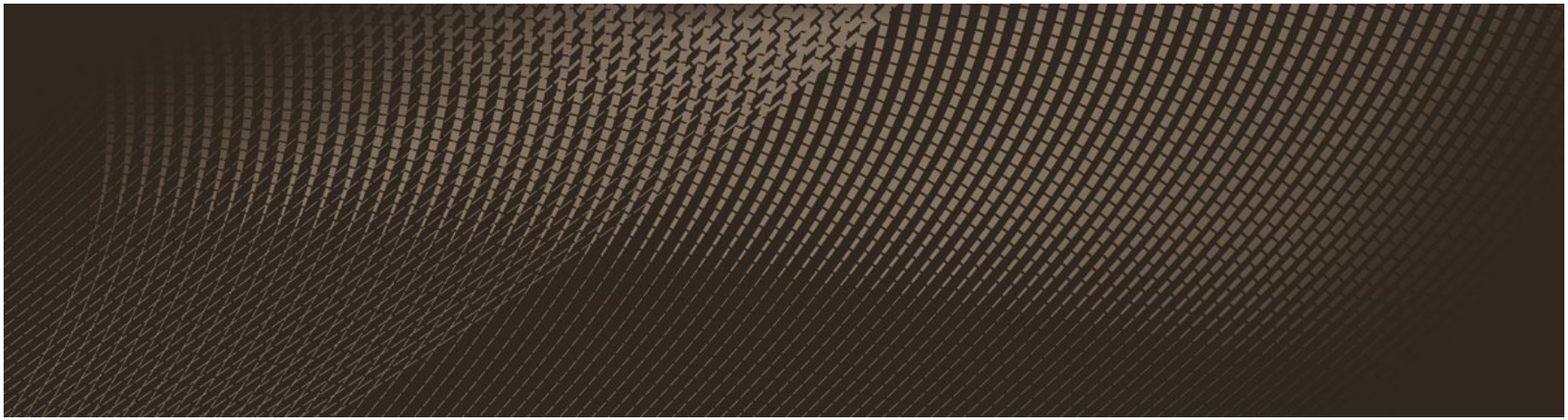


Fagkveld hos BEKK – Scala

Introduksjon til scala

28. mars 2011

Velkommen

- Hvorfor scala?
- Kurs
  - Litt foiler
  - Litt oppgaver
- Middag

Oss

Rune Eivind Ole Christian Torbjørn Kai Stein Kåre Aslak

Hvorfor scala?

Hvorfor scala?

Java er gammelt
Scala er et riktig steg videre
Bedre konsepter ≡ bedre kode

Gammelt

```
public class Person{
    private String navn;
    private String etternavn;
    private Integer alder;

    public Person(String fornavn, String etternavn, Integer alder){
        this.navn = fornavn;
        this.etternavn = etternavn;
        this.alder = alder;
    }

    public String fulltNavn(){
        return navn + " " + etternavn;
    }

    public String getNavn(){ return navn; }
    public void setNavn(String navn){ this.navn = navn; }

    public String getEtternavn(){ return etternavn; }
    public void setEtternavn(String etternavn){ this.etternavn = etternavn; }

    public Integer getAlder(){ return alder; }
    public void setAlder(Integer alder){ this.alder = alder; }
}
```

Nytt

```
class Person(
  val navn:String,
  val etternavn:String,
  val alder:Int)
{
  def fulltNavn ={
    "%s %s".format(navn, etternavn)
  }
}
```

Closure

Gammelt

```
public List<String> interesse(String starterMed){
  List<String> interesser = new ArrayList<String>();
  interesser.add("Gym");
  interesser.add("Sykkel");
  interesser.add("Scala");
  interesser.add("Vask");
  interesser.add("Mat");

  List<String> valgte = new ArrayList<String>();
  for(String interesse : interesser){
    if(interesse.startsWith(starterMed)){
      valgte.add(interesse);
    }
  }

  return valgte;
}
```

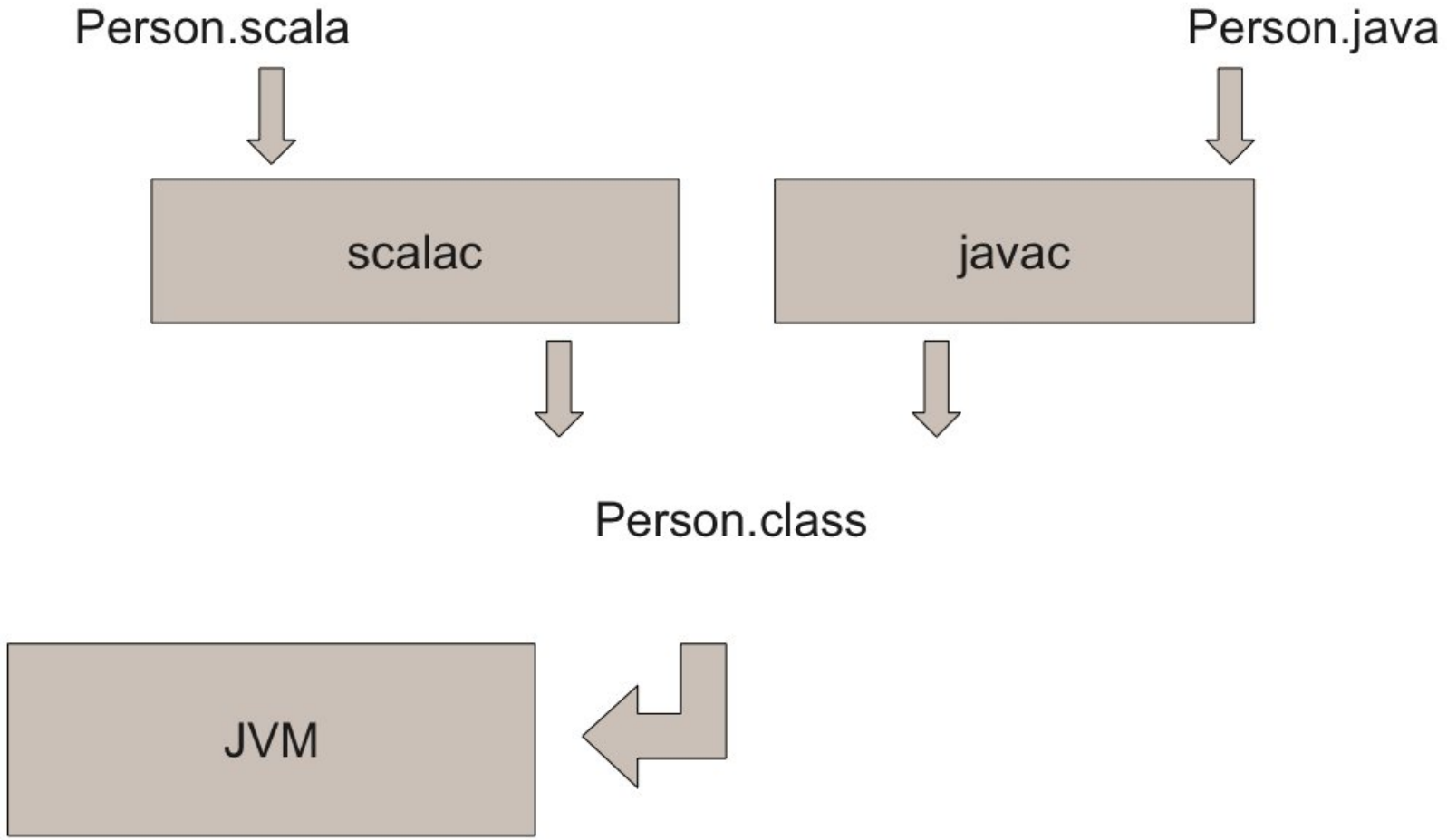
Nytt

```
val interesser = List(
   "Gym", "Sykkel", "Scala",
   "Vask", "Mat"
)

def interesse(starterMed:String) = interesser.filter(
  (s:String) => s.startsWith(starterMed)
)
```

Scala på JVM

Scala er egentlig bare java, plus en jar.



Helt grunnleggende – Klasser

```
class Person( //Konstruktørparametere og instansevariabler
   navn:String,
   etternavn:String,
   alder:Int)
{
  //Konstruktør
  val interesser = List("ting", "og", "tang")
  val cv = "%s %s -- %s\n Liker: %s".format(
    navn, etternavn, alder, interesser.mkString(", ")
  )


  def fulltNavn ={ //Funksjon
    "%s %s".format(navn, etternavn)
  }
}
```

Helt grunnleggende – val og var

```
 //Er slik for altid
  val interesser = List("ting", "og", "tang")

  def mineInteresser ={
    var interesser = "" //Kan tilordnes på nytt
    for(interesse <- interesser){
      interesser += interesse + ", "
    }

    interesser
  }
}
```

Helt grunnleggende – Oppgaver

Koden ligger her : https://github.com/aslakjo/scala_intro_kurs

- Last ned
- Gå til katalogen og kjør
  - sbt
- I sbt kjør
  - update
  - ~test-quick

Case classes

En case classe er en normal klasse med masse ekstrautstyr

- Genererer geters og seters
- Hash code og equals er implementert korrekt, og på bakgrunn av verdiene
- Det companion object er laget
- Unapply / extractors er laget

Bruk av case classes

```
case class Sykkel(val farge:String, val hjul:Int) //def

object CaseClassApp extends Application{
  //compainon og new
  assert(Sykkel("rød", 2).equals(new Sykkel("rød", 2)))
}

//unapply kommer ..
```

Patternmatching

En veldig veldig kraftig variant av switch statementet

- Gir muligheten til å velge hele eller deler av et object
- Mulig å matche på typer, verdier, arv, innhold i referanser

Bruk av pattern matching

```
case class Farge(val navn:String)

case class Bil(val farge:Farge, val hjul:Int)

object PatternMatcingApp extends Application{
  def godkjenntBil(dings: AnyRef)={
    dings match {
      //Constuctor pattarn, variabel pattern, med guard og extractor
      case Bil(_, hjul) if hjul <= 2 => println("En slik bil heter gjerne motorsykkel")
      //verdi pattern
      case Bil(_, 4) => println("helt vanelig bil")
      //sjekking i refererte objecter, variable pattern og guard
      case Bil(Farge(fargePaBil), _) if fargePaBil.equals("rød") => println("Jippi en rød bil!")
      //@ binder variabelen s til det som er på høyre side
      case s@Bil(farve, _) if farve != null => println("dette er en " + farve.navn + " bil")
      // type pattern
      case s:Bil => println("dette er en bil")
      //wildcard pattern, matcher alt
      case _ => println("dette kan være hva som helst uten om en bil")
    }
  }
}
```

Traits

Et trait spesifiser egenskap.

- Kan brukes om interface(abstract traits)
- En klasse kan få flere traits "mixet inn"
- Traits kan ikke opprettes på egenhånd

Bruk av traits

```
trait HealthCheckable{ //interface
  def isOk: Boolean
}
trait Logger { //egenskap
  def log(message: String):Unit = println(message)
}
trait LoggProcessing extends FooService{ //stackable trait
  def log(message:String):Unit

  override def process:Unit={ //ny oppførsel
    log("Starting processing")
    super.process
    log("Stopped processing")
  }
}

class FooService extends HealthCheckable with Logger{
  def isOk:Boolean = true

  def process = {
    //go allot!
  }
}

object Application{ new FooService with LoggProcessing } //mix inn ved opprettelse
```

BEKK

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