Project: Connect 4

Vanessa Predovic

Software Engineering SS 2020

Content

- Rules of Connect 4
- With 13 Tasks to a running game

Connect 4: Rules

- Board with 6 rows and 7 columns
- Each player has 21 colored pieces

Connect 4: Rules

Objective:

- Connect four pieces in a row of your color
- Possible direction: horizontal, vertical, diagonal

Connect 4: With 13 Tasks to a running game

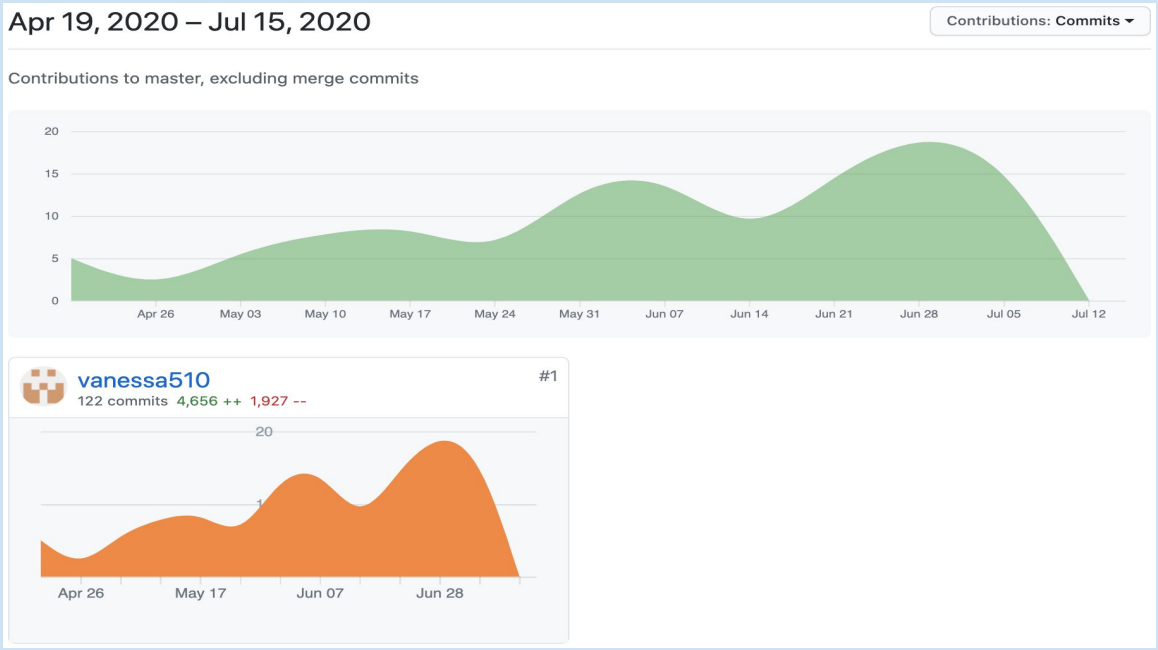
Game Project Structure and Worksheet

- Build simple Data structure

```
case class Cell (isSet:Boolean, color:Optional[String] = Optional.empty()) {
  def setColor():Unit = {}
}

val cell = Cell(false)
cell.isSet

case class Matrix[Cell](rows:Vector[Vector[Cell]]) {

  def this(sizeOfRows:Int, sizeOfCol:Int, cell: Cell) =
    this(Vector.tabulate(sizeOfRows, sizeOfCol){(row, col) => cell})

  def sizeOfRows:Int = rows.size

  def sizeOfCols: Int = rows(0).length

  def cell(row:Int, col:Int):Cell = rows(row)(col)

  def replaceCell(row:Int, col:Int, cell:Cell): Matrix[Cell] =
    copy(rows.updated(row, rows(row).updated(col, cell)))

}

case class Board(cells: Matrix[Cell]) {
  def this(sizeOfRows: Int, sizeOfCol: Int, isSet: Boolean) =
    this(new Matrix[Cell](sizeOfRows, sizeOfCol, Cell(isSet)))

  def size: Int = cells.sizeOfRows

  def cell(row: Int, col: Int): Cell = cells.rows(row)(col)

  def getBoardAsString(matrix: Matrix[Cell]): String = {
    val rows = matrix.sizeOfRows
    val cols = matrix.sizeOfCols

    var returnString = "\n"
    val oneLine = " __ " * cols
```

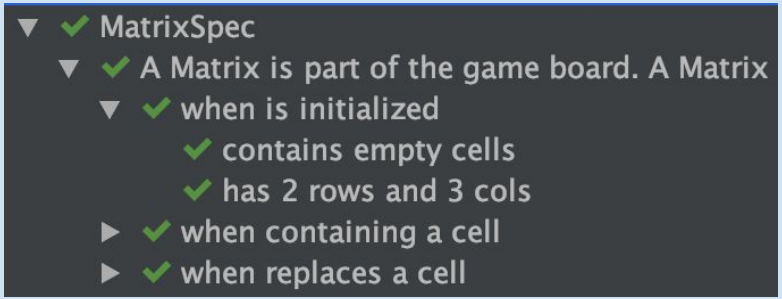
```
name := "Vier-Gewinnt"

version := "0.1"

scalaVersion := "2.12.7"

libraryDependencies += "org.scalactic" %% "scalactic" % "3.0.5"
libraryDependencies += "org.scalatest" %% "scalatest" % "3.0.5" % "test"

libraryDependencies += "org.scala-lang.modules" %% "scala-swing" % "2.1.1"

libraryDependencies += "com.google.inject" % "guice" % "4.1.0"
libraryDependencies += "net.codingwell" %% "scala-guice" % "4.1.0"

libraryDependencies += "com.typesafe.play" %% "play-json" % "2.6.6"
```

Git Repository



Agile Development: Tests and Code Coverage

aview	5% (1/18)	12% (5/40)	10% (13/125)
controller	69% (9/13)	91% (61/67)	92% (117/127)
model	71% (15/21)	92% (82/89)	93% (215/230)
util	80% (4/5)	93% (14/15)	96% (32/33)
Connect4Module	100% (1/1)	100% (2/2)	100% (5/5)



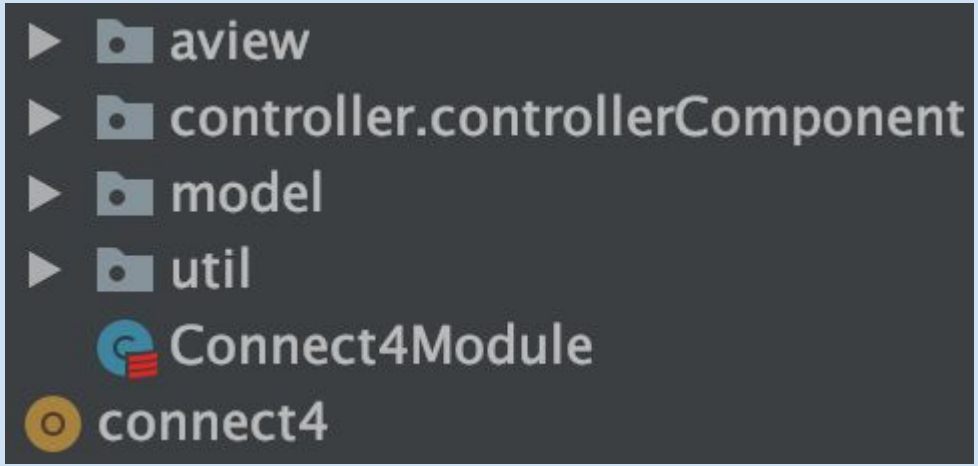
A simple Text User Interface

```scala
package de.htwg.se.connect4.aview

import de.htwg.se.connect4.controller.controllerComponent.ControllerInterface
import de.htwg.se.connect4.model.boardComponent.BoardInterface
import de.htwg.se.connect4.util.Observer

class Tui(controller: ControllerInterface) extends Observer {

  val rows: Int = 6
  val cols: Int = 7

  controller.add(this)

  def processInputLine(input: String, board: BoardInterface): String = {

    input match {
      case "q" => "exit game"
      case "n" => controller.createNewBoard(rows, cols)
      case "r" => controller.redo
      case "u" => controller.undo
      case "s" => controller.save
      case "l" => controller.load

      case _ => controller.handle(input, board)

    }
  }

  override def update: Unit = println(controller.stateString)


}
```

```
Welcome to connect 4. Please Enter your names.
test1

test2


 __ __ __ __ __ __ __
| - | - | - | - | - | - | - |
 __ __ __ __ __ __ __
| - | - | - | - | - | - | - |
 __ __ __ __ __ __ __
| - | - | - | - | - | - | - |
 __ __ __ __ __ __ __
| - | - | - | - | - | - | - |
 __ __ __ __ __ __ __
| - | - | - | - | - | - | - |
 __ __ __ __ __ __ __
| - | - | - | - | - | - | - |
It's your turn Player test1
```

Architecture: Model-View-Controller

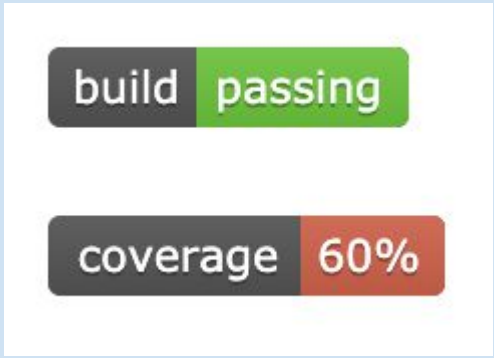
- Remove higher layer classes of lower layer ones

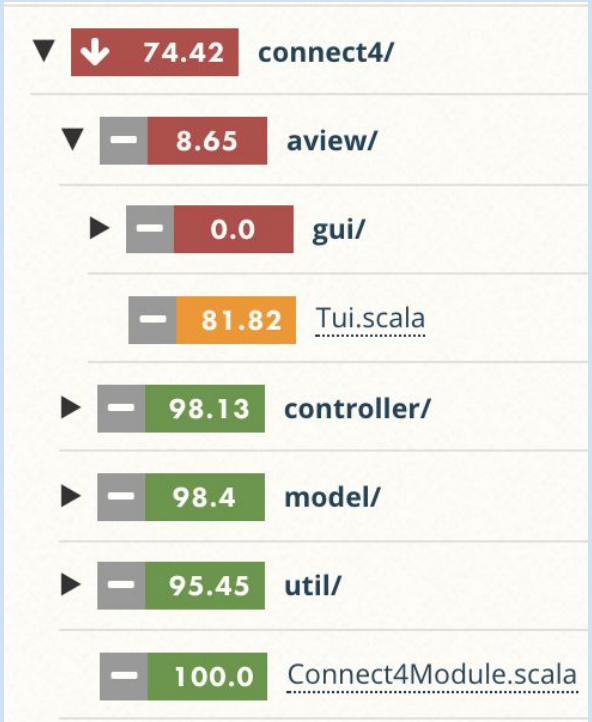


Continuous Development: Travis CI and Coveralls

.travis.yml

```
language: scala
scala:
  - 2.12.7

script:
  - sbt clean coverage test coverageReport

after_success:
  - sbt coverageReport coveralls
```





Pattern: Observer Pattern

- Remove circular dependencies

```scala
package de.htwg.se.connect4.util


  trait Observer {
    def update: Unit

  }

  class Observable {
    var subscribers: Vector[Observer] = Vector()

    def add(s: Observer): Unit = subscribers = subscribers :+ s

    def remove(s: Observer): Unit = subscribers = subscribers.filterNot(o => o == s)

    def notifyObservers: Unit = subscribers.foreach(o => o.update)

  }
```

Pattern: Strategy Pattern

- Board of different sizes

```
package de.htwg.se.connect4.model.boardComponent.boardBaseImpl

object BoardSizeStrategy {

  trait BoardSizeStrategy {
    def strategy(boardSize: (Int, Int))

    def defaultSizeStrategy(boardSize: (Int, Int)): Board

    def tinySizeStrategy(boardSize: (Int, Int)): Board

    def bigSizeStrategy(boardSize: (Int, Int)): Board

    def execute(boardSize: Int):Board
  }

  def execute(boardSize: (Int, Int)): Board = strategy(boardSize)


  def strategy(boardSize: (Int, Int)): Board = {
    boardSize match {
      case (6,7) => defaultSizeStrategy(boardSize)
      case (2,3) => tinySizeStrategy(boardSize)
      case (15,16) => bigSizeStrategy(boardSize)
      case (_,_) => defaultSizeStrategy((6,7))
    }
  }

  def defaultSizeStrategy(boardSize: (Int, Int)): Board = {
    Creator().sizeOfBoard(boardSize._1, boardSize._2)
  }

  def tinySizeStrategy(boardSize: (Int, Int)): Board = {
    Creator().sizeOfBoard(boardSize._1, boardSize._2)
  }

  def bigSizeStrategy(boardSize: (Int, Int)): Board = {
    Creator()sizeOfBoard(boardSize._1, boardSize._2)
  }

}
```

Pattern: State Pattern

- Add state to controller to manage game state and match output

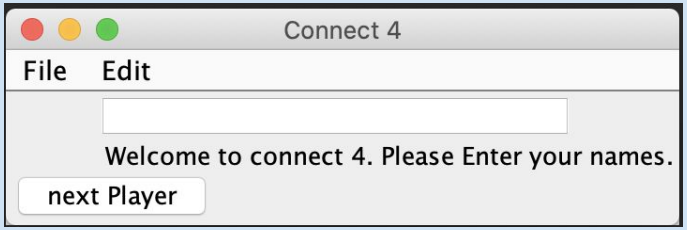
```
package de.htwg.se.connect4.controller.controllerComponent.controllerBaseImpl

import de.htwg.se.connect4.model.boardComponent.BoardInterface

abstract class ControllerState {

  def handle(input: String, board: BoardInterface): String

  def nextState(): ControllerState

  def stateString(): String

  def toString(): String

}
```

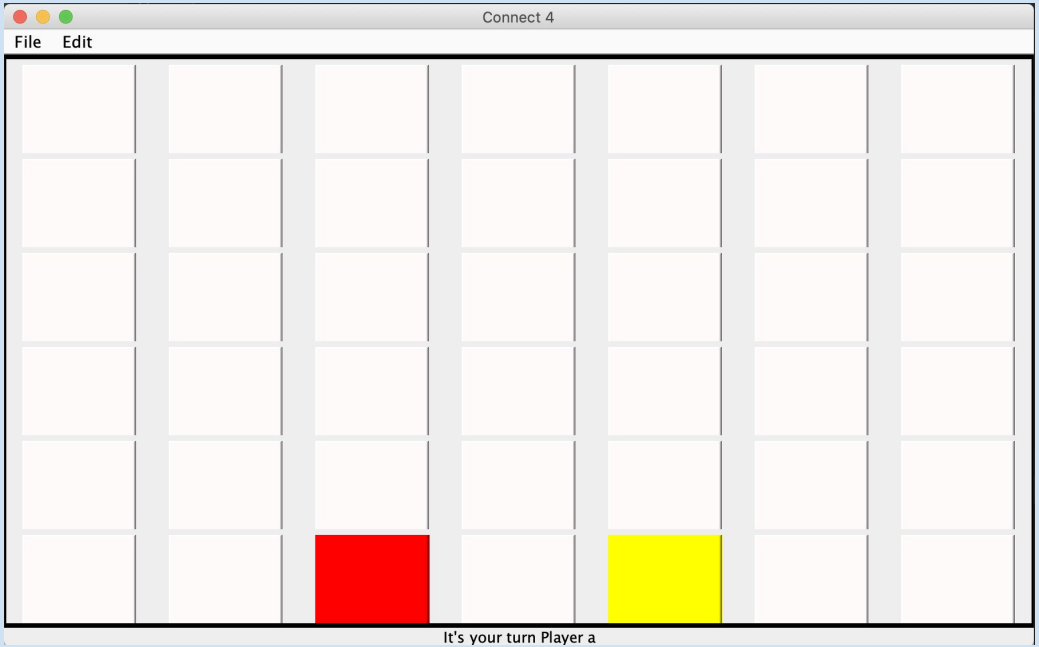
Pattern: Command-Pattern

```scala
package de.htwg.se.connect4.util

class UndoManager {
  private var undoStack: List[Command]= Nil
  private var redoStack: List[Command]= Nil

  def doStep(command: Command): Unit = {
    undoStack = command::undoStack
    command.doStep
  }
  def undoStep(): Unit  = {
    undoStack match {
      case  Nil =>
      case head::stack => {
        head.undoStep
        undoStack=stack
        redoStack= head::redoStack
      }
    }
  }
  def redoStep(): Unit = {
    redoStack match {
      case Nil =>
      case head::stack => {
        head.redoStep
        redoStack=stack
        undoStack=head::undoStack
      }
    }
  }
}
```

```scala
package de.htwg.se.connect4.controller.controllerComponent.controllerBaseImpl

import ...

class SetCommand(row: Int, col: Int, currentPlayer: Player, controller: Controller, isSet: Boolean) extends Command {

  override def doStep: Unit = controller.board = controller.board.set(row, col, currentPlayer.color, isSet)

  override def undoStep: Unit = controller.board = controller.board.set(row, col, Color.EMPTY,  isSet = false)

  override def redoStep: Unit = controller.board = controller.board.set(row, col, currentPlayer.color, isSet)
}
```

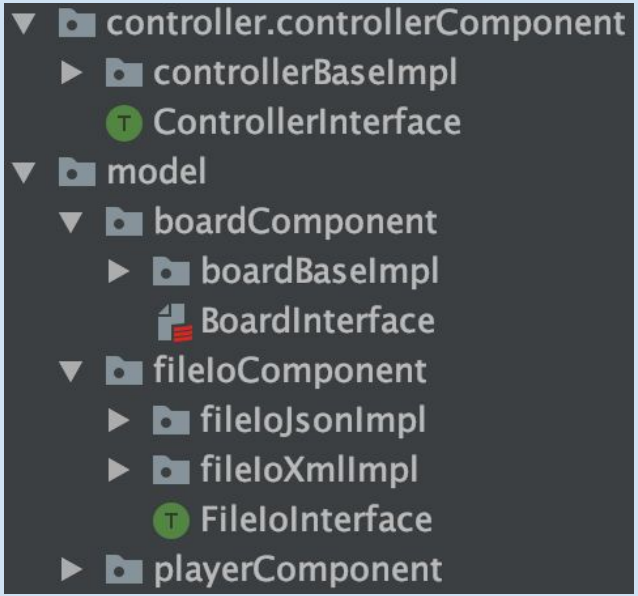
Graphical User Interface (Scala Swing)





Components

- Add interfaces and divide into components to change implementation



Dependency Injection

```
class Connect4Module extends AbstractModule with ScalaModule {
  override def configure(): Unit = {
    bind[BoardInterface].toInstance(BoardSizeStrategy.execute(6, 7))
    bind[List[Player]].toInstance(Nil)
    bind[ControllerInterface].to[controller.controllerComponent.controllerBaseImpl.Controller]

    bind[FileIoInterface].to[model.fileIoComponent.fileIoJsonImpl.FileIO]
  }
}
```

```
class Controller @Inject()(var board: BoardInterface, var players: List[Player])

  val injector = Guice.createInjector(new Connect4Module)
  val fileIo = injector.instance[FileIoInterface]
```

File IO: Json

```scala
package de.htwg.se.connect4.model.fileIoComponent.fileIoJsonImpl

import ...

class FileIO extends FileIoInterface {

  override def load: (BoardInterface, State) = {
    var board: BoardInterface = null
    val source: String = Source.fromFile("board.json").getLines.mkString
    val json: JsValue = Json.parse(source)

    val sizeOfRows = (json \ "board" \ "row").get.toString.toInt
    val sizeOfCols = (json \ "board" \ "col").get.toString.toInt

    board = BoardSizeStrategy.execute(sizeOfRows, sizeOfCols)

    val injector = {
      Guice.createInjector(new Connect4Module)
    }

    for (index <- 0 until sizeOfRows * sizeOfCols) {
      val row = (json \ "board" \ "cells" \\ "row") (index).as[Int]
      val col = (json \ "board" \ "cells" \\ "col") (index).as[Int]
      val cell = (json \\ "cell") (index)
      val isSet = (cell \ "isSet").as[Boolean]
      val color = (cell \ "color" \ "color").as[Color]

      board = board.set(row, col, color, isSet)

    }

    val currentPlayerIndex = (json \ "currentPlayerIndex").get.toString().toInt
    var players: List[Player] = Nil
```

```scala
package de.htwg.se.connect4.util

import play.api.libs.json._


object EnumRead {
  def enumReads[E <: Enumeration](enum: E): Reads[E#Value] = new Reads[E#Value] {
    def reads(json: JsValue): JsResult[E#Value] = json match {
      case JsString(s) => {
        try {
          JsSuccess(enum.withName(s))

        } catch {
          case _: NoSuchElementException =>
            JsError(s"Enumeration expected of type: '${enum.getClass}', but it does not appear to contain the value: '$s'")
        }
      }
      case _ => JsError("String value expected")
    }
  }

}
```

File IO: Json


{
"currentPlayerIndex" : 0,
"state" : "InitializationState",
"players" : [{
"name" : "test1",
"color" : {
"color" : "red"
},
"piecesLeft" : 1
}, {
"name" : "test2",
"color" : {
"color" : "yellow"
},
"piecesLeft" : 0
}],
"board" : {
"row" : 2,
"col" : 4,
"cells" : [{
"row" : 0,
"col" : 0,
"cell" : {
"isSet" : false,
"color" : {
"color" : "empty"
}
}

File IO: Xml

```scala
package de.htwg.se.connect4.model.fileIoComponent.fileIoXmlImpl

import ...

class FileIO extends FileIoInterface {


  override def load: (BoardInterface, State) = {
    var board: BoardInterface = null
    val file = scala.xml.XML.loadFile("board.xml")
    val rowAttr = file \\ "game"  \\ "board" \ "@row"
    val colAttr = file \\ "game" \\ "board" \  "@col"

    val sizeOfRows = rowAttr.text.toInt
    val sizeOfCols = colAttr.text.toInt

    board = BoardSizeStrategy.execute(sizeOfRows, sizeOfCols)

    val cellNodes = file \\ "cell"
    for (cell <- cellNodes) {
      val row: Int = (cell \ "@row").text.toInt
      val col: Int = (cell \ "@col").text.toInt
      val colorAttr: String = (cell \ "@color").text
      val color: Color = Color.toEnum(colorAttr)
      val isSet: Boolean = (cell \ "@isSet").text.toBoolean
      board = board.set(row, col, color, isSet)
    }
```

```xml
<game>
  <currentPlayerIndex currentPlayerIndex="0"> </currentPlayerIndex>
  <stateString stateString="InitializationState"> </stateString>
  <players>
    <player name="test1" piecesLeft="1" color="red"> </player>
    <player name="test2" piecesLeft="0" color="yellow"> </player>
  </players>
  <board row="2" col="4">
    <cell row="0" col="0" isSet="false" color="empty"> </cell>
    <cell row="0" col="1" isSet="false" color="empty"> </cell>
    <cell row="0" col="2" isSet="false" color="empty"> </cell>
    <cell row="0" col="3" isSet="false" color="empty"> </cell>
    <cell row="1" col="0" isSet="false" color="empty"> </cell>
    <cell row="1" col="1" isSet="false" color="empty"> </cell>
```

Docker

```
[info] Fetched artifacts of
[info] Compiling 29 Scala sources to /connect4/target/scala-2.12/classes ...
https://repo1.maven.org/maven2/org/scala-sbt/util-interface/1.3.0/util-interface-1.3.0.pom
  100.0% [##########] 2.7 KiB (123.2 KiB / s)
[info] Non-compiled module 'compiler-bridge_2.12' for Scala 2.12.7. Compiling...
[info]   Compilation completed in 12.674s.
[info] running connect4
Welcome to connect 4. Please Enter your names.
```

Thank you for your attention