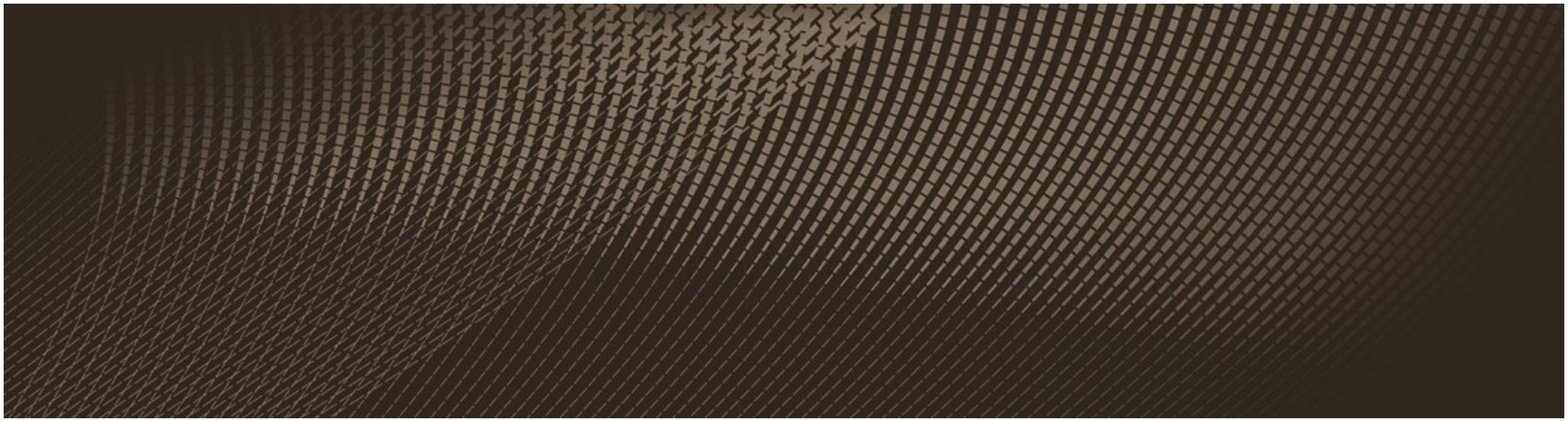


# Actors

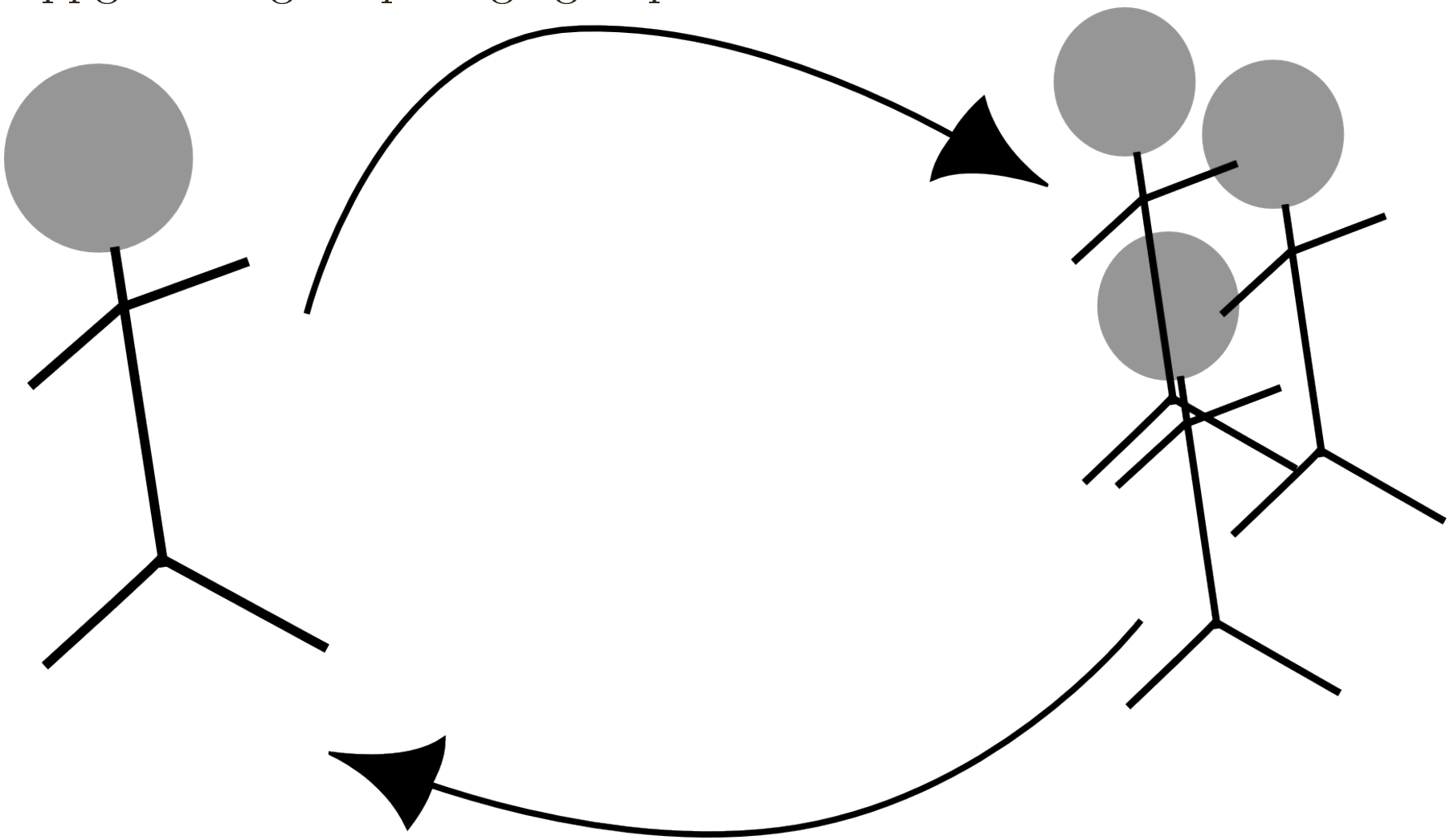
Scala -actors

17. november 2010

BEKK

# Scala quiz

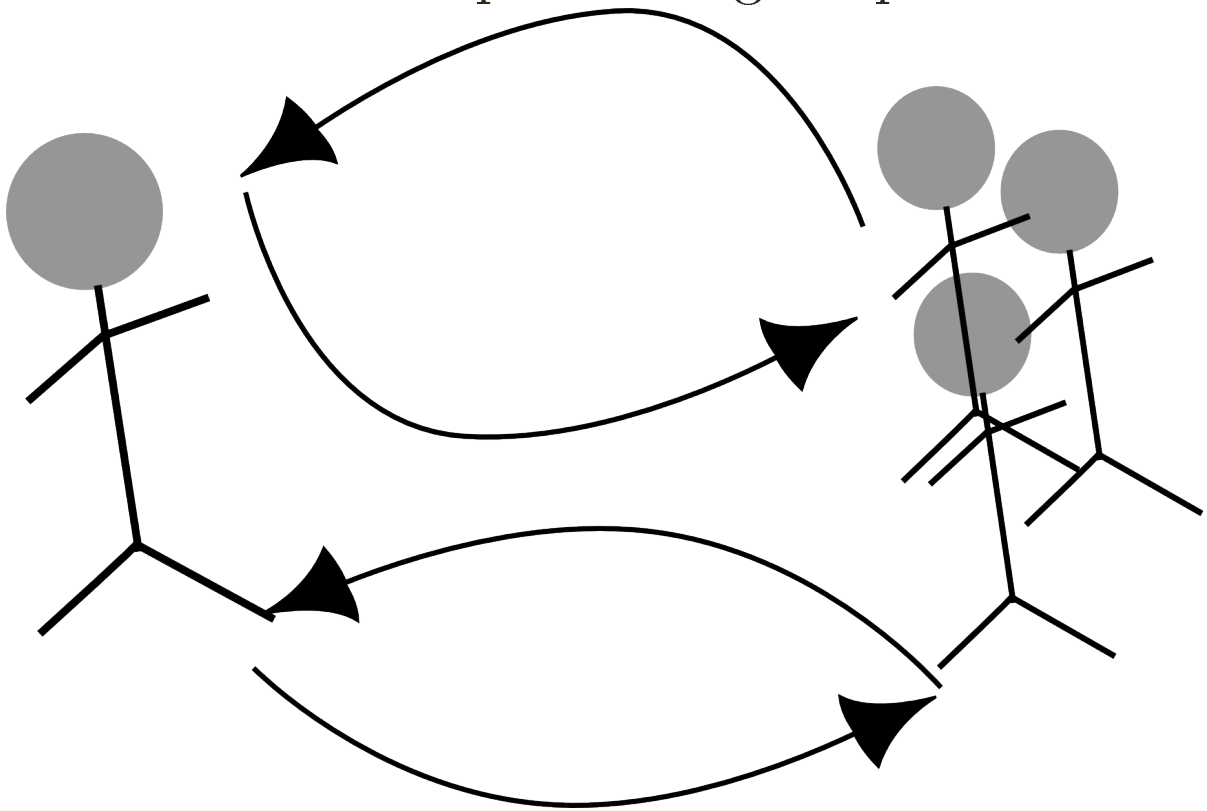
Oppgave: Lag et quizlag og implementer det som en actor



# Scala quiz

Quiz på vanlig måte, med en liten vri

1. Be om et spørsmål og få spørsmål



2. Svar på spørsmålet og få vite om det er riktig

# Actors - hva er greia

Wikipedia:

> In computer science, the Actor model is a mathematical model of concurrent computation that treats "actors"as the universal primitives of concurrent digital computation: in response to a message that it receives, an actor can make local decisions, create more actors, send more messages, and determine how to respond to the next message received. The Actor model originated in 1973. It has been used both as a framework for a theoretical understanding of concurrency, and as the theoretical basis for several practical implementations of concurrent systems.

# Actors - hva er greia

Wikipedia:

> ... concurrent computation ... in response to a message that it receives, an actor can make local decisions ... and determine how to respond to the next message received. The Actor model originated in 1973. ...

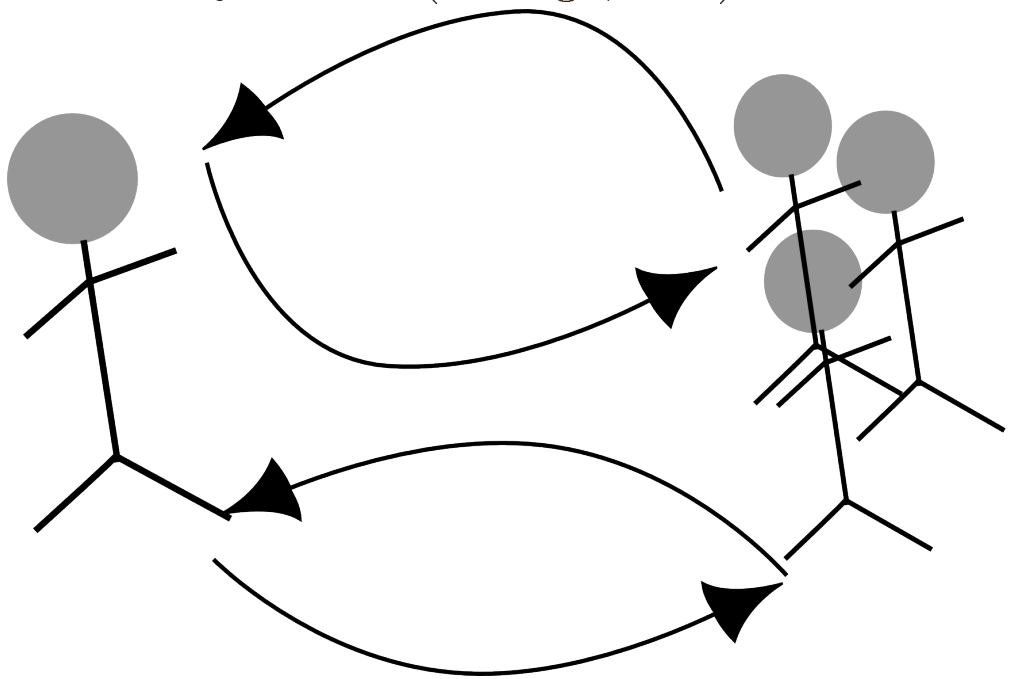
# Actors - eksempler eksempler

```
case class Tick

class Counter extends Actor {
  private var counter = 0

  def receive = {
    case Tick =>
      counter += 1
      println(counter)
  }
}
```

# Actors - in action

1. MoreQuestions(new Team("lag en"))
2. Question("Ping", Nil)

Server



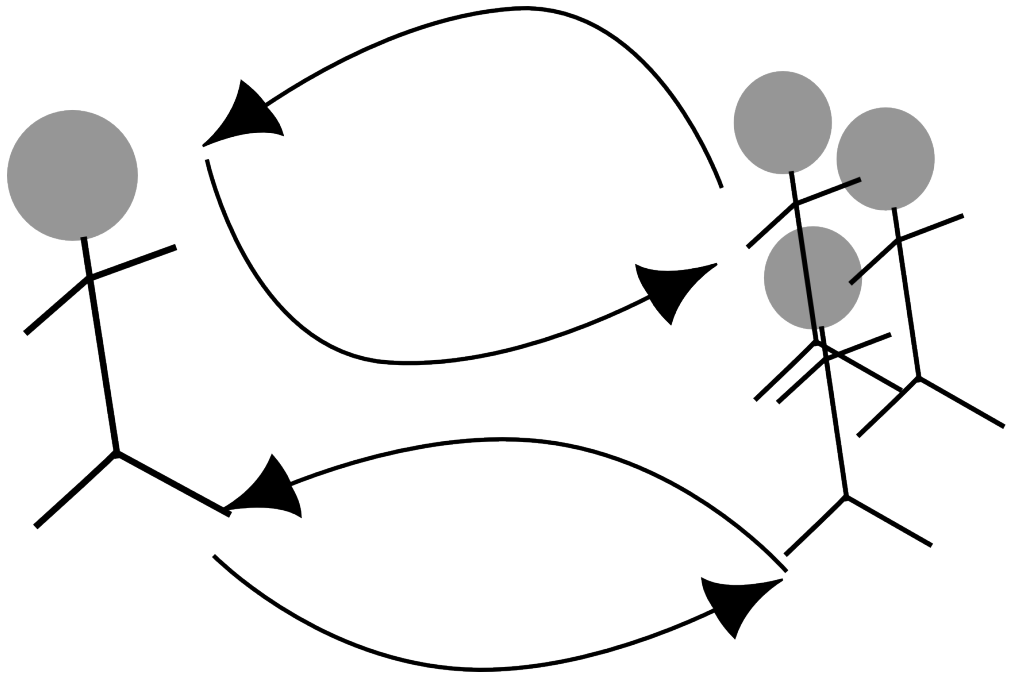
Lag

3. Answer(question, "pong")
4. Correct()

# Actors - in action

```
case class MoreQuestions(val team:Team)
case class Question(val question:String, val content:Any)
```

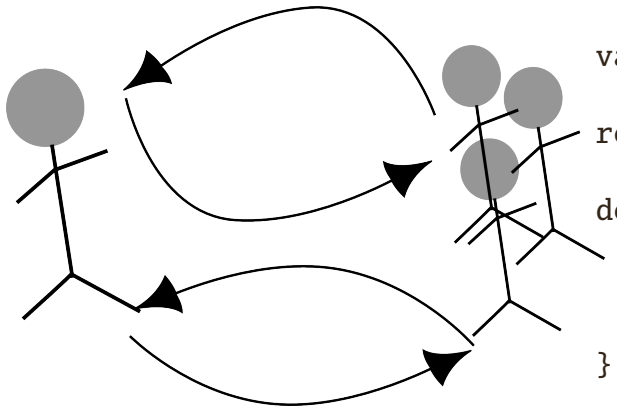
Server



Lag

```
case class Answer(val team:Team, val question:Question, val answer:Any)
case class Correct extends Verdict
```

# Actors - in action



```
val question = (remote !!
      MoreQuestions(team)).get.asInstanceOf[Question]
remote !! new Answer( team, question, giveAnswer(question) )

def giveAnswer(quiestion: Question)={
 question match {
   case x@Question("Ping", "") => "pong"
  }
}
```

# Oppgave

# Actors

## Actors - hva er det

naturlig programerings model
enkelt å paralellisere
gode biblioteker
enkelt å gjøre riktig

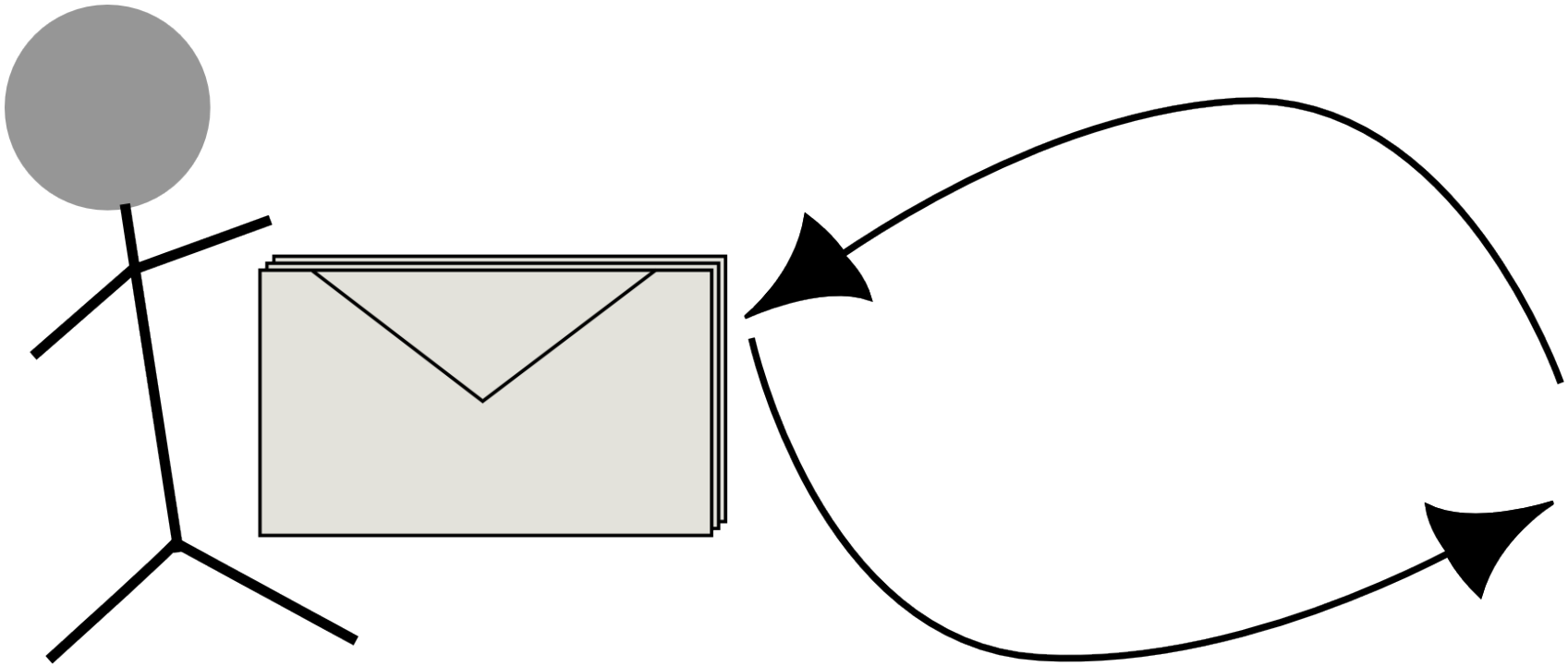
} Actors

# Hvordan?

## Actors - hva er det

Actors
- individer er normalt
- ingen deling
- akka actors erlang
- objekter

# Actors - helt enkelt



Ingen deling - meldingsboks - eksterne

# Actors - Akka

```
import se.scalablesolutions.akka.actor._
import se.scalablesolutions.akka.remote._

class Client extends GiveAnswer
{
  val team = new Team("test")
  val remote = RemoteClient.actorFor("Server", "localhost", 9999)

  def run {

    while(true)
    {
      val question = (remote !! MoreChallenges(team)).get.asInstanceOf[Question]
      remote !! new Answer(team, question, giveAnswer(question) )
    }
  }

}
```

# Videre - oppgave

# Oppgave - meldinger

```
package no.bekk.scala.messages
trait Message
trait Verdict

case class MoreChallenges(val team:Team) extends Message
case class Question(val question:String, val content:Any) extends Message

case class Answer(val temaName:Team, val question:Question, val answer: Any) extends Message

case class Correct() extends Message with Verdict
case class Wrong() extends Message with Verdict
```

# Oppgave - meldinger

```
package no.bekk.scala.messages
trait Message
trait Verdict

case class MoreChallenges(val team:Team) extends Message
case class Question(val question:String, val content:Any) extends Message

case class Answer(val temaName:Team, val question:Question, val answer: Any) extends Message

case class Correct() extends Message with Verdict
case class Wrong() extends Message with Verdict
```

Ønsker å;

- ta i mot et spørsmål Question,
- finne ut hva det handler om, og
- besvare

# Pattern matching

# Pattern matching - enkelt

```
newMessage match {
  case Tick => counter += 1
  case Tack(i:Int) => counter += i
  case message@Clear => counter = 0
  case "add a hundred" => counter += 100
  case 2 =>counter += 2
  case _ => // Do nothing
}
```

# Pattern matching - enkelt

Type pattern

```
case Tick => counter += 1
```

Matcher på typen, Tick

# Pattern matching - enkelt

Constructor pattern

```
case Tack(i:Int) => counter += i
```

Matcher på typen, Tick, med hvor den interne variablen er en Int

# Pattern matching - enkelt

Variable binding pattern

```
case message@Clear => counter = 0
case message:Clear => counter = 0
   //compile err.
```

Henter ut Clear objektet til message variablen.

# Pattern matching - enkelt

Constant pattern

```
case "add a hundred" => counter += 100
case 2 =>counter += 2
```

Matcher singletons som er lik seg selv.

# Pattern matching - enkelt

Wildcard pattern

```
case _ => // Do nothing
```

Matcher alt.

# Oppgaver

# Pattern matching - avansert

Matche på spørsmål
Pakke opp objecter, extractors

```
new Question("spørsmål", List("s", "v")) match {
  case x@Question(sp, svar:List[String]) => {
    println(sporsmal + ", " + svar)//warning!
  }
}
```

# Pattern matching - avansert

Matche på spørsmål
Pakke opp objecter, extractors

```
new Question("spørsmål", List("s", "v")) match {
  case x@Question(sp, svar@List(s:String, _*) =>{
   println(sporsmal + ", " + svar)
  }
}
```

# Pattern matching - avansert

Matche på spørsmål
Regexp

```
val Epost = "^(.*)@(.*)\.(.*)$".r
```

# Pattern matching - avansert

Matche på spørsmål
Regexp

```
val Epost = "^(.*)@(.*)\.(.*)$".r

"aslakjo@gmail.com" match {
  case Epost(user, domain, tld) => //spam!
}
```


BEKK

Aslak Johannessen
Senior Consultant
982 19 249
aslakjo@bekk.no

BEKK CONSULTING AS
SKUR 39, VIPPETANGEN. P.O. BOX 134 SENTRUM, 0102 OSLO, NORWAY. WWW.BEKK.NO