

# **EVENT SOURCING WITH JVM LANGUAGES**

Rahul Somasunderam

# **ABOUT ME**

# ABOUT ME

- Wanted Event Sourcing in 2016

# ABOUT ME

- Wanted Event Sourcing in 2016
- Didn't like anything I saw

# ABOUT ME

- Wanted Event Sourcing in 2016
- Didn't like anything I saw
- Built my own thing - Grooves

# ABOUT ME

- Wanted Event Sourcing in 2016
- Didn't like anything I saw
- Built my own thing - Grooves
- I solve problems in HealthCare

# WHAT'S WRONG WITH OTHER TOOLS

A screenshot of a web browser window displaying a CNN Politics article. The title of the article is "Trump: 'Nobody knew health care could be so complicated'". The article is by Kevin Liptak, CNN White House Producer, and was updated at 4:10 AM ET on Tuesday, February 28, 2017. The CNN logo is visible in the top left corner of the page. The browser interface shows various icons for navigation, search, and social sharing.

Trump: 'Nobody knew health care could be so complicated'

By [Kevin Liptak](#), CNN White House Producer

Updated 4:10 AM ET, Tue February 28, 2017

DEVELOPING NOW

Rahul

www.cnn.com/2017/02/27/politics/trump-health-care-complicated/index.h...

CNN politics 45 CONGRESS SECURITY THE NINE TRUMPAMERICA STATE f t i

EMAIL FACEBOOK TWITTER MORE

DEVISING NOW

TRUMP TALKS OBAMA CARE WITH INSURERS, DOCTORS

# WHERE I WORK



Used by 70+ Health Systems, 1.4k Locations, 8.5k  
Physicians.

# **ABOUT YOU**

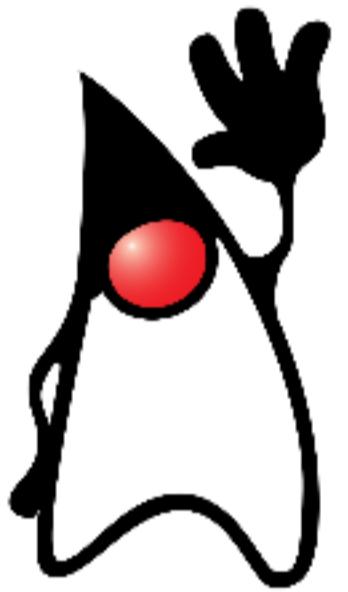
# ABOUT YOU



# ABOUT YOU



# ABOUT YOU



# ABOUT YOU



# WHAT IS EVENT SOURCING

*Treat your database like you treat your application logs*

# WHAT DOES THAT MEAN

Create

Read

~~Update~~

~~Delete~~

# BAD EXAMPLE 1

Mark and Jane share a bank account. The account has a balance of \$75. Mark goes to an ATM and withdraws \$50. At the same time Jane attempts to withdraw \$50.

This is what our code looks like

```
1 @Transactional  
boolean withdraw(String accountNumber, double amount) {  
    double balance = accountService.getBalance(accountNumber);  
    if (balance > amount) {  
        balance -= amount;  
        accountService.setBalance(accountNumber, balance);  
        return true;  
    } else {  
        return false;  
    }  
}
```

# WHAT BANKS TYPICALLY DO

Tag, Monat und Jahr der Eintragung 1	DM-Betrag der Einlage oder Rückzahlung in Buchstaben 2	Betrag der Einlage 3 DM   Pf		Betrag der Rückzahlung 4 DM   Pf	
05.05.	hiebe - d 86 adikAdi)	87			
07.05.	86 Jh	10			
26.05.	25				
12.06.	93				
28.06.					
86	Technikunivis				

1. Blatt

Guthaben 5	Unterschrift des Beamten und Tagesstempel	
	DM	Pf
87		
97		
47		
32		
16		
10		

1. Blatt

**Nr. 72.259.103**

By Deutsche Bundespost (Scan by User:Mattes) - Scan (700dpi, Millionen Farben), Public Domain

**BUT THAT WAS BANKING**

# **BUT THAT WAS BANKING**

If you can look at your logs and debug your application, you  
are already doing that.

## **BUT THAT WAS BANKING**

If you can look at your logs and debug your application, you  
are already doing that.

Banks defined their business model this way hundreds of  
years ago.

# WHAT ABOUT MY DOMAIN?

```
/* Aggregate */  
class Account {  
    String accountNumber;  
}
```

```
/* Events */  
abstract class Transaction {  
    Account account;  
}  
class AtmWithdrawal  
    extends Transaction {  
    String location;  
    double amount;  
}  
class AtmDeposit  
    extends Transaction {  
    String location;  
    double amount;  
}
```

```
/* Snapshot */  
class AccountSummary {  
    double balance;  
}
```

# WHAT ABOUT MY DOMAIN?

```
/* Aggregate */  
class Patient {  
    String identifier;  
    String system;  
}
```

```
/* Events */  
abstract class PatientEvent {  
    Patient patient;  
}  
class MedicationPrescribed  
    extends PatientEvent {  
    String code;  
    int quantity;  
}  
class ProcedurePerformed  
    extends PatientEvent {  
    String code;  
}
```

```
/* Snapshot */  
class PatientSummary {  
    List<String> medications;  
    List<String> procedures;  
}
```

# COMPUTING SNAPSHOTS

$$S_N = f(S_0, E_1 \dots E_N)$$

$S_N$  Snapshot at Version N

$S_0$  empty snapshot

$E_1 \dots E_N$  events from position 1 through N

$f$  query

**BUT IT'S NOT THAT SIMPLE**

# INCREMENTAL COMPUTATION

This is what we would like to do

$$S_N = f(S_K, E_{K+1} \dots E_N)$$

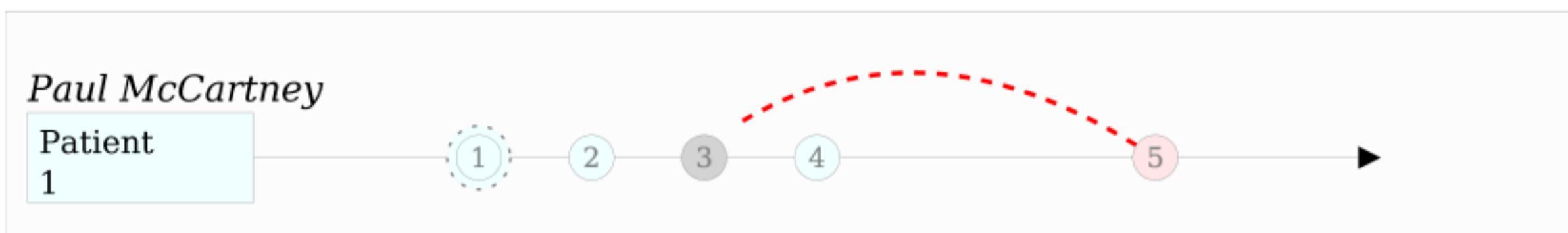
In Mathematics, it's called the distributive property

$$a + b + c = (a + b) + c$$

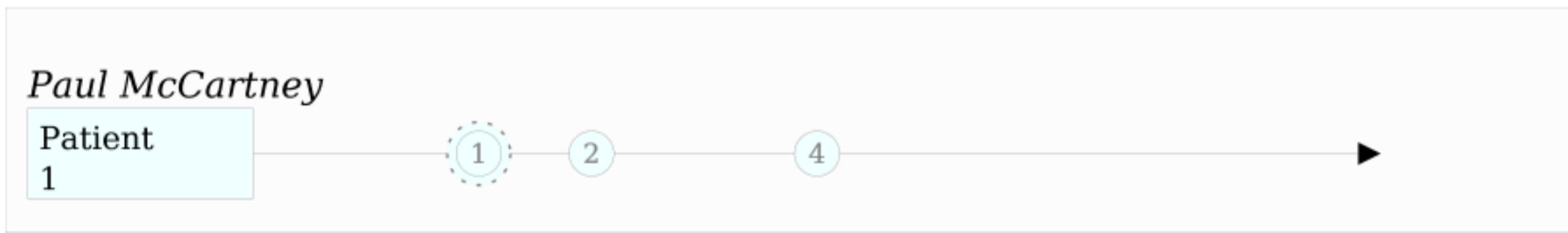
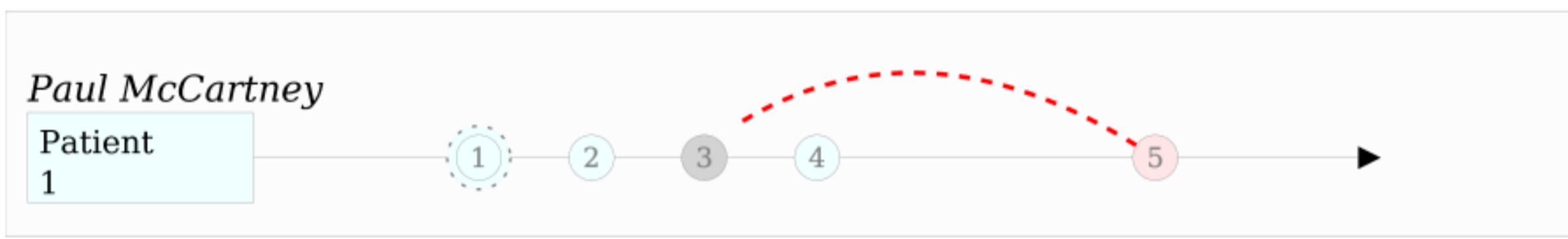
# WE'RE NOT DONE YET

There are special events

# REVERT



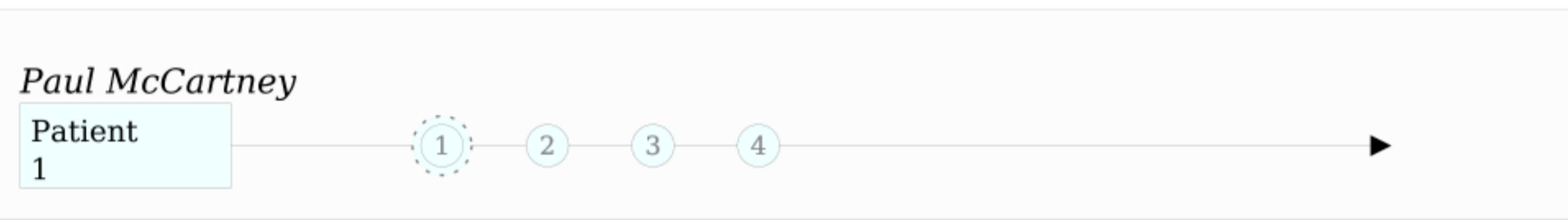
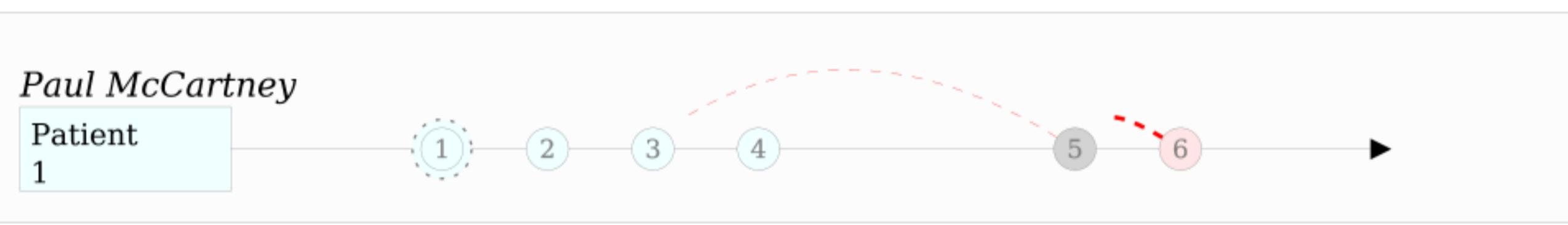
# REVERT



# REVERT AGAIN



# REVERT AGAIN



# MERGE/DEPRECATE



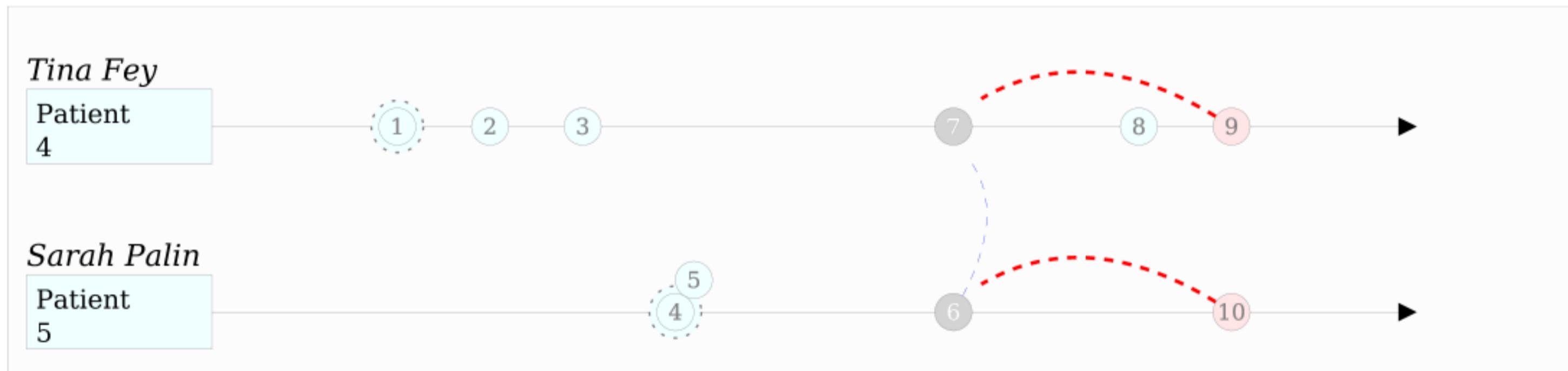
# MERGE/DEPRECATE



# REVERTING A MERGE



# REVERTING A MERGE



## BAD EXAMPLE 2

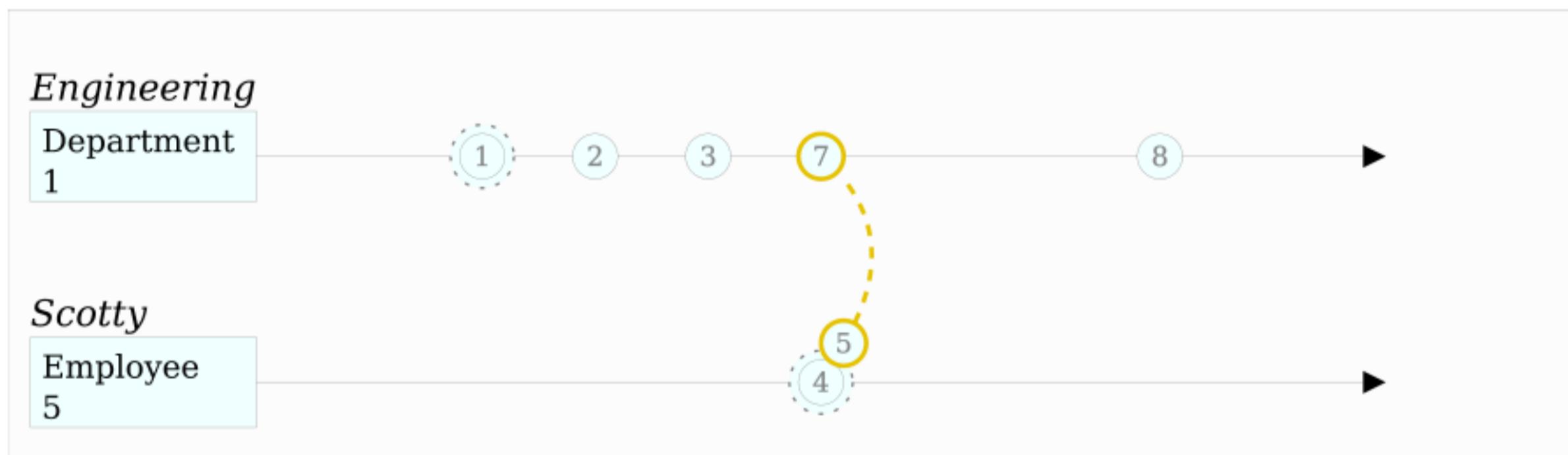
Employee

ID	EMP_NAME	DEPT_ID
1	Mr Spock	1
2	Scotty	2
3	Kirk	3
4	Janeway	3
5	La Forge	2

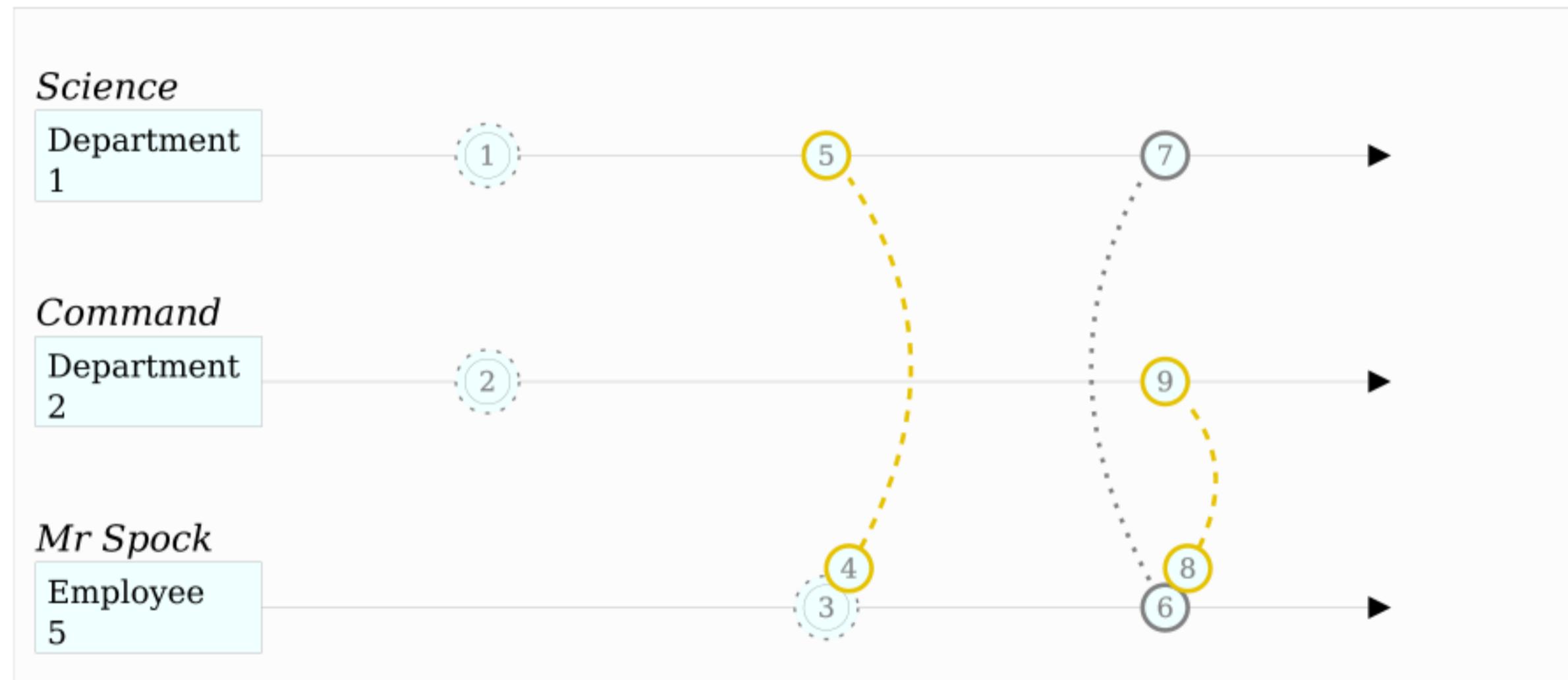
Department

ID	DEPT_NAME
1	Science
2	Engineering
3	Command

# JOINS IN EVENT SOURCING



# DISJOINS TOO



# **GROOVES DOMAIN OBJECTS**

# AGGREGATES

```
public class Patient implements AggregateType<Long> {  
    private Long id;  
    private String uniqueId;  
}
```

# EVENTS

```
public abstract class PatientEvent implements
    BaseEvent<Long, Patient, Long, PatientEvent> {
    private Patient aggregate;
    private Long id;
    private String createdBy;
    private RevertEvent<Long, Patient, Long, PatientEvent> revertedBy;
    private Date timestamp;
    private Long position;

    @Override
    @NotNull
    public Observable<Patient> getAggregateObservable() {
        return aggregate != null ? just(aggregate) : empty();
    }
}
```

# REAL EVENTS

```
public class PatientCreated extends PatientEvent {  
    private String name;  
}
```

# SPECIAL EVENTS

```
public class PatientEventReverted
    extends PatientEvent
    implements RevertEvent<Long, Patient, Long, PatientEvent> {
    private Long revertedEventId;
}
```

# SNAPSHOTS

```
public class PatientAccount
    implements JavaSnapshot<Long, Patient, Long, Long, PatientEvent>, Serializable {
    private Long id;
    private Patient aggregate;
    private Patient deprecatedBy;
    private List<Patient> deprecates = new ArrayList<>();
    private Long lastEventPosition;
    private Date lastEventTimestamp;

    11   private String name;
    12   private BigDecimal balance = new BigDecimal(0);
    13   private BigDecimal moneyMade = new BigDecimal(0);

    public Observable<Patient> getAggregateObservable() {
        return just(aggregate);
```

# GROOVES QUERIES

# SATISFYING GROOVES

```
public class PatientAccountQuery<...> extends QuerySupport<...> {  
    ...  
}
```

Could also be `VersionedQuerySupport` or  
`TemporalQuerySupport`

# FETCHING SNAPSHOTS

```
@Override  
public PatientAccount createEmptySnapshot() {  
    return new PatientAccount();  
}  
  
@Override  
public Observable<PatientAccount> getSnapshot(  
    long maxPosition, Patient aggregate) {  
    ...  
}  
  
@Override  
public Observable<PatientAccount> getSnapshot(  
    Date maxTimestamp, Patient aggregate) {  
    ...  
}
```

# FETCHING EVENTS

```
@Override  
public Observable<PatientEvent> getUncomputedEvents(  
    Patient aggregate, PatientAccount lastSnapshot, long version) {  
  
}  
  
@Override  
public Observable<PatientEvent> getUncomputedEvents(  
    Patient aggregate, PatientAccount lastSnapshot, Date snapshotTim  
  
}
```

# HANDLING ERRORS

```
@Override  
default Observable<EventApplyOutcome> onException(  
    Exception e, PatientAccount snapshot, PatientEvent event) {  
    getLog().error("Error computing snapshot", e);  
    return just(CONTINUE);  
}
```

# PROCESSING EVENTS

For languages like Java, Groovy

```
public Observable<EventApplyOutcome> applyPatientCreated(  
    PatientCreated event, PatientAccount snapshot) {  
    if (snapshot.getAggregate() == event.getAggregate()) {  
        snapshot.setName(event.getName());  
    }  
    return just(CONTINUE);  
}
```

...

# PROCESSING EVENTS

For languages with case classes

```
override fun applyEvent(  
    event: PatientEvent.Applicable, snapshot: PatientAccount) =  
    when (event) {  
        is PatientEvent.Applicable.Created -> {  
            // Your logic here  
            just(CONTINUE)  
        }  
        is PatientEvent.Applicable.ProcedurePerformed -> {  
            // Your logic here  
            just(CONTINUE)  
        }  
        is PatientEvent.Applicable.PaymentMade -> {  
            // Your logic here  
            just(CONTINUE)  
        }  
    }
```

# EVENTS

For languages with case classes

```
sealed class PatientEvent : BaseEvent<..> {
    // Properties and methods from equivalent java class

    sealed class Applicable : PatientEvent() {
        data class Created(val name: String) : Applicable()
        data class ProcedurePerformed(
            val code: String, val cost: Double) : Applicable()
        ...
    }

    data class Reverted(override val revertedEventId: String) :
        PatientEvent(), RevertEvent<..>
}
```

# COMPLETENESS FOR JAVA

...and Groovy.

```
@Aggregate public class Patient {...}  
public abstract class PatientEvent {}  
  
@Event(Patient.class)  
public class ProcedurePerformed extends PatientEvent {}  
@Event(Patient.class)  
public class PaymentMade extends PatientEvent {}  
  
@Query(aggregate=Patient.class, snapshot=PatientAccount.class)  
public class PatientAccountQuery {  
    ...  
    Observable<EventApply0utcome> applyProcedurePerformed() {...}  
    Observable<EventApply0utcome> applyPaymentMade() {...}  
}
```

# GOTCHAS



How many Tour de France General Classification Tour  
victories did this guy have on 2007-10-01?

# LANCE ARMSTRONG

1992	Joins Motorola	2002	
1993	DNF	2003	
1994	DNF	2004	
1995	36	2005	
1996	DNF	2005	Retires
1996-10-02	Diagnosed with Cancer	2009	Returns from retirement
1997-02	Declared cancer free	2009	3
1998	Joins US Postal	2010	23
1999		2010	Retires
2000		2012-10	Stripped of all wins from
2001			1998 through 2010

A professional cyclist, wearing a bright yellow jersey with the 'United States Postal Service' logo, is captured in a moment of triumph. He is wearing a white and blue helmet and sunglasses, and has his arms raised high in the air. The background is blurred, showing other cyclists and spectators, suggesting a race finish line.

1. Based on what  
we knew in 2007,  
How many TdF GC  
Tour victories did  
he have on 2007-10-  
01?

2. Based on what  
we know in 2017,  
How many TdF GC  
Tour victories did  
he have on 2007-10-  
01?



1. Based on what we knew in 2007,  
How many TdF GC  
Tour victories did he have on 2007-10-01?

2. Based on what we know in 2017,  
How many TdF GC  
Tour victories did he have on 2007-10-01?

Asking the right question is key

# ADVICE ON HANDLING DATA

- Be very careful about what your events look like.
- Don't worry much about what your snapshots look like.

# COMMUNICATION IS KEY



**Scott Bellware**  
@sbellware

 Follow

95% of the cost of event sourcing projects is explaining event sourcing

2:06 AM - Jun 11, 2017

10

142

294

i

# **NON EVENT SOURCES OF DATA**

When the intent of the user is not clear from the datasource.



# THANK YOU

Slidedeck [rahulsom/javaone-2017-eventsourcing](https://github.com/rahulsom/javaone-2017-eventsourcing)

Grooves [rahulsom/grooves](https://github.com/rahulsom/grooves)

Sun 15:00 **CON7610** Microservices Data Patterns: CQRS and ES

Mon 16:30 **CON2526** Reactive Stream Processing with Swarm and Kafka

Mon 17:30 **CON7474** ES, Distributed Systems, and CQRS with Java EE

Tue 13:15 **CON4083** Async by Default, Synchronous When Necessary

Wed 14:45 **CON4277** Three µS Patterns to Tear Down Your Monoliths