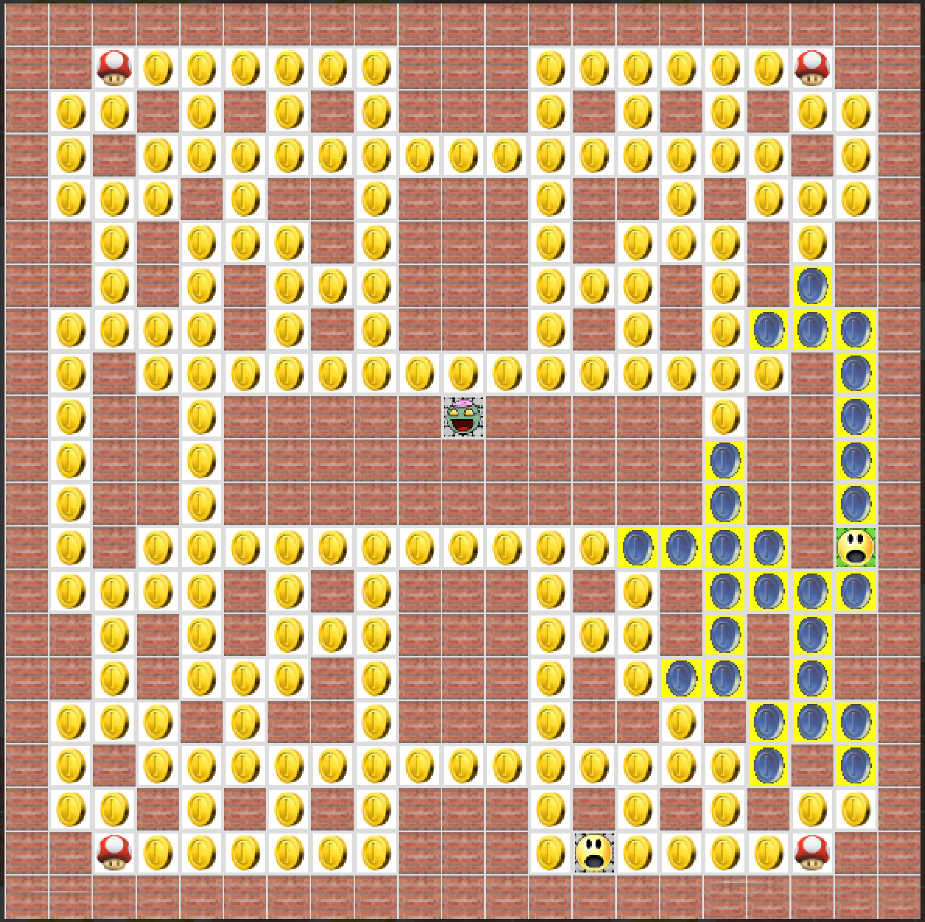
TacZombie

(a tactical Pacman clone)



Developed by Lars Eckervogt, Manuel Hieke and Stefan Junker for

the course "Moderne Programmiersprachen"

in MSI1 in WS13/14 at HTWG Konstanz

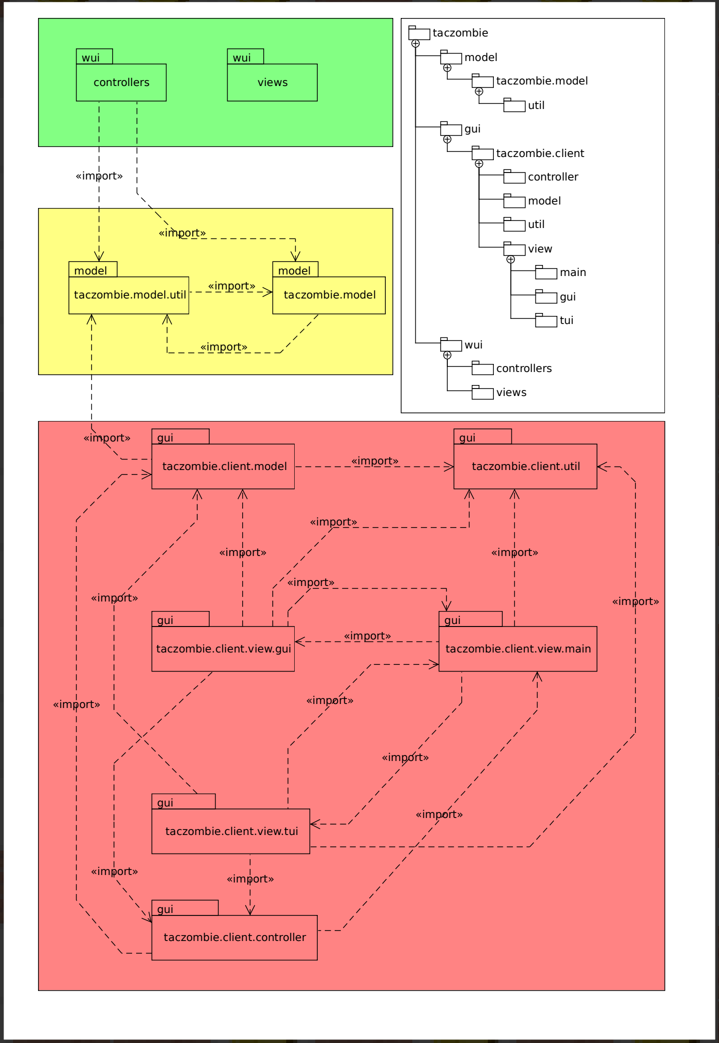
Content

Game Principle

- Round-based
- Two Players
  - **One** Human: can have multiple tokens
  - **One** Zombie: can have multiple tokens
- Goal
  - Zombie: kill Human
  - Human: survive and collect all coins
- Map
  - Pacman-like

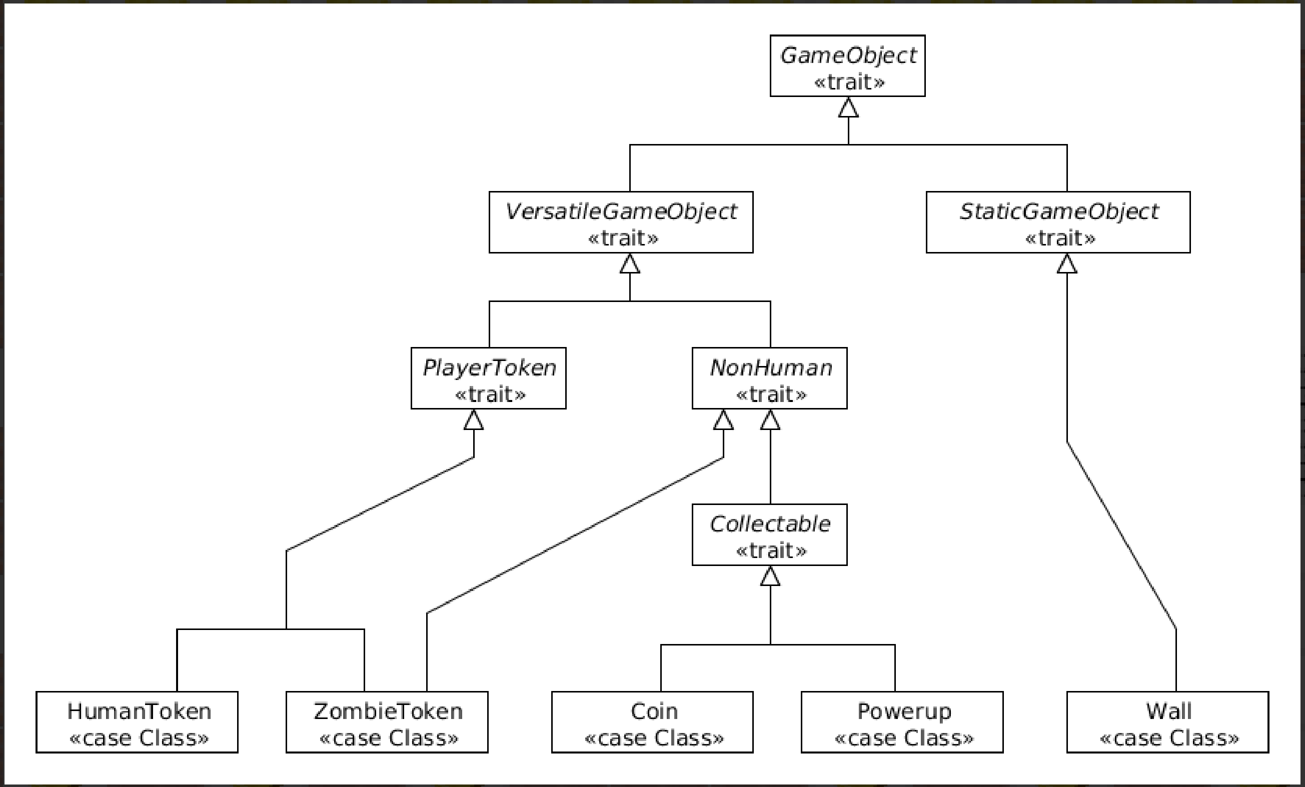
Structure

Click to maximize



Model

- Case classes
  - Easy access of subsets



Model

- Build to be extensible
  - Multiple Players (Zombie/Humans)

Views

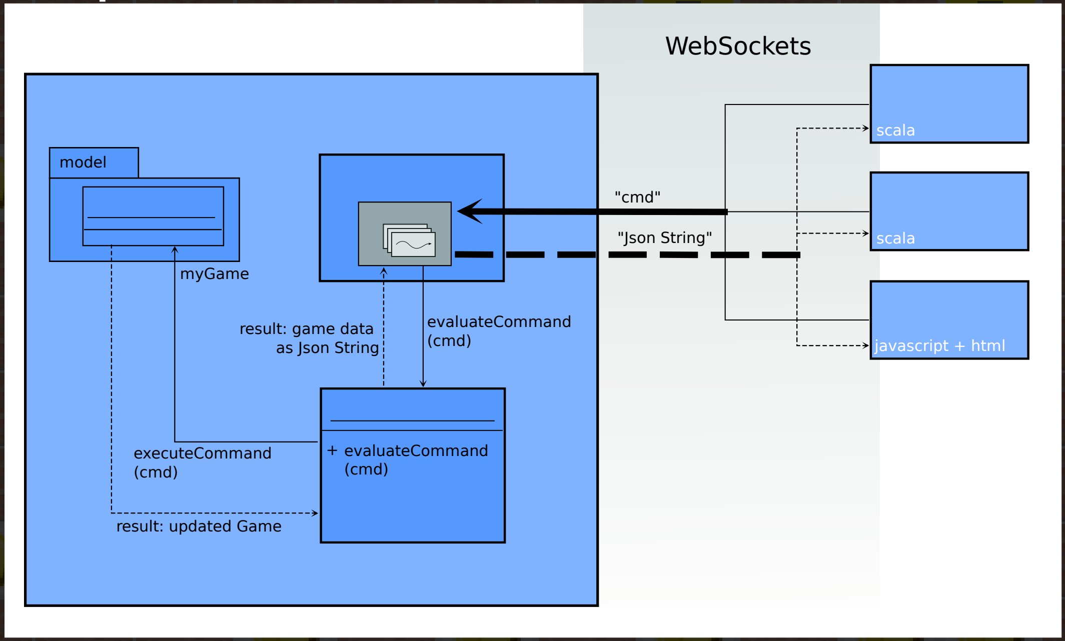
- WUI
- TUI
- GUI

- Communication through web sockets
  - Library: just.ws (https://github.com/stasimus/just.ws)
- Transport format: jSON
  - Library: io.spray (https://github.com/spray/spray-json)

Controller

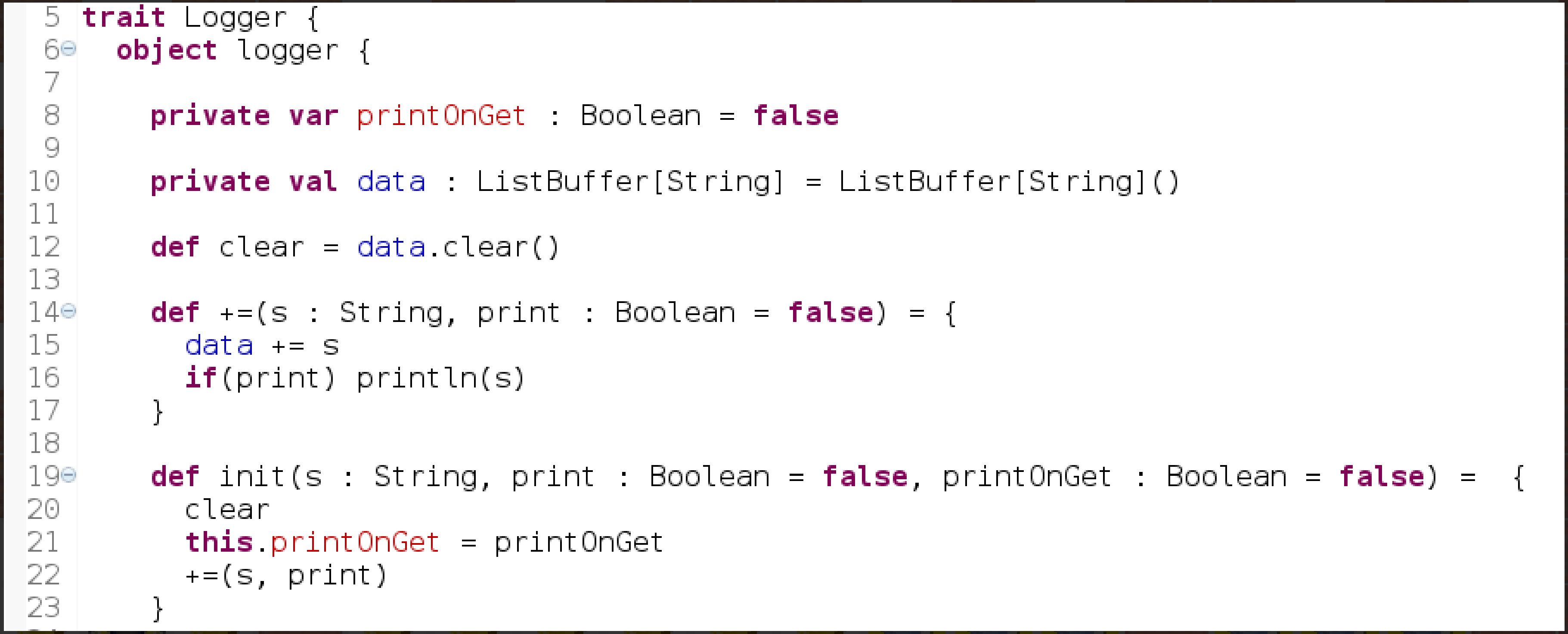
- Server-side: Play
  - Manages web socket connections
  - Manages the game
  - Handles incoming client requests
  - Servers game data to th
- Client-side:
  - GUI: Swing
  - WUI: JavaScript/Play

Click to maximize



Use of Traits

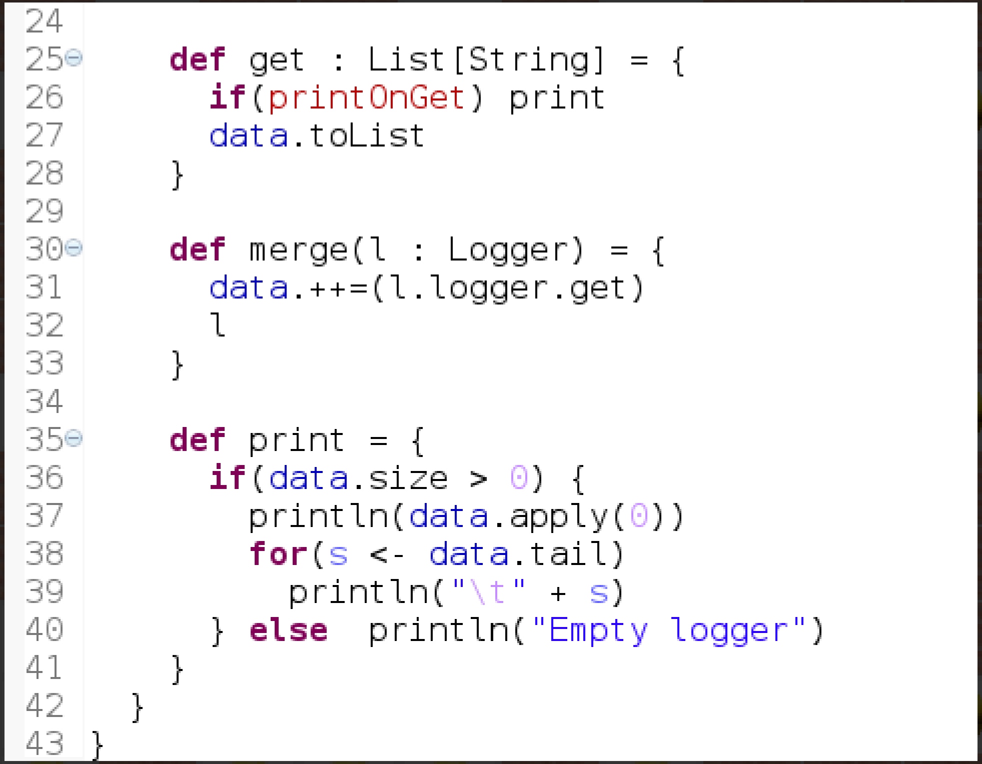
- Example: Logger



```scala
trait Logger {
  object logger {

    private var printOnGet : Boolean = false

    private val data : ListBuffer[String] = ListBuffer[String]()

    def clear = data.clear()

    def +=(s : String, print : Boolean = false) = {
      data += s
      if(print) println(s)
    }

    def init(s : String, print : Boolean = false, printOnGet : Boolean = false) =  {
      clear
      this.printOnGet = printOnGet
      +=(s, print)
    }
```

Use of Traits

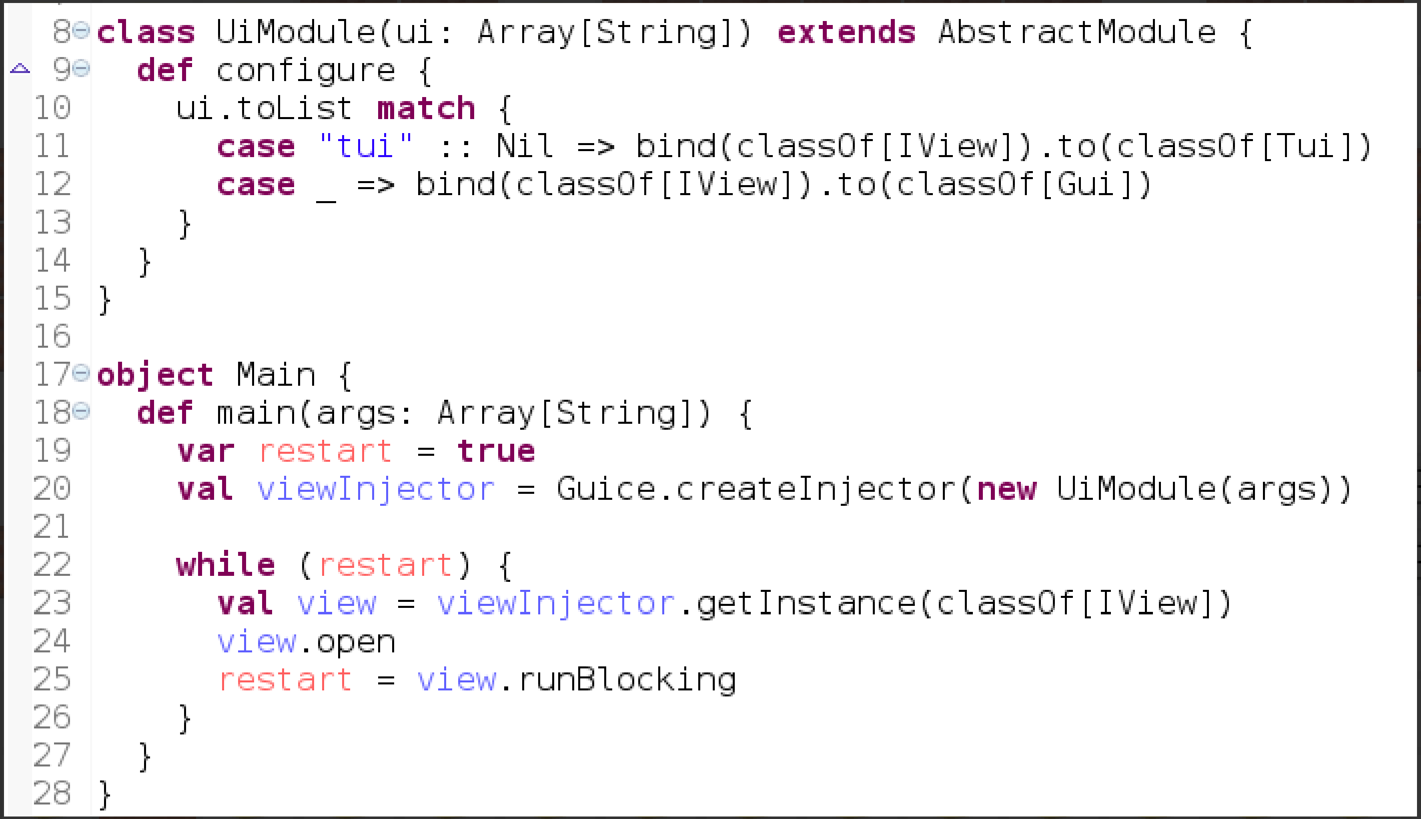
- Example: Logger



```
  def get : List[String] = {
   if(printOnGet) print
   data.toList
  }

  def merge(l : Logger) = {
   data.++=(l.logger.get)
   l
  }

  def print = {
   if(data.size > 0) {
    println(data.apply(0))
    for(s <- data.tail)
     println("\t" + s)
   } else  println("Empty logger")
  }
 }
}
```

Use of Dependency Injection

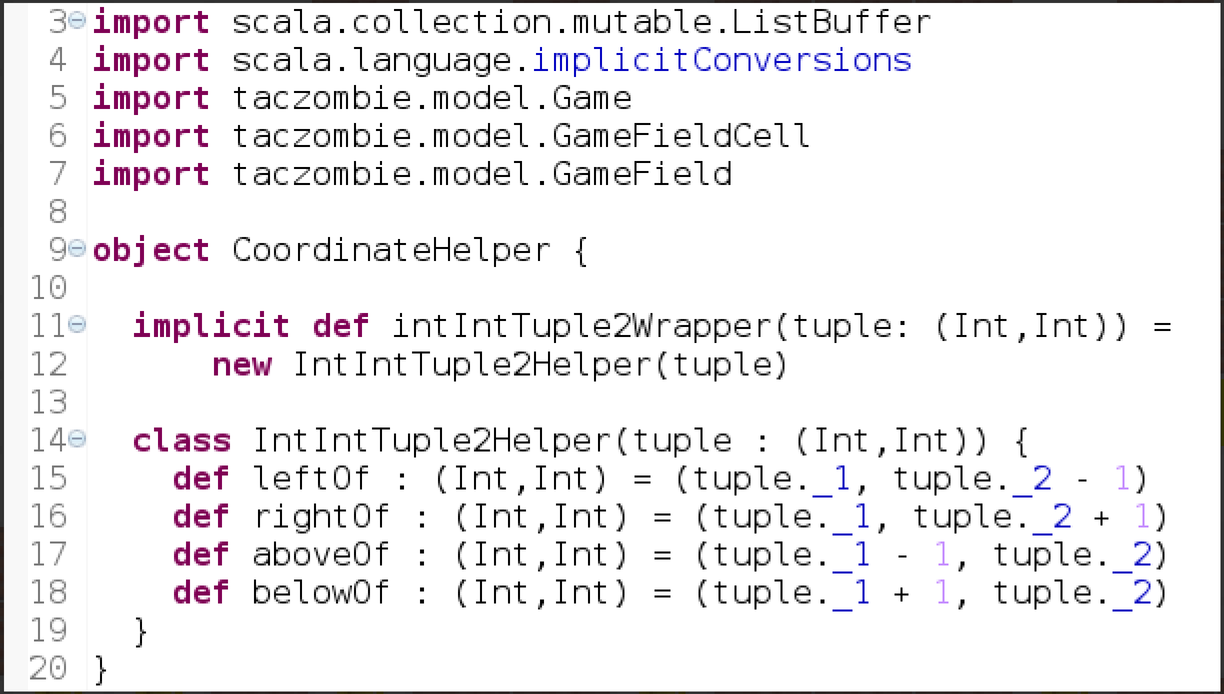
- Main Method GUI / TUI with scala-juice



```scala
class UiModule(ui: Array[String]) extends AbstractModule {
  def configure {
    ui.toList match {
      case "tui" :: Nil => bind(classOf[IView]).to(classOf[Tui])
      case _ => bind(classOf[IView]).to(classOf[Gui])
    }
  }
}

object Main {
  def main(args: Array[String]) {
    var restart = true
    val viewInjector = Guice.createInjector(new UiModule(args))

    while (restart) {
      val view = viewInjector.getInstance(classOf[IView])
      view.open
      restart = view.runBlocking
    }
  }
}
```

Use of Implicit Conversions

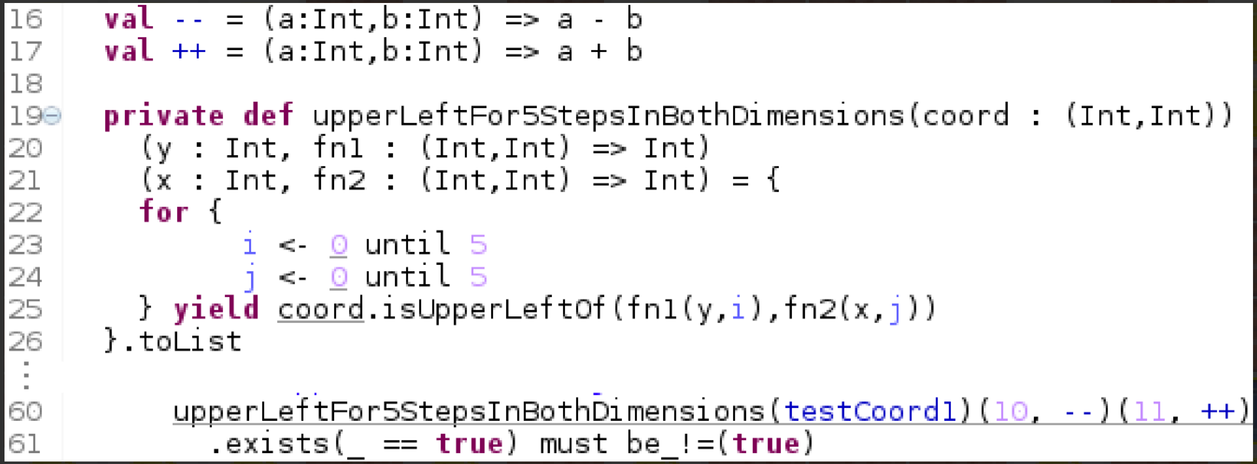
- Example: CoordinateHelper



```scala
import scala.collection.mutable.ListBuffer
import scala.language.implicitConversions
import taczombie.model.Game
import taczombie.model.GameFieldCell
import taczombie.model.GameField

object CoordinateHelper {

  implicit def intIntTuple2Wrapper(tuple: (Int,Int)) =
      new IntIntTuple2Helper(tuple)

  class IntIntTuple2Helper(tuple : (Int,Int)) {
    def leftOf : (Int,Int) = (tuple._1, tuple._2 - 1)
    def rightOf : (Int,Int) = (tuple._1, tuple._2 + 1)
    def aboveOf : (Int,Int) = (tuple._1 - 1, tuple._2)
    def belowOf : (Int,Int) = (tuple._1 + 1, tuple._2)
  }
}
```

Use of Higher Order Function

- Example: JsonHelperSpec



```scala
val -- = (a:Int,b:Int) => a - b
val ++ = (a:Int,b:Int) => a + b

private def upperLeftFor5StepsInBothDimensions(coord : (Int,Int))
  (y : Int, fn1 : (Int,Int) => Int)
  (x : Int, fn2 : (Int,Int) => Int) = {
  for {
        i <- 0 until 5
        j <- 0 until 5
  } yield coord.isUpperLeftOf(fn1(y,i),fn2(x,j))
}.toList
⋮
    upperLeftFor5StepsInBothDimensions(testCoord1)(10, --)(11, ++)
      .exists(_ == true) must be_!=(true)
```

Tests

- 200 fully automated Tests
- Code Coverage scct
  - Model 100%
  - GUI / TUI: Utils 100%
  - Controller 76%
- Coverage Report as html

Generate Standalone Game

- Server/WUI
  - sbt “project wui” dist
  - Generates zip file
  - Extract and execute Skript wui in bin/ to run server
- GUI/TUI
  - sbt assembly
  - Generates TacZombieClient.jar
  - Execute java -jar TacZombieClient.jar for GUI
  - Add parameter “tui” for TUI

Demo

- Lean back and enjoy :-)

Summary

- GUI & TUI & WUI usable
  - at the ***same*** time
  - on the ***same*** model
- Model fully automated tested
- Controller automated tested
- SBT Multi-project

- Scala is awesome!
- Swing is ok
- ScalaFx lacks of support and examples