Reactive programming: An Empirical Study on Program Comprehension

Technical University of Darmstadt

March 2014
1 Introduction

In the reminder of the report we present the tasks we used for the empirical study. We initially show the tasks in the RP version. All the respective OO versions follow. For each task we show the comprehension question we asked, the proposed options and the code of the application.

Finally we present the preliminary questions we used to assess the programming skills of the subjects.

2 Tasks

Task 1 - RP

Question:

What is the final value of k after the execution of the following code

Options:

- 0
- false
- 3
- true

```java
1 val a = Var(0)
2 val b = Var(0)
3 val c = Signal { 2 * a() }
4 val d = Signal { 3 * a() }
5 val e = Signal { d() == b() }
6 val k = Signal { e() || false }
7 a() = 1
8 b() = 1
9 b() = 3
11 // value of k here ?
```

Task 2 - RP

Question:

Which property is always true (invariant) during the execution of the following application?

Options:

- c contains the value 0
- a is updated after d
- if d is updated both c and b have been already updated
- a is updated after c

```scala
val a = Var(0)
val b = Signal{ a() + 1 }
val c = Signal{ 2 * a() }
val d = Signal{ b() + c() }

a() = 3
```

**Task 3 - RP**

**Question:**

What is the sequence of values held by y during the execution of the program? (Do not count repeated contiguous values, for example, 1,2,2,3,2 counts as 1,2,3,2)

**Options:**

- 1,2,3
- 0,2
- 0,2,4
- 0,1,2

```scala
val a = Var(0)
val b = Var(0)
val c = Signal{ a() + b() }
val d = Signal{ 2 * a() }
val e = Signal{ d() == b() }
val y = Signal{ if (e()) c() else d() }

a()=1
b()=1
a()=2
```

**Task 4 - RP**

**Question:**

Which variables can potentially change after a has been updated?

**Options:**

- e, f, g
- f, g
- c, f, g
- c, e, g
val a = Var(0)
val b = Var(1)

val c = Signal { a() + 1 }
val d = Signal { b() * 2 }
val e = Signal { c() + d() % 3 }
val f = Signal {
  if (b() % 2 == 0)
    d() * 3
  else
    10
}
val g = Signal { a() / f() }

---

Task 5 - RP

Question:

The application displays four particles, i.e. four black circles on the screen. Which of the following sentences is correct for the application?

Options:

- The particles do not change over time.
- The particles become ellipses, i.e. they become flatter over time.
- The particles periodically get bigger and then small again.
- The particles increase in size indefinitely.

---

object Particles1_Reactive extends SimpleSwingApplication {

  // — APPLICATION LOGIC ——

  class Particle{
    val center: Point,
    val size1: Double,
    val size2: Double,
    val interval: Time) {

    val time1 = Signal { time() % interval }
    val progress = Signal { time1() / interval }

    val normalSize = Signal { 2 * (if (progress() < 0.5) progress() else 1 - progress()) }

}
val size = Signal { size1 + (size2 - size1) * normalSize() }

val particle0 = new Particle(Point(100, 100), 20, 50, 2.0.s)
val particle1 = new Particle(Point(200, 100), 5, 20, 1.8.s)
val particle2 = new Particle(Point(100, 200), 10, 30, 2.1.s)
val particle3 = new Particle(Point(200, 200), 15, 40, 1.5.s)

// painting component
particle0.size.changed ||
particle1.size.changed ||
particle2.size.changed ||
particle3.size.changed += { _ => Swing.onEDT { top.repaint } }

// — GUI stuff ————————————————————

// lazy val panel: RePanel = new RePanel {
   override def paintComponent(g: Graphics2D) {
      super.paintComponent(g)
      for (particle <- Seq(particle0, particle1, particle2, particle3))
         g.fillOval((particle.center.x - particle.size.getValue / 2).toInt,
                    (particle.center.y - particle.size.getValue / 2).toInt,
                    particle.size.getValue.toInt,
                    particle.size.getValue.toInt)
   }

lazy val top = new MainFrame {
   preferredSize = new Dimension(500, 400)
   contents = panel
}

Task 6 - RP

Question:

The particles in the following applications are moving. They:

Options:

• Circle around the same center
• Circle around different centers on the same horizontal line
• Get bigger and smaller regularly
• Move horizontally form left to right and from right to left at regular intervals

import commons._
import react._
import react.Signal
import 'macro'.SignalMacro.{SignalM => Signal}
import react.events._
import swing.{ Panel, MainFrame, SimpleSwingApplication }
import swing.Swing
import java.awt.{Color, Graphics2D, Dimension}
import reswing.RePanel
import math.{sin, cos, Pi}
import Animation._

object Particles2Reactive extends SimpleSwingApplication {

// — APPLICATION LOGIC ——

class Particle(val center: Point, val radius: Double, val period: Time) {
  val time1 = Signal { time() % period }
  val angle = Signal { 2 * Pi * (time1() / period) }
  val position1 = Signal { Point(cos(angle()), sin(angle())) }
  val position2 = Signal { position1() * radius + center }
}

val particle0 = new Particle(Point(100, 150), 40, 1.5.s)
val particle1 = new Particle(Point(300, 150), 50, 1.3.s)
val particle2 = new Particle(Point(500, 150), 45, 1.7.s)
val particle3 = new Particle(Point(700, 150), 30, 1.2.s)

// painting component
particle0.position2.changed ||
particle1.position2.changed ||
particle2.position2.changed ||
particle3.position2.changed += { _ => Swing.onEDT { top.repaint } }

// — Draw the particles ——
lazy val panel: RePanel = new RePanel {
  override def paintComponent(g: Graphics2D) {
    super.paintComponent(g)
    for (particle <- Seq(particle0, particle1, particle2, particle3))
      g.fillOval(
        particle.position2.getValue.x.toInt - 8,
        particle.position2.getValue.y.toInt - 8,
        16, 16)
  }
}

lazy val top = newMainFrame {
  preferredSize = new Dimension(800, 400)
  contents = panel
}

Task 7 - RP

Question:

The following application draws two squares. Which of the following sentences is true?
Options:

- The squares are moving at the same speed
- The first square is moving at constant speed, the second at increasing speed
- The squares have a fixed position
- The first square is fixed, the second is moving

```scala
object Squares_Reactive extends SimpleSwingApplication {

  // — APPLICATION LOGIC —---------------

  object square1 {
    val position = Signal { Point(time().s * 100, 100) }
  }

  object square2 {
    val v = Signal { time().s * 100 }
    val position = Signal { Point(time().s * v(), 200) }
  }

  // painting components
  (square1.position.changed || square2.position.changed) += {
    _ => Swing.onEDT { top.repaint }
  }

  // — Graphic Stuff —---------------

  lazy val panel: RePanel = new RePanel {
    override def paintComponent(g: Graphics2D) {
      super.paintComponent(g)
      g.fillRect(square1.position.getValue.x.toInt - 8, square1.position.getValue.y.toInt - 8, 16, 16)
      g.fillRect(square2.position.getValue.x.toInt - 8, square2.position.getValue.y.toInt - 8, 16, 16)
    }
  }

  lazy val top = new MainFrame {
    preferredSize = new Dimension(800, 400)
    contents = panel
  }
}
```
Task 8 - RP

Question:

The following GUI application displays two buttons and three labels (1) (2) and (3) (Labels (1) and (2) are displayed on top of the two buttons), Which of the following is correct?

Options:

- (1) and (2) both display the number of clicks of the left button. (3) says if the number of clicks of the left button and the number of clicks of the right button are equal.

- (1) does not change over time (2) displays the number of clicks of the left button. (3) says if the number of clicks of the left button and the number of clicks of the right button are equal.

- (1) displays the number of clicks of the left button. (2) displays the number of clicks of the right button. (3) says if the number of clicks of the left button and the number of clicks of the right button are equal.

- (1) displays the number of clicks of the left button. (2) displays the number of clicks of the right button. (3) says if the number of clicks of the left button and the number of clicks of the right button are NOT equal.

```scala
object Interactive1-Reactive extends SimpleSwingApplication {
  val leftButton: ReButton = new ReButton(
    text = Signal { leftClicks() + " times clicked" },
    background = Color.RED)
  val rightButton: ReButton = new ReButton(
    text = Signal { rightClicks() + " times clicked" },
    background = Color.GREEN)
  lazy val leftClicks = leftButton.clicked.count
  lazy val rightClicks = rightButton.clicked.count

  val label = new ReLabel(
    text = Signal { if (rightClicks() == leftClicks()) "EQUALS" else "" })

  def top = new MainFrame {
    preferredSize = new Dimension(400, 150)
    contents = new GridPanel(2, 1) {
```

}}
contents += new GridPanel(1, 2) {
    contents += leftButton
    contents += rightButton
}  
contents += label

Task 9 - RP

Question:

In the application, a vertical line slides form left to right. Which of the following is true?

Options:

- The line becomes visible when the user clicks the mouse, remains visible for some time while sliding, then disappears.
- The line initially is visible and is sliding.
- After the user’s mouse click the line remains visible indefinitely.
- The line is initially vertical and starts rotating.

```
object LineReactive extends SimpleSwingApplication {
  val SPEED = 4
  val SIZE = 200

  // APPLICATION LOGIC

  object line {
    val t0: Signal[Time] = panel.mouse.clicks.clicked snapshot: time
    val progress = Signal { (time() since t0()).s / SPEED }
    val x = Signal { progress() * SIZE }
    val visible = panel.mouse.clicks.clicked.switchOnce(Signal[false], Signal[progress()<1])
```
// painting component
line.x.changed += { _ => Swing.onEDT { top.repaint } }

// Drawing stuff

lazy val panel: RePanel = new RePanel {
  override def paintComponent(g: Graphics2D) {
    super.paintComponent(g)
    g.drawString("Click somewhere here", 20, 20)
    if (line.visible.getValue)
      g.drawLine(line.x.getValue.toInt, 0, line.x.getValue.toInt, panel.bounds.getValue.height)
  }
}

lazy val top = newMainFrame {
  preferredSize = new Dimension(400, 400)
  contents = panel
}

Task 10 - RP

Question:

The application draws a square (box) on a canvas. When a certain sequence of events occurs, the color of the box changes from black (the color of the box when the application is started) to red. The sequence is:

Options:

- The user clicks once on the box
- The user drags the mouse staying inside the box
- The user drags the mouse across the box borders
- The user drags the mouse staying outside the box
object MouseSquare_Reactive extends SimpleSwingApplication {

class Box(val topLeft: Point, val bottomRight: Point) {
  def isInside(p: Point): Boolean =
  p.x > topLeft.x && p.y > topLeft.y &&
  p.x < bottomRight.x && p.y < bottomRight.y
}

// --- APPLICATION LOGIC --------------------------

val box = new Box(new Point(100, 100), new Point(200, 200))

val mousePosition: Signal[Point] = (panel.mouse.moves.moved || panel.mouse.moves.dragged) map
  ((_, MouseEvent).point) latest new Point(0, 0)

val isMousepressed: Signal[Boolean] = (panel.mouse.clicks.pressed || panel.mouse.clicks.released) toggle
  (Signal{false}, Signal{true})

val positionWhilePressing: Signal[Option[Point]] = Signal {
  if(isMousepressed())
    Some(mousePosition())
  else
    None
}

val overlapping: Signal[Option[Boolean]] = Signal {
  positionWhilePressing().map(box.isInside(_))
}

val pressingChange: Signal[(Int, Int)] =
  overlapping.changed.fold((0, 0)) {
    (_, _) match {
      case ((count, acc), Some(_)) => (count, acc +1)
      case ((count, acc), None) => (acc, 0)
    }
  }

val color: Signal[Color] = Signal {
  pressingChange() match {
    case (count, acc) if count == 3 => Color.RED
    case _ => Color.BLACK
  }
}

// painting component

// ---- GRAPHIC STUFF --------------------------

lazy val panel: RePanel = new RePanel {
  override def paintComponent(g: Graphics2D) {
    super.paintComponent(g)
    g.setColor(color.getValue)
    g.fillRect(box.topLeft.x, box.topLeft.y, box.bottomRight.x - box.topLeft.x, box.bottomRight.y - box.topLeft.y)
  }
}

lazy val top = new MainFrame {

}
preferredSize = new Dimension(400, 400)
contents = panel
}
}

Task 1 - OO

Question:

What is the final value of k after the execution of the following code

Options:

- 0
- false
- 3
- true

val a = Observable[Int](0)
val b = Observable[Int](0)
val c = Observable[Int](0)
val d = Observable[Int](0)
val e = Observable[Boolean](false)
val k = Observable[Boolean](false)

a.addObserver(
  x => c.set(2 * x)
)

a.addObserver(
  x => d.set(3 * x)
)

b.addObserver(
  x => e.set(d.get == b.get)
)

d.addObserver(
  x => e.set(d.get == b.get)
)

e.addObserver(
  x => k.set(e.get || false)
)

a.set(1)
b.set(1)
b.set(3)
// value of k here ?

Task 2 - OO

Question:
Which property is always true (invariant) during the execution of the following application?

Options:

- c contains the value 0
- a is updated after d
- if d is updated both c and b have been already updated
- a is updated after c

```kotlin
val a = Observable[Int](0)
val b = Observable[Int](0)
val c = Observable[Int](0)
val d = Observable[Int](0)

a.addObserver{
  x => b.set(x + 1)
}
a.addObserver{
  x => c.set(2 * x)
}

var secondEvent = false
b.addObserver{
  x =>
    if (secondEvent) {
      d.set(b.get + c.get)
      secondEvent = false
    } else
      secondEvent = true
}
c.addObserver{
  x =>
    if (secondEvent) {
      d.set(b.get + c.get)
      secondEvent = false
    } else
      secondEvent = true
}
a.set(3)
```

Task 3 - OO

Question:

What is the sequence of values held by y during the execution of the program? (Do not count repeated contiguous values, for example, 1,2,2,3 counts as 1,2,3,2)

Options:

- 1,2,3
• 0.2
• 0.2,4
• 0.1,2

```kotlin
val a = Observable[Int](0)
val b = Observable[Int](0)
val c = Observable[Int](a.get + b.get)
val d = Observable[Int](2 * a.get)
val e = Observable[Boolean](d.get == b.get)
val y = Observable[Int](if (e.get) c.get else d.get)

a.addObserver({
  x => c.set(a.get + b.get)
})
a.addObserver({
  x => d.set(2 * x)
})
b.addObserver({
  x => c.set(a.get + b.get)
})
b.addObserver({
  x => e.set(d.get == b.get)
})
d.addObserver({
  x => e.set(d.get == b.get)
})
c.addObserver({
  x =>
    if (e.get)
      y.set(c.get)
    else
      y.set(d.get)
})
e.addObserver({
  x =>
    if (!e.get)
      y.set(d.get)
    else
      y.set(if (e.get) c.get else d.get)
})
a.set(1)
b.set(1)
a.set(2)
```

**Task 4 - OO**

**Question:**

Which variables can potentially change after a has been updated?

**Options:**

- 0.2
- 0.2,4
- 0.1,2
Task 5 - OO

Question:

The application displays four particles, i.e. four black circles on the screen. Which of the following sentences is correct for the application?

Options:

- The particles do not change over time.
• The particles become ellipses, i.e. they become flatter over time.

• The particles periodically get bigger and then small again.

• The particles increase in size indefinitely.

```java
import commons._
import swing.{ Panel, MainFrame, SimpleSwingApplication }
import swing.Swing
import java.awt.{ Color, Graphics2D, Dimension }
import math.{ sin, cos, Pi }
import Animation._

object Particles1Observer extends SimpleSwingApplication {

  class Particle(
    val center: Point,
    val size1: Double,
    val size2: Double,
    val interval: Time)
  {
    val time1 = Observable { now }
    val progress = Observable { 0.0 }
    val normalSize = Observable { 0.0 }
    val size = Observable { 0.0 }
    addTimeChangedHandler { time =>
      time1 set (time % interval)
    }
    time1 addObserver { time =>
      progress set (time / interval)
    }
    progress addObserver { progress =>
      normalSize set (2 * (if (progress < 0.5) progress else 1 - progress))
    }
    normalSize addObserver { normalSize =>
      size set (size1 + (size2 - size1) * normalSize)
    }
  }

  val particle0 = new Particle(Point(100, 100), 20, 50, 2.0.s)
  val particle1 = new Particle(Point(200, 100), 5, 20, 1.8.s)
  val particle2 = new Particle(Point(100, 200), 10, 30, 2.1.s)
  val particle3 = new Particle(Point(200, 200), 15, 40, 1.5.s)

  // painting component
  particle0.size addObserver { _ => repaint }
  particle1.size addObserver { _ => repaint }
  particle2.size addObserver { _ => repaint }
  particle3.size addObserver { _ => repaint }
  def repaint = Swing onEDT { top.repaint }
```
lazy val panel: Panel = new Panel {
  override def paintComponent(g: Graphics2D) {
    super.paintComponent(g)
    g.fillOval(
      (particle.center.x - particle.size.get / 2).toInt,
      (particle.center.y - particle.size.get / 2).toInt,
      particle.size.get.toInt,
      particle.size.get.toInt)
  }
}

lazy val top = new MainFrame {
  preferredSize = new Dimension(400, 400)
  contents = panel
}

Task 6 - OO

Question:

The particles in the following applications are moving. They:

Options:

- Circle around the same center
- Circle around different centers on the same horizontal line
- Get bigger and smaller regularly
- Move horizontally form left to right and from right to left at regular intervals

object Particles2Observer extends SimpleSwingApplication {
  // — APPLICATION LOGIC ——
  class Particle(val center: Point, val radius: Double, val period: Time) {
    val time1 = Observable { now }
    val angle = Observable { 0.0 }
    val position1 = Observable { Point(0, 0) }
    val position2 = Observable { Point(0, 0) }

    // APPLICATION LOGIC
```scala
addTimeChangedHandler { time =>
  time1 set (time % period)
}
time1.addObserver { time =>
  angle set (2 * Pi * (time / period))
}
angle.addObserver { angle =>
  position1 set Point(cos(angle), sin(angle))
}
position1.addObserver { pos =>
  position2 set pos * radius + center
}
val particle0 = new Particle(Point(100, 150), 40, 1.5.s)
val particle1 = new Particle(Point(300, 150), 50, 1.3.s)
val particle2 = new Particle(Point(500, 150), 45, 1.7.s)
val particle3 = new Particle(Point(700, 150), 30, 1.2.s)
// painting components
particle0.position2 addObserver { _ => repaint }
particle1.position2 addObserver { _ => repaint }
particle2.position2 addObserver { _ => repaint }
particle3.position2 addObserver { _ => repaint }
def repaint = Swing onEDT { top.repaint }
// — Draw the particles ————————————————————————
lazy val panel: Panel = new Panel {
  override def paintComponent(g: Graphics2D) {
    super.paintComponent(g)
    for (particle <- Seq(particle0, particle1, particle2, particle3))
      g.fillOval(particle.position2.get.x.toInt - 8,
                 particle.position2.get.y.toInt - 8,
                 16, 16)
  }
}
lazy val top = new MainFrame {
  preferredSize = new Dimension(800, 400)
  contents = panel
}
```

**Task 7 - OO**

**Question:**

The following application draws two squares. Which of the following sentences is true?

**Options:**

- The squares are moving at the same speed
• The first square is moving at constant speed, the second at increasing speed
• The squares have a fixed position
• The first square is fixed, the second is moving

```scala
import commons._
import swing.{ Panel, MainFrame, SimpleSwingApplication }  
import swing.Swing
import java.awt.{ Color, Graphics2D, Dimension }
import Animation._

object SquaresObserver extends SimpleSwingApplication {

  // — APPLICATION LOGIC ————

  object square1 {
    val position = Observable { Point(0, 0) }
    addTimeChangedHandler { time =>
      position set Point(time.s * 100, 100)
    }
  }

  object square2 {
    val v = Observable { 0.0 }
    val position = Observable { Point(0, 0) }
    addTimeChangedHandler { time =>
      v set time.s * 100
      updatePos(time, v.get)
    }
    v addObserver { v =>
      updatePos(now, v)
    }
    def updatePos(time: Time, v: Double) {
      position set Point(time.s * v, 200)
    }
  }

  // painting components
  square1.position addObserver { _ => repaint }
  square2.position addObserver { _ => repaint }
  def repaint = Swing onEDT { top.repaint }

  // — Graphic Stuff ————

  lazy val panel: Panel = new Panel {
    override def paintComponent(g: Graphics2D) {
      super.paintComponent(g)
      g.fillRect(square1.position.get.x.toInt - 8,
                 square1.position.get.y.toInt - 8,
                 16, 16)
      g.fillRect(square2.position.get.x.toInt - 8,
                 square2.position.get.y.toInt - 8,
                 16, 16)
    }
  }
}
lazy val top = new MainFrame {
  preferredSize = new Dimension(800, 400)
  contents = panel
}

Task 8 - OO

Question:

This GUI application displays two buttons and three labels (1) (2) and (3) (Labels (1) and (2) are displayed on top of the two buttons). Which of the following is correct?:

Options:

- (1) and (2) both display the number of clicks of the left button. (3) says if the number of clicks of the left button and the number of clicks of the right button are equal.

- (1) does not change over time (2) displays the number of clicks of the left button. (3) says if the number of clicks of the left button and the number of clicks of the right button are equal.

- (1) displays the number of clicks of the left button. (2) displays the number of clicks of the right button. (3) says if the number of clicks of the left button and the number of clicks of the right button are equal.

- (1) displays the number of clicks of the left button. (2) displays the number of clicks of the right button. (3) says if the number of clicks of the left button and the number of clicks of the right button are NOT equal.
var rightClicks = 0
rightButton.reactions += {
case ButtonClicked(_,) =>
  rightClicks += 1
  updateLabels
}
def updateLabels {
  leftButton.text = leftClicks + " times clicked"
  rightButton.text = rightClicks + " times clicked"
  label.text = if (rightClicks == leftClicks) "EQUALS" else ""
}
val label = new Label
updateLabels

// — GUI stuff ————————————————————
def top = new MainFrame {
  preferredSize = new Dimension(400, 150)
  contents = new GridPanel(2, 1) {
    contents += new GridPanel(1, 2) {
      contents += leftButton
      contents += rightButton
    }
    contents += label
  }
}

Task 9 - OO

Question:

In the application, a vertical line slides form left to right. Which of the following is true?

Options:

- The line becomes visible when the user clicks the mouse, remains visible for some time while sliding, then disappears.
- The line initially is visible and is sliding.
- After the user’s mouse click the line remains visible indefinitely.
- The line is initially vertical and starts rotating.

object LineObserver extends SimpleSwingApplication {
  val SPEED = 4
val SIZE = 200

// — APPLICATION LOGIC ———————————

panel.mouse.clicks.listenTo(panel.mouse.clicks)

object line {
  val t0 = Observable { now }
  val progress = Observable { 0.0 }
  val x = Observable { 0.0 }
  val visible = Observable { false }

  panel.mouse.clicks.reactions += {
    caseMouseClicked(_, _, _, _) => 
      t0 set now
      updateProgress(now)
  }

  addTimeChangedHandler { time =>
    updateProgress(time)
  }

  def updateProgress(time: Time) {
    progress set ((time since t0.get).s / SPEED)
  }

  progress addObserver {
    progress =>
      x set (progress * SIZE)
      visible set (progress < 1)
  }
}

// painting component
line.x addObserver { _ => Swing.onEDT { top.repaint } }

// — Drawing stuff ———————————

lazy val panel: Panel = new Panel {
  override def paintComponent(g: Graphics2D) {
    super.paintComponent(g)
    g.drawString("Click somewhere here", 20, 20)
    if (line.visible.get)
      g.drawLine(
        line.x.get.toInt,
        0,
        line.x.get.toInt,
        panel.bounds.height)
  }
}

lazy val top = new MainFrame {
  preferredSize = new Dimension(400, 400)
  contents = panel
}

Task 10 - OO

Question:
The application draws a square (box) on a canvas. When a certain sequence of events occurs, the color of the box changes from black (the color of the box when the application is started) to red. The sequence is:

**Options:**

- The user clicks once on the box
- The user drags the mouse staying inside the box
- The user drags the mouse across the box borders
- The user drags the mouse staying outside the box

```java
import commons._
import java.awt.Color
import scala.swing.SimpleButton
import scala.swing.ColorBox
import scala.swing.MainFrame
import scala.swing.event.{MouseMoved, MouseDragged, MousePressed, MouseReleased}
import scala.swing.Reactor

object MouseSquareObserver extends SimpleSwingApplication {
  class Box(val topLeft: Point, val bottomRight: Point) {
    def isInside(p: Point): Boolean = 
      p.x > topLeft.x && p.y > topLeft.y && 
      p.x < bottomRight.x && p.y < bottomRight.y
  }

  val box = new Box(new Point(100, 100), new Point(200, 200))
  val mousePosition: Observable[Point] = Observable(new Point(0, 0))
  val isMousepressed: Observable[Boolean] = Observable(false)

  new Reactor {
    reactions += {
      case MouseMoved(_, point, _) => 
        mousePosition.set(point)
      case MouseDragged(_, point, _) => 
        mousePosition.set(point)
      case MousePressed(_, _, _, _) => 
        isMousepressed.set(true)
      case MouseReleased(_, _, _, _) => 
        isMousepressed.set(false)
    }
    listenTo(panel.mouse.moves, panel.mouse.clicks)
  }

  mousePosition.addObserver(updatePointWhilePressing)
  isMousepressed.addObserver(updatePointWhilePressing)

  val positionWhilePressing = Observable(Option.empty[Point])
  def updatePointWhilePressing(v: Any) {
```
positionWhilePressing.set(
  if(isMousepressed.get)
    Some(mousePosition.get)
  else
    None
)

val overlapping = Observable(Option.empty[Boolean])
positionWhilePressing.addObserver {_ =>
  val newIsPressingOverlappingBox = positionWhilePressing.get.map(box.isInside(_))
  if (newIsPressingOverlappingBox != overlapping.get)
    overlapping.set(newIsPressingOverlappingBox)
}

val pressingChange = Observable((0, 0))
overlapping.addObserver {_ =>
  val (count, acc) = pressingChange.get
  pressingChange.set(
    overlapping.get match {
      case Some(_) => (count, acc +1)
      case None => (acc, 0)
    }
  )
}

val color = Observable(Color.BLACK)
pressingChange.addObserver {_ =>
  color.set(
    pressingChange.get match {
      case (count, acc) if count == 3 => Color.RED
      case _ => Color.BLACK
    }
  )
}

color.addObserver {_ => top.repaint }

// --- Graphic stuff ---------------------

lazy val panel: Panel = new Panel {
  override def paintComponent(g: Graphics2D) {
    super.paintComponent(g)
    g.setColor(color.get)
    g.fillRect(box.topLeft.x, box.topLeft.y, box.bottomRight.x - box.topLeft.x, box.bottomRight.y - box.topLeft.y)
  }
}

lazy val top = new MainFrame {
  preferredSize = new Dimension(400, 400)
  contents = panel
}

23
3 Preliminary Questions

Question 1

Question:

What is the value of v at the end of the execution of the program?

Options:

• 0
• 3
• 4
• 5
• Don’t know

```scala
class A {
  def foo() = { 3 }
}
class B extends A {
  override def foo() = { 4 }
}
class C extends B {
  def bar() = { 5 }
}
val o = new C()
val v = o.foo()
```

Question 2

Question:

What are the values of v1 and v2 at the end of the execution of the program?

Options:

• 0,0
• 3,3
• 4,4
• 3,4
• Don’t know

```scala
class A {
  def foo() = { 3 }
}
class B extends A {
  override def foo() = { 4 }
}
```
val o1: A = new A()
val o2: A = new B()
def bar(o: A) = {
o.foo()
}
val v1 = bar(o1)
val v2 = bar(o2)

**Question 3**

**Question:**

What is the value of v at the end of the execution of the program?

**Options:**

- 1
- 2
- 3
- 6
- Don’t know

class A {
def foo() = { 1 }
}
class B extends A {
  override def foo() = { 2 + super.foo() }
}
class C extends B {
  override def foo() = { 3 + super.foo() }
}
val o = new C()
val v = o.foo()

**Question 4**

**Question:**

What is the value of v at the end of the execution of the program?

**Options:**

- AC
- BA
- BCA
- CBA

25
• Don’t know

class A {
  def foo() = { "A" }
}

trait B extends A {
  override def foo() = { "B" + super.foo() }
}

trait C extends A {
  override def foo() = { "C" + super.foo() }
}

val o = new A() with C with B
val v = o.foo()

Question 5
Question:
What is the value of v at the end of the execution of the program?

Options:
• No value. The compilation fails due to an error at line YYY
• A
• Animal
• No value. The compilation fails due to an error at line XXX
• Don’t know

trait Printable[A] {
  def print(): String
}

class Animal { // YYY
  def getName() = { "Animal" }
}

implicit def impl(p: Animal) = new Animal() with Printable[Animal] {
  def print() = { getName() }
}

def toOutput[A](obj: Printable[A]): String = {
  obj.print()
}

val v = toOutput(new Animal()) // XXX

Question 6
Question:
Does this program compile?

Options:
• The compilation fails due to an error in line YYY
• It compiles
• The compilation fails due to errors in both lines XXX and YYY
• The compilation fails due to an error in line XXX
• Don’t know

```scala
class ContainerSimple[A] {}
class ContainerPlus[A] {}
class Father
class Son extends Father
val c1 = new ContainerSimple[Son]()
val c2 = new ContainerPlus[Son]()
val v1: ContainerSimple[Father] = c1 // YYY
val v2: ContainerPlus[Father] = c2 // XXX
```

**Question 7**

**Question:**

What is the value of v at the end of the execution of the program?

**Options:**

• 0
• 21
• 11
• 10
• Don’t know

```scala
var v = 0
val a = new ImperativeEvent[Int]()
a(10)
a += { (x: Int) => v = v + x }
a(11)
```

**Question 8**

**Question:**

What is the value of v at the end of the execution of the program?

**Options:**

• 0
val v = 0
val e1 = new ImperativeEvent[Int]()
val e2 = new ImperativeEvent[Int]()
val e1 OR e2 = e1 || e2
  e1 OR e2 += ((x: Int) => v = v + x)
e1(10)
e2(11)

Question 9

Question:

What is the TYPE of s in the program?

Options:

- Signal[Int]
- Signal[List[String]]
- Signal[String]
- Signal[Int => String]
- Don’t know

val a = new ImperativeEvent[Int]()
val s = a.fold("")((acc, x) => (acc + x.toString))

Question 10

Question:

What is the TYPE of t in the program?

Options:

- Event[Unit]
- Event[(Int,Int)]
- Signal[Int]
- Signal[Unit]
- Don’t know

val a = Var(1)
val b = Var(2)
val s = Signal { a() + b() }
val t = s.change
**Question 11**

**Question:**

What are the values of t1, t2, t3 at the end of the program?

**Options:**

- 1,1,1
- 1,2,1
- 1,2,2
- 10,10,10
- Don’t know

```scala
val e = new ImperativeEvent[Int]()
val s: Signal[Int] = e.latest(10)
  e(1)
val t1 = s.getValue
  e(2)
val t2 = s.getValue
  e(1)
val t3 = s.getValue
```

**Question 12**

**Question:**

What are the values of t1 and t2 at the end of the program?

**Options:**

- 1,2
- 2,2
- 2,3
- 3,3
- Don’t know

```scala
val e = new ImperativeEvent[Int]()
val v = Var(1)
val s1 = Signal { v() + 1 }
val s = e.snapshot(s1)
e(1)
v.setVal(2)
val t1 = s.getValue
  e(1)
val t2 = s.getValue
```

29
**Question 13**

**Question:**

What is the type of f in the program?

**Options:**

- List[String]
- Int => List[String]
- String => List[String]
- String => List
- Don’t know

```scala
val a = (x: Int) => x.toString
val b = (x: String) => { List(x) }
val f = { (x: Int) => b(a(x)) }
```

**Question 14**

**Question:**

What is the value of s at the end of the execution of this program?

**Options:**

- false
- 10
- 0
- 4
- Don’t know

```scala
val list = List(1, 2, 4, 3)
val s = list.fold(0)((acc, x) => acc + x)
```

**Question 15**

**Question:**

What is the TYPE of s in this program?

**Options:**

- List[Unit]
- List[Int]
• Wp
• List[Wp]
• Don’t know

```scala
class Wp(i: Int)
val list = List(1, 2, 4, 3)
val s = list.map(new Wp(_))
```

**Question 16**

**Question:**
What is the value of a at the end of the execution of the program?

**Options:**
• 0
• 1
• 2
• 3
• Don’t know

```scala
def check[T](list: List[T]): Int = list match {
  case Nil => 0
  case x :: xs => 1 + check(xs)
}
val l = List(1, 3, 2)
val a = check(l)
```

**Question 17**

**Question:**
Currying: What is the type of f1 in the program?

**Options:**
• (Int,Int) => Int
• (Int => Int) => Int
• Int => Int
• Int => (Int => Int)
• Don’t know

```scala
def add(x: Int)(y: Int)(z: Int) = x + y + z
val f1 = add(5)
```
Question 18

Question:

What is the type of `z` in the following program?

Options:

- `List[Int]`
- `List[(Int,Int)]`
- `Int`
- `List[(Int => Int)]`
- `List[Int => Int]`

```scala
val l1 = List(1, 2, 3)
val l2 = List(10, 11, 12)
val z = l1.zip(l2)
```