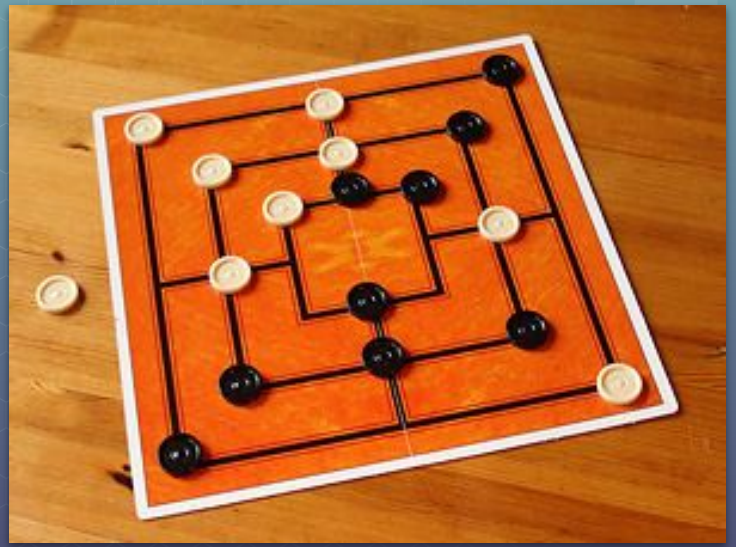
Project: Mill

Group 9/13 - Manuel Günter
Kevin Schoch
Software Engineering - 17.07.2020

TABLE OF CONTENTS

Mill

- Two Players: White vs. Black
- 9 stones per player
- move stones
- remove other player's stones with mill
- first to have less than 3 stones loses



Development of the Game Mill

13 tasks of Software Engineering

Project Setup & Worksheet

build.sbt

```
name          := "htwg-scala-mill"
version       := "0.13"
scalaVersion  := "2.13.2"

libraryDependencies += "org.scalactic" %% "scalactic" % "3.1.2"

libraryDependencies += "org.scalatest" %% "scalatest" % "3.1.2" % "test"

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

libraryDependencies += "com.google.inject" % "guice" % "4.2.3"

libraryDependencies += "net.codingwell" %% "scala-guice" % "4.2.10"

libraryDependencies += "org.scala-lang.modules" %% "scala-xml" % "1.2.0"

libraryDependencies += "com.typesafe.play" %% "play-json" % "2.9.0"

```

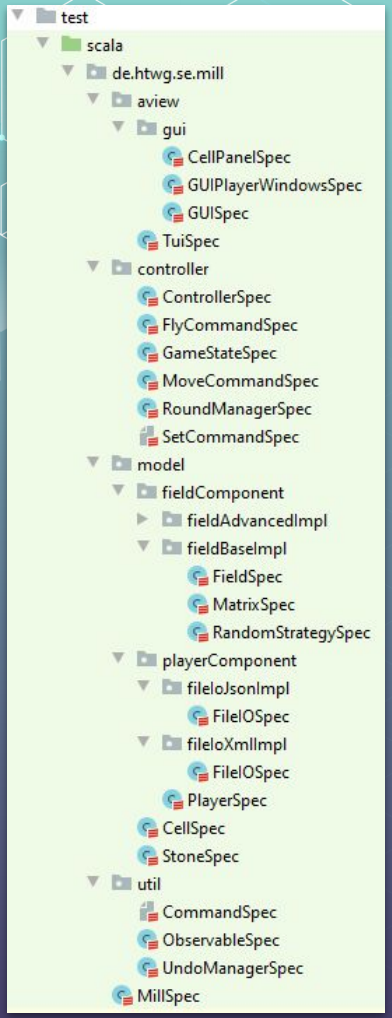
```
import de.htwg.se.mill.model.fieldComponent.{C
import de.htwg.se.mill.model.fieldComponent.fi


val stone = Stone("w")

val colorset = Color.values.toIndexedSeq
val h = colorset.apply(0)
val s = Color.black
println(s)

var field = new Field( size = 7)
field = field.set( row = 0, col = 0, Cell("ce"))
field = field.set( row = 0, col = 3, Cell("cw"))
field = field.set( row = 0, col = 6, Cell("wb"))
```

build.sbt

Worksheet

Git & Github

Contributions



Tests & Code Coverage

79% classes, 93% lines covered in package 'de.htwg.se.mill'

Element	Class, %	Method, %	Line, %
aview	100% (28/28)	100% (83/83)	95% (229/241)
controller	61% (27/44)	85% (109/127)	92% (306/331)
model	84% (21/25)	97% (117/120)	98% (326/331)
util	100% (2/2)	100% (11/11)	100% (24/24)
Mill	0% (0/2)	0% (0/8)	0% (0/16)
MillModule	100% (9/9)	100% (11/11)	100% (16/16)

Our version of the game Mill

build passing coverage 94%

TUI - Text User Interface

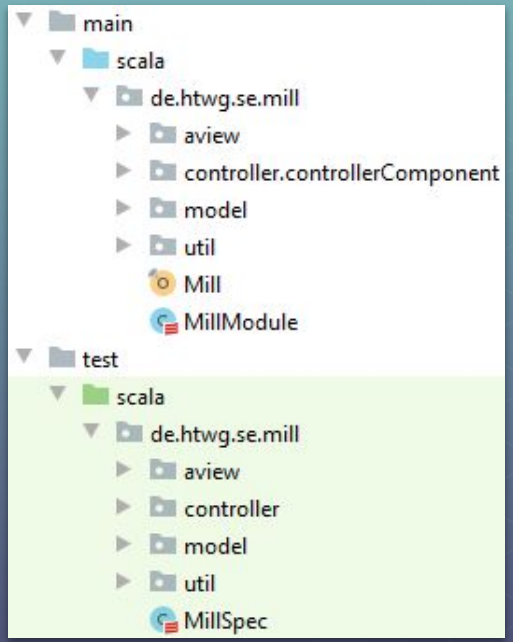
```
Mill Gameboard:
 o  -  -  o  -  -  o
 -  o  -  o  -  o  -
 -  -  o  o  o  -  -
 o  o  o  -  o  o  o
 -  -  o  o  o  -  -
 -  o  -  o  -  o  -
 o  -  -  o  -  -  o

New field
Possible commands: new, random, place <location,0/1>, undo, redo, exit  -->00
Mill Gameboard:
 w  -  -  o  -  -  o
 -  o  -  o  -  o  -
 -  -  o  o  o  -  -
 o  o  o  -  o  o  o
 -  -  o  o  o  -  -
 -  o  -  o  -  o  -
 o  -  -  o  -  -  o

Black's turn
No Mill
valid command: 00
Possible commands: new, random, place <location,0/1>, undo, redo, exit  -->
```

set stone

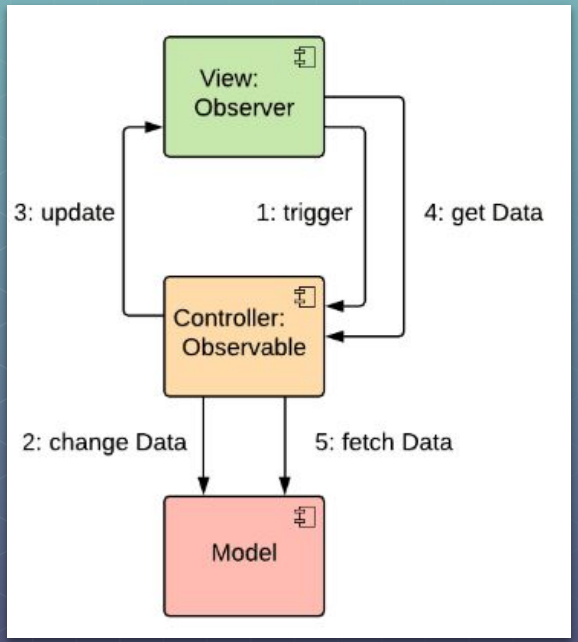
```
Possible commands: new, random, place <location,0/1>, undo, redo, exit  -->36
No Mill
valid command: 36
Possible commands: new, random, place <location,0/1>, undo, redo, exit  -->66
Mill Gameboard:
 w  -  -  b  -  -  w
 -  b  -  w  -  b  -
 -  -  w  b  w  -  -
 b  w  b  -  w  b  o
 -  -  b  w  b  -  -
 -  o  -  o  -  o  -
 o  -  -  o  -  -  w
```

move stone

Model-View-Controller



architecture



pattern

Continuous Deployment

```
language: scala
scala:
  - 2.13.2

services:
  - xvfb

env:
  - DISPLAY=:99.0

script:
  - sbt clean coverage test coverageReport

after_success:
  - sbt coverageReport coveralls
```

travis.yml

master Improved coverage

Commit a480404

Compare 39f6e95..a480404

Branch master

KzuDemEvin

Scala: 2.13.2

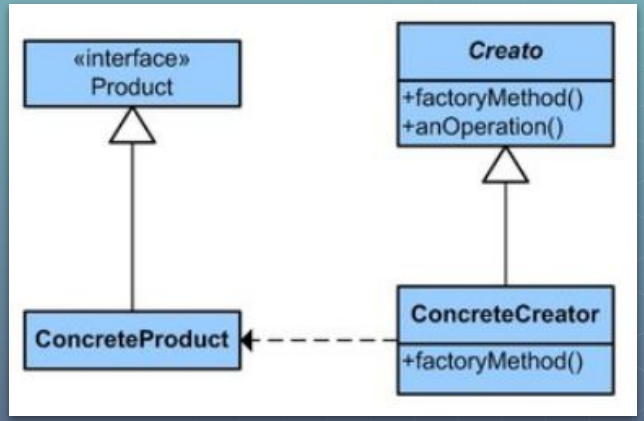
AMD64

DISPLAY=:99.0

build passing coverage 94%

Travis & Coveralls

Design Pattern - Factory Method



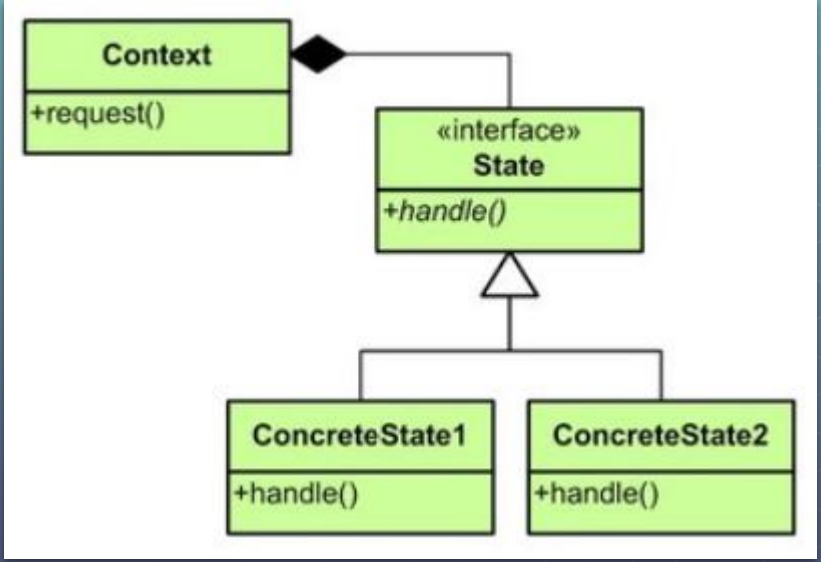
- Stone
- Cell
- Player

```
trait Stone {
  def isSet: Boolean
  def whichColor: Color.Value
}
```

```
object Stone {
  def apply(kind: String):Stone = kind match{
    case "w+" => new WhiteStone( value = 1, Color.white)
    case "w-" => new WhiteStone( value = 0, Color.white)
    case "b+" => new BlackStone( value = 1, Color.black)
    case "b-" => new BlackStone( value = 0, Color.black)
    case "n" => new ColorLessStone( value = 0, Color.noColor)
  }
}
```

Stone - Factory Method

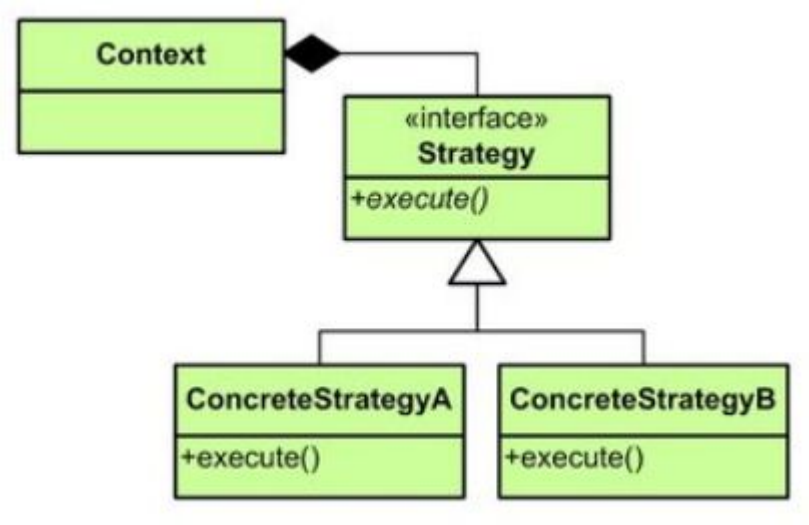
Design Pattern - State




```
trait ModeState {
  def handle:String
  def whichState:ModeState
}
```

- GameState
- MillState
- ModeState

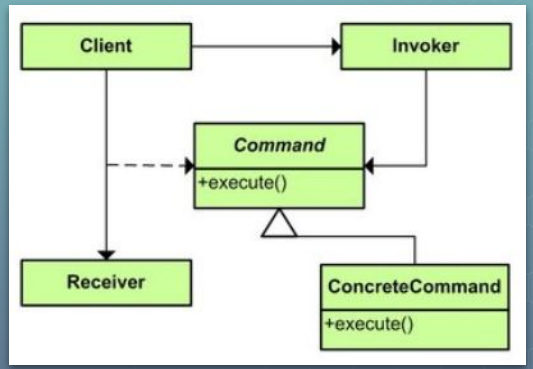
Design Pattern - Strategy




```
trait Strategy {

  def createNewField(size:Int): Field = {
    var field  = new Field(size)
    field = fill(field)
    field
  }

  def fill(field: Field) : Field
}
```

- RandomStrategy

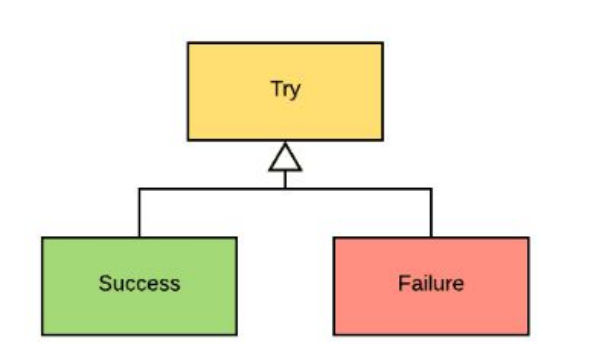
Design Pattern - Command Pattern



- SetCommand
- MoveCommand
- FlyCommand

```
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
      }
    }
  }
}
```

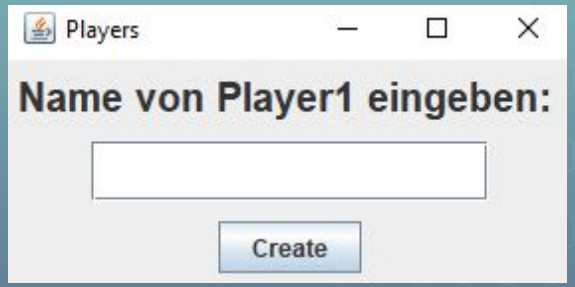
Design Pattern - Try



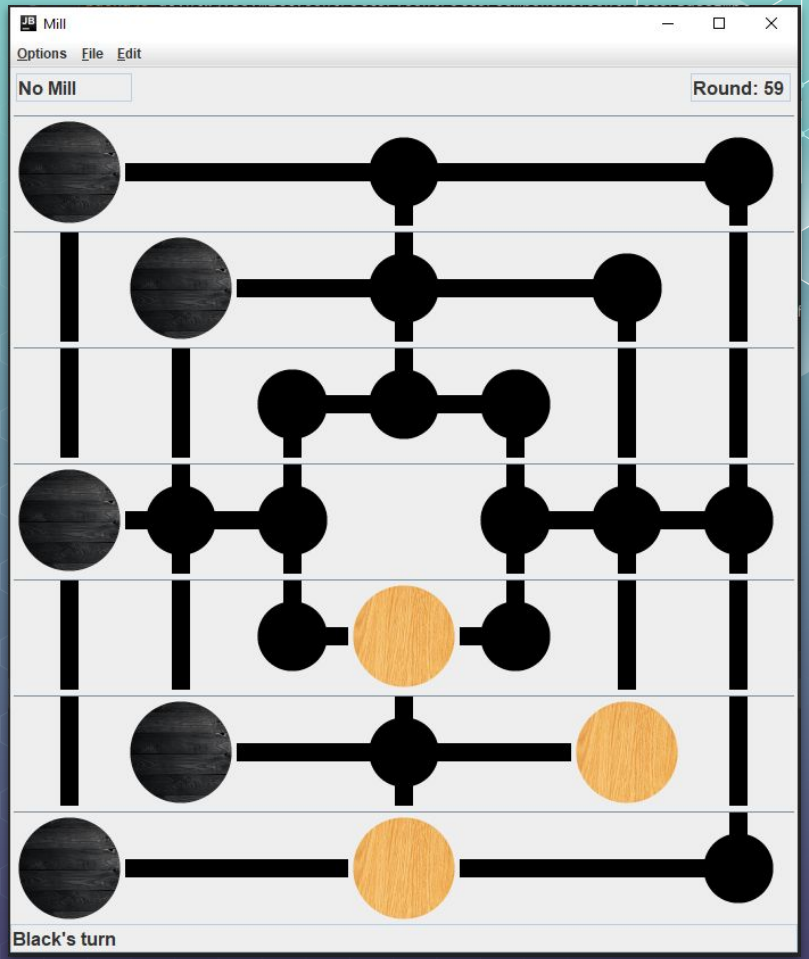
used in TUI

```
def execInput(input: String): Try[String] = {
  input match {
    case "new" => controller.createEmptyField(size)
      Success("valid command: " + input)
    case "random" => controller.createRandomField(size)
      Success("valid command: " + input)
    case "undo" => controller.undo
      Success("valid command: " + input)
    case "redo" => controller.redo
      Success("valid command: " + input)
    case "save" => controller.save
      Success("valid command: " + input)
    case "load" => controller.load
      Success("valid command: " + input)
    case "exit" =>
      Success(input)
    case _ => input.toList.filter(p => p != ' ').filter(_.isDigit).map(p
      case row :: column :: Nil => controller.handleClick(row, column)
        println(controller.millState)
        Success("valid command: " + input)
    }
    case _ =>
      Failure(new IllegalArgumentException("Wrong input: " + input))
  }
```

GUI



Endscreen

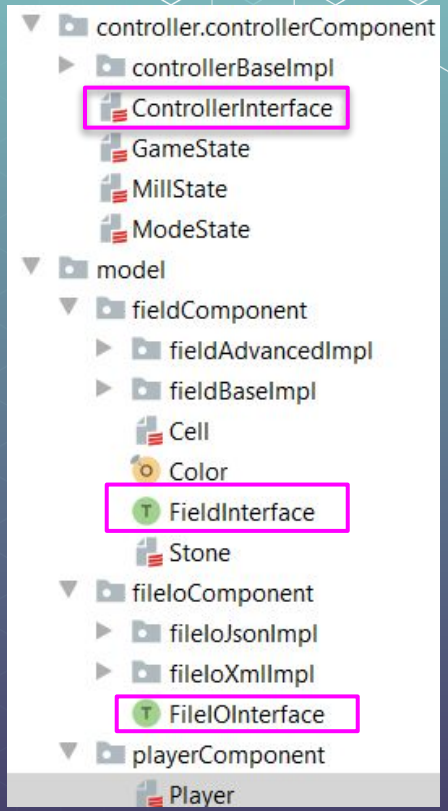


Components

implement Interfaces for communication:

- ControllerInterface
- FieldInterface
- FileIOInterface

➢ Dependency only to abstractions
➢ Implementation can change



Dependency Injection

```scala
class MillModule extends AbstractModule with ScalaModule {

  val defaultSize:Int = 7

  override def configure(): Unit = {
    bindConstant().annotatedWith(Names.named( name = "DefaultSize")).to(defaultSize)
    bind[FieldInterface].to[Field]
    bind[ControllerInterface].to[controllerBaseImpl.Controller]


    bind[FieldInterface].annotatedWithName( name = "normal").toInstance(new Field(defaultSize))
    bind[FieldInterface].annotatedWithName( name = "random").toInstance((new RandomStrategy).createNewField(defaultSize))

    bind[FileIOInterface].to[fileIoXmlImpl.FileIO] //XML
    //bind[FileIOInterface].to[fileIoJsonImpl.FileIO] //JSON
  }
```

MillModule

```scala
class Controller @Inject() (var field: FieldInterface) extends ControllerInterface with Publisher {
```

Injection

FileIO - XML

```scala
def save(field: FieldInterface): Unit = saveString(field)

def saveString(field: FieldInterface): Unit = {
  import java.io._
  val pw = new PrintWriter(new File( pathname = "field.xml"))
  val prettyPrinter = new PrettyPrinter(120, 4)
  val xml = prettyPrinter.format(fieldToXml(field))
  pw.write(xml)
  pw.close
}

def fieldToXml(field: FieldInterface): Node = {
  <field roundCounter={ field.savedRoundCounter.toString } player1Mode={ field.player1Mode }
         player2Mode={ field.player2Mode }>
    {
    for {
      row <- 0 until field.size
      col <- 0 until field.size
    } yield cellToXml(field, row, col)
    }
  </field>
}

def cellToXml(field: FieldInterface, row: Int, col: Int): Node = {
  <cell row={ row.toString } col={ col.toString }>
    { field.cell(row, col).getContent.whichColor }
  </cell>
}
```

saving method

```
<field roundCounter="0" player1Mode="SetMode" player2Mode="SetMode">
      <cell row="0" col="0">
      white
    </cell><cell row="0" col="1">
      noColor
    </cell><cell row="0" col="2">
      noColor
    </cell><cell row="0" col="3">
      noColor
    </cell><cell row="0" col="4">
      noColor
    </cell><cell row="0" col="5">
      noColor
    </cell><cell row="0" col="6">
```

field.xml

FileIO - JSON

```
override def save(field: FieldInterface): Unit = {
  import java.io._
  val pw = new PrintWriter(new File( pathname = "field.json"))
  pw.write(Json.prettyPrint(fieldToJson(field)))
  pw.close()
}

def fieldToJson(field: FieldInterface): JsValue = {
  Json.obj(
    fields = "field" -> Json.obj(
      fields = "roundCounter" -> JsNumber(field.savedRoundCounter),
      "player1Mode" -> JsString(field.player1Mode),
      "player2Mode" -> JsString(field.player2Mode),
      "cells" -> Json.toJson(
        for {
          row <- 0 until field.size
          col <- 0 until field.size
        } yield {
          Json.obj(
            fields = "row" -> row,
            "col" -> col,
            "color" -> Json.toJson(field.cell(row, col).getContent.whichColor)
          )
        }
```

saving method

field.json

Docker

```
[info] running de.htwg.se.mill.Mill
Mill Gameboard:
 o  -  -  o  -  -  o
 -  o  -  o  -  o  -
 -  -  o  o  o  -  -
 o  o  o  -  o  o  o
 -  -  o  o  o  -  -
 -  o  -  o  -  o  -
 o  -  -  o  -  -  o

New field
Possible commands: new, random, place <location,0/1>, undo, redo, exit  -->00
Mill Gameboard:
 w  -  -  o  -  -  o
 -  o  -  o  -  o  -
 -  -  o  o  o  -  -
 o  o  o  -  o  o  o
 -  -  o  o  o  -  -
 -  o  -  o  -  o  -
 o  -  -  o  -  -  o

Black's turn
No Mill
valid command: 00
Possible commands: new, random, place <location,0/1>, undo, redo, exit  -->03
Mill Gameboard:
 w  -  -  b  -  -  o
 -  o  -  o  -  o  -
 -  -  o  o  o  -  -
 o  o  o  -  o  o  o
 -  -  o  o  o  -  -
 -  o  -  o  -  o  -
 o  -  -  o  -  -  o

White's turn
No Mill
valid command: 03
Possible commands: new, random, place <location,0/1>, undo, redo, exit  -->
```

Docker executed in shell

```
1  FROM hseeberger/scala-sbt:8u222_1.3.5_2.13.1
2  WORKDIR /mill
3  ADD . /mill
4  CMD sbt run
```

Dockerfile

Thank you for your attention !