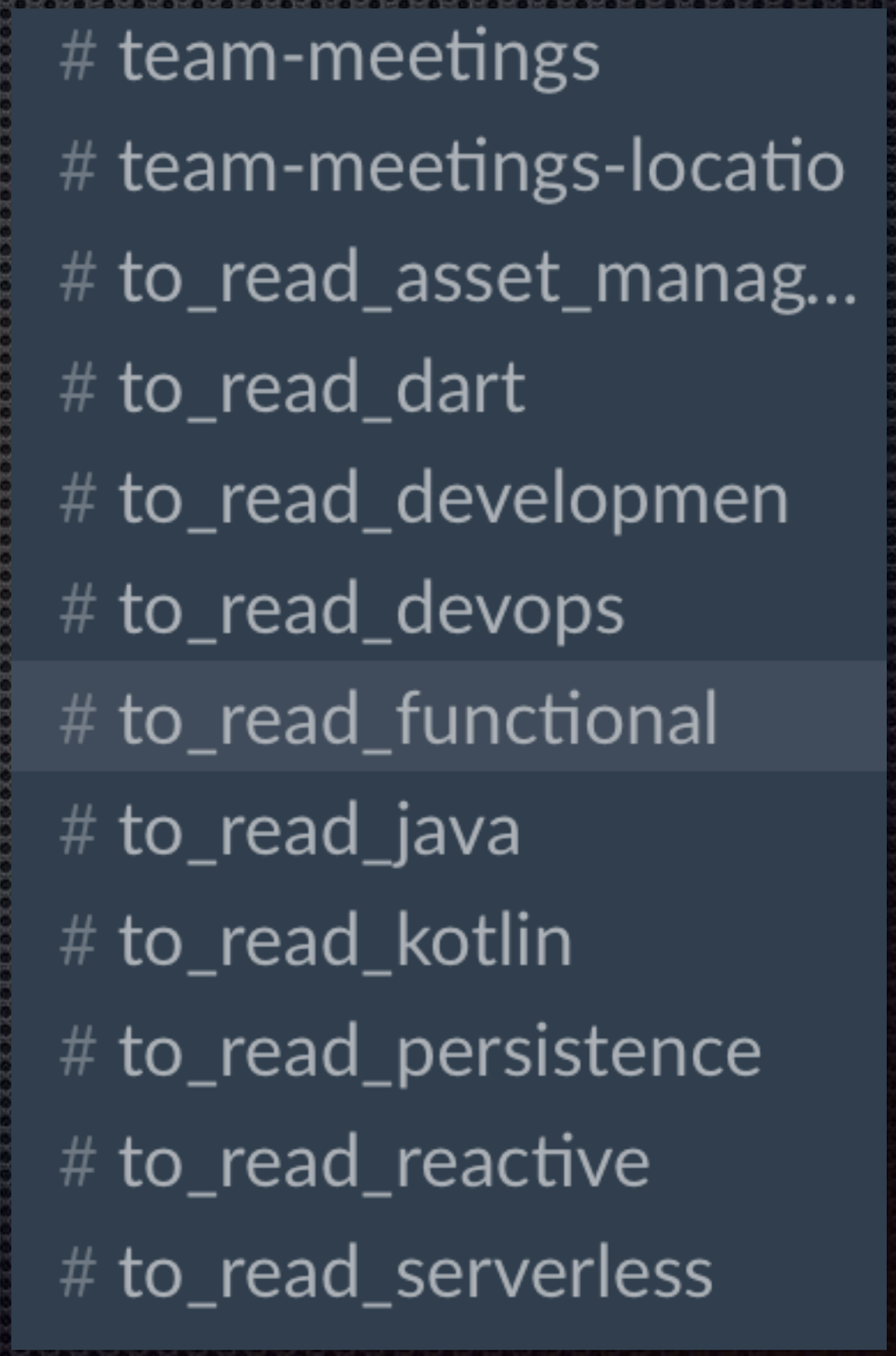
@SvenRuppert

LTS - Developers

Consultants

Core - Developers

build a source of knowledge



Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers

build a source of knowledge



team-meetings
team-meetings-locatio
to_read_asset_manag...
to_read_dart
to_read_developmen
to_read_devops
to_read_functional
to_read_java
to_read_kotlin
to_read_persistence
to_read_reactive
to_read_serverless

Knowledge Sharing

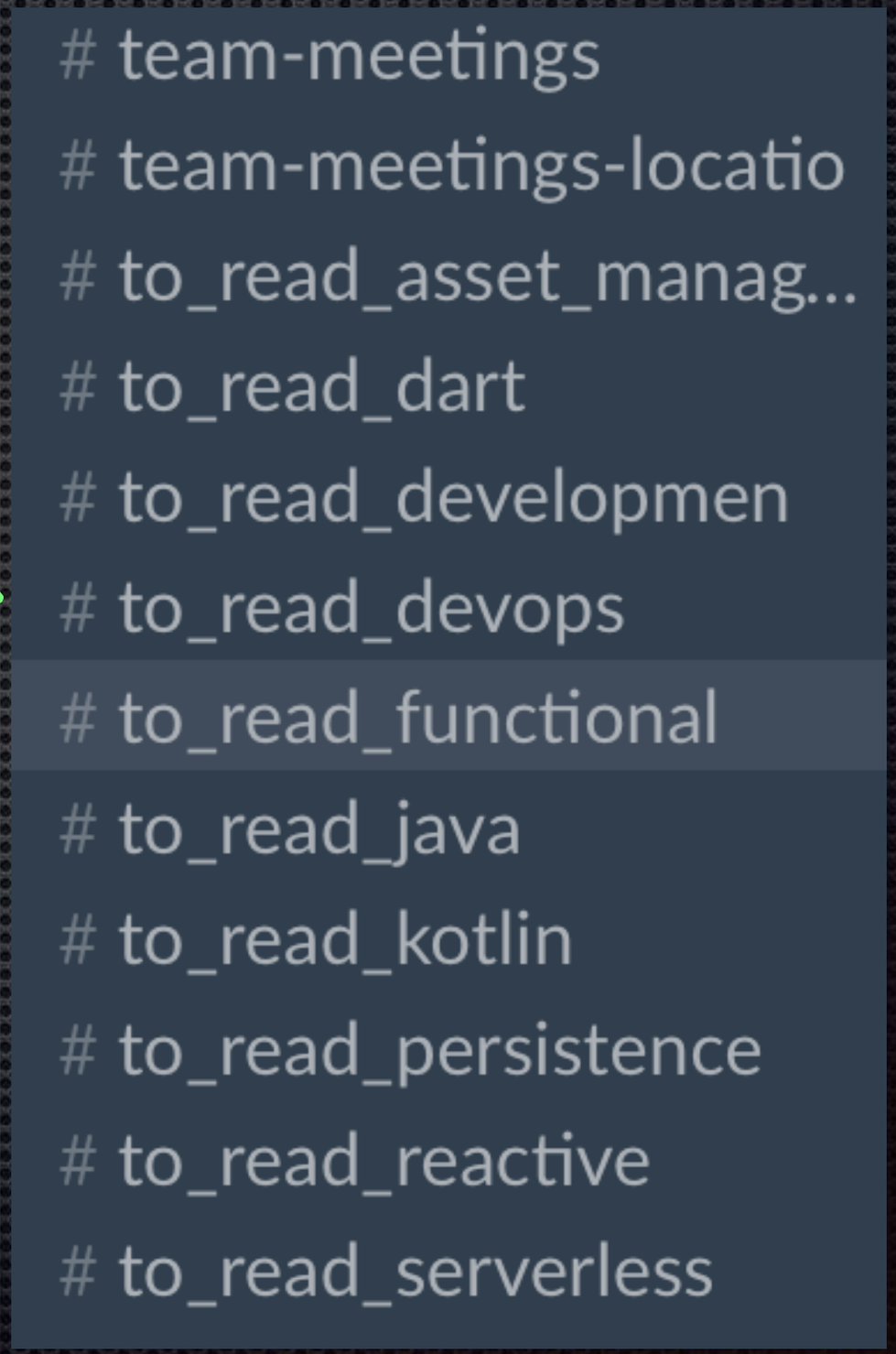
@SvenRuppert

LTS - Developers

Consultants

Core - Developers

build a source of knowledge

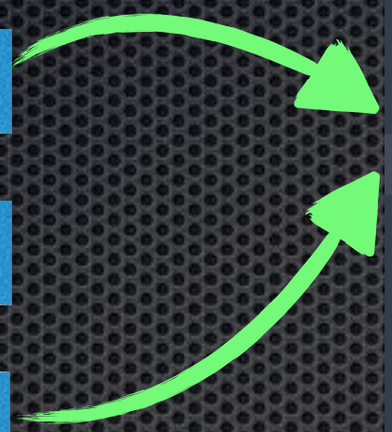


Knowledge Sharing

LTS - Developers

Consultants

Core - Developers





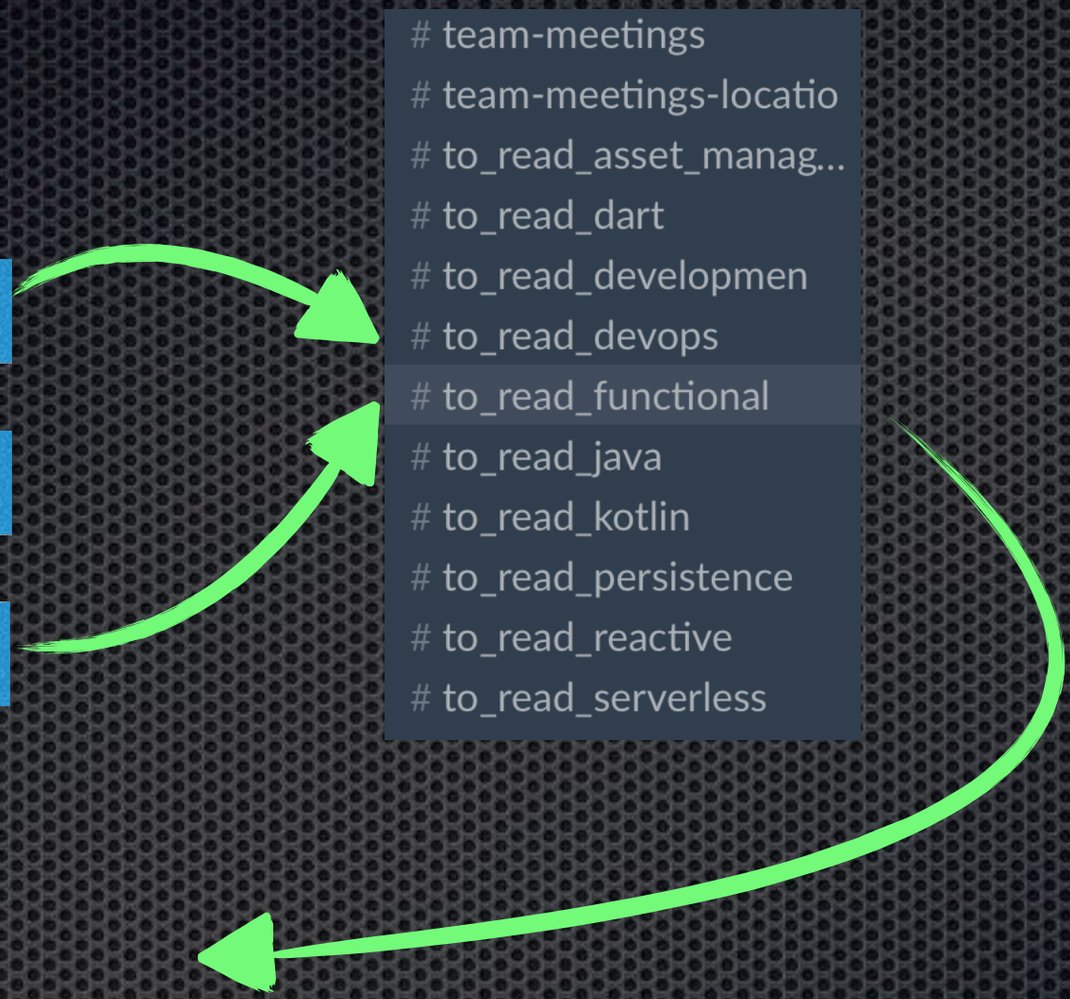
Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

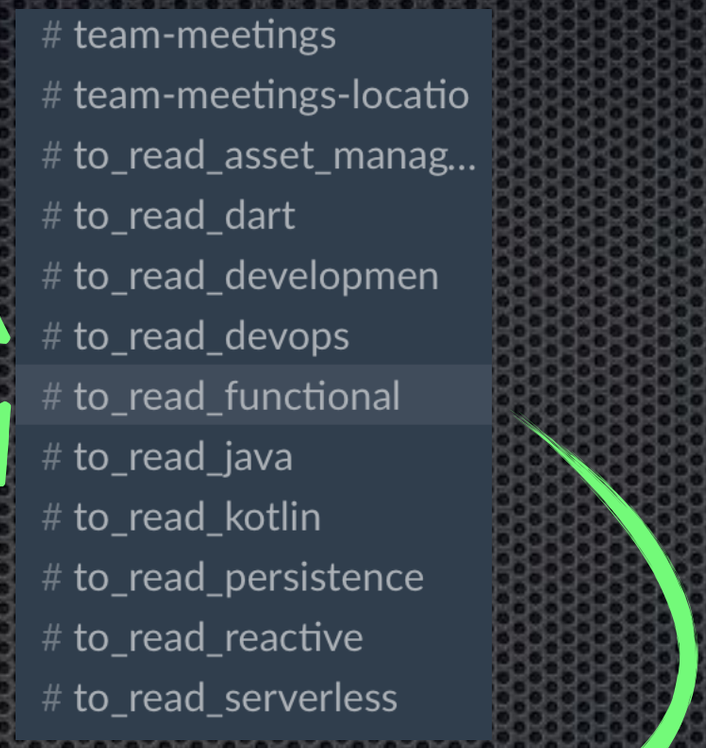
Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

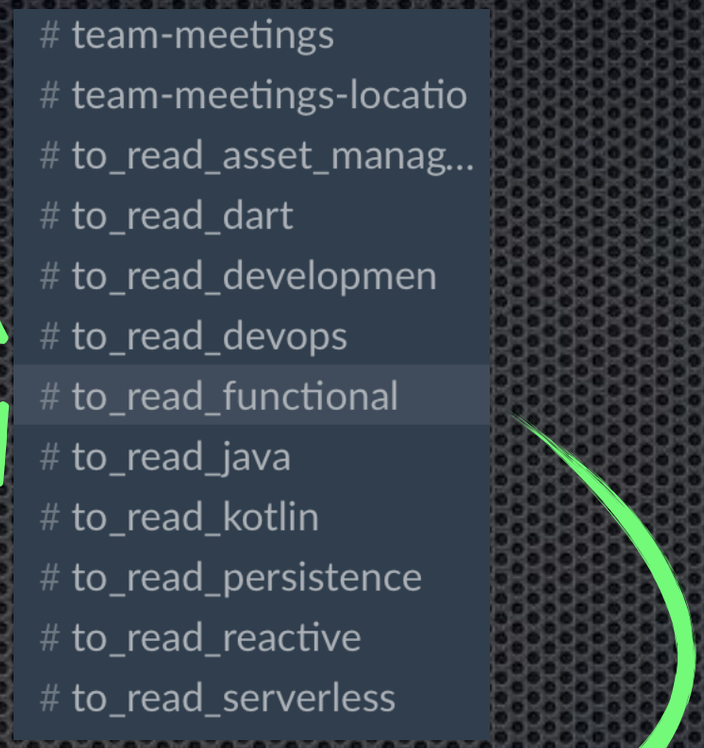
Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

Articles / Blogs

Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

Articles / Blogs

Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

Articles / Blogs

Screencasts

Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

Articles / Blogs

Screencasts

Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

Articles / Blogs

Screencasts

POC

Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers

team-meetings
team-meetings-locatio
to_read_asset_manag...
to_read_dart
to_read_developmen
to_read_devops
to_read_functional
to_read_java
to_read_kotlin
to_read_persistence
to_read_reactive
to_read_serverless

Hacking session

Articles / Blogs

Screencasts

POC

Knowledge Sharing

@SvenRuppert

LTS - Developers

Consultants

Core - Developers



Hacking session

Articles / Blogs

Screencasts

POC

Refactoring Sessions

Knowledge Sharing

How to motivate your team to do it ?

Knowledge Sharing

How to motivate your team to do it ?



Knowledge Sharing

How to motivate your team to do it ?



prepare for

Refactoring - Technologies

choose your weapons

@SvenRuppert

Technical Migrations

@SvenRuppert

Technical Migrations

if you ask the consultants...

Technical Migrations

if you ask the consultants...

to scale or change you need at least .... ;-)

Technical Migrations

if you ask the consultants...

to scale or change you need at least .... ;-)

NoSQL

Technical Migrations

if you ask the consultants...

to scale or change you need at least .... ;-)

NoSQL Akka / Scala

Technical Migrations

@SvenRuppert

if you ask the consultants...

to scale or change you need at least .... ;-)

NoSQL Akka / Scala

Events

Technical Migrations

if you ask the consultants...

to scale or change you need at least .... ;-)

NoSQL

Akka / Scala

Events

Reactive

Technical Migrations

if you ask the consultants...

to scale or change you need at least .... ;-)

NoSQL Akka / Scala

Events Reactive

Functional

Technical Migrations

if you ask the consultants...

to scale or change you need at least .... ;-)

NoSQL Akka / Scala

Events Reactive

Functional

maybe ;-)

Technical Transformation

@SvenRuppert

Technical Transformation

@SvenRuppert

Java OOP

Technical Transformation

Java OOP 

Technical Transformation

@SvenRuppert

Java OOP  Java FRP

Technical Transformation

@SvenRuppert

Java OOP  Java FRP

JDBC - ResultSet

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet →

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing →

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing → Vaadin

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing → Vaadin

HashMap

Technical Transformation

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing → Vaadin

HashMap →

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing → Vaadin

HashMap → Hazelcast - Map

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing → Vaadin

HashMap → Hazelcast - Map

what is the right technology for your team?

Technical Transformation

@SvenRuppert

Java OOP → Java FRP

JDBC - ResultSet → Speedment - Streams

Swing → Vaadin

HashMap → Hazelcast - Map

what is the right technology for your team?

what is the right order?

Technical Transformation

@SvenRuppert

what is the right technology for your team?

Technical Transformation

what is the right technology for your team?

what is the right order?

Technical Transformation

what is the right technology for your team?

- reduce complexity

what is the right order?

Technical Transformation

@SvenRuppert

what is the right technology for your team?

reduce complexity

the team could work with

what is the right order?

Technical Transformation

@SvenRuppert

what is the right technology for your team?

reduce complexity

scale

the team could work with

what is the right order?

Technical Transformation

@SvenRuppert

what is the right technology for your team?

reduce complexity

scale

the team could work with

a lot more....

what is the right order?

Technical Transformation

what is the right technology for your team?

reduce complexity

scale

the team could work with

a lot more....

what is the right order?

start with new modules ?

Technical Transformation

@SvenRuppert

what is the right technology for your team?

reduce complexity

the team could work with

scale

a lot more....

what is the right order?

start with new modules ?

start with the oldest one ?

Technical Transformation

what is the right technology for your team?

reduce complexity

the team could work with

scale

a lot more....

what is the right order?

start with new modules ?

start with the oldest one ?

most komplex one?

Technical Transformation

what is the right technology for your team?

reduce complexity

scale

the team could work with

a lot more....

what is the right order?

start with new modules ?

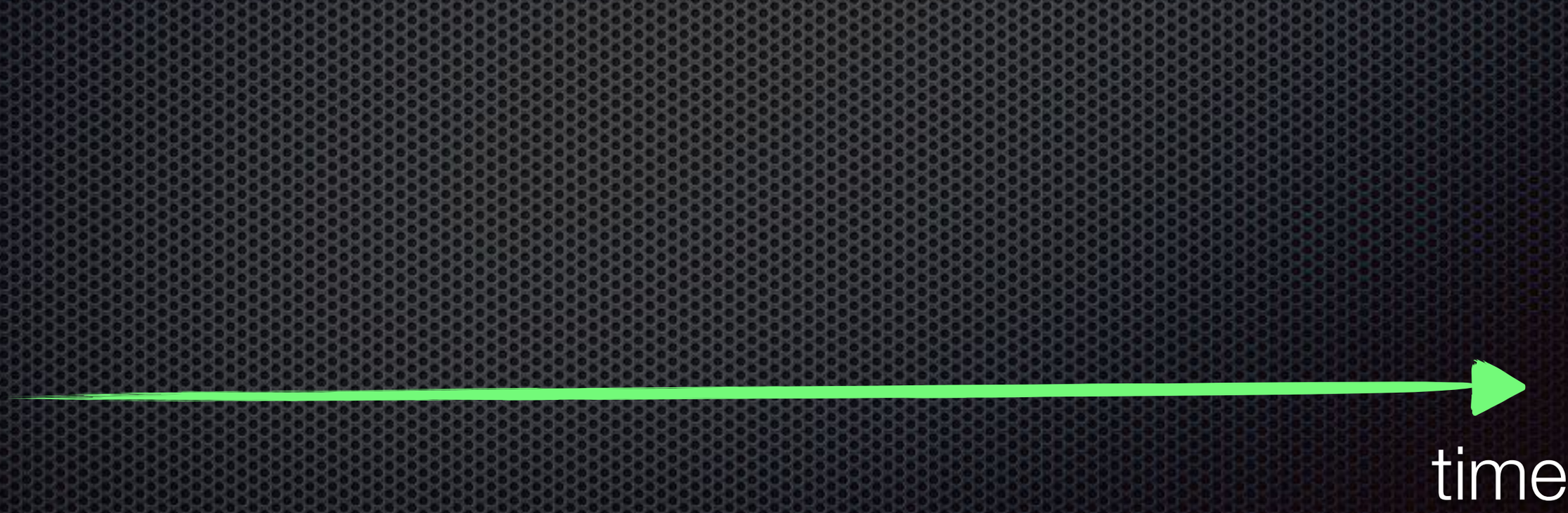
start with the oldest one ?

most komplex one?

.... or ....

on thing that would limit your speed…

on thing that would limit your speed…



Technical Transformation

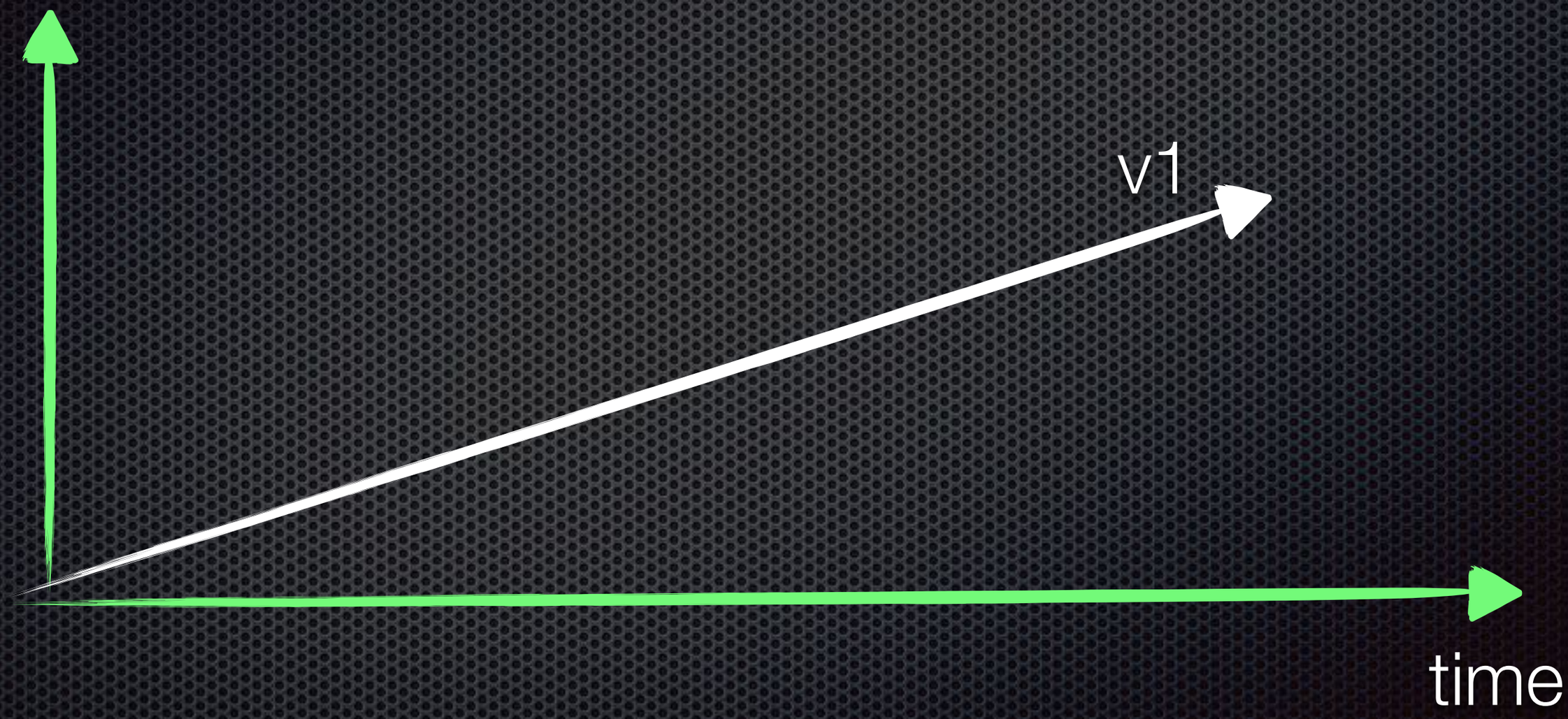
on thing that would limit your speed…

number of versions



Technical Transformation

on thing that would limit your speed…



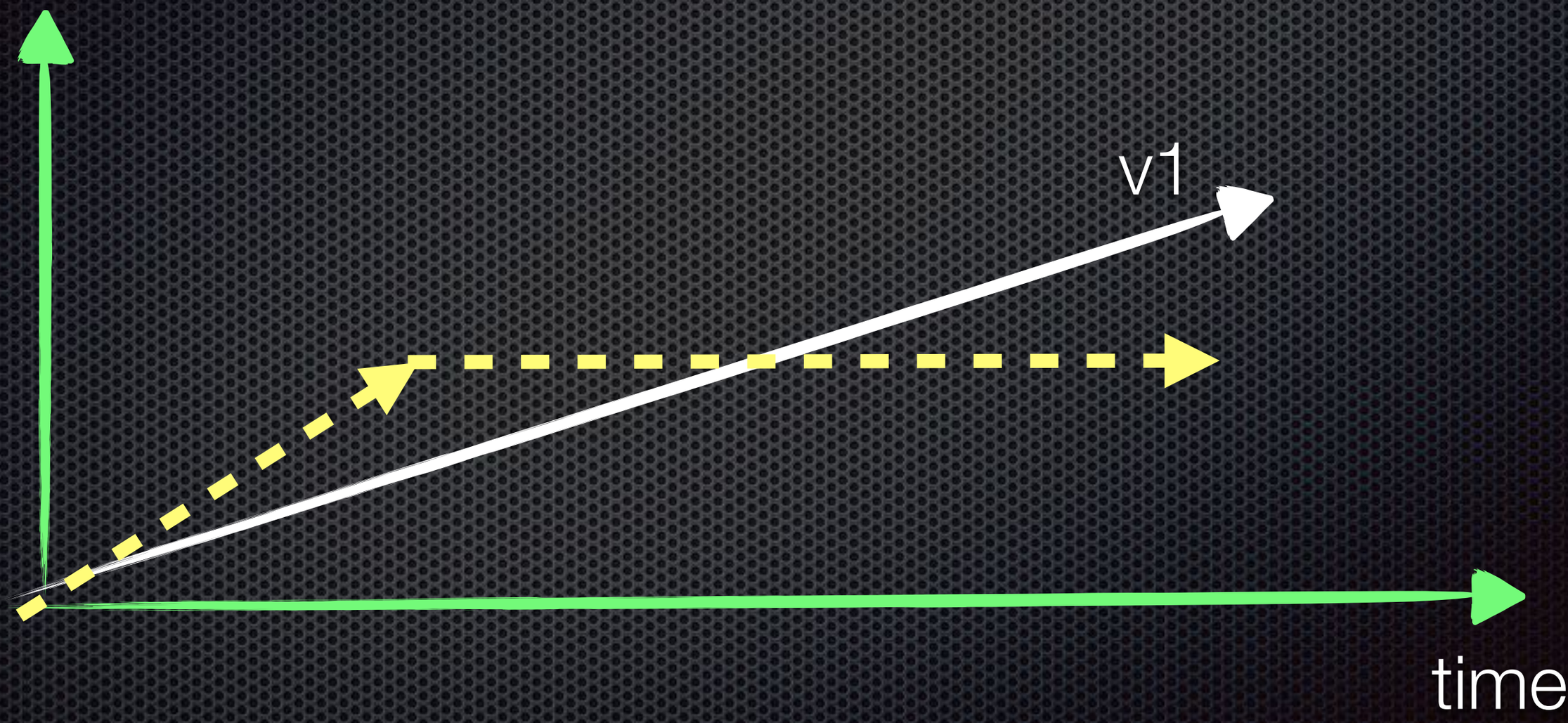
Technical Transformation

on thing that would limit your speed…



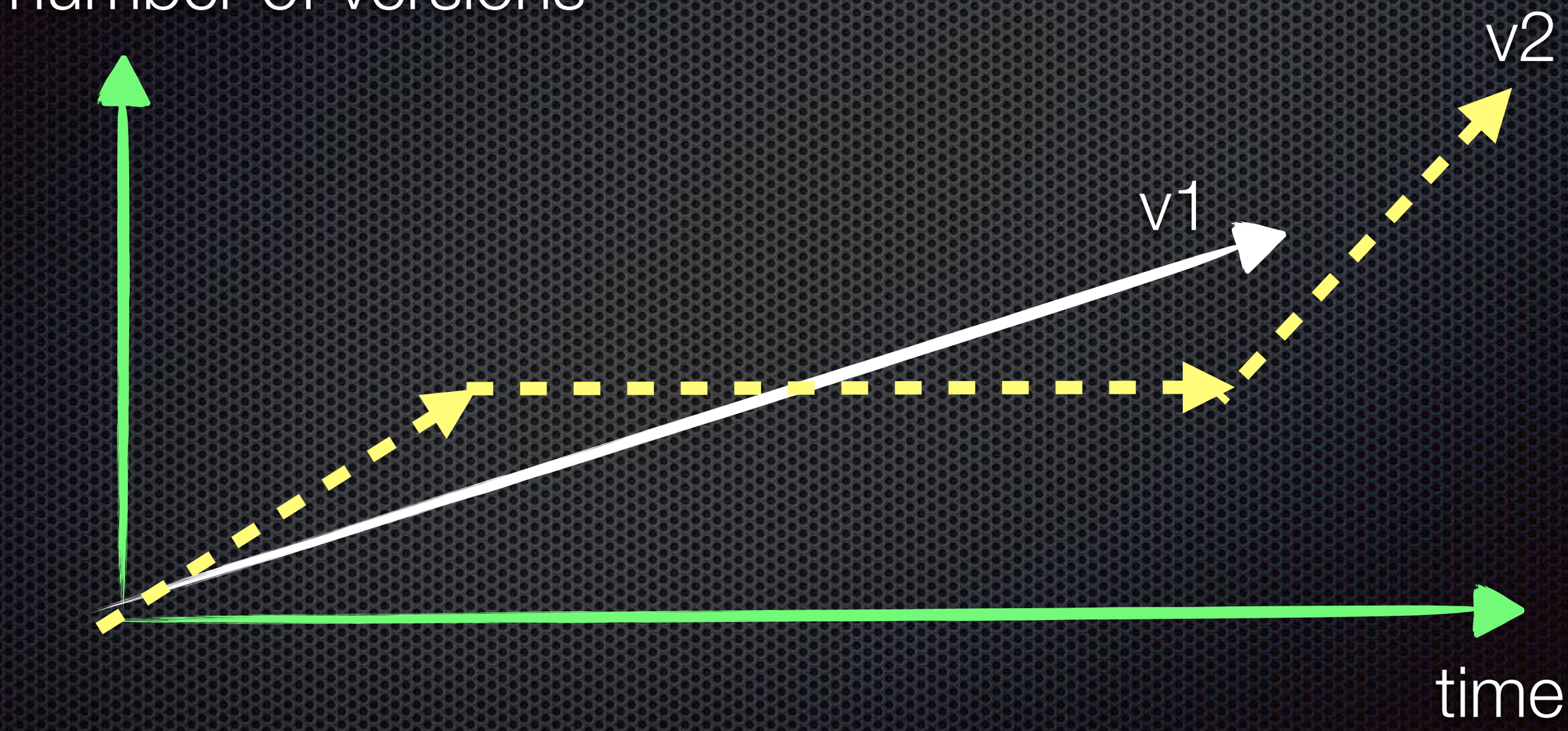
Technical Transformation

on thing that would limit your speed…



Technical Transformation

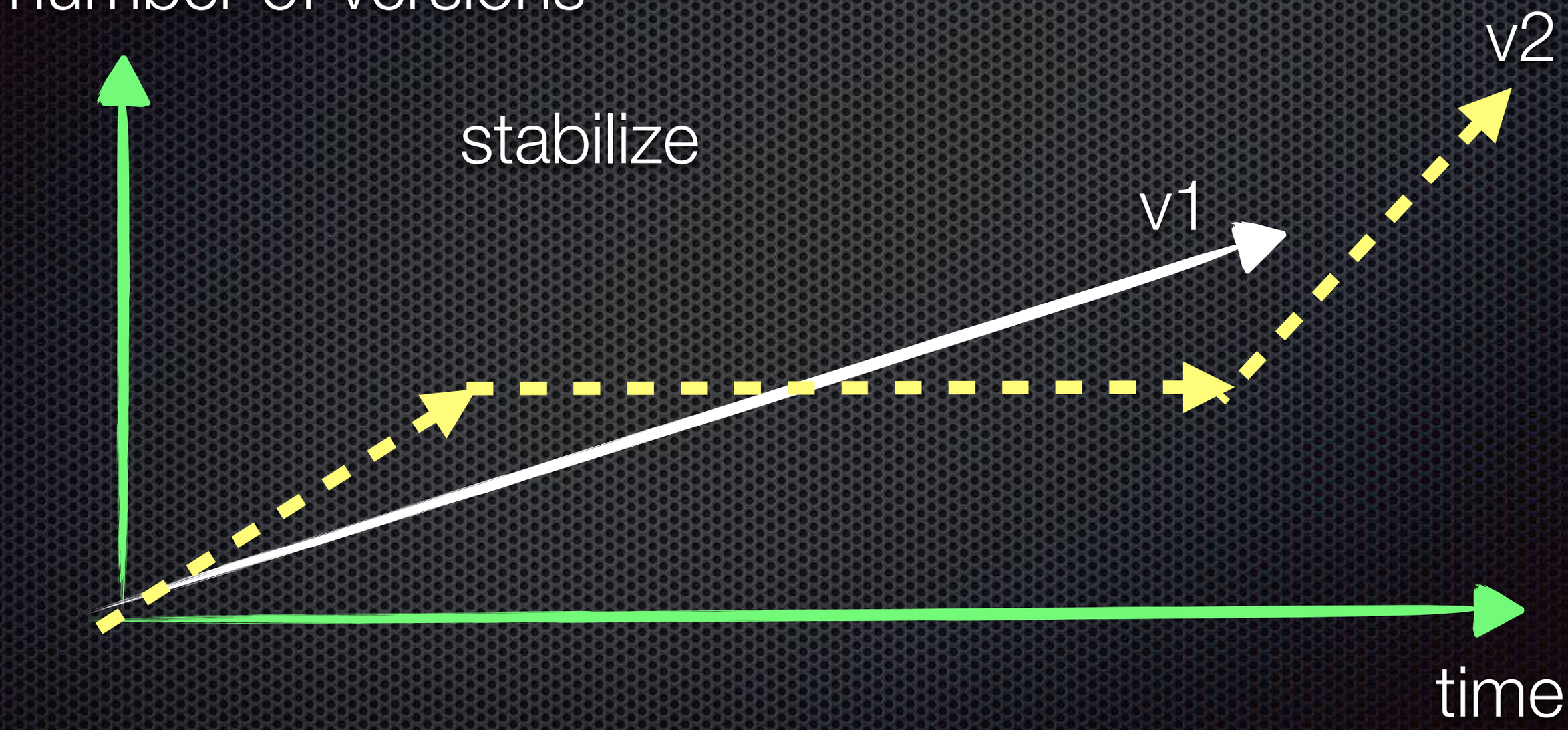
on thing that would limit your speed…



Technical Transformation

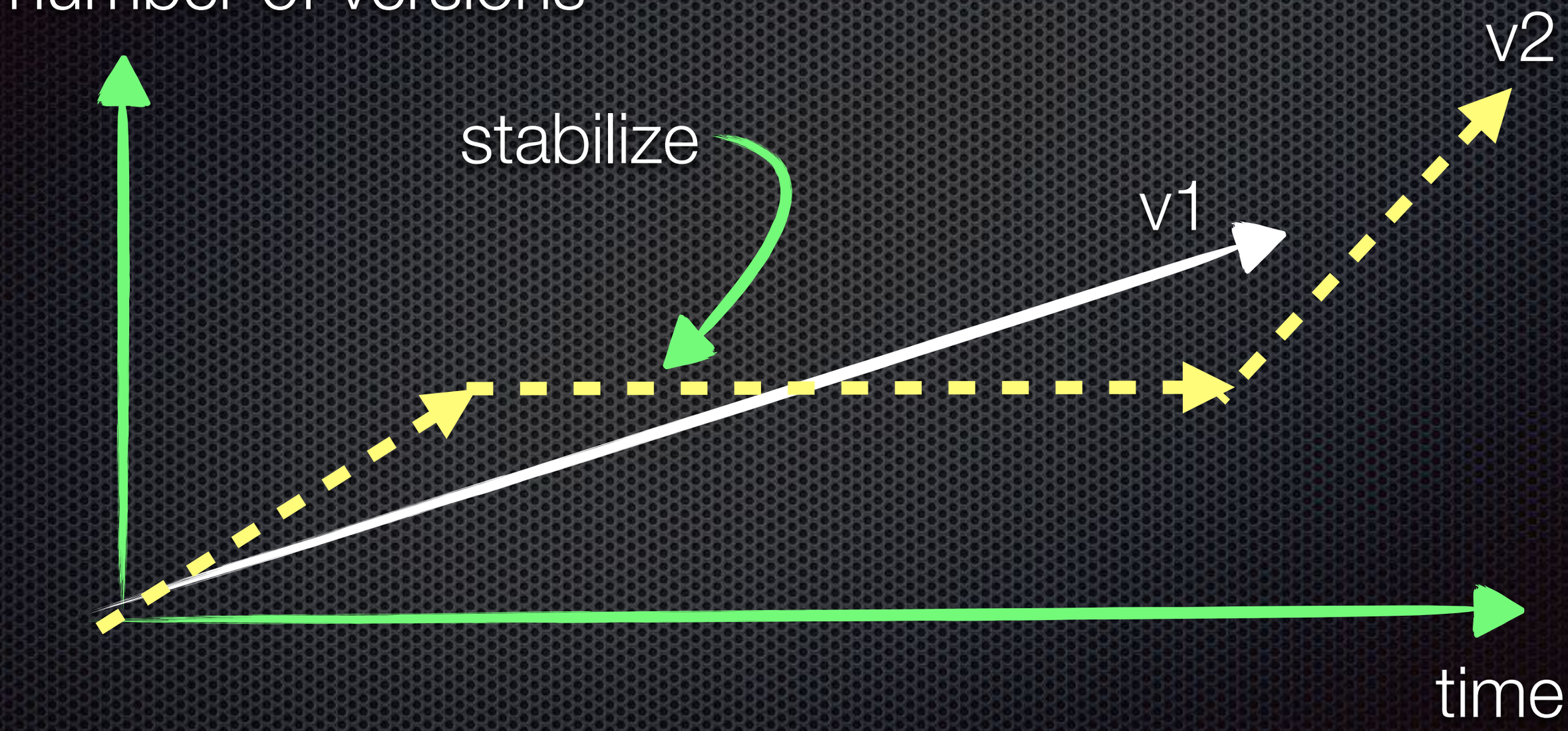
@SvenRuppert

on thing that would limit your speed…



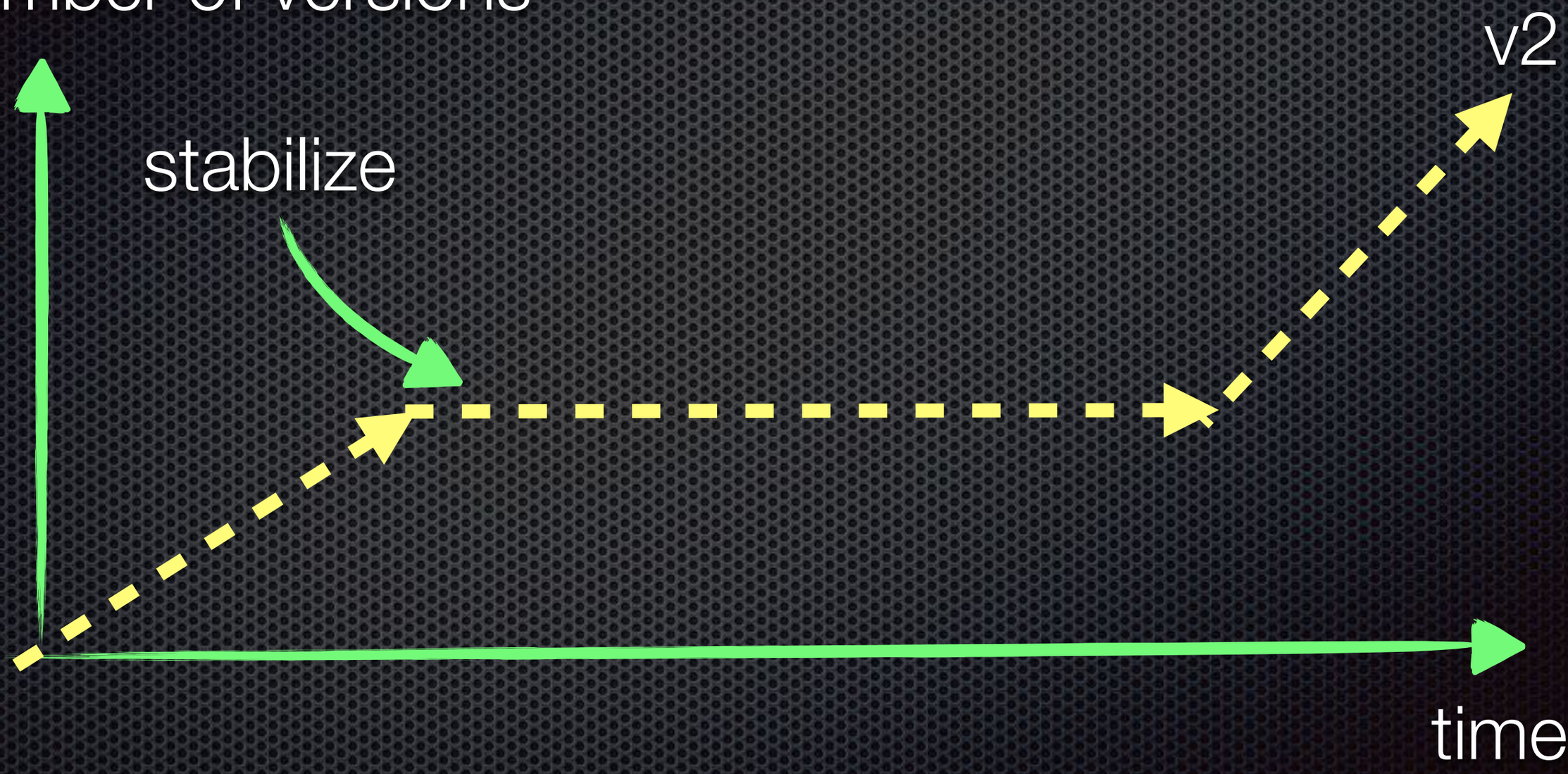
Technical Transformation

on thing that would limit your speed…



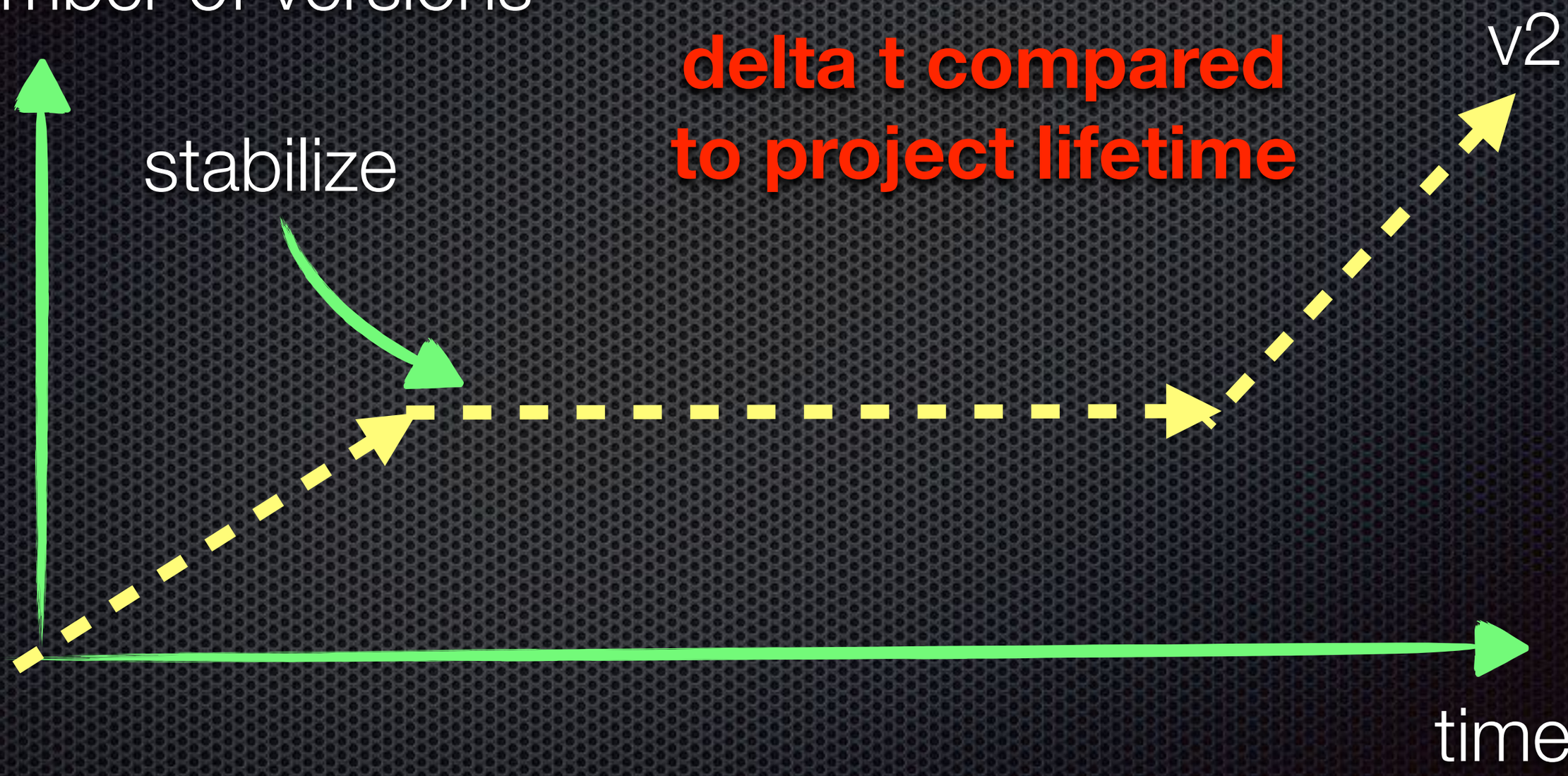
Technical Transformation

on thing that would limit your speed…



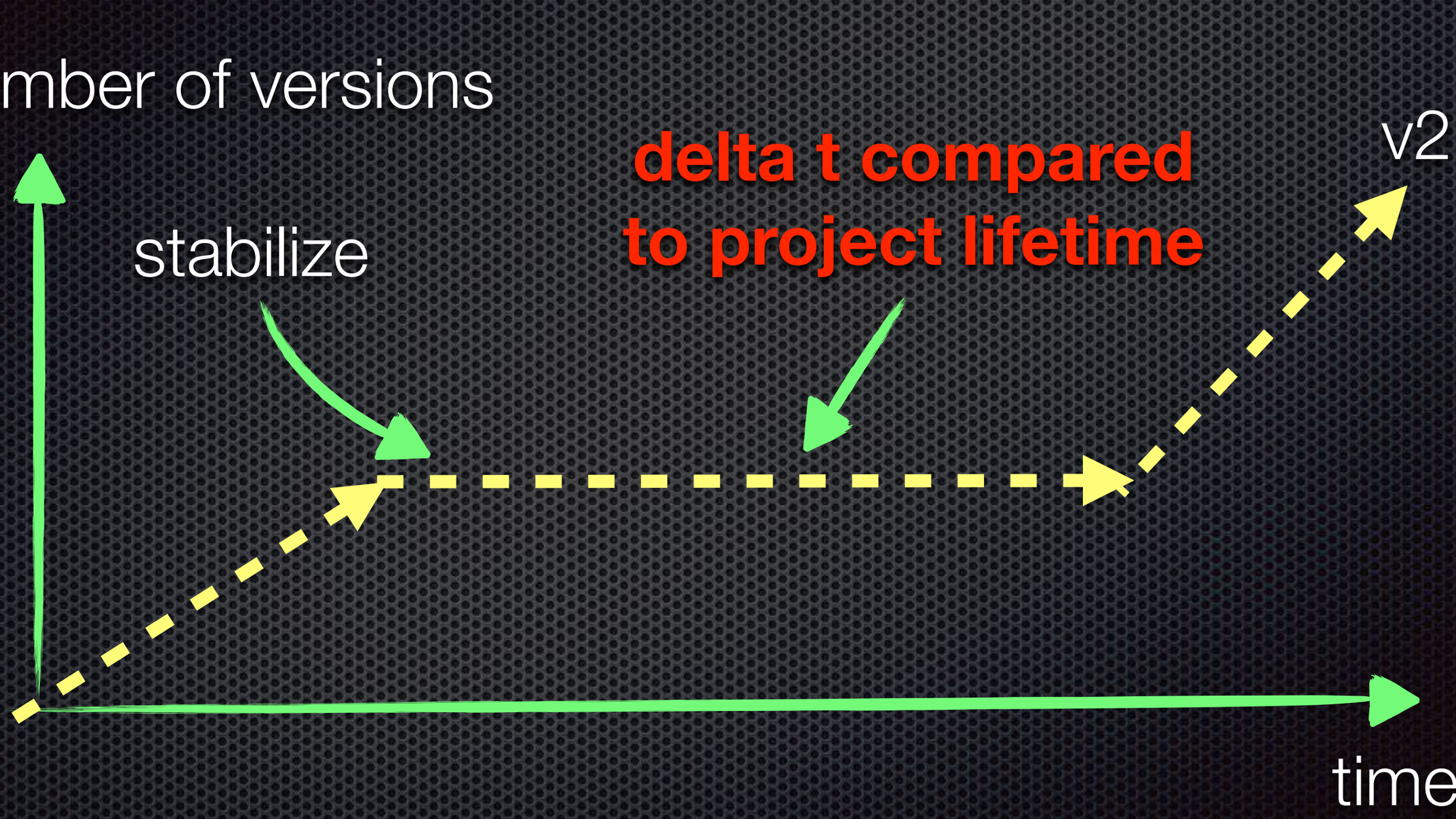
Technical Transformation

on thing that would limit your speed…



Technical Transformation

on thing that would limit your speed…



prepare for

Refactoring - Safety belt

choose your weapons

@SvenRuppert

QM / QS - TDD

Do you have bugs in your code ?

Do you have bugs in your code ?

no

Do you have bugs in your code ?

no

since years

you are
working hard
on this.....

Do you have bugs in your code ?

no

since years you are working hard on this....



QM / QS - TDD

remember....

Codebase is > 13 years old

how to start?

what is the right point to start?

how to increase the quality of the tests?

QM / QS - TDD

remember....

Codebase is > 13 years old

no test coverage

how to start?

what is the right point to start?

how to increase the quality of the tests?

QM / QS - TDD

remember....

Codebase is > 13 years old

no test coverage

how to start?

what is the right point to start?

how to increase the quality of the tests?

how to decrease complexity?

TDD with jUnit

@SvenRuppert

TDD with jUnit

are you using jUnit?

are you using jUnit?

assume that the following would make sense.. ;-)

are you using jUnit?

assume that the following would make sense.. ;-)

```
public class Service {
    public int add(int a, int b){
        if(a<2){
            return (a+b) * -1;
        } else {
            return a+b;
        }
    }
}
```

are you using jUnit?

assume that the following would make sense.. ;-)

```
public class Service {
    public int add(int a, int b){
        if(a<2){
            return (a+b) * -1;
        } else {
            return a+b;
        }
    }
}
```

How many tests you will need ?

are you using jUnit?

assume that the following would make sense.. ;-)

```
public class Service {
    public int add(int a, int b){
        if(a<2){
            return (a+b) * -1;
        } else {
            return a+b;
        }
    }
}
```

How many tests you will need ?

it depends ;-)

TDD with jUnit


```
public class Service {
  public int add(int a, int b){
    if(a<2){
      return (a+b) * -1;
    } else {
      return a+b;
    }
  }
}
```

How many tests you will need ?

it depends ;-)

for line 100% coverage 2

but will this be enough? **maybe ;-)**

TDD with jUnit


```
public class Service {
  public int add(int a, int b){
    if(a<2){
      return (a+b) * -1;
    } else {
      return a+b;
    }
  }
}
```

How many tests you will need ?

it depends ;-)

for line 100% coverage 2

but will this be enough? **maybe ;-)**

how to find out, what will be enough?

TDD with jUnit


```
public class Service {
  public int add(int a, int b){
    if(a<2){
      return (a+b) * -1;
    } else {
      return a+b;
    }
  }
}
```

How many tests you will need ?

it depends ;-)

for line 100% coverage 2

but will this be enough? **maybe ;-)**

how to find out, what will be enough?

how to find the right tests?

practical TDD with Mutation Testing

practical TDD with Mutation Testing

generating the mutants and

Mutation Testing

practical TDD with Mutation Testing

generating the mutants and
running all junit tests

Mutation Testing

practical TDD with Mutation Testing

generating the mutants and

running all junit tests

check the reports

Mutation Testing

practical TDD with Mutation Testing

generating the mutants and

running all junit tests

check the reports

write more / better tests

Mutation Testing

practical TDD with Mutation Testing

generating the mutants and

running all junit tests

check the reports

write more / better tests

loop until quality target reached

Mutation Testing - Hello World

@SvenRuppert

```
public class Service {
  public int add(int a, int b){
    if (a<2) { return (a+b) * -1; }
    else     { return a+b;        }
  }
}
```

100% Line Coverage... and...

Mutation Testing - Hello World

@SvenRuppert

```
public class Service {
  public int add(int a, int b){
    if (a<2) { return (a+b) * -1; }
    else     { return a+b;        }
  }
}
```

100% Line Coverage... and...

```
@Test
public void testAdd001() throws Exception {
  final int add = new Service().add(0, 0);
  Assertions.assertThat(add).isEqualTo(0);
}
```

Mutation Testing - Hello World


```
public class Service {
  public int add(int a, int b){
    if (a<2) { return (a+b) * -1; }
    else     { return a+b;        }
  }
}
```

100% Line Coverage... and...

```
@Test
public void testAdd001() throws Exception {
  final int add = new Service().add(0, 0);
  Assertions.assertThat(add).isEqualTo(0);
}
  @Test
  public void testAdd002() throws Exception {
    final int add = new Service().add(3, 0);
    Assertions.assertThat(add).isEqualTo(3);
  }
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
Assertions.assertThat(add).isEqualTo(0);
```

```
final int add = new Service().add(3, 0);
Assertions.assertThat(add).isEqualTo(3);
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
Assertions.assertThat(add).isEqualTo(0);
```

```
 6. public class Service {
 7.   public int add(int a, int b) {
 8.     if (a < 2) {
 9.       return (a + b) * -1;
10.     } else {
11.       return a + b;
12.     }
13.   }
```

```
final int add = new Service().add(3, 0);
Assertions.assertThat(add).isEqualTo(3);
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
Assertions.assertThat(add).isEqualTo(0);
```

```
 6. public class Service {
 7.   public int add(int a, int b) {
 8.     if (a < 2) {
 9.       return (a + b) * -1;
10.     } else {
11.       return a + b;
12.     }
13.   }
```

```
final int add = new Service().add(3, 0);
Assertions.assertThat(add).isEqualTo(3);
```

```
 6. public class Service {
 7.   public int add(int a, int b) {
 8.     if (a < 2) {
 9.       return (a + b) * -1;
10.     } else {
11.       return a + b;
12.     }
13.   }
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
final int add = new Service().add(3, 0);
```

>> Generated mutations Killed 3

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
final int add = new Service().add(3, 0);
```

>> Generated mutations Killed 3

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Service.java	100%	4/4	43%	3/7

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
final int add = new Service().add(3, 0);
```

>> Generated mutations Killed 3

```
6     public class Service {
7       public int add(int a, int b) {
8   2       if (a < 2) {
9   3           return (a + b) * -1;
10          } else {
11  2           return a + b;
12          }
13      }
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
final int add = new Service().add(3, 0);
```

>> Generated mutations Killed 3

```
8  2   if (a < 2) {
9  3     return (a + b) * -1;
10     } else {
11 2     return a + b;
```

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
final int add = new Service().add(3, 0);
```

>> Generated mutations Killed 3

```
8  2    if (a < 2) {
9  3      return (a + b) * -1;
10     } else {
11 2      return a + b;
```

8
1. changed conditional boundary → SURVIVED
2. negated conditional → KILLED

9
1. Replaced integer addition with subtraction → SURVIVED
2. Replaced integer multiplication with division → SURVIVED
3. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

11
1. Replaced integer addition with subtraction → SURVIVED
2. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

Mutation Testing - Hello World

@SvenRuppert

```
final int add = new Service().add(0, 0);
final int add = new Service().add(3, 0);
```

>> Generated mutations Killed 3

```
8  2    if (a < 2) {
9  3      return (a + b) * -1;
10     } else {
11 2      return a + b;
```

8
1. changed conditional boundary → SURVIVED
2. negated conditional → KILLED

9
1. Replaced integer addition with subtraction → SURVIVED
2. Replaced integer multiplication with division → SURVIVED
3. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

11
1. Replaced integer addition with subtraction → SURVIVED
2. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED

Mutation Testing - in short words

@SvenRuppert

Mutation Testing - in short words

mutation testing is an add on to normal jUnit TDD

Mutation Testing - in short words

mutation testing is an add on to normal jUnit TDD

tools are supporting it well

Mutation Testing - in short words

@SvenRuppert

mutation testing is an add on to normal jUnit TDD

tools are supporting it well

generating and running all tests are time consuming

Mutation Testing - in short words

mutation testing is an add on to normal jUnit TDD

tools are supporting it well

generating and running all tests are time consuming

but most important

Mutation Testing - in short words

mutation testing is an add on to normal jUnit TDD

tools are supporting it well

generating and running all tests are time consuming

but most important

will effect your project structure

Mutation Testing - Lesson Learned

@SvenRuppert

Mutation Testing - Lesson Learned

mutation tests are often leading to

Mutation Testing - Lesson Learned

mutation tests are often leading to

…cleaner code compared to jUnit only

Mutation Testing - Lesson Learned

@SvenRuppert

```
public static final String[] discardCommonPrefix(String a, String b) {
  String[] ret = { a, b };
  int l = a.length() < b.length() ? a.length() : b.length();
  for (int i = 0; i < l; i++) {
    if (a.charAt(i) == b.charAt(i)) {
      if (i + 1 < l) { ret[0] = a.substring(i + 1); ret[1] = b.substring(i + 1); }
      else {
        if (a.length() < b.length())  { ret[0] = ""; ret[1] = b.substring(i + 1); }
        if (a.length() == b.length()) { ret[0] = ""; ret[1] = „"; }
        if (a.length() > b.length())  { ret[0] = a.substring(i + 1); ret[1] = „"; }
      }
    } else break;
  }
  return ret;
}
```

Mutation Testing - Lesson Learned

@SvenRuppert

```
public String[] discardCommonPrefix(String a, String b) {
  final String[] ret = new String[2];
  int l;
  if (a.length() < b.length()) { l = a.length(); }
  else                         { l = b.length(); }

  int position = 0;
  for (; position < l; position++) {
    final char charA = a.charAt(position);
    final char charB = b.charAt(position);
    if (charA != charB) { break; }
  }

  if (position >= a.length()) { ret[0] = ""; }
  else                        { ret[0] = a.substring(position); }

  if (position >= b.length()) { ret[1] = ""; }
  else                        { ret[1] = b.substring(position); }
  return ret;
}
```

Mutation Testing - Lesson Learned

Version 1

Version 2

Mutation Testing - Lesson Learned

@SvenRuppert

Version 1

```
for {
  if {
    if
    else{
      if
      if
      if
    }
  }else
}
```

Version 2

Mutation Testing - Lesson Learned

@SvenRuppert

Version 1

```
for {
  if {
    if
    if
    if
    if
    if
    if
    if
  }
 }else
}
```

Version 2

```
if else

for {
    if
}

if else

if else
```

Example of useless Code

Example of useless Code

@SvenRuppert

```
12  public class ReflectionUtils extends org.reflections.ReflectionUtils {
13
14    public boolean checkInterface(final Type aClass, Class targetInterface) {
15 2    if (aClass.equals(targetInterface)) return true;
16
17     final Type[] genericInterfaces = ((Class) aClass).getGenericInterfaces();
18 2    if (genericInterfaces.length > 0) {
19 3      for (Type genericInterface : genericInterfaces) {
20 2        if (genericInterface.equals(targetInterface)) return true;
21
22         final Type[] nextLevBackArray = ((Class) genericInterface).getGenericInterfaces();
23 2        if (nextLevBackArray.length > 0)
24 3          for (Type type : nextLevBackArray) {
25 2            if (checkInterface(type, targetInterface)) return true;
26           }
27       }
28     }
29     final Type genericSuperclass = ((Class) aClass).getGenericSuperclass();
30 1    if (genericSuperclass != null) {
31 2      if (checkInterface(genericSuperclass, targetInterface)) return true;
32     }
33
34
35 1    return false;
36   }
```

Example of useless Code

@SvenRuppert

```
18 2     if (genericInterfaces.length > 0) {
19 3         for (Type genericInterface : genericInterfaces) {
```

Functions

how use Functions…

@SvenRuppert

Functions - Java8

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
names
    .stream()
    .filter(s -> s.contains("i"))
    .forEach(out::println);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
        "Hugo",
        "Willy",
        "Simon",
        "Erwin",
        "Sigfried"
);
```

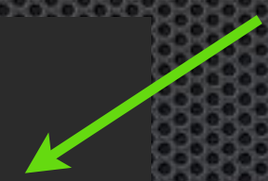
```
names
        .stream()
        .filter(s -> s.contains("i"))
        .forEach(out::println);
```

```
names
        .stream()
        .filter(s -> s.contains("g"))
        .forEach(out::println);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
names
    .stream()
    .filter(s -> s.contains("i"))
    .forEach(out::println);
```



```
names
    .stream()
    .filter(s -> s.contains("g"))
    .forEach(out::println);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
names
    .stream()
    .filter(s -> s.contains("i"))
    .forEach(out::println);
```

```
names
    .stream()
    .filter(s -> s.contains("g"))
    .forEach(out::println);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

```
final String v = "i";
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

```
final String v = "i";
```

```
final Predicate<String> predicateI = s -> s.contains(v);
```

Functions - Java8

@SvenRuppert

```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

```
final String v = "i";
```

```
final Predicate<String> predicateI = s -> s.contains(v);
```

```
names
    .stream()
    .filter(predicateI)
    .forEach(println);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

```
final String v = "i";
```

```
final Predicate<String> predicateI = s -> s.contains(v);
```

```
names
    .stream()
    .filter(predicateI)
    .forEach(println);
```

```
final Predicate<String> predicateG = s -> s.contains("g");
names
    .stream()
    .filter(predicateG)
    .forEach(println);
```

Functions - Java8

@SvenRuppert

```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

```
final String v = "i";
```

```
final Predicate<String> predicateI = s -> s.contains(v);
```

```
names
    .stream()
    .filter(predicateI)
    .forEach(println);
```

```
final Predicate<String> predicateG = s -> s.contains("g");
names
    .stream()
    .filter(predicateG)
    .forEach(println);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

```
final String v = "i";
```

```
final Predicate<String> predicateI = s -> s.contains(v);
```

```
names
    .stream()
    .filter(predicateI)
    .forEach(println);
```

```
final Predicate<String> predicateG = s -> s.contains("g");
names
    .stream()
    .filter(predicateG)
    .forEach(println);
```

Functions - Java8


```
final List<String> names = Arrays.asList(
    "Hugo",
    "Willy",
    "Simon",
    "Erwin",
    "Sigfried"
);
```

```
final Consumer<String> println = out::println;
```

```
final String v = "i";
```

```
final Predicate<String> predicateI = s -> s.contains(v);
```

```
names
    .stream()
    .filter(predicateI)
    .forEach(println);
```

```
final Predicate<String> predicateG = s -> s.contains("g");
names
    .stream()
    .filter(predicateG)
    .forEach(println);
```

Functions - Java8


```
final Function<String, Predicate<String>> funPredicate
    = str -> (String s) -> s.contains(str);
```

Functions - Java8


```
final Function<String, Predicate<String>> funPredicate
    = str -> (String s) -> s.contains(str);
```



Functions - Java8


```
final Function<String, Predicate<String>> funPredicate
    = str -> (String s) -> s.contains(str);
```





Functions - Java8

@SvenRuppert

```
final Function<String, Predicate<String>> funPredicate
    = str -> (String s) -> s.contains(str);
```





```
names
    .stream()
    .filter(funPredicate.apply( t: "i"))
    .forEach(println);
```

Functions - Java8


```
final Function<String, Predicate<String>> funPredicate
    = str -> (String s) -> s.contains(str);
```

```
names
    .stream()
    .filter(funPredicate.apply( t: "i"))
    .forEach(println);
```

```
names
    .stream()
    .filter(funPredicate.apply( t: "g"))
    .forEach(println);
```

Functions - Java8


```
final Function<String, Predicate<String>> funPredicate
    = str -> (String s) -> s.contains(str);
```

```
names
    .stream()
    .filter(funPredicate.apply( t: "i"))
    .forEach(println);
```

```
names
    .stream()
    .filter(funPredicate.apply( t: "g"))
    .forEach(println);
```

prepare for

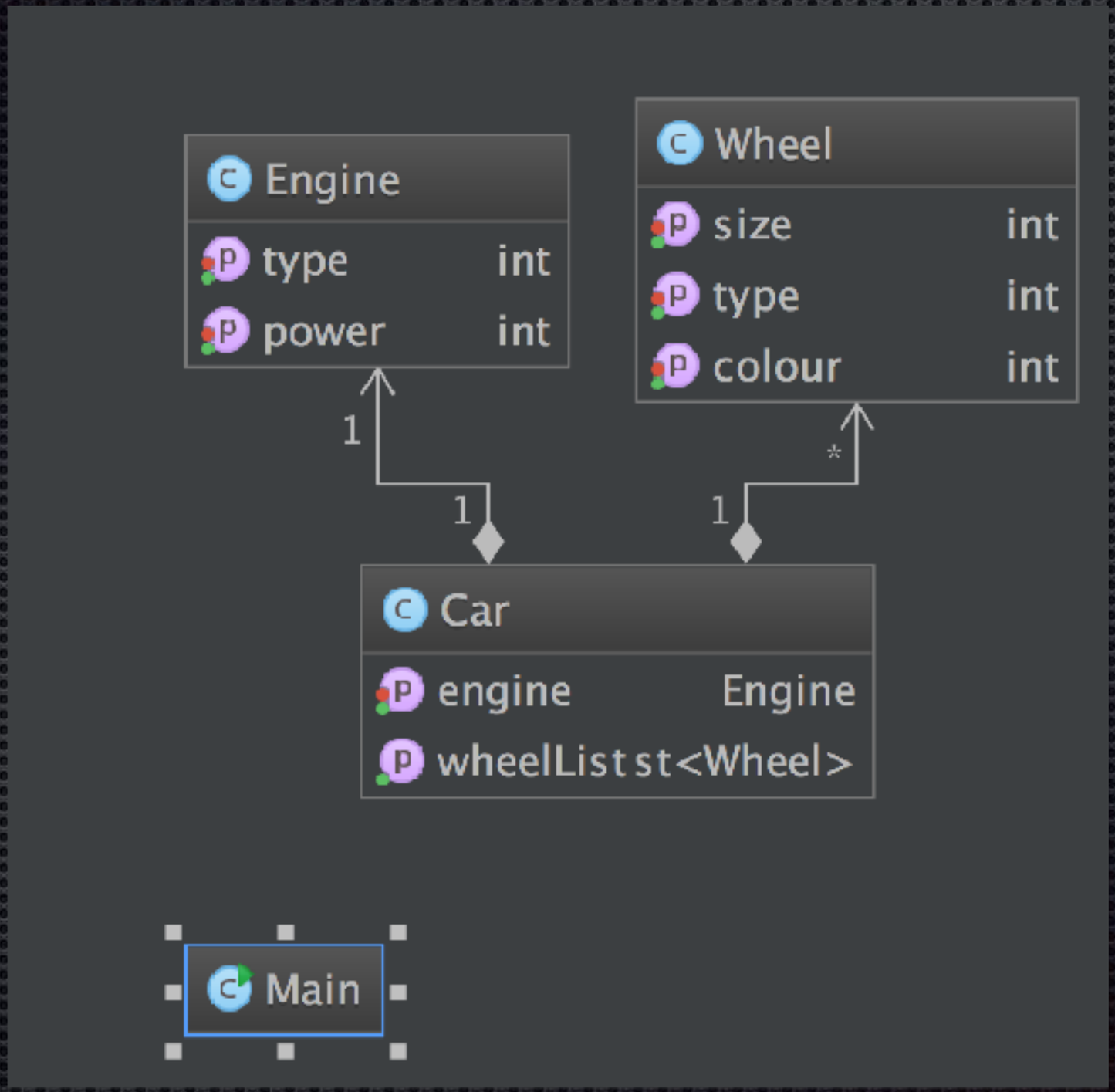
Refactoring - Design Pattern

start developing design pattern again

@SvenRuppert

NestedBuilder - The Beginning

@SvenRuppert



NestedBuilder - The Beginning

@SvenRuppert

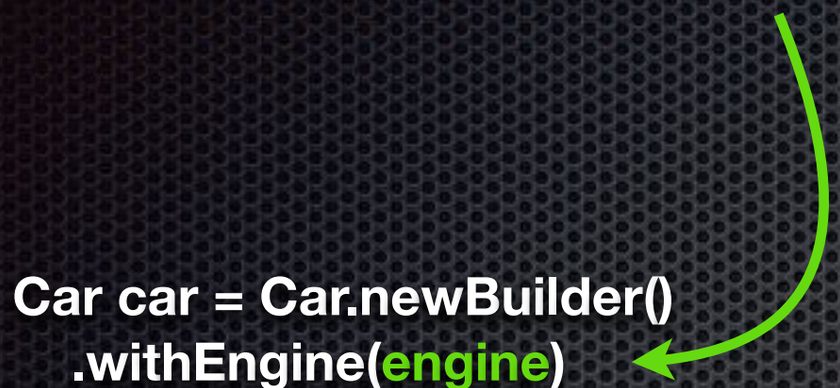
NestedBuilder - The Beginning

@SvenRuppert

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert



```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();

List<Wheel> wheels = new ArrayList<>();
    wheels.add(wheel1);
    wheels.add(wheel2);
    wheels.add(wheel3);
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();

List<Wheel> wheels = new ArrayList<>();
{
    wheels.add(wheel1);
    wheels.add(wheel2);
    wheels.add(wheel3);
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();

List<Wheel> wheels = new ArrayList<>();
{
    wheels.add(wheel1);
    wheels.add(wheel2);
    wheels.add(wheel3);
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();

List<Wheel> wheels = new ArrayList<>();
{
    wheels.add(wheel1);
    wheels.add(wheel2);
    wheels.add(wheel3);
}

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

List<Wheel> wheels = new ArrayList<>();
   wheels.add(wheel1);
   wheels.add(wheel2);
   wheels.add(wheel3);

Car car = Car.newBuilder()
   .withEngine(engine)
   .withWheelList(wheels)
   .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

List<Wheel> wheels = new ArrayList<>();
   wheels.add(wheel1);
   wheels.add(wheel2);
   wheels.add(wheel3);
```

```
Car car = Car.newBuilder()
   .withEngine(engine)
   .withWheelList(wheels)
   .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
```

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

//List<Wheel> wheels = new ArrayList<>();
//    wheels.add(wheel1);
//    wheels.add(wheel2);
//    wheels.add(wheel3);



Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

//List<Wheel> wheels = new ArrayList<>();
//    wheels.add(wheel1);
//    wheels.add(wheel2);
//    wheels.add(wheel3);
 List<Wheel> wheelList = WheelListBuilder.newBuilder()
     .withNewList()
     .addWheel(wheel1)
     .addWheel(wheel2)
     .addWheel(wheel3)
     .build();


Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

//List<Wheel> wheels = new ArrayList<>();
//    wheels.add(wheel1);
//    wheels.add(wheel2);
//    wheels.add(wheel3);

List<Wheel> wheelList = WheelListBuilder.newBuilder()
    .withNewList()
    .addWheel(wheel1)
    .addWheel(wheel2)
    .addWheel(wheel3)
    .build();   //more robust if you add tests at build()


Car car = Car.newBuilder()
   .withEngine(engine)
   .withWheelList(wheels)
   .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

//List<Wheel> wheels = new ArrayList<>();
//    wheels.add(wheel1);
//    wheels.add(wheel2);
//    wheels.add(wheel3);

List<Wheel> wheelList = WheelListBuilder.newBuilder()
    .withNewList()
    .addWheel(wheel1)
    .addWheel(wheel2)
    .addWheel(wheel3)
    .build();   //more robust if you add tests at build()


Car car = Car.newBuilder()
   .withEngine(engine)
   .withWheelList(wheels)
   .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

//List<Wheel> wheels = new ArrayList<>();
//    wheels.add(wheel1);
//    wheels.add(wheel2);
//    wheels.add(wheel3);

List<Wheel> wheelList = WheelListBuilder.newBuilder()
    .withNewList()
    .addWheel(wheel1)
    .addWheel(wheel2)
    .addWheel(wheel3)
    .build();    //more robust if you add tests at build()


Car car = Car.newBuilder()
   .withEngine(engine)
   .withWheelList(wheels)
   .build();
```

how to combine ?

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();

List<Wheel> wheelList = WheelListBuilder.newBuilder()
    .withNewList()
    .addWheel(wheel1)
    .addWheel(wheel2)
    .addWheel(wheel3)
    .build();
```

how to combine ?

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

// Wheel wheel1 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
// Wheel wheel2 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
// Wheel wheel3 = Wheel.newBuilder().withType(2).withColour(3).withSize(4).build();
// List<Wheel> wheelList = WheelListBuilder.newBuilder()
    .withNewList()
    .addWheel(wheel1)
    .addWheel(wheel2)
    .addWheel(wheel3)
    .build();



Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();
```

```
// List<Wheel> wheelList = WheelListBuilder.newBuilder()
    .withNewList()
    .addWheel(wheel1)
    .addWheel(wheel2)
    .addWheel(wheel3)
    .build();
```

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning


```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();
```

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();
```

```
List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();

Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();
```

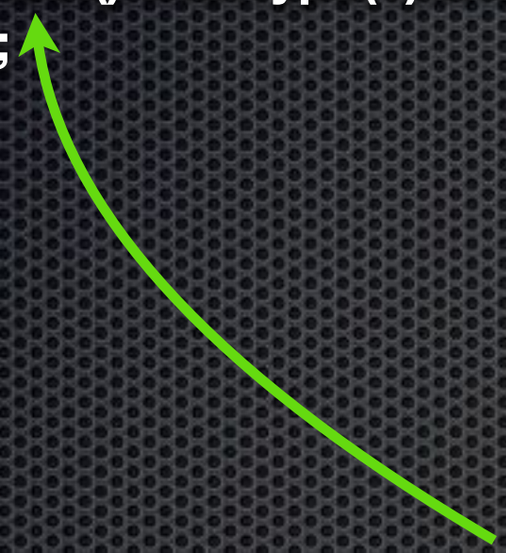
```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();
```



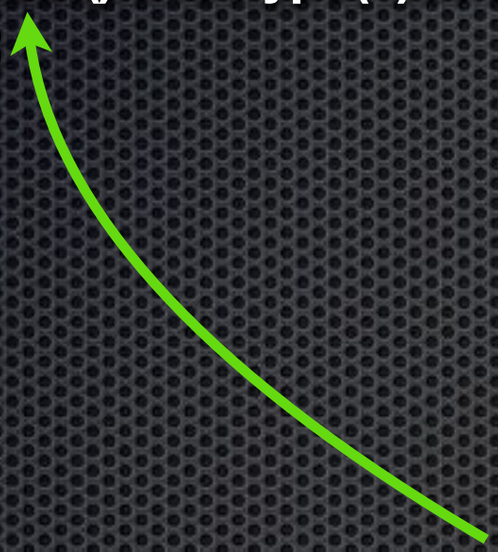
```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();
```



WheelBuilder

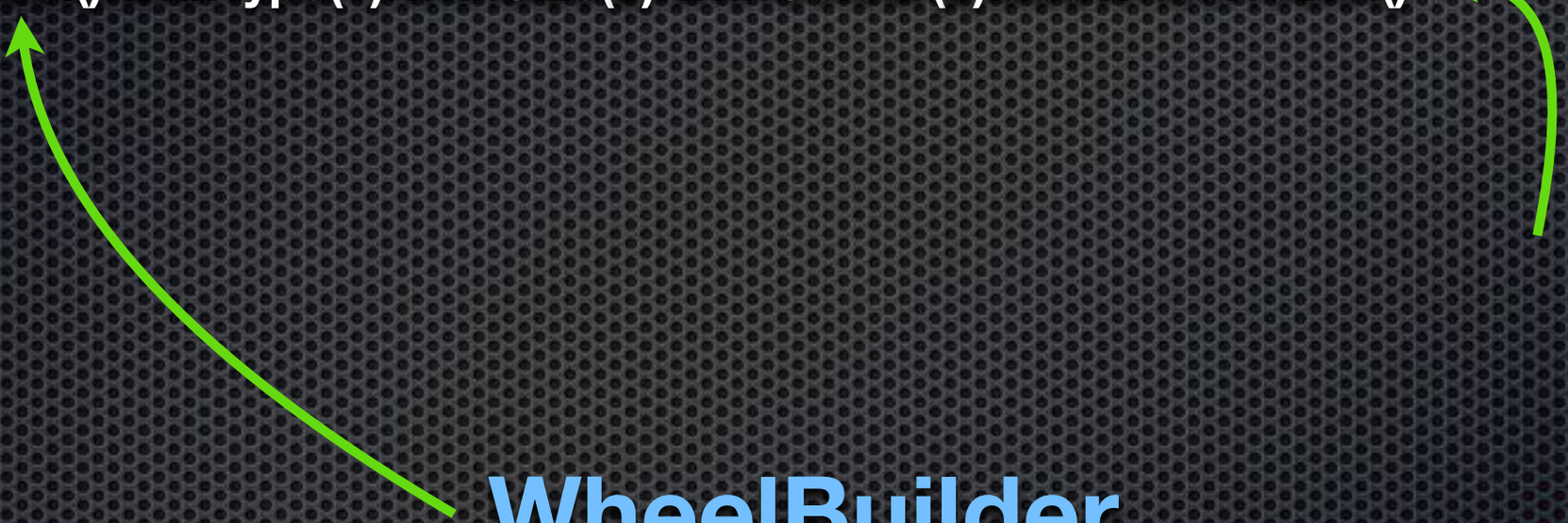
```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();
```



WheelBuilder

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();
```

WheelListBuilder

WheelBuilder

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();

Car car = Car.newBuilder()
    .withEngine(engine)
.withWheelList(wheels)
.build();
```

NestedBuilder - The Beginning

@SvenRuppert

```
Engine engine = Engine.newBuilder().withPower(100).withType(5).build();

List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();

Car car = Car.newBuilder()
    .withEngine(engine)
.withWheelList(wheels)
.build();
```

now we have to combine all

NestedBuilder - The Beginning

@SvenRuppert

```
List<Wheel> wheels = wheelListBuilder
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .build();

Car car = Car.newBuilder()
    .withEngine(engine)
.withWheelList(wheels)
.build();
```

now we have to combine all

NestedBuilder - The Beginning

@SvenRuppert

```
Car car = Car.newBuilder()
    .withEngine(engine)
    .withWheelList(wheels)
    .build();
```

now we have to combine all

NestedBuilder - The Beginning

@SvenRuppert

now we have to combine all

NestedBuilder - The Beginning

now we have to combine all

NestedBuilder - The Beginning

now we have to combine all

```
Car car = Car.newBuilder()
    .addEngine().withPower(100).withType(5).done()
    .addWheels()
        .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
        .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
        .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
        .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .done()
    .build();
```

NestedBuilder - The Beginning

@SvenRuppert

now we have to combine all

```
Car car = Car.newBuilder()
    .addEngine().withPower(100).withType(5).done()
    .addWheels()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
    .addWheel().withType(1).withSize(2).withColour(2).addWheelToList()
.done()
.build();
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {

  public abstract V build()
  protected T parent;

  public <P extends NestedBuilder<T, V>> P
  withParentBuilder(T parent) {
      this.parent = parent;
      return (P) this;
  }
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {

  public abstract V build()
  protected T parent;

  public <P extends NestedBuilder<T, V>> P
  withParentBuilder(T parent) {
      this.parent = parent;
      return (P) this;
  }
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {

  public abstract V build()
  protected T parent;

  public <P extends NestedBuilder<T, V>> P
  withParentBuilder(T parent) {
      this.parent = parent;
      return (P) this;
  }
```

parent will connect itself…

NestedBuilder - The Pattern

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {

  public T done() {
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {

  public T done() {

    Class<?> parentClass = parent.getClass();
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {

  public T done() {

    Class<?> parentClass = parent.getClass();



    try {
      V build = this.build();
      String methodname = "with" + build.getClass().getSimpleName();
      Method method = parentClass.getDeclaredMethod(
                                          methodname, build.getClass());
      method.invoke(parent, build);
    } catch (NoSuchMethodException
            | IllegalAccessException | InvocationTargetException e) {
              e.printStackTrace();
    }
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {

  public T done() {

    Class<?> parentClass = parent.getClass();



    try {
      V build = this.build();
      String methodname = "with" + build.getClass().getSimpleName();
      Method method = parentClass.getDeclaredMethod(
                                          methodname, build.getClass());
      method.invoke(parent, build);
    } catch (NoSuchMethodException
            | IllegalAccessException | InvocationTargetException e) {
              e.printStackTrace();
    }
  }
```

NestedBuilder - The Pattern


```
public abstract class NestedBuilder<T, V> {
  public T done() {
    Class<?> parentClass = parent.getClass();

    try {
      V build = this.build();
      String methodname = "with" + build.getClass().getSimpleName();
      Method method = parentClass.getDeclaredMethod(
                                          methodname, build.getClass());
      method.invoke(parent, build);
    } catch (NoSuchMethodException
             | IllegalAccessException | InvocationTargetException e) {
        e.printStackTrace();
    }
  }
```

NestedBuilder - The Pattern

@SvenRuppert

```
public abstract class NestedBuilder<T, V> {

  public T done() {

    Class<?> parentClass = parent.getClass();

    try {
      V build = this.build();
      String methodname = "with" + build.getClass().getSimpleName();
      Method method = parentClass.getDeclaredMethod(
                                          methodname, build.getClass());
      method.invoke(parent, build);
    } catch (NoSuchMethodException
             | IllegalAccessException | InvocationTargetException e) {
               e.printStackTrace();
    }
  }
```

connect itself with parent

NestedBuilder - The Pattern

NestedBuilder - The Pattern

the basic steps in short words

NestedBuilder - The Pattern

the basic steps in short words

the Parent-Builder will hold the Child-Builder

NestedBuilder - The Pattern

the basic steps in short words

the Parent-Builder will hold the Child-Builder
the Parent-Builder will have a addChild - Method

NestedBuilder - The Pattern

the basic steps in short words

the Parent-Builder will hold the Child-Builder
the Parent-Builder will have a addChild - Method
the Child-Builder will extend the NestedBuilder

the basic steps in short words

the Parent-Builder will hold the Child-Builder
the Parent-Builder will have a addChild - Method
the Child-Builder will extend the NestedBuilder

the rest could be generated with a default Builder-Generator

NestedBuilder - The Pattern


```
public class Parent {
  private KidA kidA;
  private KidB kidB;
  //snipp.....
  public static final class Builder {
    private KidA kidA;
    private KidB kidB;
    // to add manually
    private KidA.Builder builderKidA = KidA.newBuilder().withParentBuilder(this);
    private KidB.Builder builderKidB = KidB.newBuilder().withParentBuilder(this);

    public KidA.Builder addKidA() {
      return this.builderKidA;
    }
    public KidB.Builder addKidB() {
      return this.builderKidB;
    }
    //---------
```

NestedBuilder - The Pattern


```
public class Parent {
  private KidA kidA;
  private KidB kidB;
  //snipp.....
  public static final class Builder {
    private KidA kidA;
    private KidB kidB;
    // to add manually
    private KidA.Builder builderKidA = KidA.newBuilder().withParentBuilder(this);
    private KidB.Builder builderKidB = KidB.newBuilder().withParentBuilder(this);

    public KidA.Builder addKidA() {
      return this.builderKidA;
    }
    public KidB.Builder addKidB() {
      return this.builderKidB;
    }
    //---------
```

NestedBuilder - The Pattern

@SvenRuppert

```
public class Parent {
  private KidA kidA;
  private KidB kidB;
  //snipp.....
  public static final class Builder {
    private KidA kidA;
    private KidB kidB;
    // to add manually
    private KidA.Builder builderKidA = KidA.newBuilder().withParentBuilder(this);
    private KidB.Builder builderKidB = KidB.newBuilder().withParentBuilder(this);

    public KidA.Builder addKidA() {
      return this.builderKidA;
    }
    public KidB.Builder addKidB() {
      return this.builderKidB;
    }
    //---------
```

connect itself with child

NestedBuilder - The Pattern


```
public class Parent {
  private KidA kidA;
  private KidB kidB;
  //snipp.....
  public static final class Builder {
    private KidA kidA;
    private KidB kidB;
    // to add manually
    private KidA.Builder builderKidA = KidA.newBuilder().withParentBuilder(this);
    private KidB.Builder builderKidB = KidB.newBuilder().withParentBuilder(this);

    public KidA.Builder addKidA() {
      return this.builderKidA;
    }
    public KidB.Builder addKidB() {
      return this.builderKidB;
    }
    //---------
```

connect itself with child

NestedBuilder - The Pattern

@SvenRuppert

```
public class Parent {
  private KidA kidA;
  private KidB kidB;
  //snipp.....
  public static final class Builder {
    private KidA kidA;
    private KidB kidB;
    // to add manually
    private KidA.Builder builderKidA = KidA.newBuilder().withParentBuilder(this);
    private KidB.Builder builderKidB = KidB.newBuilder().withParentBuilder(this);

    public KidA.Builder addKidA() {
      return this.builderKidA;
    }
    public KidB.Builder addKidB() {
      return this.builderKidB;
    }
    //---------
```

connect itself with child

switch to Child-Builder

NestedBuilder - The Pattern


```
public static final class Builder
                    extends NestedBuilder<Parent.Builder, KidA> {
```

NestedBuilder - The Pattern

@SvenRuppert

```
public static final class Builder
                    extends NestedBuilder<Parent.Builder, KidA> {
```



NestedBuilder - The Pattern


```
public static final class Builder
                    extends NestedBuilder<Parent.Builder, KidA> {
```



only extends on Child-Builder

NestedBuilder - The Pattern

NestedBuilder - The Pattern

@SvenRuppert

```
Parent build = Parent.newBuilder()

   .addKidA().withNote("A").done()
   .addKidB().withNote("B").done()

   .build();
System.out.println("build = " + build);
```

NestedBuilder - The Pattern


```
Parent build = Parent.newBuilder()

   .addKidA().withNote("A").done()
   .addKidB().withNote("B").done()

   .build();
System.out.println("build = " + build);
```

and a child could be a parent in the same time

NestedBuilder - The Pattern


```
Parent build = Parent.newBuilder()

   .addKidA().withNote("A").done()
   .addKidB().withNote("B").done()

   .build();
System.out.println("build = " + build);
```

and a child could be a parent in the same time

```
Parent build = Parent.newBuilder()

    .addKidA().withNote("A")
                .addKidB().withNote("B").done()
    .done()

    .build();
System.out.println("build = " + build);
```

Refactoring example

based on Java8

@SvenRuppert

Java8 - Functional Interfaces - Example

@SvenRuppert

typical legacy implementation

```
public void writeUser(final User user){
  final int update = update(user, connectionPool());
  // check result
}
```

```
private int update(User user, final JDBCConnectionPool connectionPool) {
  final DataSource dataSource = connectionPool.getDataSource();
  try {
    final Connection connection = dataSource.getConnection();
    final int count;
    try (final Statement statement = connection.createStatement()) {
      final String sql = createUpdteSQL(user);
      count = statement.executeUpdate(sql);
      statement.close();
    }
    dataSource.evictConnection(connection);
    return count;
  } catch (final SQLException e) {
    e.printStackTrace();
  }
  return -1;
}
```

Java8 - Functional Interfaces - Example

typical legacy implementation

```
private String createUpdteSQL(final User user) {
  return ""; // vendor SQL , MySQL/Oracle...
}


public static class User { }

public abstract JDBCConnectionPool connectionPool();

public interface JDBCConnectionPool {...}

public interface DataSource {...}
```

Java8 - Functional Interfaces - Example

Java8 - Functional Interfaces - Example


```
@FunctionalInterface
public interface BasicOperation {
  @NotNull
  String createSQL();
}
```

Java8 - Functional Interfaces - Example

@SvenRuppert

```java
@FunctionalInterface
public interface BasicOperation {
  @NotNull
  String createSQL();
}
```

```java
@FunctionalInterface
public interface Update extends BasicOperation {

  default int update(final JDBCConnectionPool connectionPool) {
    final HikariDataSource dataSource = connectionPool.getDataSource();
    try {
      final Connection connection = dataSource.getConnection();
      final int count;
      try (final Statement statement = connection.createStatement()) {
        final String sql = createSQL();
        count = statement.executeUpdate(sql);
        statement.close();
      }
      dataSource.evictConnection(connection);
      return count;
    } catch (final SQLException e) {
      e.printStackTrace();
    }
    return -1;
  }

}
```

Java8 - Functional Interfaces - Example


```java
@FunctionalInterface
public interface BasicOperation {
  @NotNull
  String createSQL();
}
```

```java
@FunctionalInterface
public interface Update extends BasicOperation {

  default int update(final JDBCConnectionPool connectionPool) {
    final HikariDataSource dataSource = connectionPool.getDataSource();
    try {
      final Connection connection = dataSource.getConnection();
      final int count;
      try (final Statement statement = connection.createStatement()) {
        final String sql = createSQL();
        count = statement.executeUpdate(sql);
        statement.close();
      }
      dataSource.evictConnection(connection);
      return count;
    } catch (final SQLException e) {
      e.printStackTrace();
    }
    return -1;
  }

}
```

Java8 - Functional Interfaces - Example


```java
@FunctionalInterface
public interface BasicOperation {
  @NotNull
  String createSQL();
}
```

```java
@FunctionalInterface
public interface Update extends BasicOperation {

  default int update(final JDBCConnectionPool connectionPool) {
    final HikariDataSource dataSource = connectionPool.getDataSource();
    try {
      final Connection connection = dataSource.getConnection();
      final int count;
      try (final Statement statement = connection.createStatement()) {
        final String sql = createSQL();
        count = statement.executeUpdate(sql);
        statement.close();
      }
      dataSource.evictConnection(connection);
      return count;
    } catch (final SQLException e) {
      e.printStackTrace();
    }
    return -1;
  }

}
```

Java8 - Functional Interfaces - Example


```java
@FunctionalInterface
public interface BasicOperation {
  @NotNull
  String createSQL();
}
```

```java
@FunctionalInterface
public interface Update extends BasicOperation {

  default int update(final JDBCConnectionPool connectionPool) {
    final HikariDataSource dataSource = connectionPool.getDataSource();
    try {
      final Connection connection = dataSource.getConnection();
      final int count;
      try (final Statement statement = connection.createStatement()) {
        final String sql = createSQL();
        count = statement.executeUpdate(sql);
        statement.close();
      }
      dataSource.evictConnection(connection);
      return count;
    } catch (final SQLException e) {
      e.printStackTrace();
    }
    return -1;
  }

}
```

Java8 - Functional Interfaces - Example

Java8 - Functional Interfaces - Example


```
@FunctionalInterface
public interface DAO {
  JDBCConnectionPool connectionPool();
}
```

Java8 - Functional Interfaces - Example


```
@FunctionalInterface
public interface DAO {
  JDBCConnectionPool connectionPool();
}
```

```
public abstract class GenericDAO implements DAO {

  private JDBCConnectionPool connectionPool;

  public DAO workOnPool(final JDBCConnectionPool connectionPool) {
    this.connectionPool = connectionPool;
    return this;
  }

  @Override
  public JDBCConnectionPool connectionPool() { return connectionPool; }
}
```

Java8 - Functional Interfaces - Example

@SvenRuppert

```java
@FunctionalInterface
public interface DAO {
  JDBCConnectionPool connectionPool();
}
```



```java
public abstract class GenericDAO implements DAO {

  private JDBCConnectionPool connectionPool;

  public DAO workOnPool(final JDBCConnectionPool connectionPool) {
    this.connectionPool = connectionPool;
    return this;
  }

  @Override
  public JDBCConnectionPool connectionPool() { return connectionPool; }
}
```

Java8 - Functional Interfaces - Example


```java
@FunctionalInterface
public interface DAO {
  JDBCConnectionPool connectionPool();
}
```

```java
public abstract class GenericDAO implements DAO {

  private JDBCConnectionPool connectionPool;

  public DAO workOnPool(final JDBCConnectionPool connectionPool) {
    this.connectionPool = connectionPool;
    return this;
  }

  @Override
  public JDBCConnectionPool connectionPool() { return connectionPool; }
}
```

Java8 - Functional Interfaces - Example

Java8 - Functional Interfaces - Example

@SvenRuppert

```
public interface UserDAO extends DAO {

  default void writeUser(final User user) {
    ((Update) () -> createInsert(user)).update(connectionPool());
  }
```

```
String createInsert(User user);
```

Java8 - Functional Interfaces - Example

@SvenRuppert



```
public interface UserDAO extends DAO {

  default void writeUser(final User user) {
    ((Update) () -> createInsert(user)).update(connectionPool());
  }
```

```
String createInsert(User user);
```

Java8 - Functional Interfaces - Example

Java8 - Functional Interfaces - Example

@SvenRuppert

```
public class UserDAOHSQL extends GenericDAO implements UserDAO {

  public String createInsert(final User user) {
    final String sql = "INSERT INTO CUSTOMER " +
        "( CUSTOMER_ID, FIRSTNAME, LASTNAME, EMAIL ) " +
        " VALUES " +
        "(  " +
        user.getCustomerID() + ", " +
        "'" + user.getFirstname() + "', " +
        "'" + user.getLastname() + "', " +
        "'" + user.getEmail() + "' " +
        ")";
    return sql;
  }
```

Java8 - Functional Interfaces - Example

@SvenRuppert



```
public class UserDAOHSQL extends GenericDAO implements UserDAO {

  public String createInsert(final User user) {
    final String sql = "INSERT INTO CUSTOMER " +
        "( CUSTOMER_ID, FIRSTNAME, LASTNAME, EMAIL ) " +
        " VALUES " +
        "(  " +
        user.getCustomerID() + ", " +
        "'" + user.getFirstname() + "', " +
        "'" + user.getLastname() + "', " +
        "'" + user.getEmail() + "' " +
        ")";
    return sql;
  }
```

Java8 - Functional Interfaces - Example





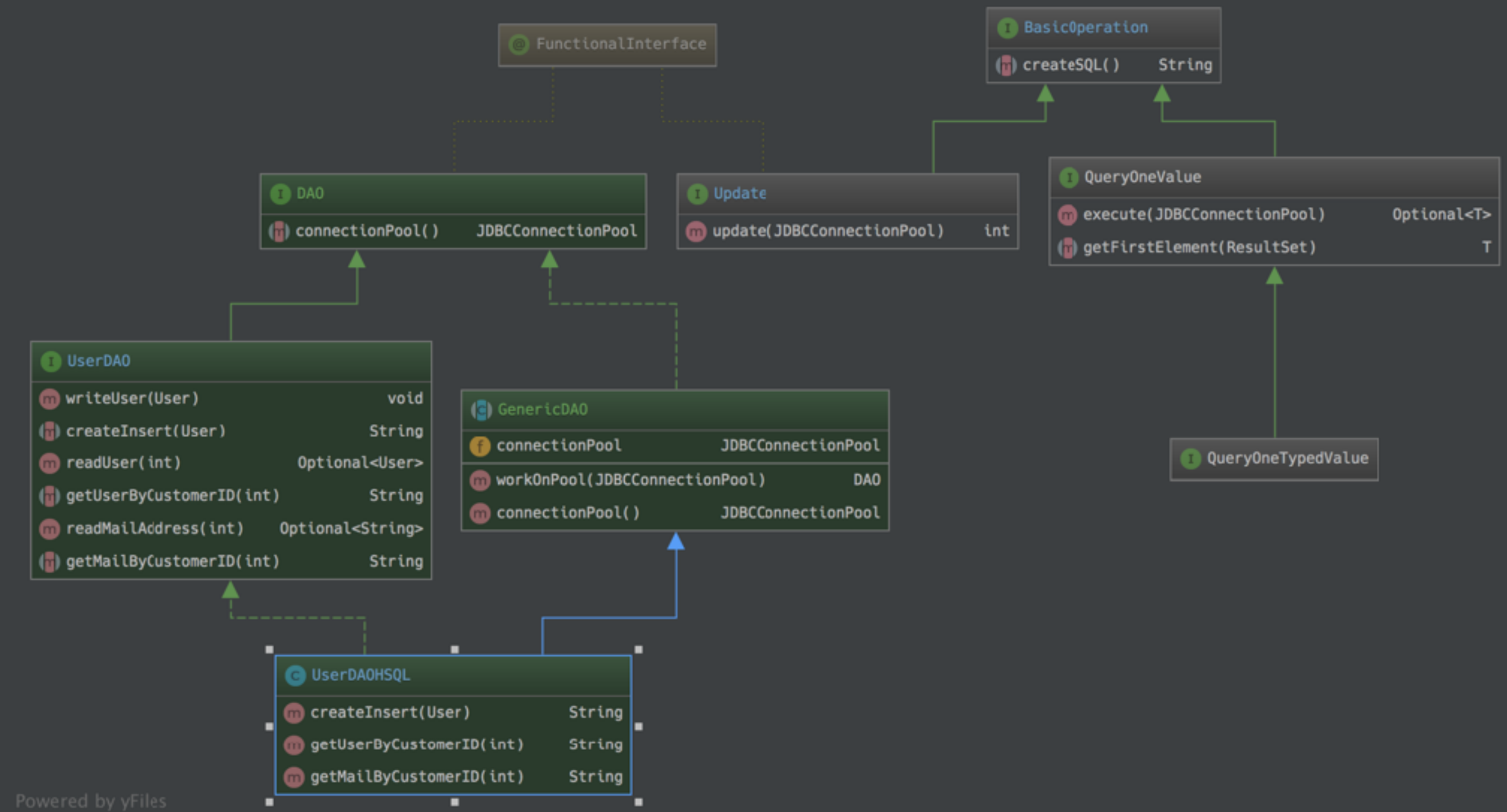
```
public class UserDAOHSQL extends GenericDAO implements UserDAO {

  public String createInsert(final User user) {
    final String sql = "INSERT INTO CUSTOMER " +
        "( CUSTOMER_ID, FIRSTNAME, LASTNAME, EMAIL ) " +
        " VALUES " +
        "(  " +
        user.getCustomerID() + ", " +
        "'" + user.getFirstname() + "', " +
        "'" + user.getLastname() + "', " +
        "'" + user.getEmail() + "' " +
        ")";
    return sql;
  }
```

Java8 - Functional Interfaces - Example

Java8 - Functional Interfaces - Example

@SvenRuppert



Java8 - Functional Interfaces - Example

Sourcecode is on github

Java8 - Functional Interfaces - Example

Sourcecode is on github

svenruppert / JDBC-DAO

Code | Issues 1 | Pull requests 0 | Projects 1

minimalistic JDBC based DAO

connection-pool | dao | design-pattern | functional | java | java8

Interpreter - pre Java8

Interpreter - pre Java8

Interpreter - pre Java8


```
interface Expression {
  int interpret();
}

public static abstract class Operator implements Expression {
  protected final Expression leftExpression;
  protected final Expression rightExpression;

  public Operator(final Expression leftExpression, final Expression rightExpression) {
    this.leftExpression = leftExpression;
    this.rightExpression = rightExpression;
  }
}
```

Interpreter - pre Java8


```
interface Expression {
  int interpret();
}

public static abstract class Operator implements Expression {
  protected final Expression leftExpression;
  protected final Expression rightExpression;

  public Operator(final Expression leftExpression, final Expression rightExpression) {
    this.leftExpression = leftExpression;
    this.rightExpression = rightExpression;
  }
}

public static class Add extends Operator {
  public Add(final Expression leftExpression, final Expression rightExpression) {
    super(leftExpression, rightExpression);
  }

  @Override
  public int interpret() {
    return leftExpression.interpret() + rightExpression.interpret();
  }
}
```

Interpreter - pre Java8

Interpreter - pre Java8


```
public static class Number implements Expression {
  private final int n;

  public Number(int n) { this.n = n; }

  @Override
  public int interpret() { return n; }
}
```

Interpreter - pre Java8


```
public static class Number implements Expression {
  private final int n;

  public Number(int n) { this.n = n; }

  @Override
  public int interpret() { return n; }
}
```

```
public static Expression getOperator(String s, Expression left, Expression right) {
  return
      (s.equals("+")) ? new Add(left, right) :
      (s.equals("-")) ? new Subtract(left, right) :
      (s.equals("*")) ? new Product(left, right) :
                        null; // not nice ;-)
}
```

Interpreter - pre Java8

Interpreter - pre Java8


```
public static boolean isOperator(String s) {
  return s.equals("+") || s.equals("-") || s.equals("*");
}
```

Interpreter - pre Java8


```
public static boolean isOperator(String s) {
  return s.equals("+") || s.equals("-") || s.equals("*");
}
```

```
public static int evaluate(String expression) {
  final Stack<Expression> stack = new Stack<>();
  for (final String s : expression.split( regex: " ")) {
    if (isOperator(s)) {
      Expression right = stack.pop();
      Expression left = stack.pop();
      stack.push(getOperator(s, left, right));
    } else {
      Expression i = new Number(Integer.parseInt(s));
      stack.push(i);
    }
  }
  return stack.pop().interpret();
}
```

Interpreter - pre Java8


```
public static boolean isOperator(String s) {
  return s.equals("+") || s.equals("-") || s.equals("*");
}
```

```
public static int evaluate(String expression) {
  final Stack<Expression> stack = new Stack<>();
  for (final String s : expression.split( regex: " ")) {
    if (isOperator(s)) {
      Expression right = stack.pop();
      Expression left = stack.pop();
      stack.push(getOperator(s, left, right));
    } else {
      Expression i = new Number(Integer.parseInt(s));
      stack.push(i);
    }
  }
  return stack.pop().interpret();
}
```

```
public static void main(String[] args) {
  String expression = "7 3 - 2 1 + *";
  System.out.println(evaluate(expression));
}
```

Interpreter - Java8 - Java9

Interpreter - Java8 - Java9


```
private static final Map<String, IntBinaryOperator> OPERATOR_MAP = new HashMap<>();

static {
  OPERATOR_MAP.put("+", (a, b) -> a + b);
  OPERATOR_MAP.put("*", (a, b) -> a * b);
  OPERATOR_MAP.put("-", (a, b) -> a - b);
}
```

Interpreter - Java8 - Java9


```
private static final Map<String, IntBinaryOperator> OPERATOR_MAP = new HashMap<>();

static {
  OPERATOR_MAP.put("+", (a, b) -> a + b);
  OPERATOR_MAP.put("*", (a, b) -> a * b);
  OPERATOR_MAP.put("-", (a, b) -> a - b);
}
```

```
public static int evaluate(String expression) {
  Stack<Integer> stack = new Stack<>();
  for (final String s : expression.split( regex: " ")) {
    IntBinaryOperator op = OPERATOR_MAP.get( s );
    if (op != null) {
      int right = stack.pop();
      int left = stack.pop();
      stack.push(op.applyAsInt( left, right ));
    } else {
      stack.push(Integer.parseInt(s));
    }
  }
  return stack.pop();
}
```

Main Idea - DataStructure to Function

Interpreter - Java8 - Java9

@SvenRuppert

Main Idea - DataStructure to Function

```
private static final Map<String, IntBinaryOperator> OPERATOR_MAP = new HashMap<>();

static {
  OPERATOR_MAP.put("+", (a, b) -> a + b);
  OPERATOR_MAP.put("*", (a, b) -> a * b);
  OPERATOR_MAP.put("-", (a, b) -> a - b);
}
```

Interpreter - Java8 - Java9

Main Idea - DataStructure to Function

```
private static final Map<String, IntBinaryOperator> OPERATOR_MAP = new HashMap<>();

static {
  OPERATOR_MAP.put("+", (a, b) -> a + b);
  OPERATOR_MAP.put("*", (a, b) -> a * b);
  OPERATOR_MAP.put("-", (a, b) -> a - b);
}
```

```
private static final Function<String, Optional<IntBinaryOperator>> str2OpFNC = (str) ->
    (isNull(str))      ? empty() :
    (str.equals("+")) ? of((a, b) -> a + b) :
    (str.equals("*")) ? of((a, b) -> a * b) :
    (str.equals("-")) ? of((a, b) -> a - b) :
                        empty();
```

Memoizing

partial cache-ing ?

full example and code
http://www.functional-reactive.org/

@SvenRuppert

prepare for

Refactoring - Clean your code

nano refactoring on language level

@SvenRuppert

clean your code

@SvenRuppert

clean your code

@SvenRuppert

delete JavaDoc

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

replace loops with streams

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

replace loops with streams

extract Functional Interfaces

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

replace loops with streams

extract Functional Interfaces

remove anonymous classes

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

replace loops with streams

extract Functional Interfaces

remove anonymous classes

use everywhere the same language level

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

replace loops with streams

extract Functional Interfaces

remove anonymous classes

use everywhere the same language level

Why ?

clean your code

@SvenRuppert

delete JavaDoc

Why ?

use final for Attributes as much as possible

replace loops with streams

extract Functional Interfaces

remove anonymous classes

use everywhere the same language level

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

replace loops with streams

extract Functional Interfaces

remove anonymous classes

use everywhere the same language level

Why ?

know the part you want to refactor

clean your code

@SvenRuppert

delete JavaDoc

use final for Attributes as much as possible

replace loops with streams

extract Functional Interfaces

remove anonymous classes

use everywhere the same language level

Why ?

know the part you want to refactor

homogeneous sources are better for machines

prepare for

Refactoring - Summary

the full picture

@SvenRuppert

Refactoring - Summary

prepare your team

change the way of working

work on your project management

build your safety belt - mutation testing

divide and conquer - @Inject

build design pattern again

extract cross functionality - Proxy's

clean your code

prepare for

Management - Summary

the full picture for the C-Level

@SvenRuppert

Management - Summary

@SvenRuppert

Now it is time to relax again ;-)



Management - Summary

@SvenRuppert

Now it is time to relax again ;-)



Summary

places to read more about it

or contact me ;-) @SvenRuppert

Summary

places to read more about it

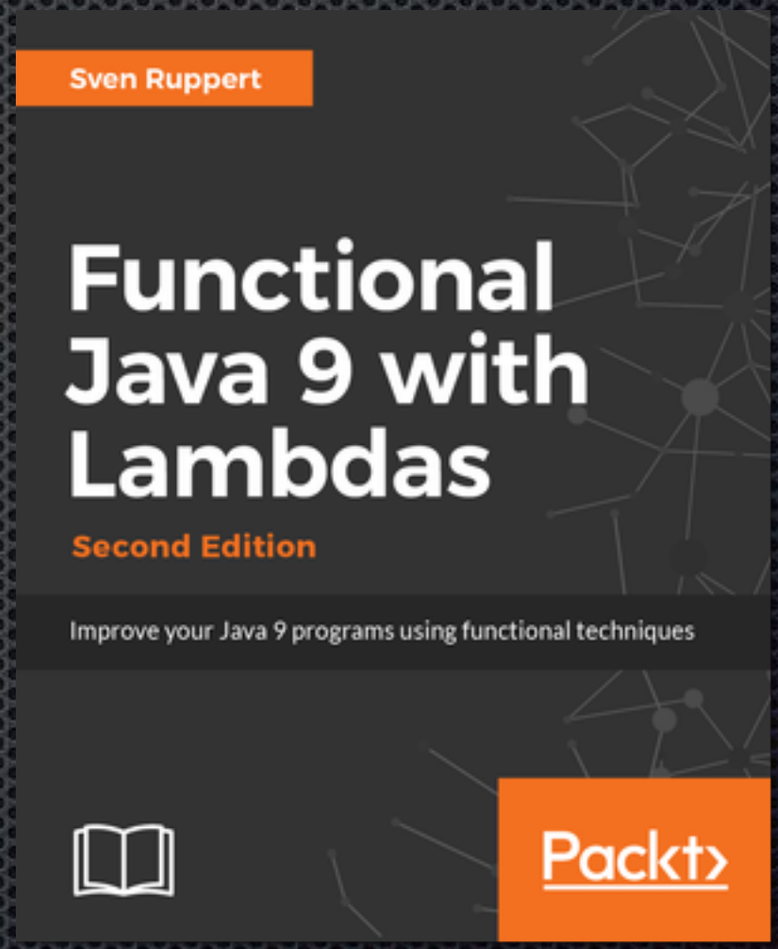
www.functional-reactive.org

or contact me ;-) @SvenRuppert

Summary

places to read more about it

www.functional-reactive.org

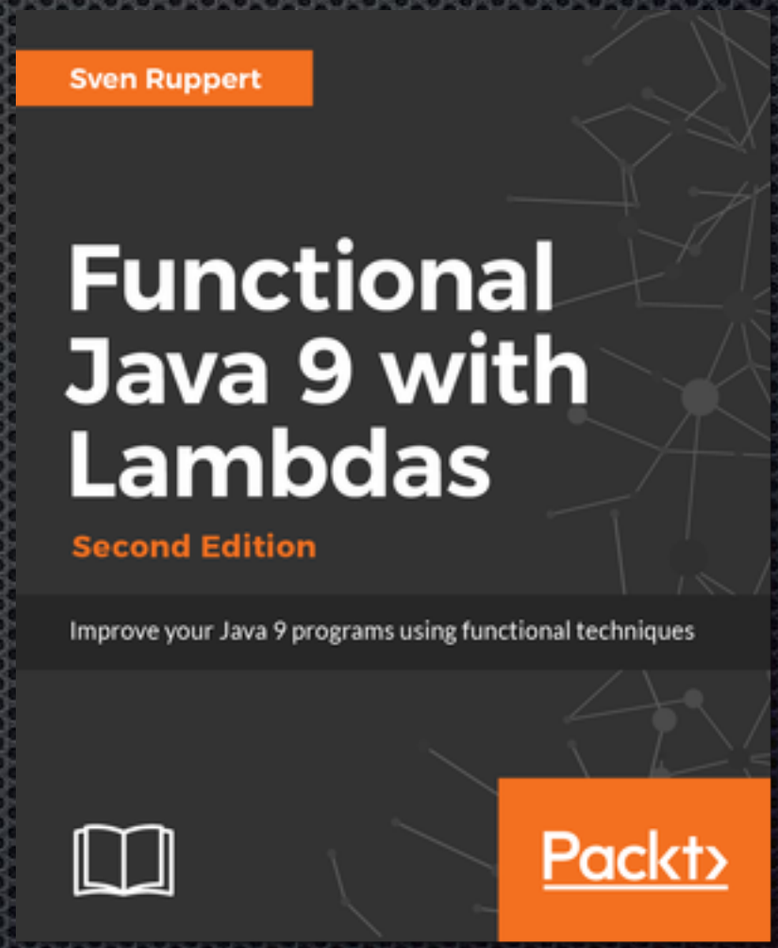


or contact me ;-) @SvenRuppert

Summary

places to read more about it

www.functional-reactive.org



www.java-9.org

or contact me ;-) @SvenRuppert

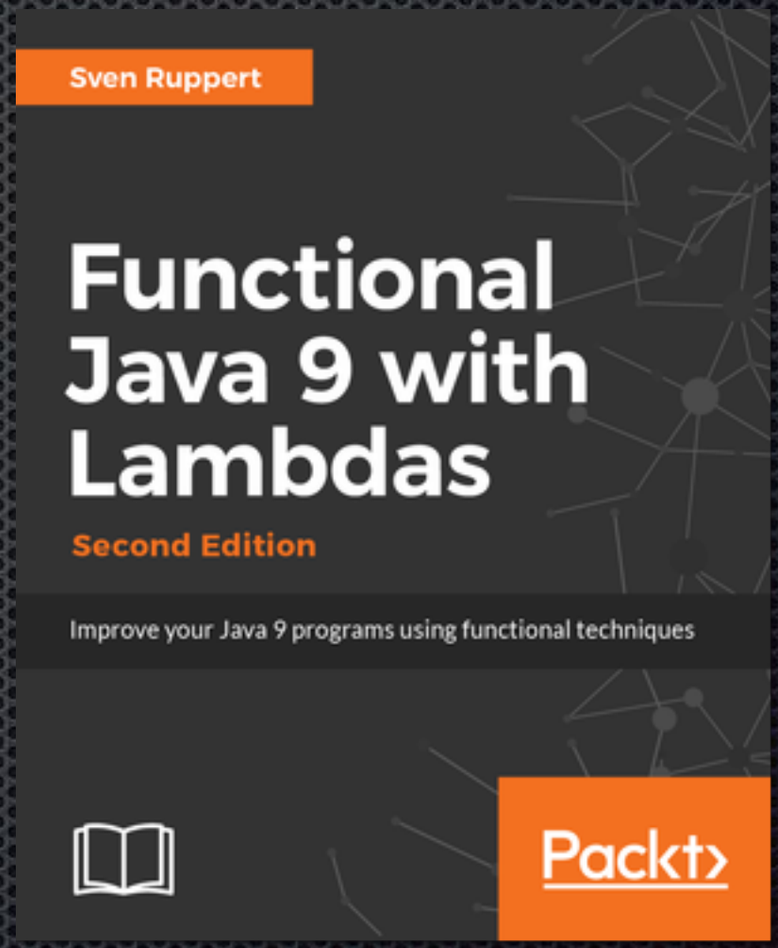
Summary

places to read more about it

www.functional-reactive.org

Thank You !!!

www.java-9.org



or contact me ;-) @SvenRuppert