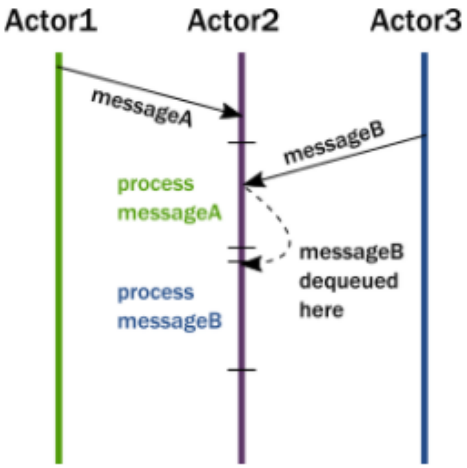
Actors

> Actor .
>
> Actor / / .



,

.

: Actor

import akka.actor.ActorSystem

```
ActorSystem system = ActorSystem.create("helloakka");
```

,

.

import akka.actor.AbstractActor

```
public class MyActor extends AbstractActor {
  private final LoggingAdapter log = Logging.getLogger(getContext().getSystem(), this);

  @Override
  public Receive createReceive() {
    return receiveBuilder()
      .match(String.class, s -> {
        log.info("Received String message: {}", s);
      })
      .matchAny(o -> log.info("received unknown message"))
      .build();
  }
}
```

.

AbstaractActor .

UntypedActor If/Switch .

import akka.actor.UntypedActor

```
public class TestActor extends UntypedActor {
    private final LoggingAdapter log = Logging
            .getLogger(getContext().system(), "TestActor");

    @Override
    public void onReceive(Object message) throws Exception {
            if(message instanceof String) {        //String   Java
                    log.info("Incommessage {}", message);
                    sender().tell("  ", ActorRef.noSender());
            }else {
                    log.info("Unhandle Message {}", message);
            }

    }
}
```

? , Java/C#

Scala . Java

8 . .

import akka.actor.Props

```
Props props1 = Props.create(MyActor.class);
Props props2 = Props.create(ActorWithArgs.class,
  () -> new ActorWithArgs("arg")); // careful, see below
Props props3 = Props.create(ActorWithArgs.class, "arg");
```

,

.

import akka.actor.ActorRef

```
ActorRef howdyGreeter = system.actorOf( props1  , "howdyGreeter");
howdyGreeter.tell(new WhoToGreet("Akka"), ActorRef.noSender());
```

, .

,

 .

    .