

ReSwing – A Reactive Interface for Scala Swing



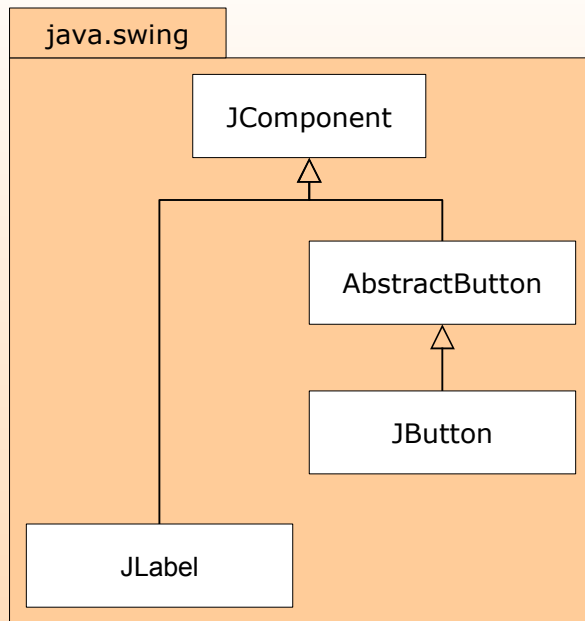
TECHNISCHE
UNIVERSITÄT
DARMSTADT

Pascal Weisenburger

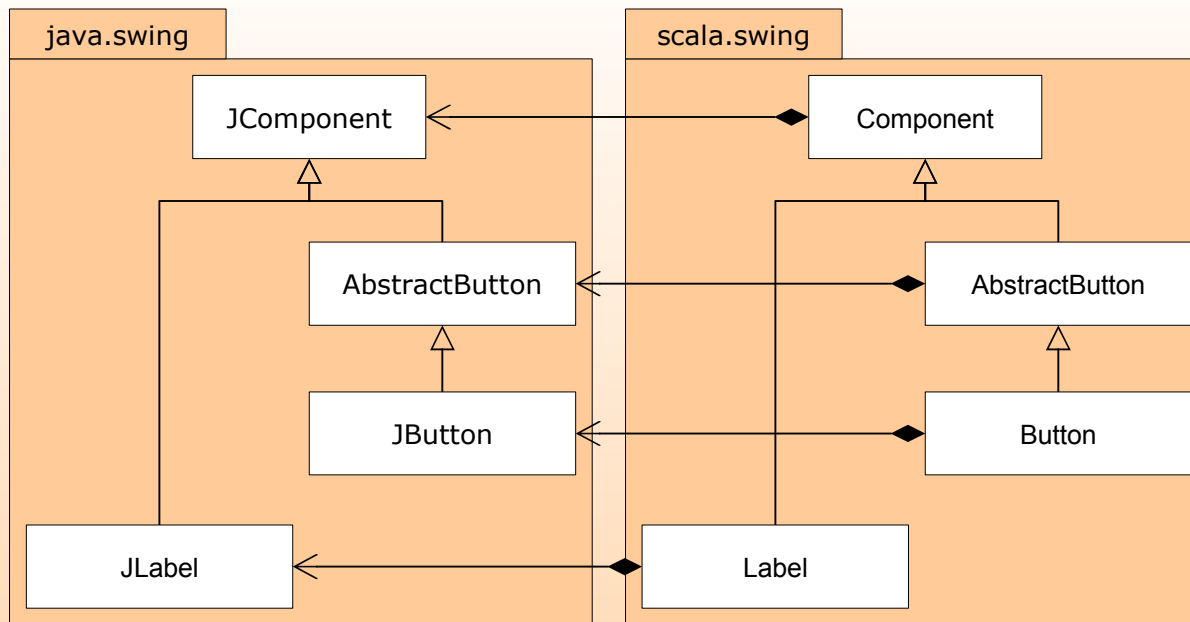
Java Swing / Scala Swing / ReSwing



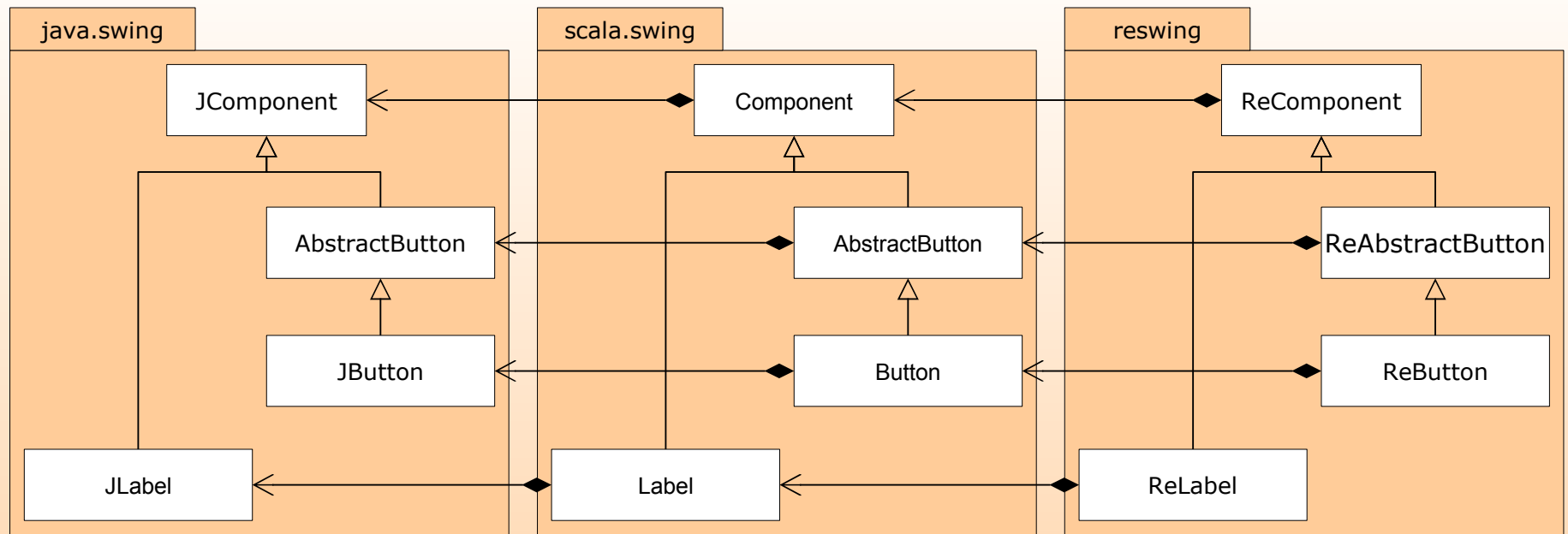
TECHNISCHE
UNIVERSITÄT
DARMSTADT



Java Swing / Scala Swing / ReSwing



Java Swing / Scala Swing / ReSwing



Swing vs ReSwing



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Swing

```
val label = new Label  
label.text = "foobar"  
label.preferredSize = new Dimension(400, 40)
```

Swing vs ReSwing



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Swing

```
val label = new JLabel  
label.text = "foobar"  
label.preferredSize = new Dimension(400, 40)
```

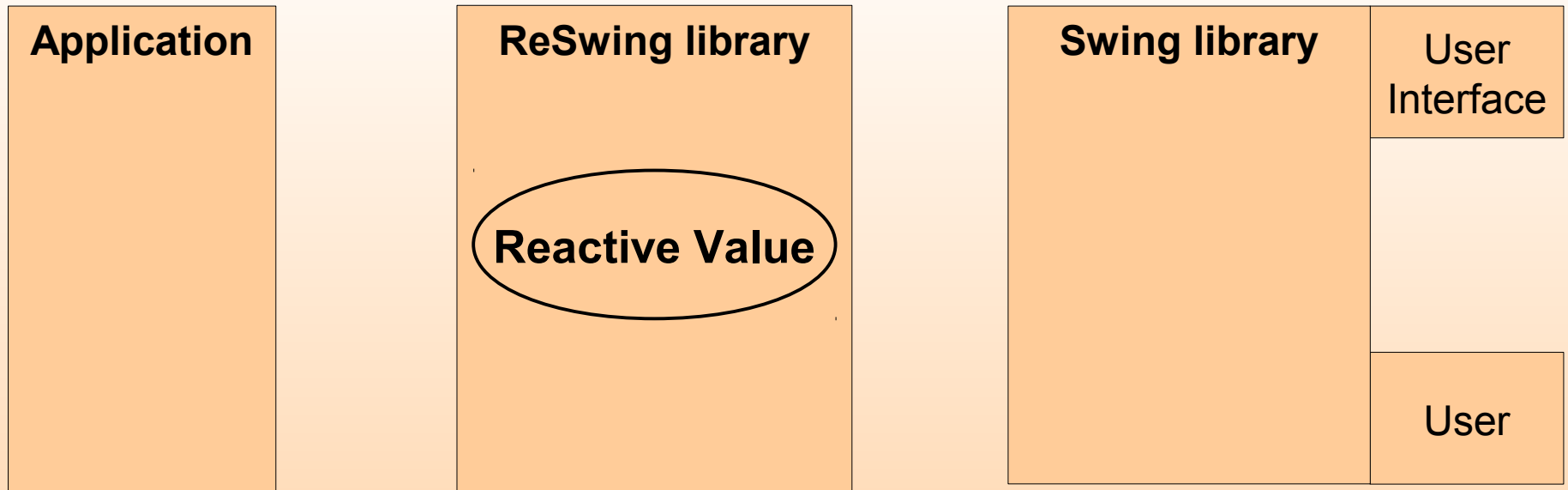
ReSwing

```
val label = new ReLabel(  
    text = "foobar"  
    preferredSize = new Dimension(400, 40)  
)
```

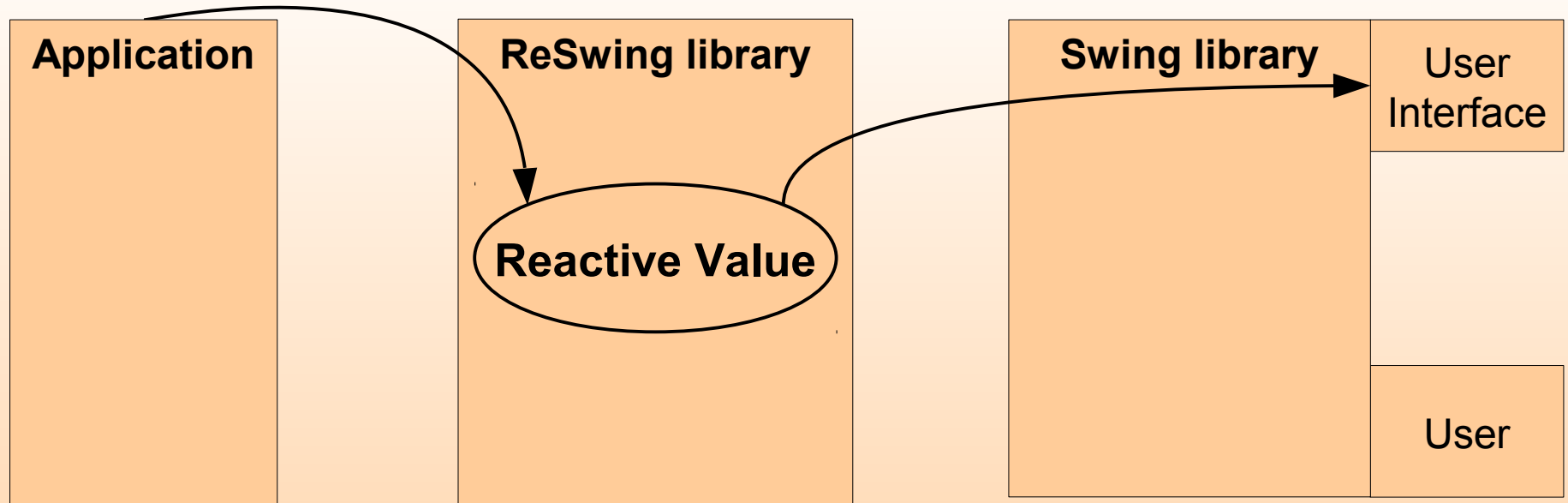
ReSwing Reactive Values



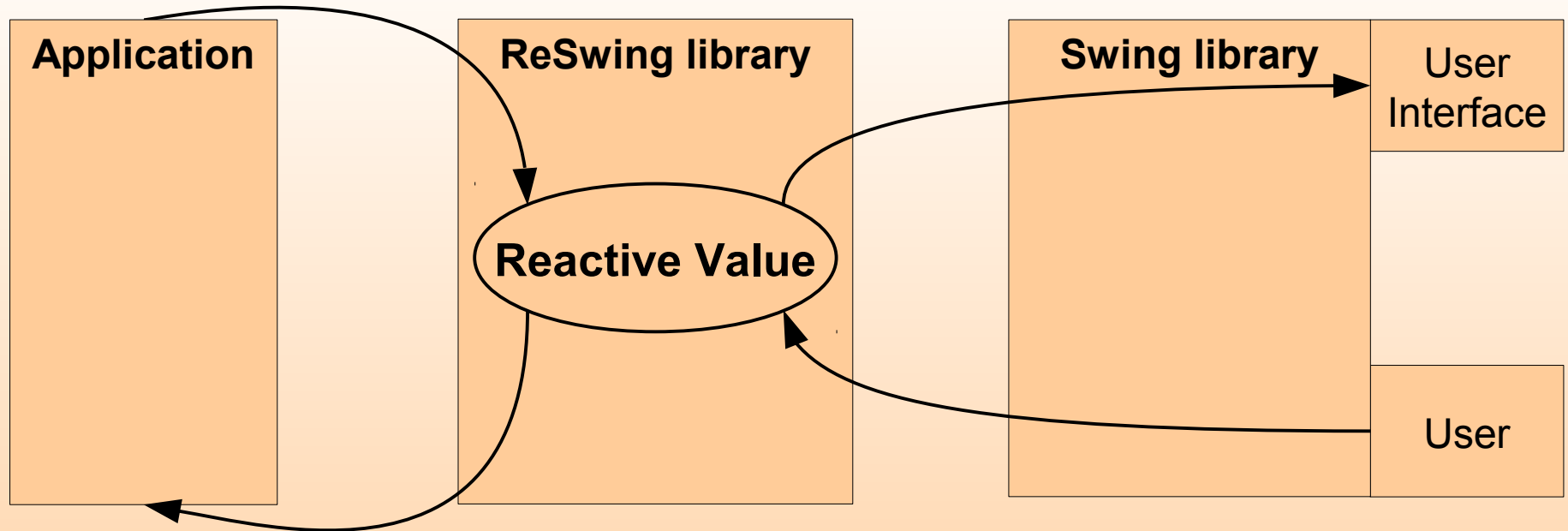
TECHNISCHE
UNIVERSITÄT
DARMSTADT



ReSwing Reactive Values



ReSwing Reactive Values



Setting Signals



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
val value: String
val label = new ReTextArea(
    text = value
)
```

```
val event: Event[String]
val label = new ReTextArea(
    text = event
)
```

```
val signal: Signal[String]
val label = new ReTextArea(
    text = signal
)
```

Setting Signals

```
val value: String
val label = new ReTextArea(
    text = value
)
```

- initializes the reactive value
- user can change the value

```
val event: Event[String]
val label = new ReTextArea(
    text = event
)
```

```
val signal: Signal[String]
val label = new ReTextArea(
    text = signal
)
```

Setting Signals

```
val value: String
val label = new ReTextArea(
    text = value
)
```

- initializes the reactive value
- user can change the value

```
val event: Event[String]
val label = new ReTextArea(
    text = event
)
```

- updates the reactive value on each event occurrence
- user can change the value

```
val signal: Signal[String]
val label = new ReTextArea(
    text = signal
)
```

Setting Signals

```
val value: String
val label = new ReTextArea(
    text = value
)
```

- initializes the reactive value
- user can change the value

```
val event: Event[String]
val label = new ReTextArea(
    text = event
)
```

- updates the reactive value on each event occurrence
- user can change the value

```
val signal: Signal[String]
val label = new ReTextArea(
    text = signal
)
```

- the signal defines the reactive value
- user *cannot* change the value

Defining Signals



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
class ReLabel(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
  
}
```

Defining Signals



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
class ReLabel(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
    text using (peer.text _, peer.text_= _, "text")  
  }
```

Defining Signals



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
class ReLabel(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
    text using (peer.text _, peer.text_= _, "text")  
  }
```

```
class ReTextComponent(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
    text using (peer.text _, peer.text_= _, classOf[ValueChanged])  
  }
```


Defining Signals



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
class ReLabel(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
    text using (peer.text _, peer.text_ = _, "text")  
  }
```

```
class ReTextComponent(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
    (text using (peer.text _, peer.text_ = _, classOf[ValueChanged]))  
      force ("editable", peer.editable_ = _, false)  
  }
```

Defining Signals



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
class ReLabel(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
    text using (peer.text _, peer.text_ = _, "text")  
  }
```

```
class ReTextComponent(val text: ReSwingValue[String] = ())  
  extends ReComponent {  
    (text using (peer.text _, peer.text_ = _, classOf[ValueChanged]))  
      force ("editable", peer.editable_ = _, false))  
  }
```

```
abstract class ReComponent extends ReUIElement {  
  val hasFocus: ReSwingValue[Boolean] = ()  
  hasFocus using (peer.hasFocus _, classOf[FocusGained],  
                  classOf[FocusLost])  
}
```

Use Case



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
object ApplicationSwingTextArea extends SimpleSwingApplication {
  val textArea = new ReTextArea("Lorem ipsum dolor sit amet")

  val selectionLabel = new ReLabel(
    text = Signal { "Selection Length: " +
      (if (textArea.selected() != null)
        textArea.selected().length else 0) })

  val countLabel = new ReLabel(
    text = Signal { "Character Count: " + textArea.text().length })

  def top = new ReMainFrame(
    preferredSize = new Dimension(400, 400),
    contents = new BorderPanel {
      layout(new JScrollPane(textArea)) = Position.Center
      layout(new GridPanel(1, 0) {
        contents += selectionLabel
        contents += countLabel
      }) = Position.South
    }
  )
}
```