Case classes

En case classe er en normal klasse med masse ekstrautstyr

- 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

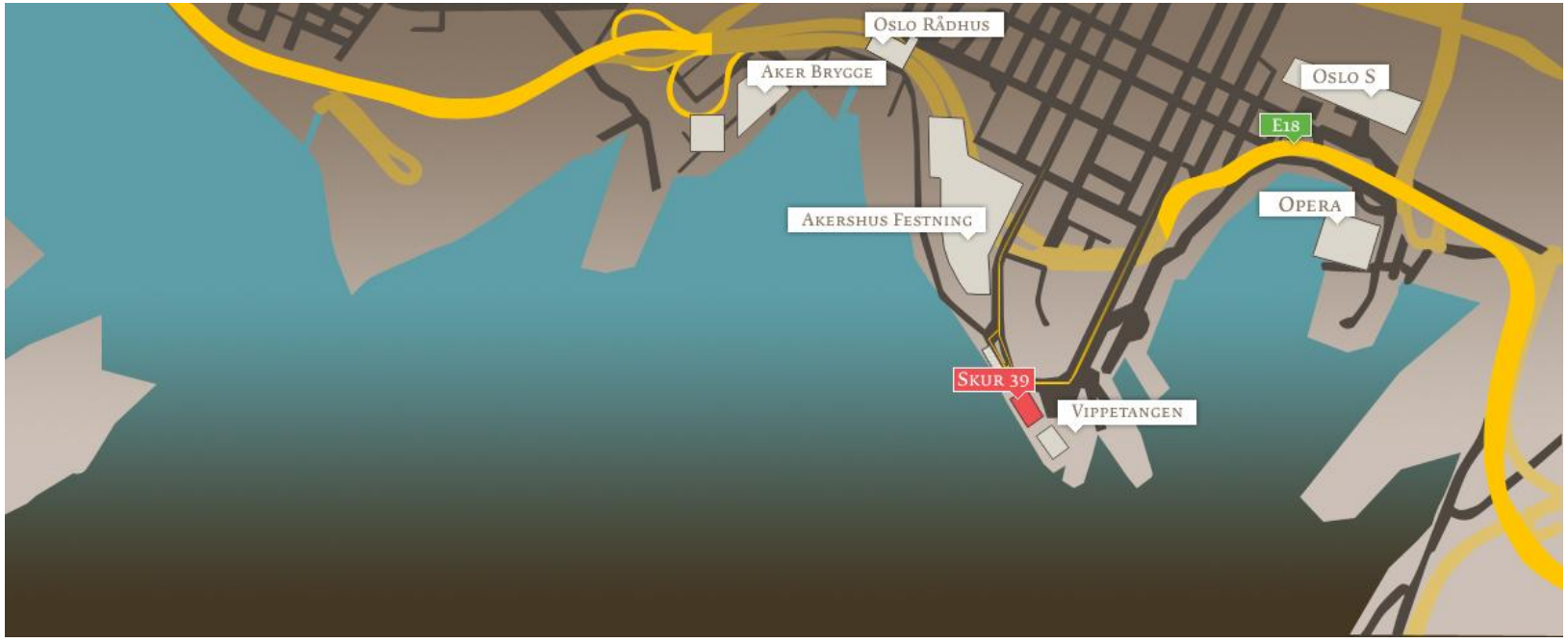
```
trait HealthCheckable{ //interface
  def isOk: Boolean
}
trait Logger { //egenskap
  def log(message: String):Unit = println(message)
}
trait LoggProcessing extends FooService{ //stackable trait
  def log(message:String):Unit

  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