Java Actoren

Contents

[Akka 1](#_Toc68695404)

[Wichtige Momente 1](#_Toc68695405)

[Aktorsystem estellen 1](#_Toc68695406)

[Aktor Class erstllen 1](#_Toc68695407)

[Aktor ohne Parameter erstellen 2](#_Toc68695408)

[Aktor mit Parameter erstellen 2](#_Toc68695409)

[Child Actor mit context() erstellen 2](#_Toc68695410)

[Aktor createReceive() example 2](#_Toc68695411)

[match() 3](#_Toc68695412)

[matchAny() 3](#_Toc68695413)

[unhandled() 3](#_Toc68695414)

[tell() 3](#_Toc68695415)

[sender() 4](#_Toc68695416)

[self() 4](#_Toc68695417)

[Alle Objekte sind Aktoren 4](#_Toc68695418)

[Folien 4](#_Toc68695419)

[Aufgaben 9](#_Toc68695420)

[SS17 9](#_Toc68695421)

[Zyklische Verteilung von Queries 10](#_Toc68695422)

# 

# Akka

## Wichtige Momente

### Aktorsystem estellen

ActorSystem actorSystem = ActorSystem.create("TeaseLisa");

### Aktor Class erstllen

extends **AbstractActor**, implementiere methode **public Receive createReceive()**

public class PingPong extends AbstractActor {  
 ActorRef partner;  
 String name;  
  
 public PingPong(ActorRef sendTo, String name) {  
 this.partner = sendTo;  
 this.name = name;  
 }

@Override  
public Receive createReceive() {  
 return receiveBuilder()  
 .match(Integer.class, message -> beat(message))  
 .build();  
}

### Aktor ohne Parameter erstellen

ActorRef lisa = actorSystem.actorOf(Props.create(Kid.class));

### Aktor mit Parameter erstellen

ActorRef philosoher = actorSystem.actorOf(  
 Props.create(Philosopher.class, table, name));

Struktur: Props.create(class, args);

### Child Actor mit context() erstellen

Man verwendet getContext().getSystem(), um actorSystem zu finden

public class Kid extends AbstractActor {  
 private int cursed = 0;  
 private ActorRef mommy;  
  
 public Kid() {  
 mommy = getContext().getSystem()

.actorOf(Props.create(Mommy.class));  
 }

### Aktor createReceive() example

Falls message ein String ist und message==“printHello“ ist, wird „Hello World!“ gedruckt

Otherwise unhandled(message)



### match()

Nur dann, wenn class passt und bedingung erfüllt (optional)

match(class, bedingung, aktion);

match(class, aktion);

### matchAny()

Beliebige message.

mathcAny(aktion);

### unhandled()

Mache nichts, default case

### tell()

sender sends a message to actor:

actor.tell(message, sender);

ping.tell(0, ActorRef.noSender());

message ist 0, sender() in ping gibt Null-Sender zurück

ping.tell(“abc”, pong);

message ist „abc“, sender() in pong gibt ActorRef von pong zurück

ping.tell(“abc”, self());

getSender() gibt ActorRef von current object.

### sender()

Mit sender kann man ActorRef auf Sender vom Message bekommen, falls es beim tell() gesetzt wurde, ActorRef.noSender, falls es nicht gesetzt wurde.

if (partner != null) {  
 partner.tell(message, self());  
} else {  
 sender().tell(message, self());  
}

### self()

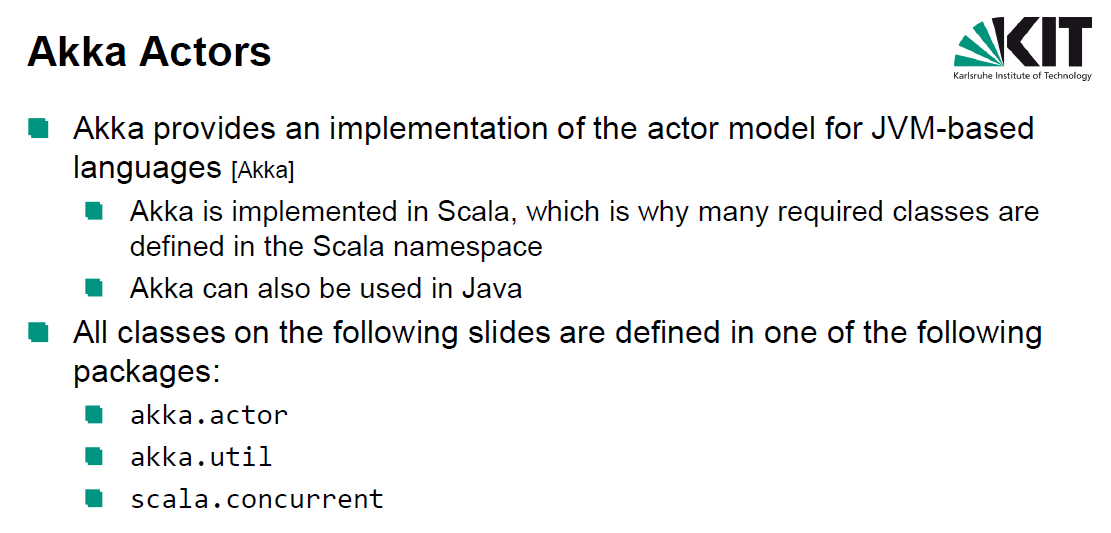
“”returns ActorRef of **current Actor**””

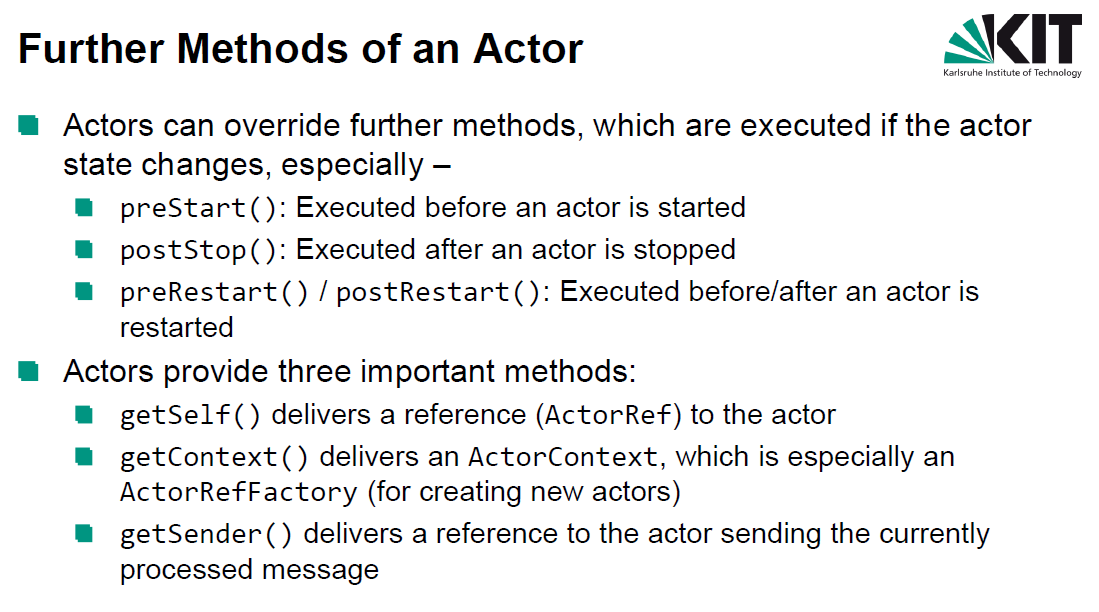
Kann man für tell verwenden

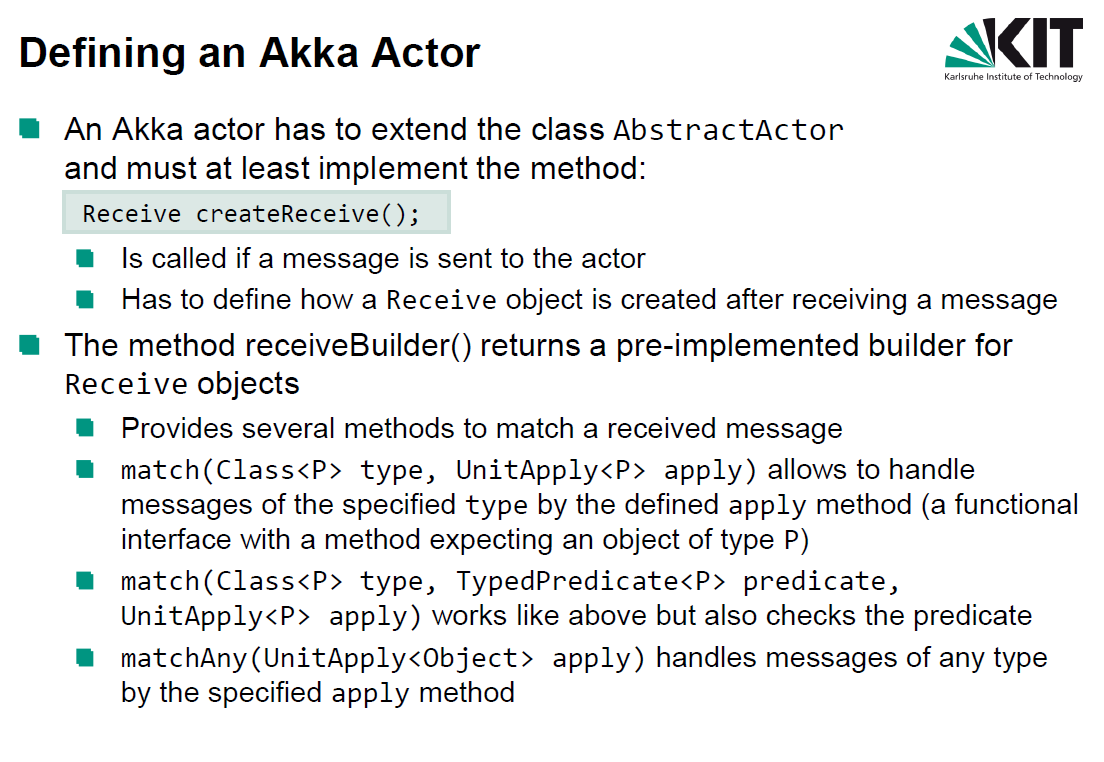
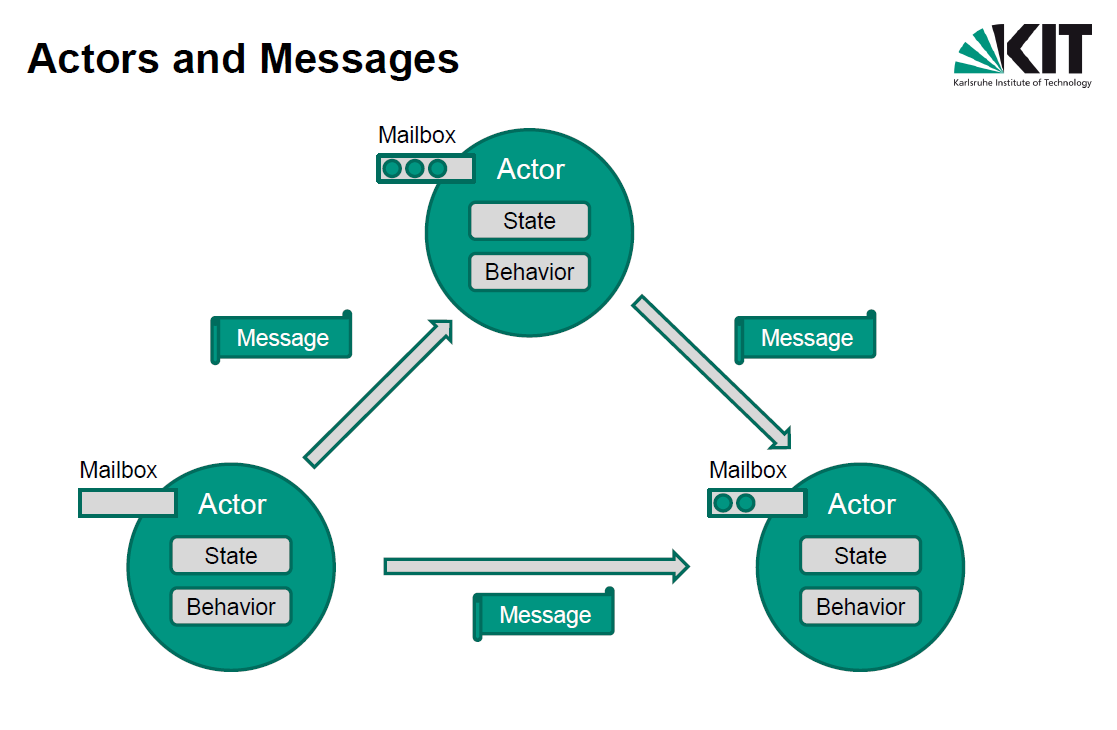
### Alle Objekte sind Aktoren

Speischende Philosopher: Alle Philosopher sind Aktoren, und Tisch ist auch ein Aktor

## Folien







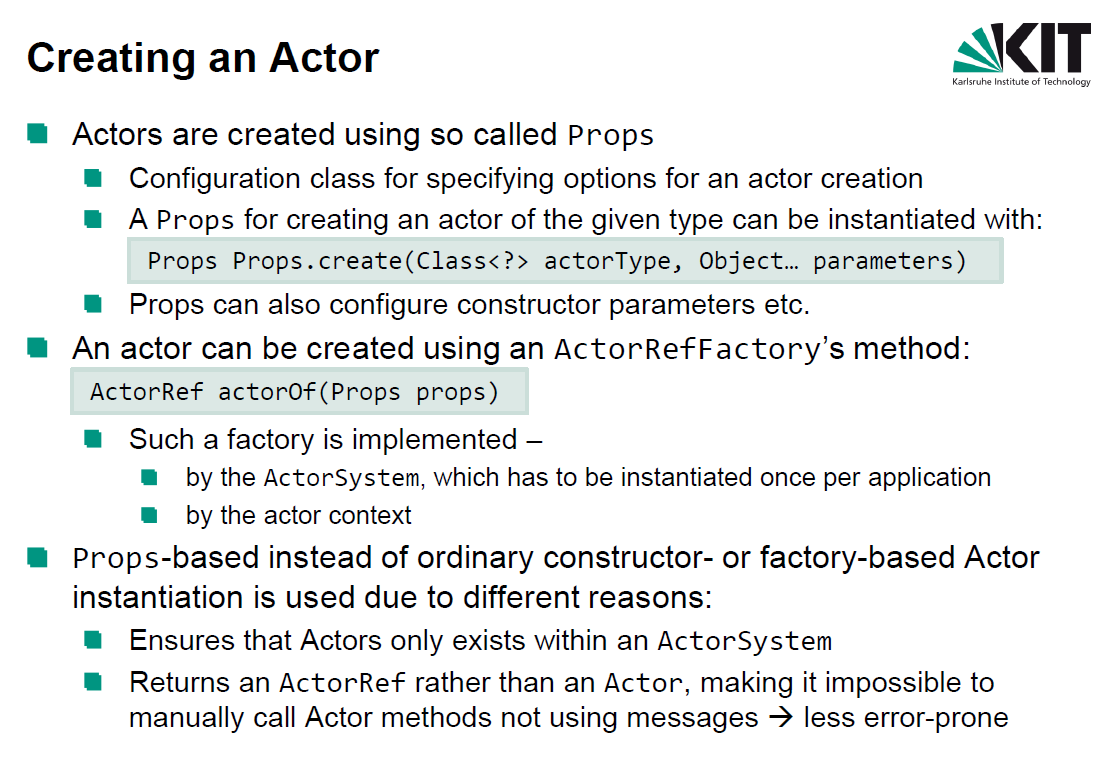
Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

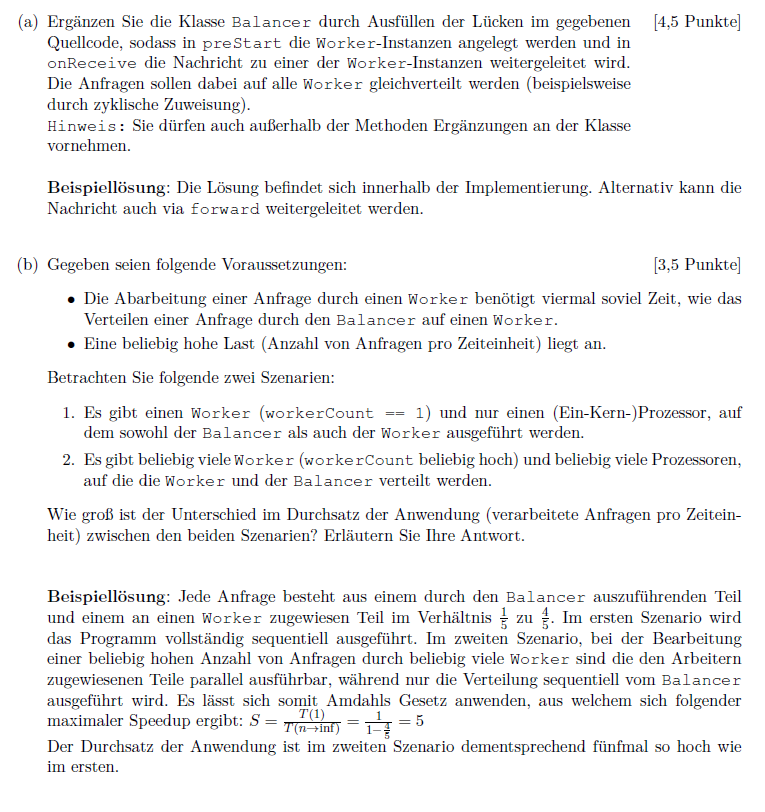
Description automatically generated

# Aufgaben

## SS17

Text

Description automatically generated



### Zyklische Verteilung von Queries

1) int counter Field in Balancer definieren

2) onReceive:

ActorRef current = this.worker.get(counter % workerCount);

current.tell(query, self());

counter++;