

Vorlesung Softwaretechnik I

Übung 4

SWT I – Sommersemester 2019

Walter F. Tichy, Sebastian Weigelt, Tobias Hey

IPD Tichy, Fakultät für Informatik

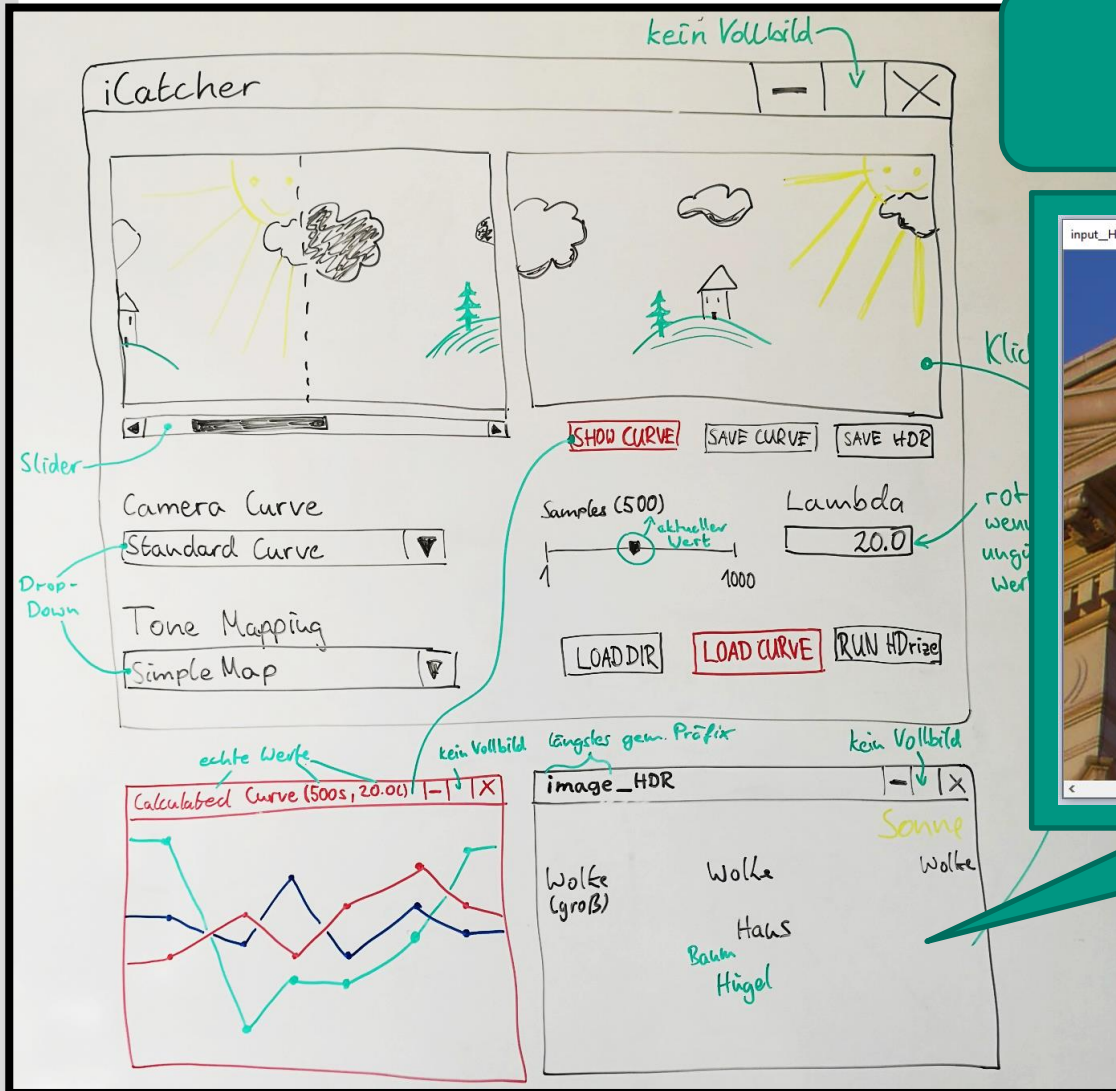


Grafische Benutzeroberfläche für iMage

AUFGABE 1

Aufgabe 1

GB0 für iMage „iCatcher“



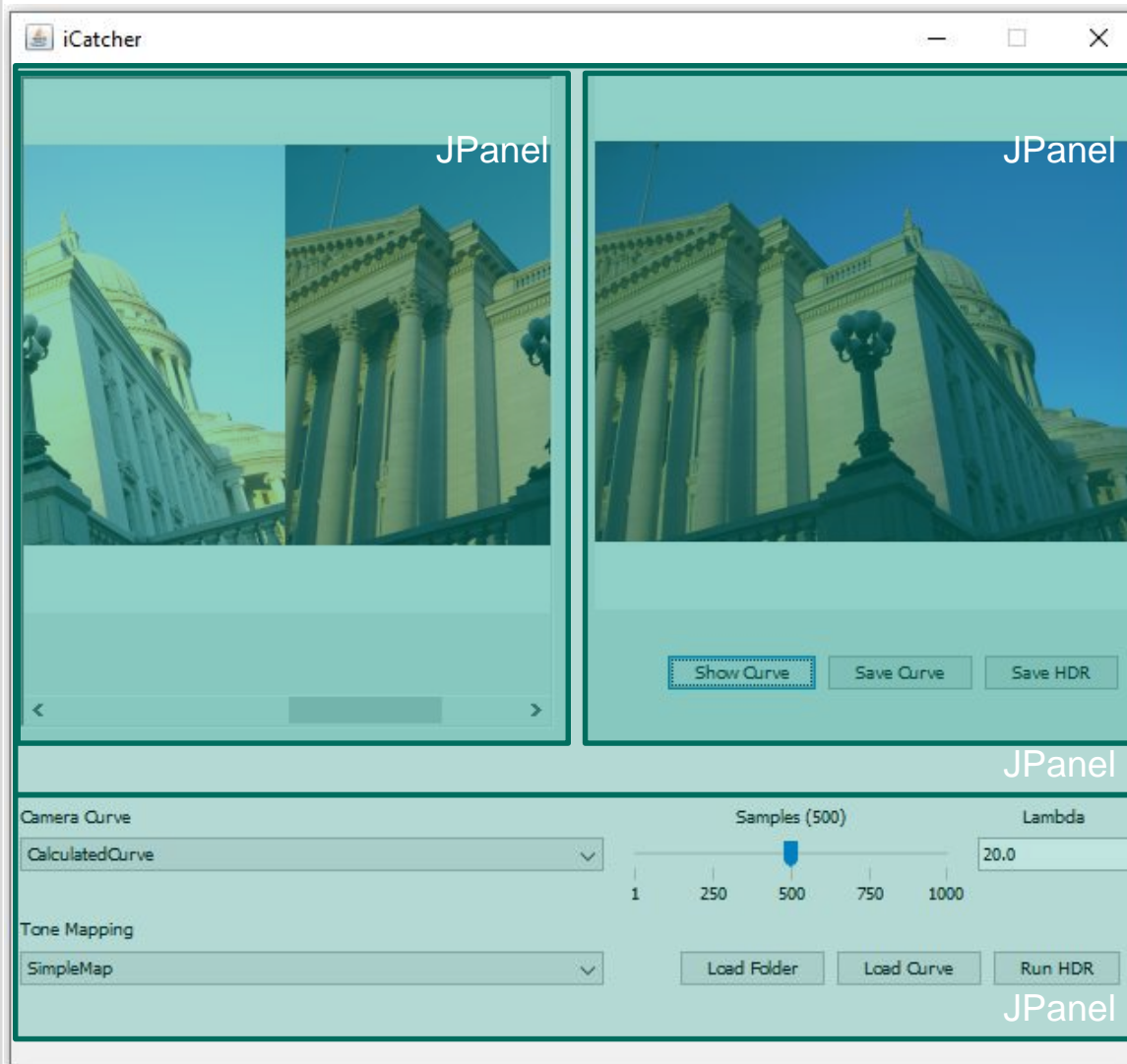
JDialog



JScrollPane

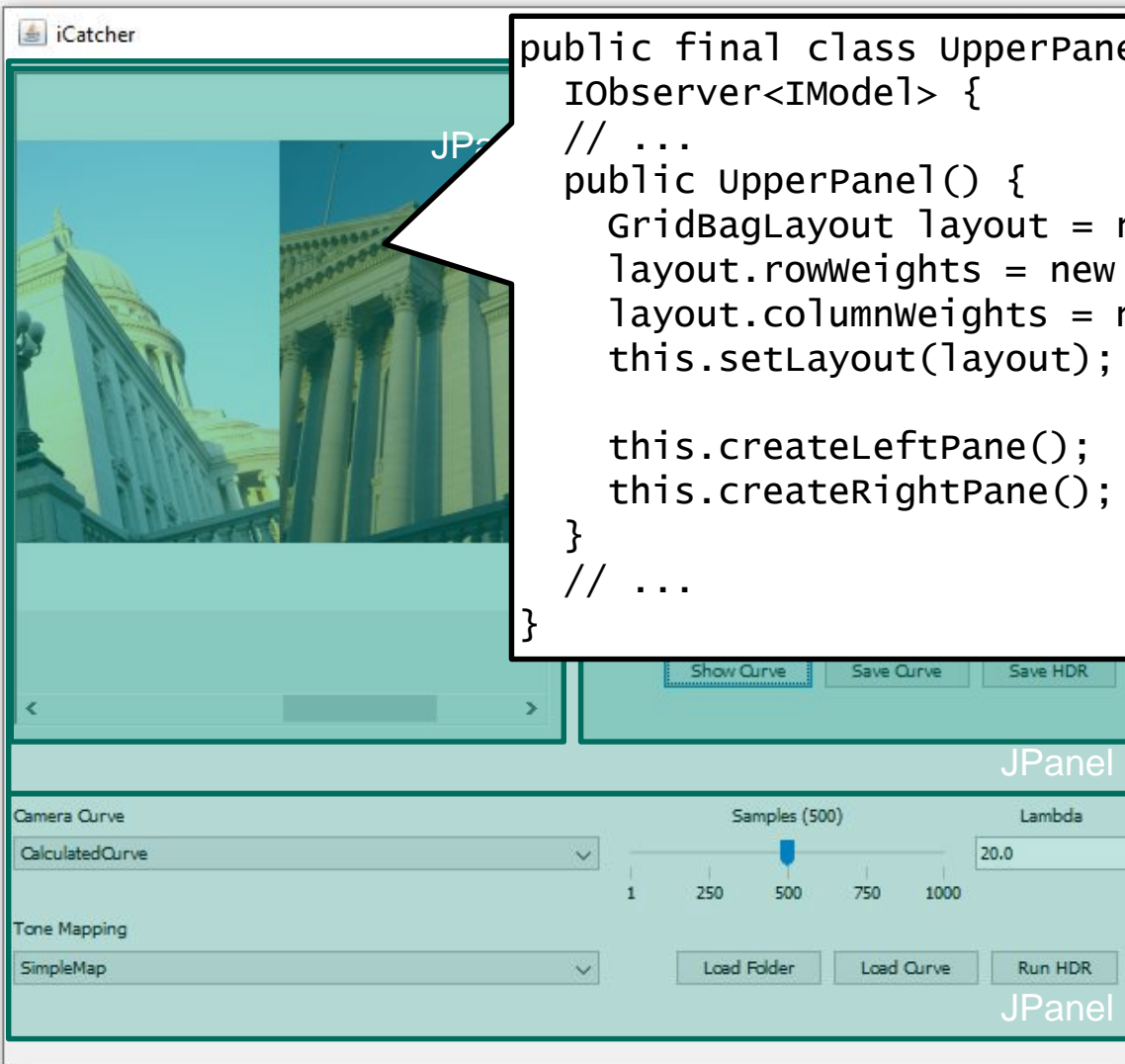
Aufgabe 1

GBO für iMage „iCatcher” – Aufbau



Aufgabe 1

GBO für iMage „iCatcher” – Aufbau

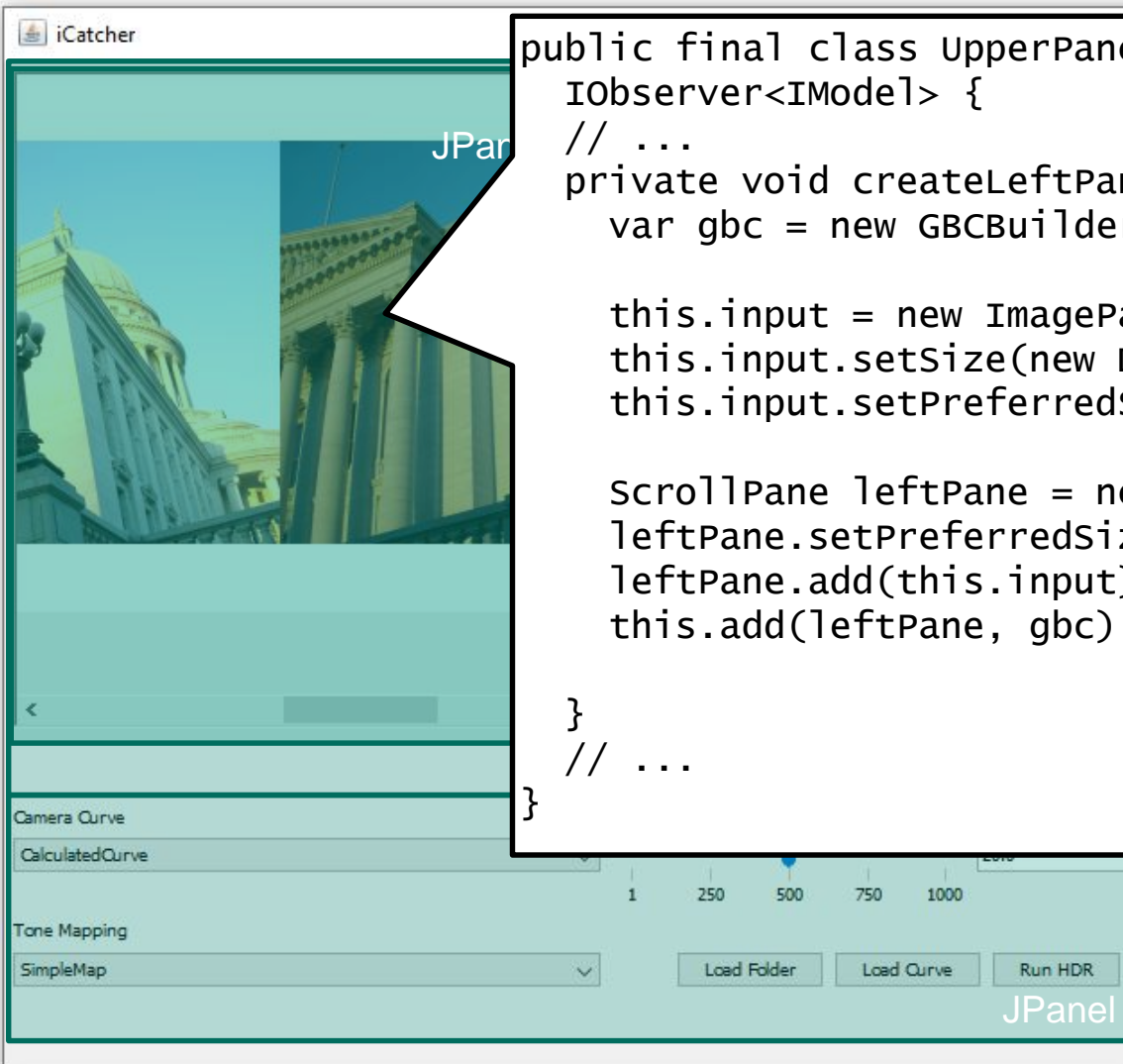


```
public final class UpperPanel extends JPanel implements
IObserver<IModel> {
    // ...
    public UpperPanel() {
        GridBagLayout layout = new GridBagLayout();
        layout.rowWeights = new double[] { 1.0 };
        layout.columnWeights = new double[] { 1.0, 1.0 };
        this.setLayout(layout);

        this.createLeftPane();
        this.createRightPane();
    }
    // ...
}
```


Aufgabe 1

GBO für iMage „iCatcher” – Aufbau



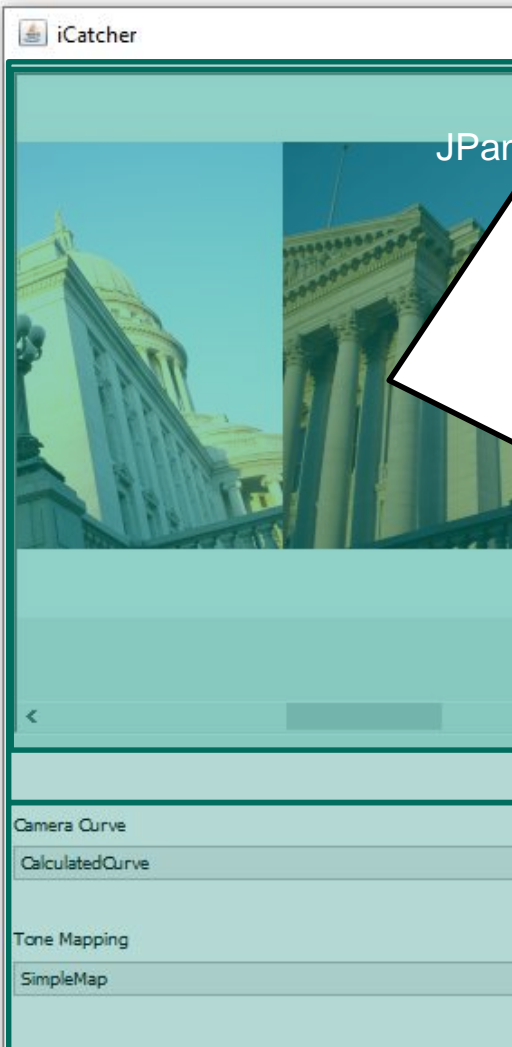
```
public final class UpperPanel extends JPanel implements
    IObservable<IModel> {
    // ...
    private void createLeftPane() {
        var gbc = new GridBagConstraints().anchor(...);

        this.input = new ImagePanel(null);
        this.input.setSize(new Dimension(300, 280));
        this.input.setPreferredSize(new Dimension(300, 300));

        JScrollPane leftPane = new JScrollPane();
        leftPane.setPreferredSize(new Dimension(300, 300));
        leftPane.add(this.input);
        this.add(leftPane, gbc);
    }
    // ...
}
```

Aufgabe 1

GBO für iMage „iCatcher” – Aufbau



```
public final class UpperPanel extends JPanel implements IObservable<IModel> {  
    // ...  
    private void setInput(IModel model) {  
        BufferedImage[] inputs = model.getInput();  
        if (inputs == this.currentInputs) {  
            return;  
        }  
  
        if (inputs == null) {  
            this.input.setImage(null);  
            return;  
        }  
  
        this.currentInputs = inputs;  
  
        BufferedImage input = new BufferedImage(inputs[0].getWidth() * inputs.length,  
            inputs[0].getHeight(), BufferedImage.TYPE_INT_RGB);  
        Graphics2D g2d = (Graphics2D) input.getGraphics();  
        for (int i = 0; i < inputs.length; i++) {  
            g2d.drawImage(inputs[i], i * inputs[0].getWidth(), 0, null);  
        }  
  
        input.flush();  
        input = ImageUtils.getImage(input, 300 * inputs.length, 300, Color.WHITE, 0);  
        this.input.setPreferredSize(new Dimension(300 * inputs.length, 300));  
        this.input.setImage(input);  
    }  
    // ...  
}
```

Aufgabe 4

```
public final class UpperPanel extends JPanel implements IObservable<IModel> {
    // ...
    private void createButtonRight(JPanel rightPane) {

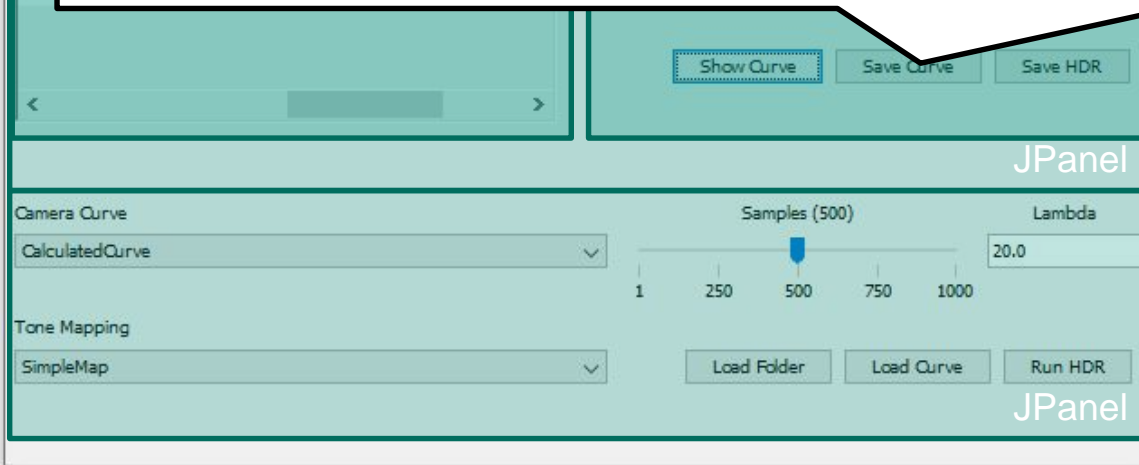
        // ...

        this.btnShowCurve = new JButton("Show Curve");
        gbc = new GridBagConstraints().insets(0, 0, 0, 5).build();
        buttonRight.add(this.btnShowCurve, gbc);

        this.btnSaveCurve = new JButton("Save Curve");
        gbc = new GridBagConstraints().insets(0, 0, 0, 5).gridx(1).build();
        buttonRight.add(this.btnSaveCurve, gbc);

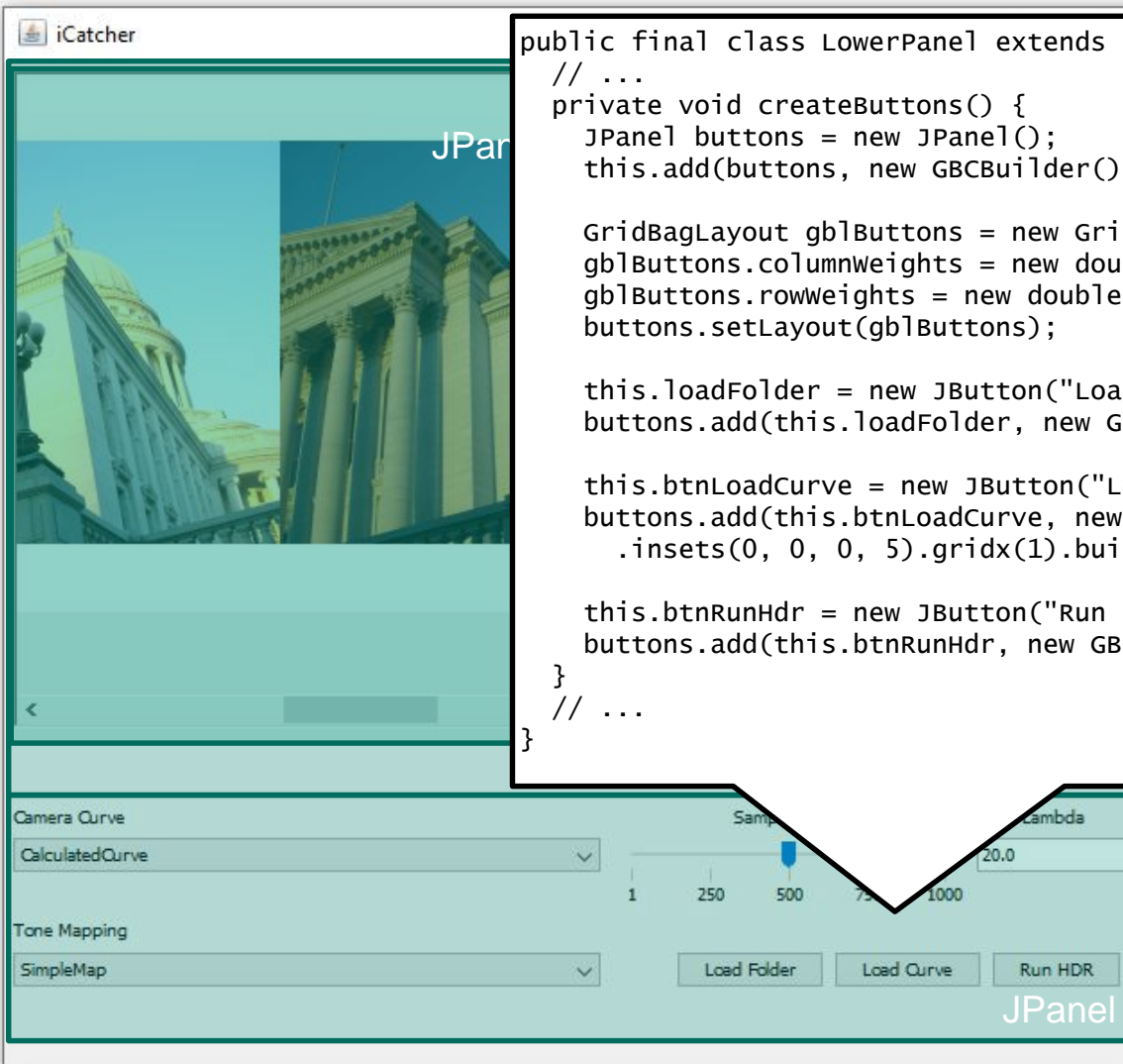
        this.btnSaveHDR = new JButton("Save HDR");
        gbc = new GridBagConstraints().insets(0, 0, 0, 5).gridx(2).build();
        buttonRight.add(this.btnSaveHDR, gbc);

    }
    // ...
}
```



Aufgabe 1

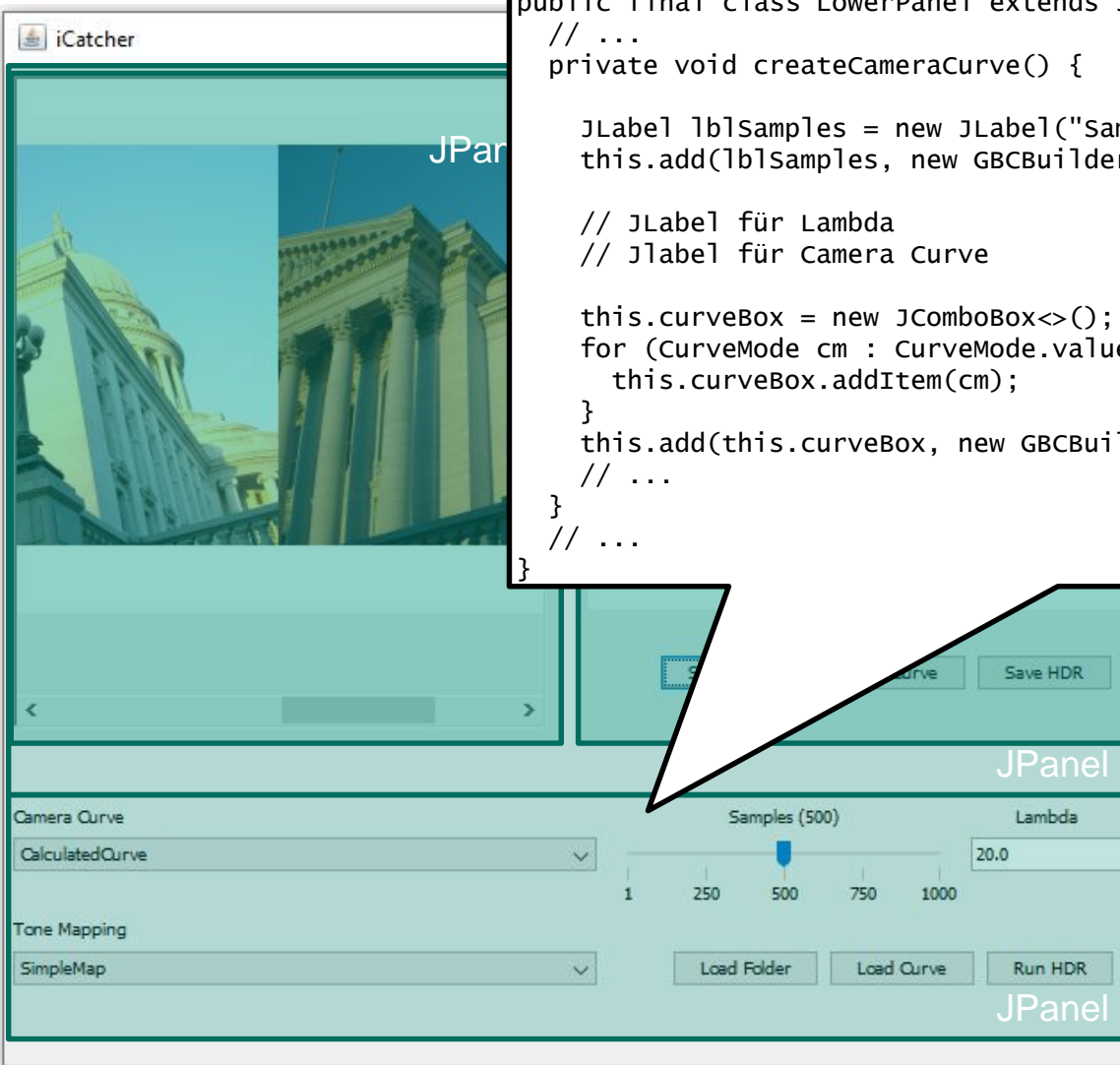
GBO für iMage „iCatcher” – Aufbau



```
public final class LowerPanel extends JPanel implements IObservable<IModel> {  
    // ...  
    private void createButtons() {  
        JPanel buttons = new JPanel();  
        this.add(buttons, new GBCBuilder().anchor(...));  
  
        GridBagLayout gblButtons = new GridBagLayout();  
        gblButtons.columnWeights = new double[] { 0.0, 0.0, 0.0, Double.MIN_VALUE };  
        gblButtons.rowWeights = new double[] { 0.0, Double.MIN_VALUE };  
        buttons.setLayout(gblButtons);  
  
        this.loadFolder = new JButton("Load Folder");  
        buttons.add(this.loadFolder, new GBCBuilder().insets(0, 0, 0, 5).build());  
  
        this.btnLoadCurve = new JButton("Load Curve");  
        buttons.add(this.btnLoadCurve, new GBCBuilder()  
            .insets(0, 0, 0, 5).gridx(1).build());  
  
        this.btnRunHdr = new JButton("Run HDR");  
        buttons.add(this.btnRunHdr, new GBCBuilder().gridx(2).build());  
    }  
    // ...  
}
```

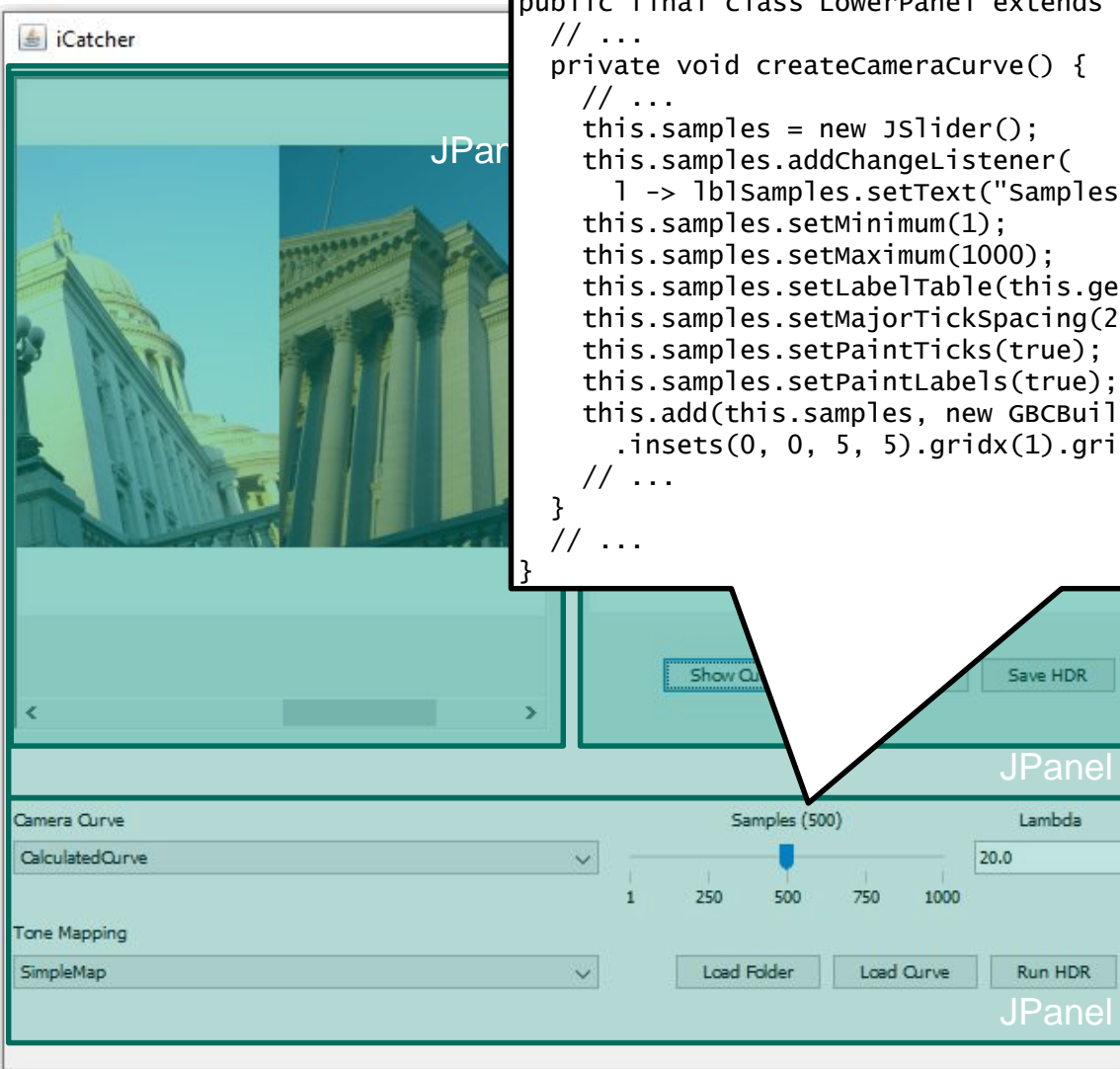
Aufgabe 1

GBO für iMage „iCatcher” – Aufbau



Aufgabe 1

GBO für iMage „iCatcher” – Aufbau



```
public final class LowerPanel extends JPanel implements IObserver<IMode> {  
    // ...  
    private void createCameraCurve() {  
        // ...  
        this.samples = new JSlider();  
        this.samples.addChangeListener(  
            1 -> 1b1Samples.setText("Samples (" + this.samples.getValue() + ")");  
        this.samples.setMinimum(1);  
        this.samples.setMaximum(1000);  
        this.samples.setLabelTable(this.getSliderDictionary());  
        this.samples.setMajorTickSpacing(249);  
        this.samples.setPaintTicks(true);  
        this.samples.setPaintLabels(true);  
        this.add(this.samples, new GBCBuilder()  
            .insets(0, 0, 5, 5).gridx(1).gridy(1).build());  
        // ...  
    }  
    // ...  
}
```

Aufgabe 1

GB0 für iMage „iCatcher” – Aufbau

```
iCat
public final class LowerPanel extends JPanel implements IObservable<IModel> {
    // ...
    private void createCameraCurve() {
        // ...
        this.lambdaText = new JTextField();
        this.lambdaText.setColumns(10);
        this.add(this.lambdaText, new GridBagConstraints().anchor(...));
    }
    // ...
}
```

```
public final class LowerPanel ... {
    // ...
    public void setLambdaToInvalid(
        boolean invalid) {

        this.lambdaText.setForeground(
            invalid ? Color.RED : Color.BLACK);
    }
    // ...
}
```

```
final class Model implements IModel {
    // ...
    @Override
    public boolean setLambda(float lambda) {
        if (this.lambda == lambda) {
            return true;
        }

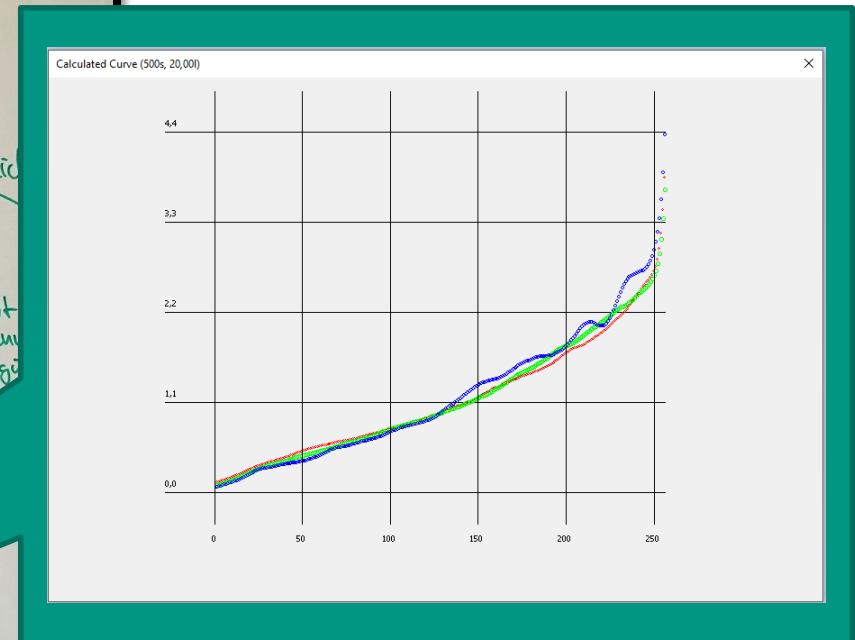
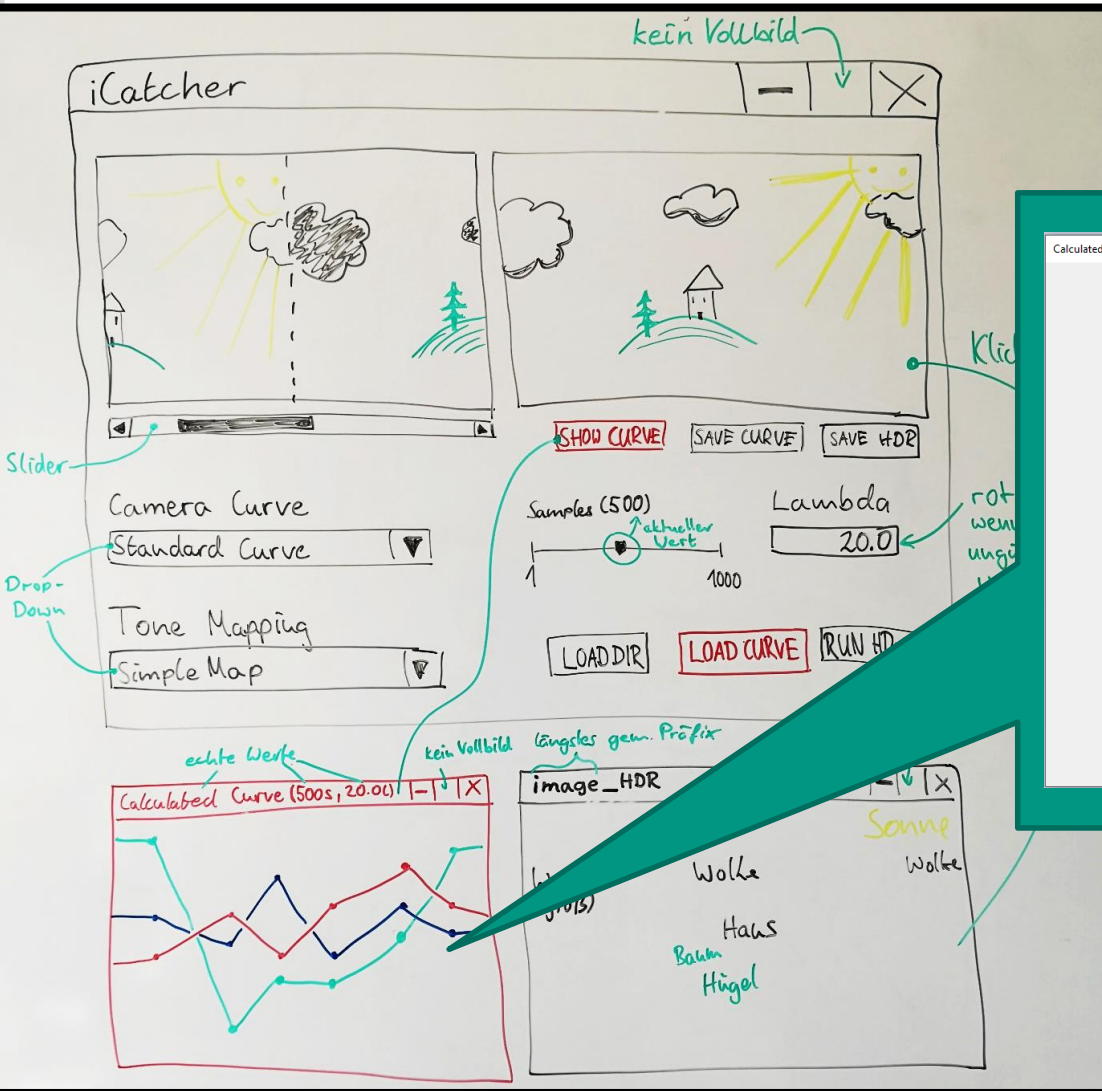
        if (lambda <= 0 || lambda > 100) {
            return false;
        }
        this.lambda = lambda;
        if (this.cm == CurveMode.CalculatedCurve) {
            this.result = null;
            this.resultRGB = null;
        }
        this.informObservers();
        return true;
    }
    // ...
}
```

```
public final class ParameterUpdateHandler
    extends Handler implements IObservable<IModel> {
    // ...
    private boolean setLambda() {
        String lambda = this.view.getLambda();
        Float l;
        try {
            l = Float.parseFloat(lambda);
        } catch (NumberFormatException
            | NullPointerException e) {
            l = null;
        }

        boolean invalid = l == null
            || !this.model.setLambda(l);
        this.view.setLambdaToInvalid(invalid);
        return !invalid;
    }
    // ...
}
```

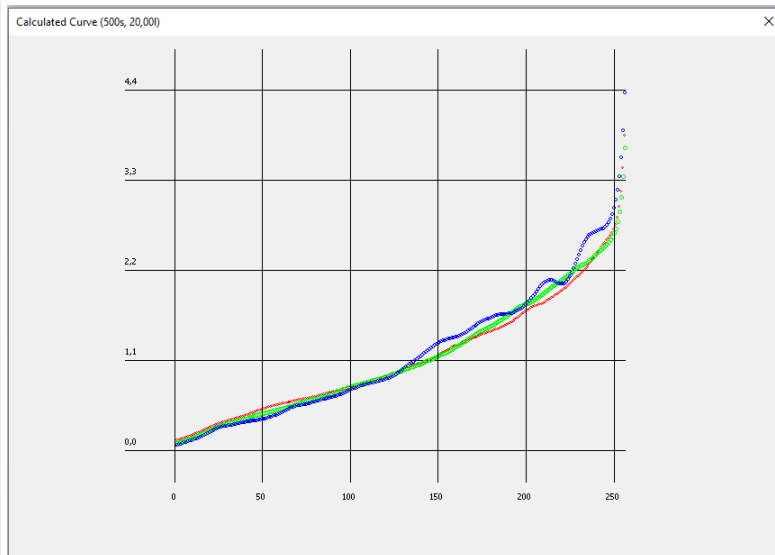
Aufgabe 1

GBO für iMage „iCatcher“



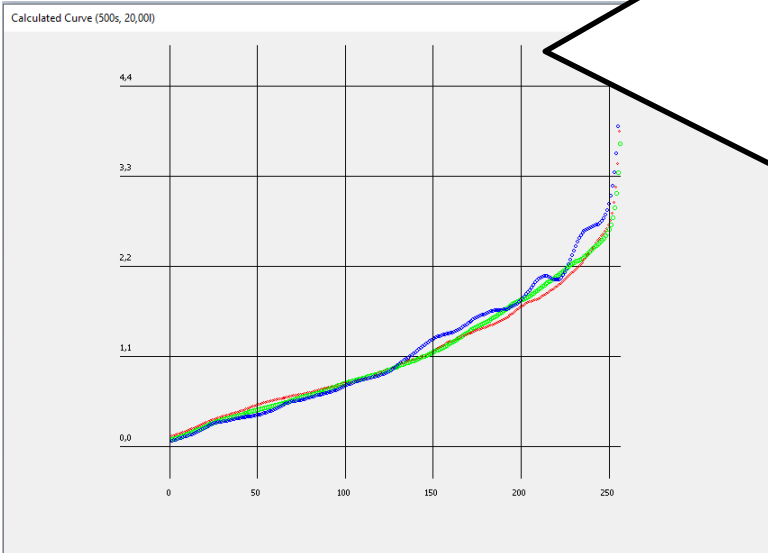
Aufgabe 1

GBO für iMage „iCatcher”



Aufgabe 1

GB0 für iMage „iCatcher“

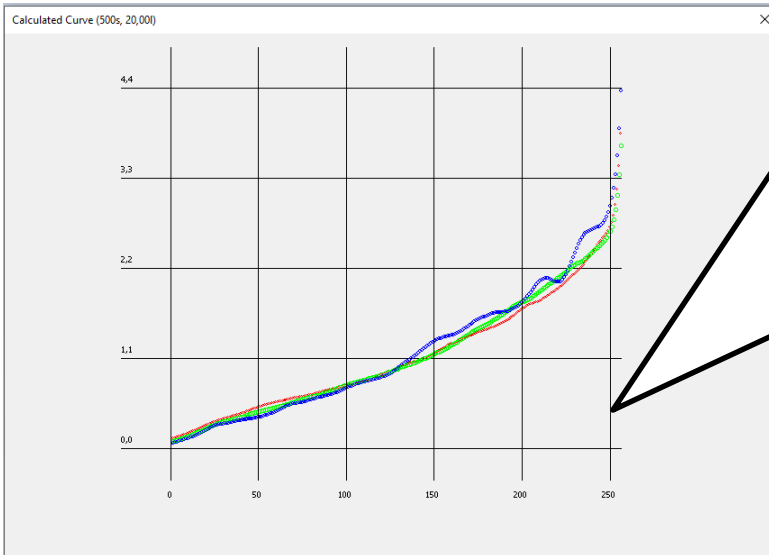


```
public final class Graph extends JDialog {
    // ...
    private void drawGraph(Graphics2D g2d) {
        double[] r = new double[256];
        double[] g = new double[256];
        double[] b = new double[256];
        double max = Double.MIN_VALUE;

        // Berechne Wertebereich
        for (int i = 0; i < 256; i++) {
            float[] resp = this.cc.getResponse(new int[] { i, i, i });
            r[i] = resp[0];
            if (r[i] > max) {
                max = r[i];
            }
            g[i] = resp[1];
            if (g[i] > max) {
                max = g[i];
            }
            b[i] = resp[2];
            if (b[i] > max) {
                max = b[i];
            }
        }
        max = Math.ceil(max) + 0.5;
        // ...
    }
    // ...
}
```

Aufgabe 1

GBO für iMage „iCatcher“

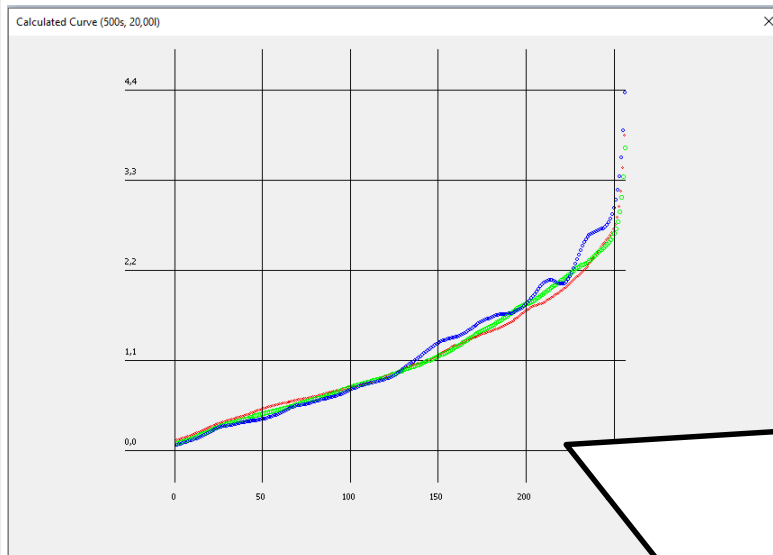


```
public final class Graph extends JDialog {
    // ...
    private void drawGraph(Graphics2D g2d) {
        // ...
        g2d.setStroke(new BasicStroke(1));

        // Zeichne Linien in y-Richtung
        g2d.setColor(Color.BLACK);
        for (int i = 0; i < Graph.Y_STEPS; i++) {
            double yVal = max / Graph.Y_STEPS * i;
            int y = Graph.GRAPH_SIZE - (int) ((yVal / max)
                * Graph.GRAPH_SIZE) - Graph.XY_OFFSET;
            String text = String.format("%.1f", yVal);
            g2d.drawString(text, 0, y - Graph.TEXT_SHIFT_PER_LETTER
                * text.length());
            g2d.drawLine(0, y, Graph.GRAPH_SIZE + Graph.XY_OFFSET, y);
        }
        // ...
    }
    // ...
}
```

Aufgabe 1

GBO für iMage „iCatcher“



```
public final class Graph extends JDialog {
    // ...
    private void drawGraph(Graphics2D g2d) {
        // ...
        for (int i = 0; i < 256; i++) {
            // Berechne x-Koordinate
            int x = (int) ((i) * (Graph.GRAPH_SIZE / 256.f))
                + Graph.XY_OFFSET;

            // Zeichne Kreis in entsprechender Farbe...
            g2d.setColor(Color.RED);
            int y = Graph.GRAPH_SIZE - (int) ((r[i] / max)
                * Graph.GRAPH_SIZE) - Graph.XY_OFFSET;
            g2d.drawOval(x, y, 2, 2);

            g2d.setColor(Color.GREEN);
            y = Graph.GRAPH_SIZE - (int) ((g[i] / max)
                * Graph.GRAPH_SIZE) - Graph.XY_OFFSET;
            g2d.drawOval(x, y, 4, 4);

            g2d.setColor(Color.BLUE);
            y = Graph.GRAPH_SIZE - (int) ((b[i] / max)
                * Graph.GRAPH_SIZE) - Graph.XY_OFFSET;
            g2d.drawOval(x, y, 3, 3);

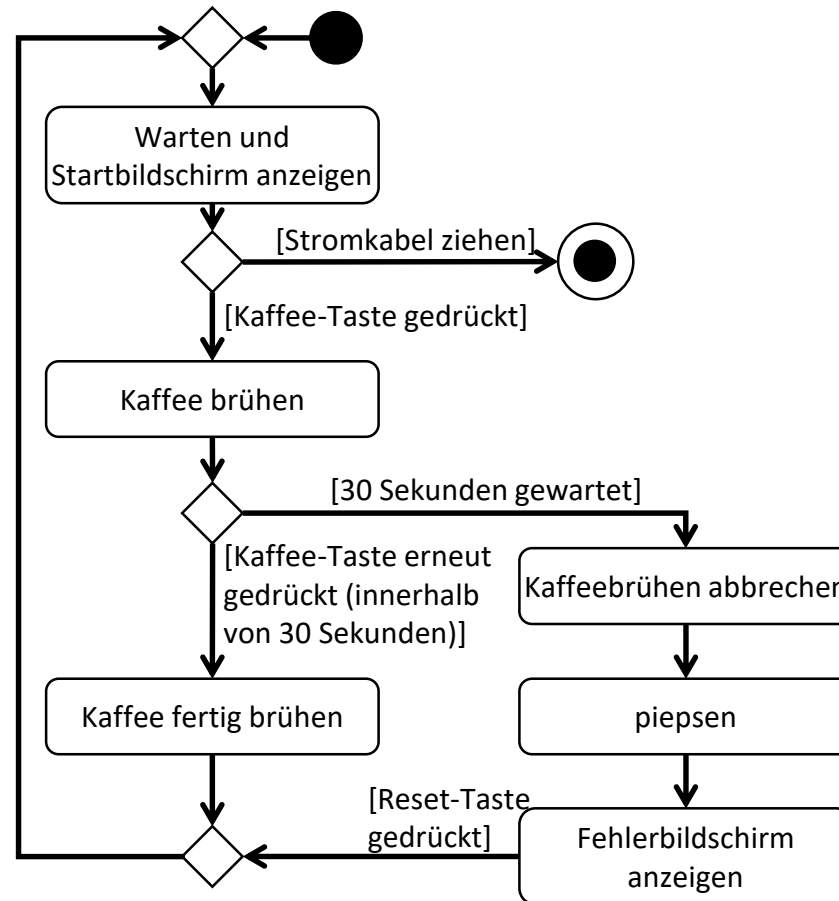
            // Zeichne alle 50 x-Werte eine Linie
            if (i % 50 == 0) {
                g2d.setColor(Color.BLACK);
                g2d.drawString(String.valueOf(i),
                    x - Graph.TEXT_SHIFT_PER_LETTER
                    * String.valueOf(i).length(), Graph.GRAPH_SIZE);
                g2d.drawLine(x, 0, x, Graph.GRAPH_SIZE - 20);
            }
        }
    }
}
```

Diagramme überführen

AUFGABE 2

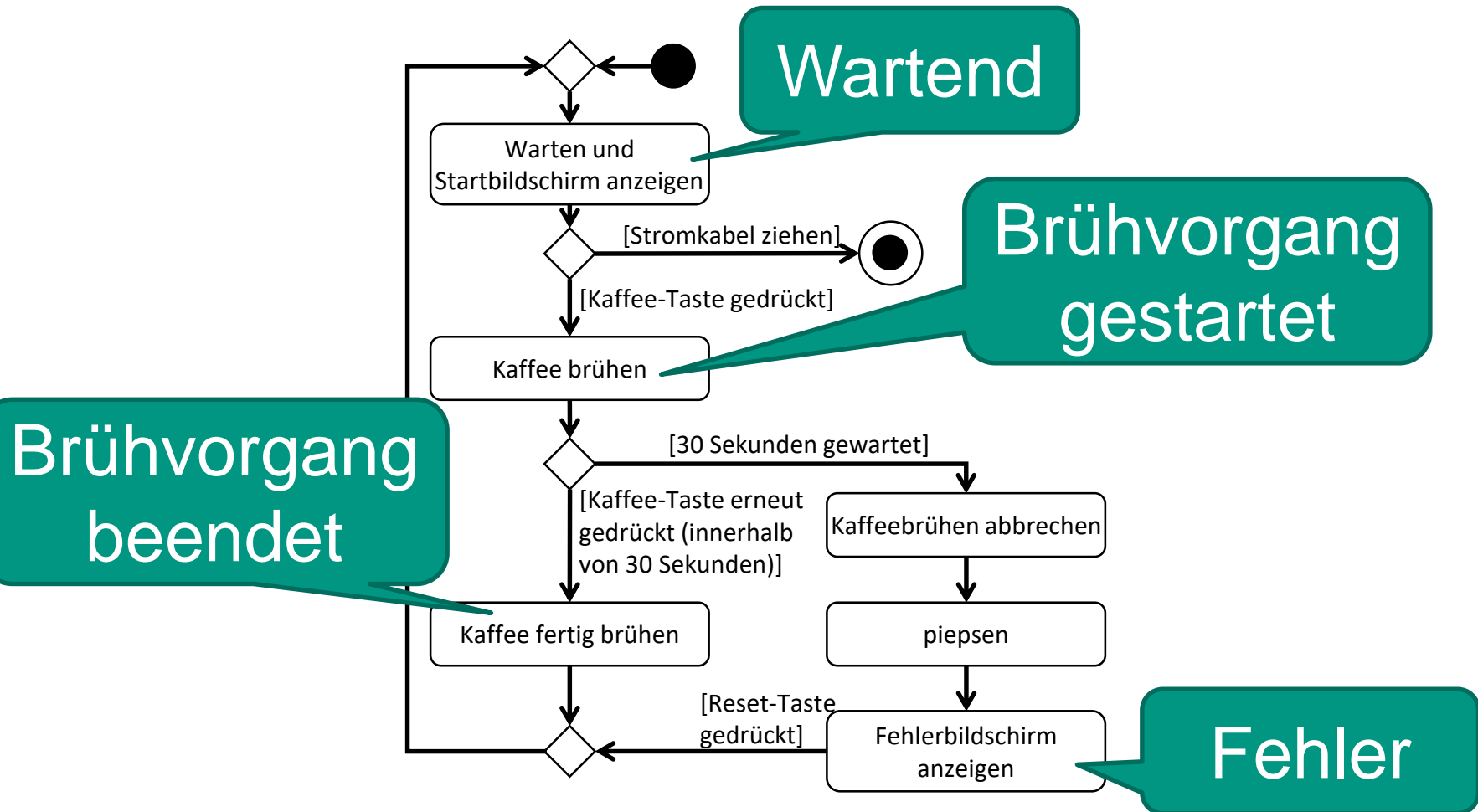
Aufgabe 2

Diagramme überführen: Aktivitätsdiagramm



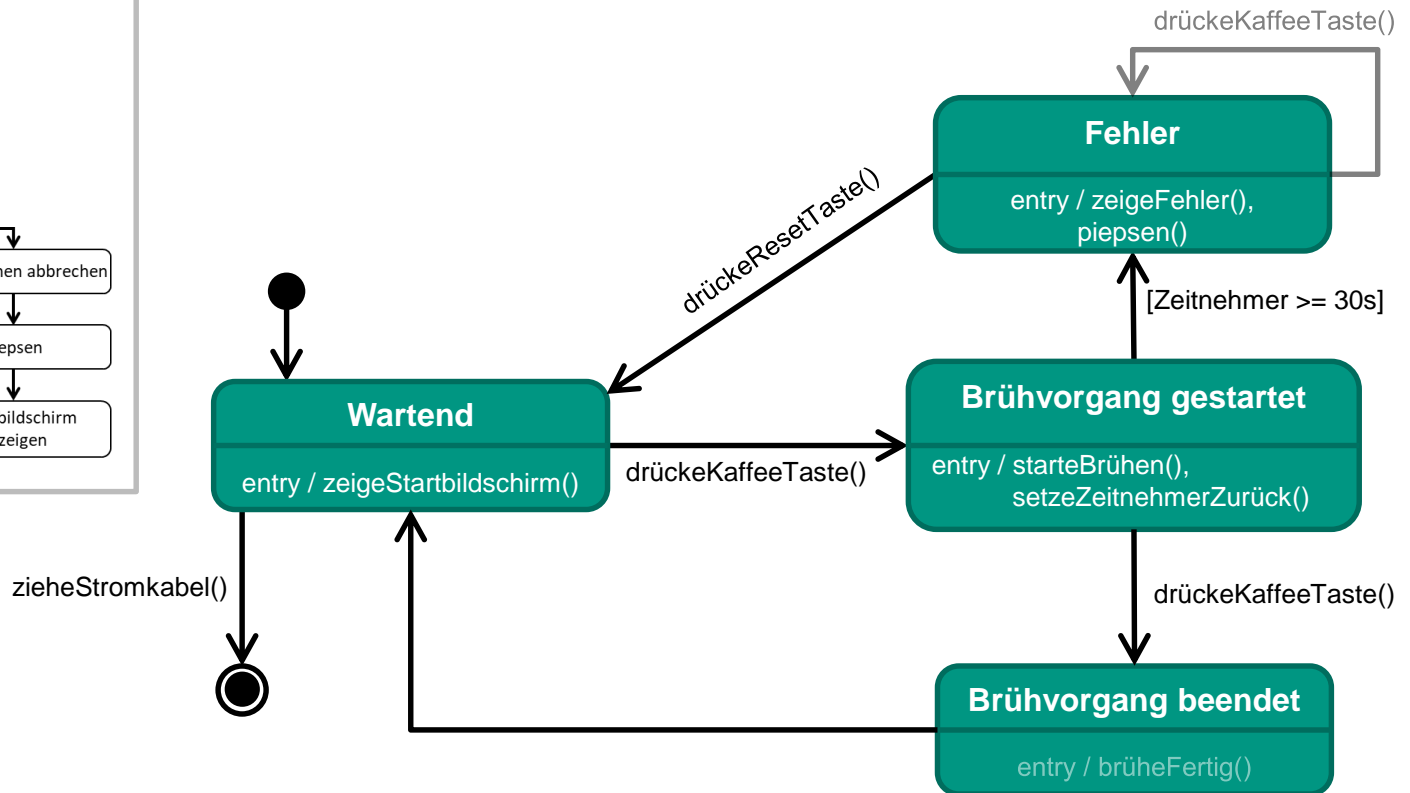
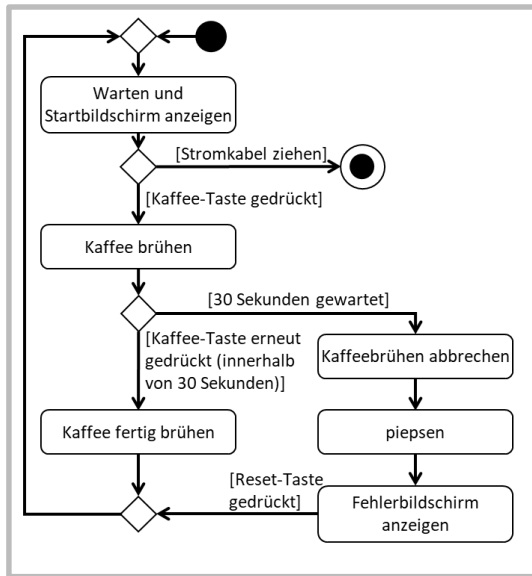
Aufgabe 2

Diagramme überführen: Zustände



Aufgabe 2

Diagramme überführen: Zustände



Geheimnisprinzip

AUFGABE 3

Aufgabe 3

Geheimnisprinzip

- `java.io.Closeable`
- `java.lang.Comparable<T>`
- `java.util.List`
- `java.util.Locale`

Aufgabe 3

Geheimnisprinzip – `java.io.Closeable`

■ Was?

- Implementierung von Strom-artigen Ressourcen (die beendet/geschlossen werden können)
- Wie funktioniert der Zugriff auf die konkreten Ressourcen?

■ Wie?

- `close()` kapselt den Zugriff auf die Ressource
- Das Schließen wird somit abstrahiert
- (Bei Benutzung ist es egal, wie die Ressource implementiert ist)

Aufgabe 3

Geheimnisprinzip – `java.lang.Comparable<T>`

■ Was?

- Vergleich zwei Objekte gleichen Typs
- Verborgenen werden die Vergleichskriterien (der Algorithmus)

■ Wie?

- `compare()` kapselt den Vergleich
- Algorithmus/Vergleichskriterien wird nicht offengelegt
- Objekte können verglichen werden, ohne Aufbau zu kennen
- (Bei Benutzung erhält man Ordnung, ohne zu Wissen über Vergleichen)

Aufgabe 3

Geheimnisprinzip – `java.util.List`

■ Was?

- Die Implementierung der Liste, d.h.,
- ...verborgen wird die konkret verwendete Datenstruktur

■ Wie?

- Die Methoden der Schnittstelle kapseln Zugriff auf die Liste
- Liste kann immer als solche verwendet werden, egal wie konkrete Implementierung...
- ... und die konkrete Datenstruktur aussieht
- (Bei Benutzung erhält man eine abstrakte Listenrepräsentation)

Aufgabe 3

Geheimnisprinzip – `java.util.Locale`

■ Was?

- Gebietsschemaparameter, d.h.
- ...Zeichensatz, Formate, etc.

■ Wie?






- Gebietsparameter gekapselt in Objekt
- Zugriff auf regionsspezifische Informationen über einheitliche Schnittstelle
- Bei Benutzung ist es egal welches konkrete Locale-Objekt verwendet wird

Benutztrelation

AUFGABE 4

Aufgabe 4

Benutztrelation – Relationstypen






- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

Benutztrelation – Blatt 1 iMage JMJRST

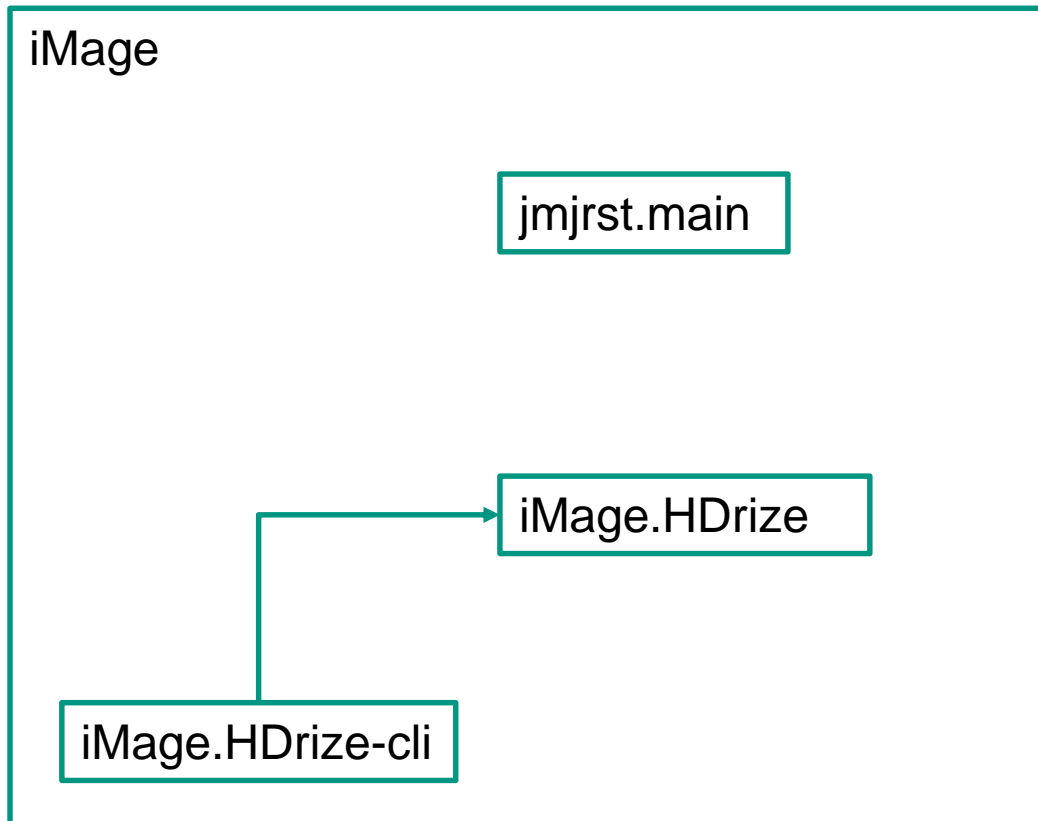
iMage






jmjrst.main

- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

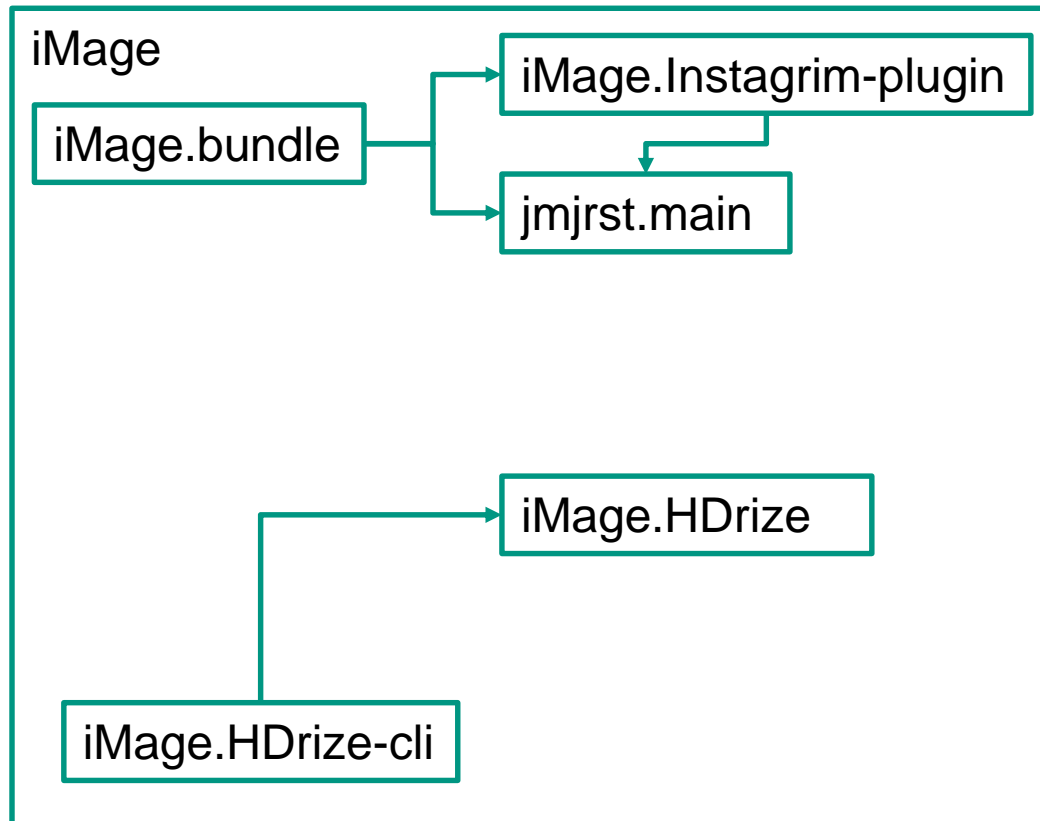
Benutztrelation – Blatt 2 HDrize und KZS








- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

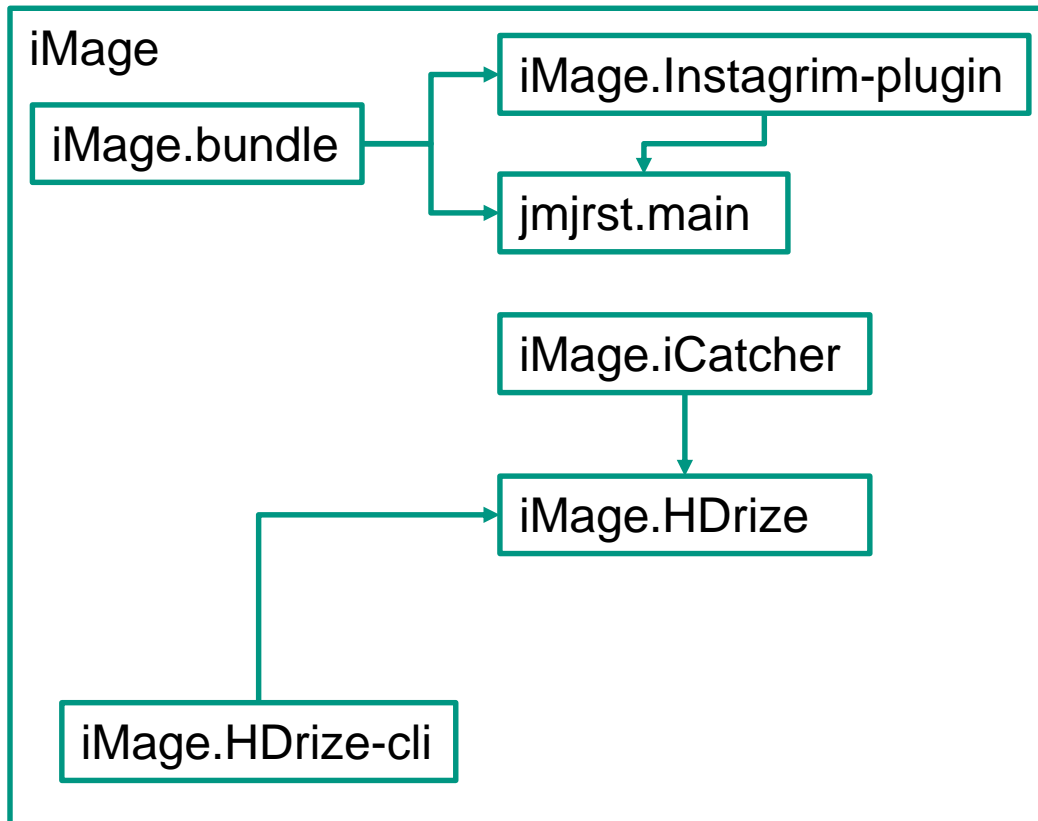
Benutzrelation – Blatt 3 Einschübe und Bündel.








- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

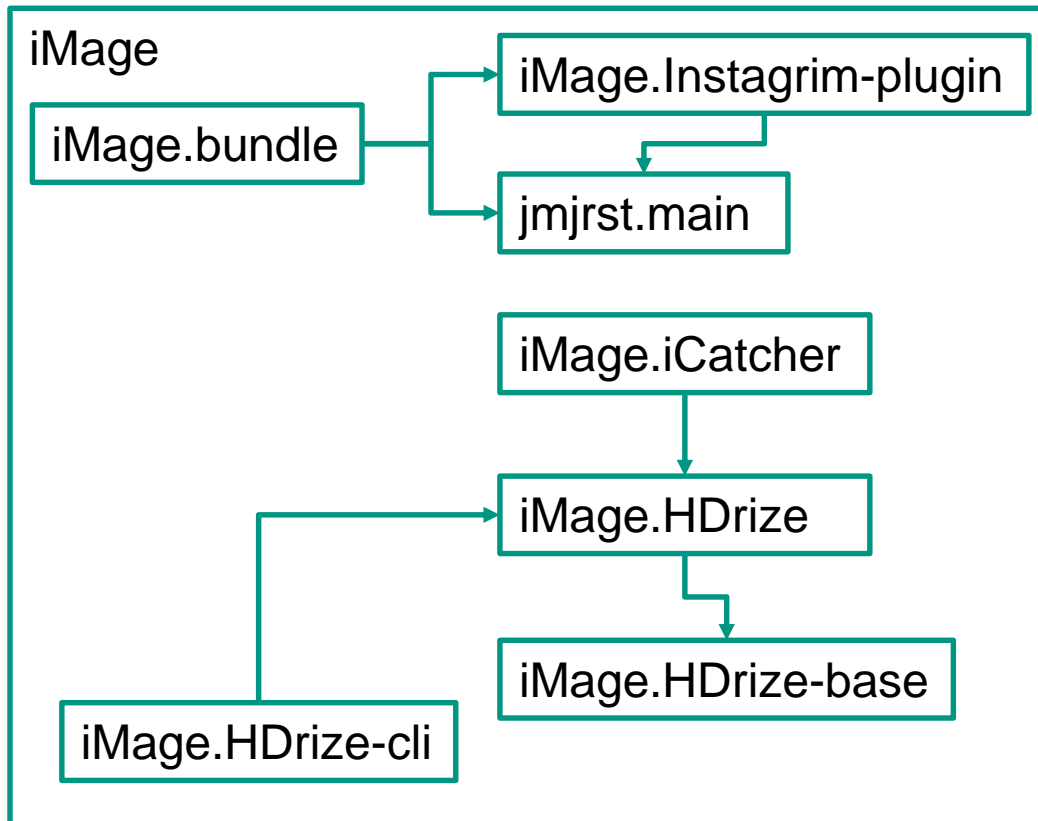
Benutztrelation – Blatt 4 GBO








- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

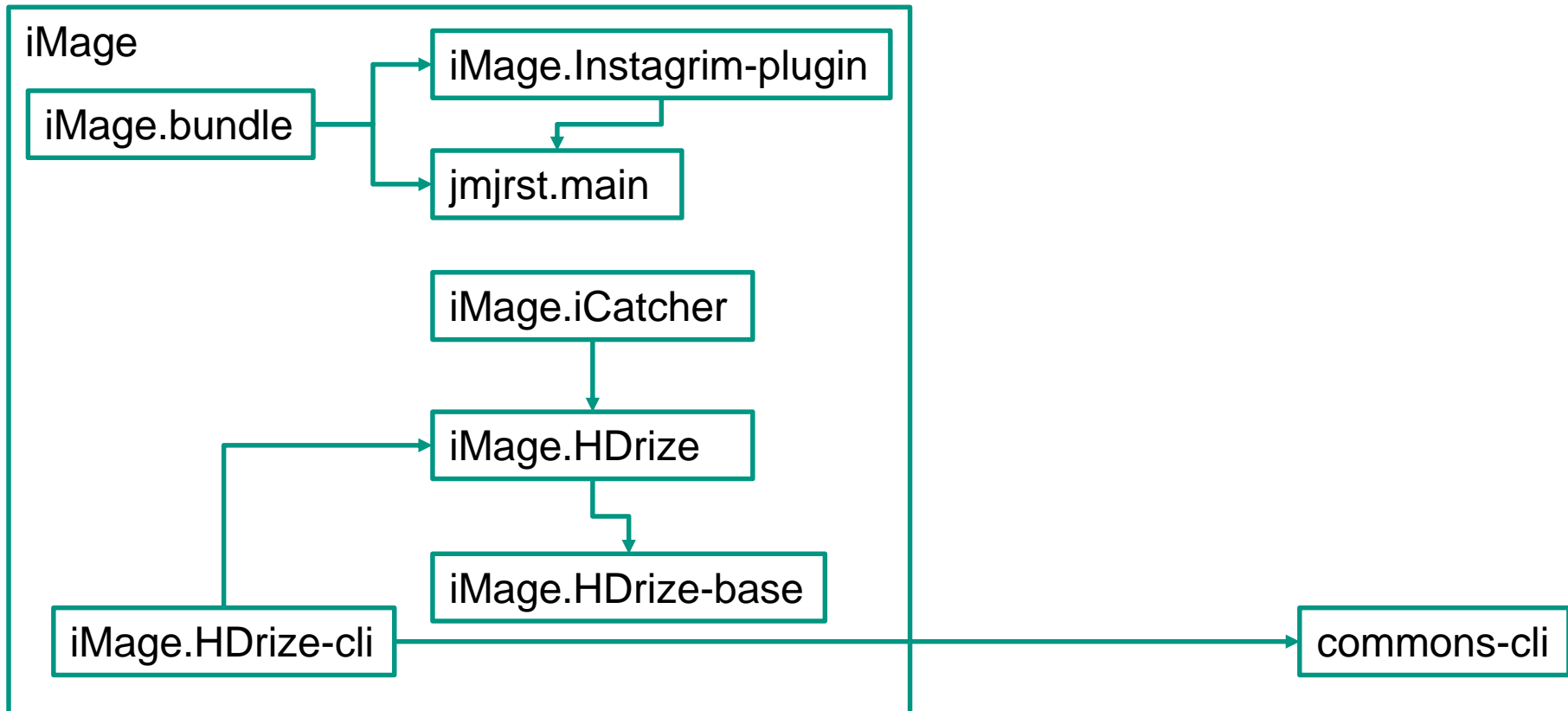
Benutzrelation – (nicht so) externe Abhängigk.








- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

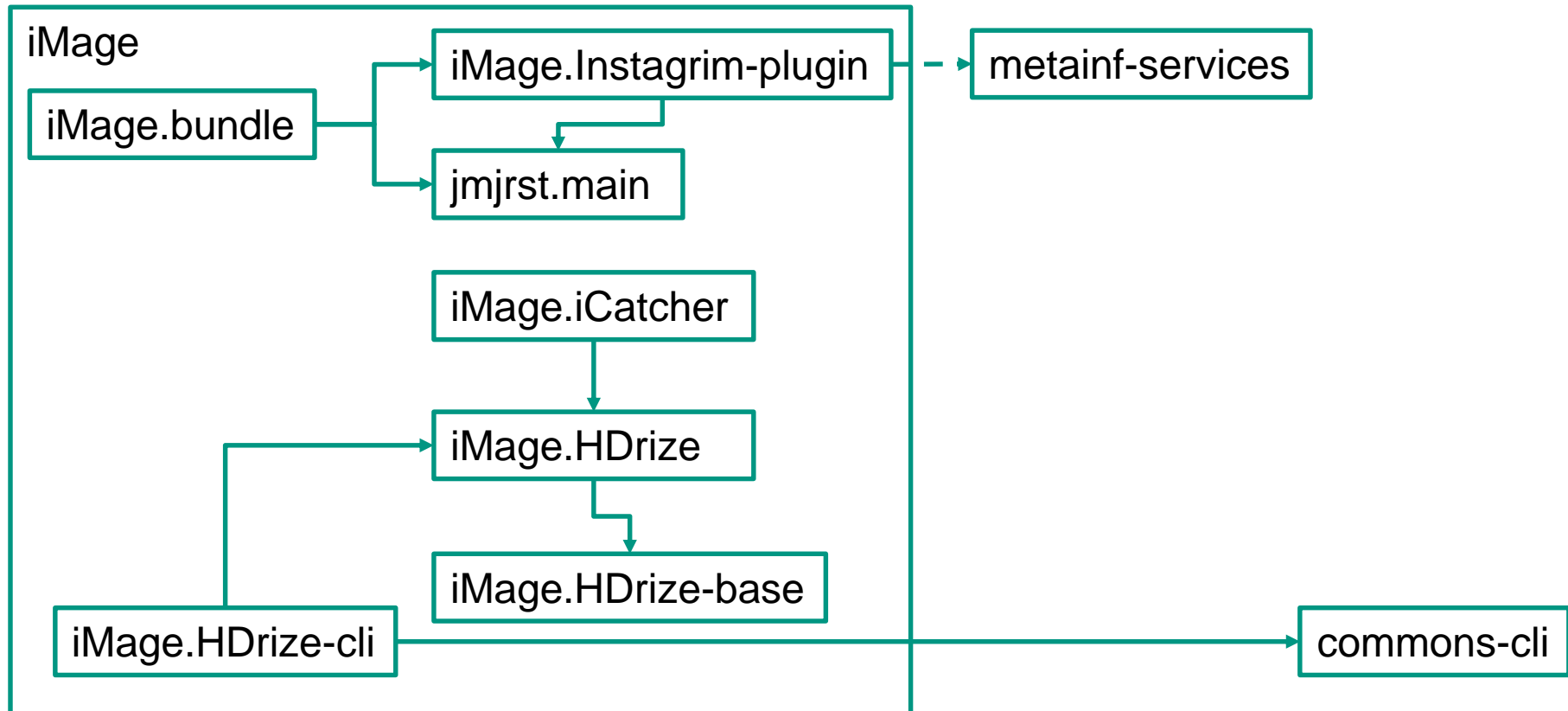
Benutzrelation – Externe Abhängigkeiten








- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

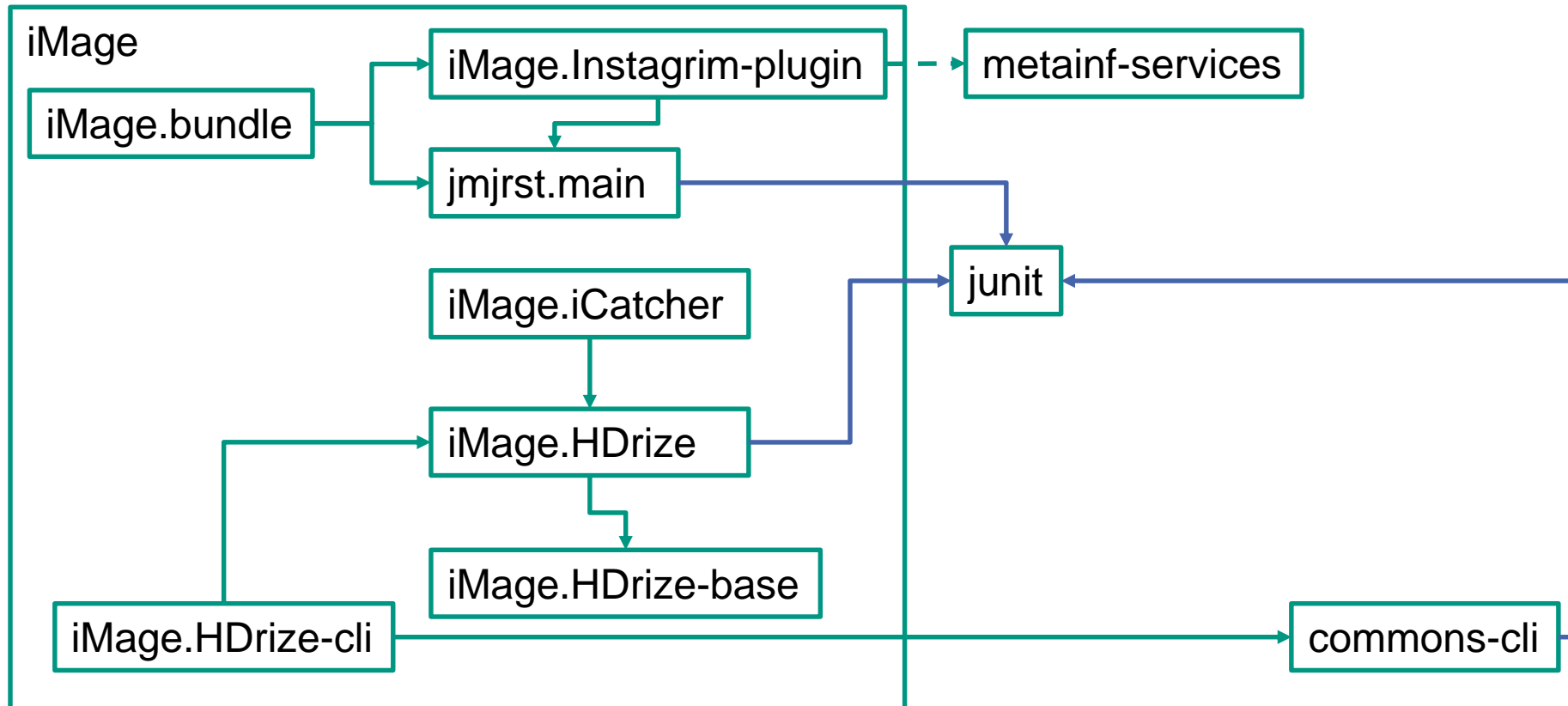
Benutzrelation – Externe Abhängigkeiten








- A  B: A hängt von B ab; - -  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

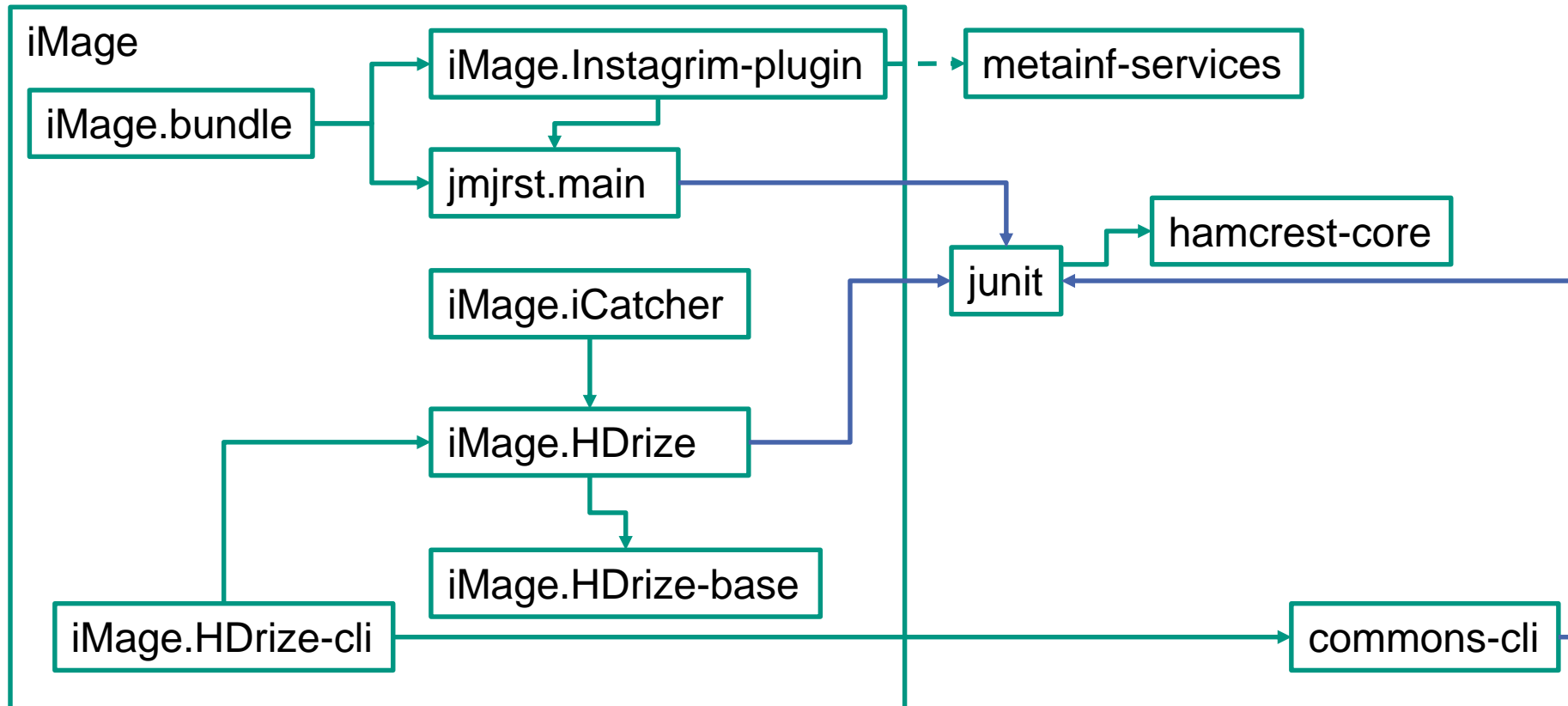
Benutzrelation – Externe Abhängigkeiten








- A  B: A hängt von B ab; - -  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

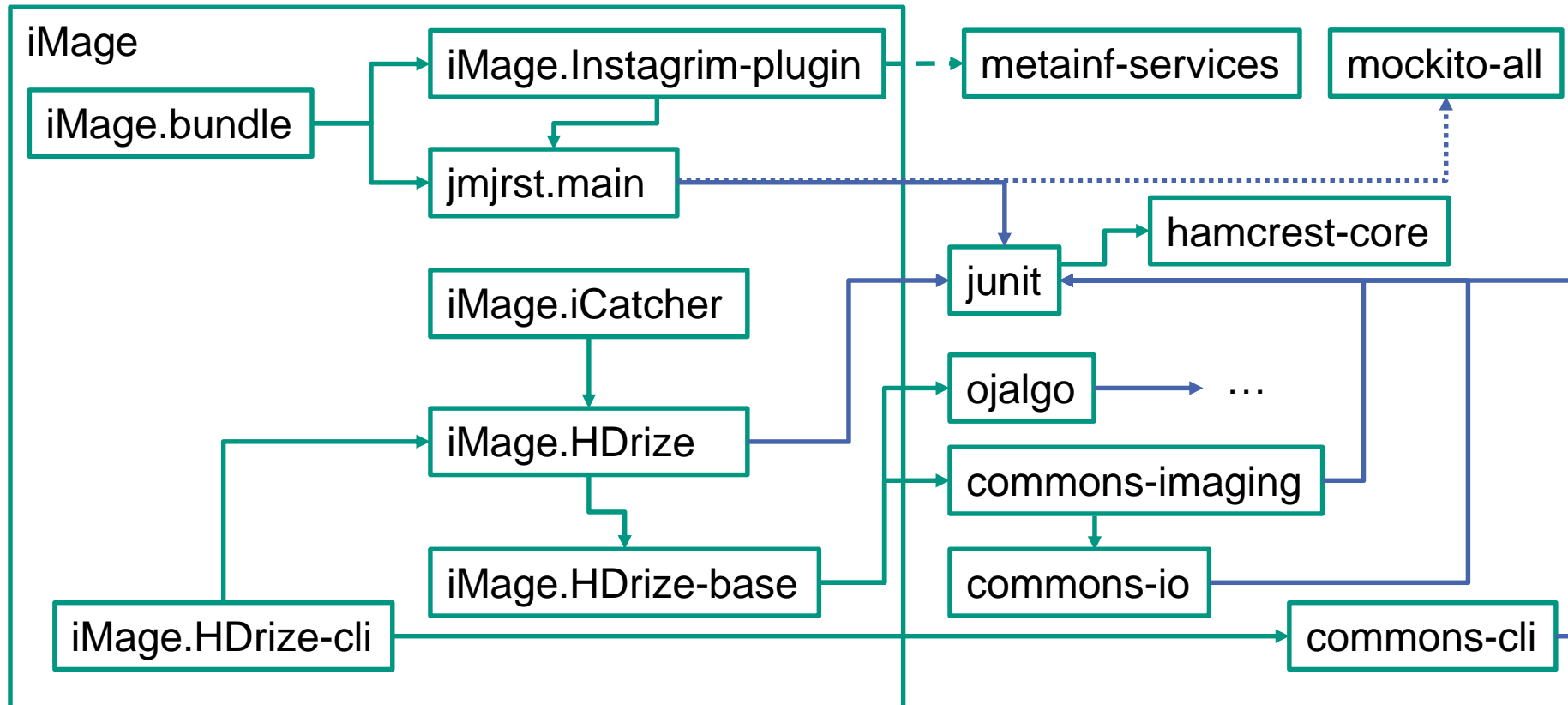
Benutzrelation – Externe Abhängigkeiten








- A  B: A hängt von B ab; - -  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

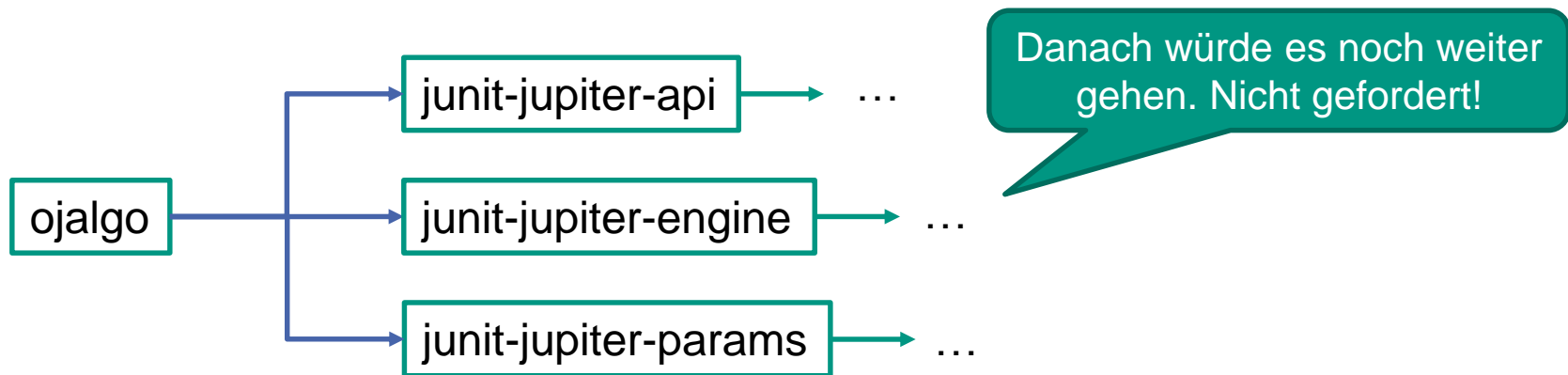
Benutzrelation – Externe Abhängigkeiten








- A  B: A hängt von B ab; - -  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Aufgabe 4

Benutzrelation – Externe Abhängigkeiten



- A  B: A hängt von B ab;  in pom.xml als optional gekennzeichnet
- C  D: C hängt von D ab (Bonusaufgabe)
- E  F: E hängt von F ab (nur für Test)
- G  H: G hängt von H ab (nur für Test, Bonusaufgabe)

Entwurfsmuster

AUFGABE 5

Aufgabe 5

Entwurfsmuster – A

„Die ganzen Matrix-Operationen selbst zu implementieren, scheint mir nicht sinnvoll. Warum nehmen wir nicht die Matrix-Bibliothek von org.ojalgo? Die speichern allerdings Matrizen in einem PrimitiveDenseStore (oder so). Ich weiß nicht, ob das zu unserer Matrix-Modellierung passt. Die Modellierung können wir aber auch nicht mehr ändern, glaub ich, denn die Klicki-Bunti-Leute (das GUI-Team) verwenden die Schnittstellen ja schon.“

Aufgabe 5

Entwurfsmuster – A: Auflösung

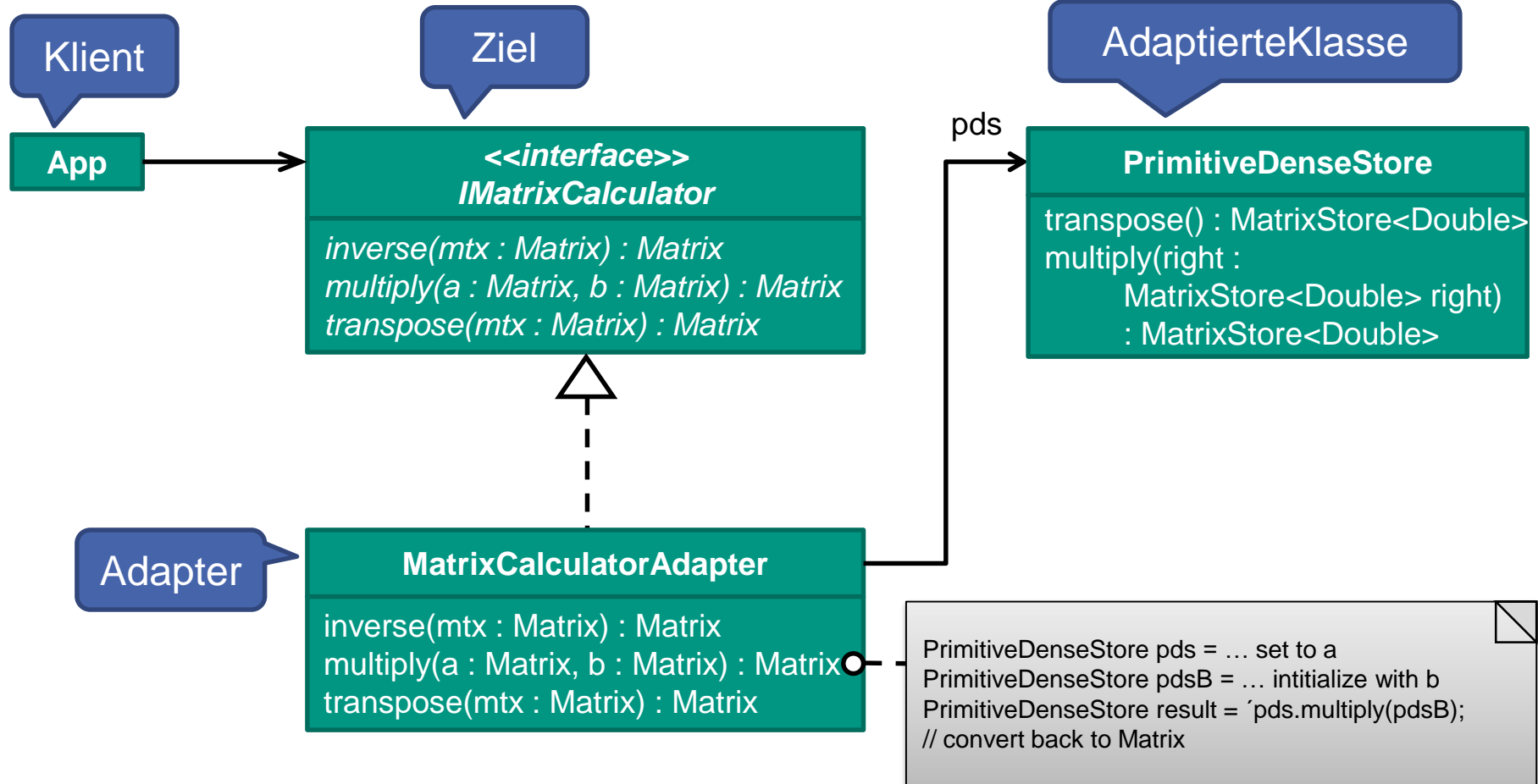
„Die ganzen Matrix-Operationen selbst zu implementieren, scheint mir nicht sinnvoll. Warum nehmen wir nicht die Matrix-Bibliothek von org.ojalgo? Die speichern allerdings Matrizen in einem PrimitiveDenseStore (oder so). Ich weiß nicht, ob das zu unserer Matrix-Modellierung passt. Die Modellierung können wir aber auch nicht mehr ändern, glaub ich, denn die Klicki-Bunti-Leute (das GUI-Team) verwenden die Schnittstellen ja schon.“



Adapter!

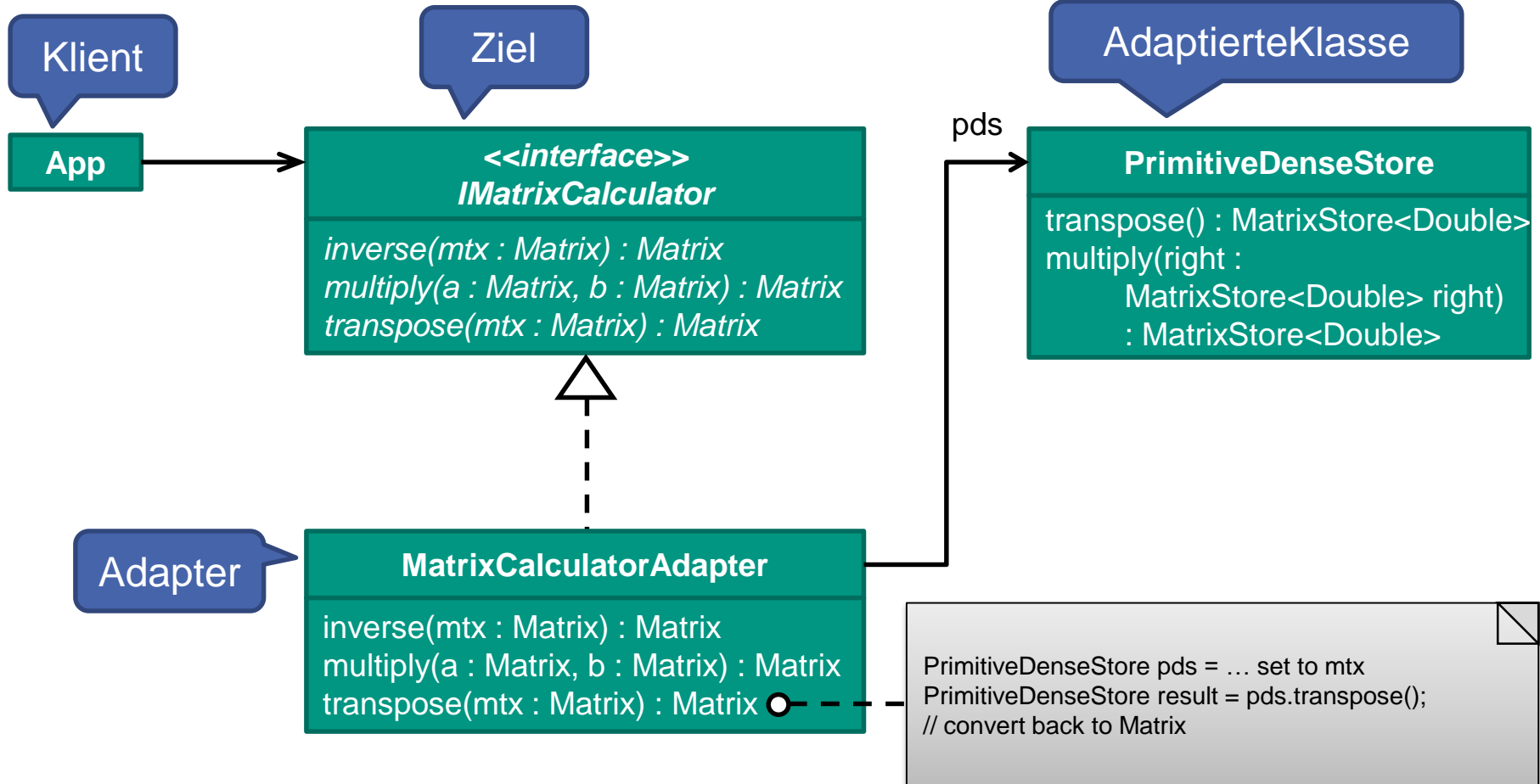
Aufgabe 5

Entwurfsmuster – A: Umsetzung



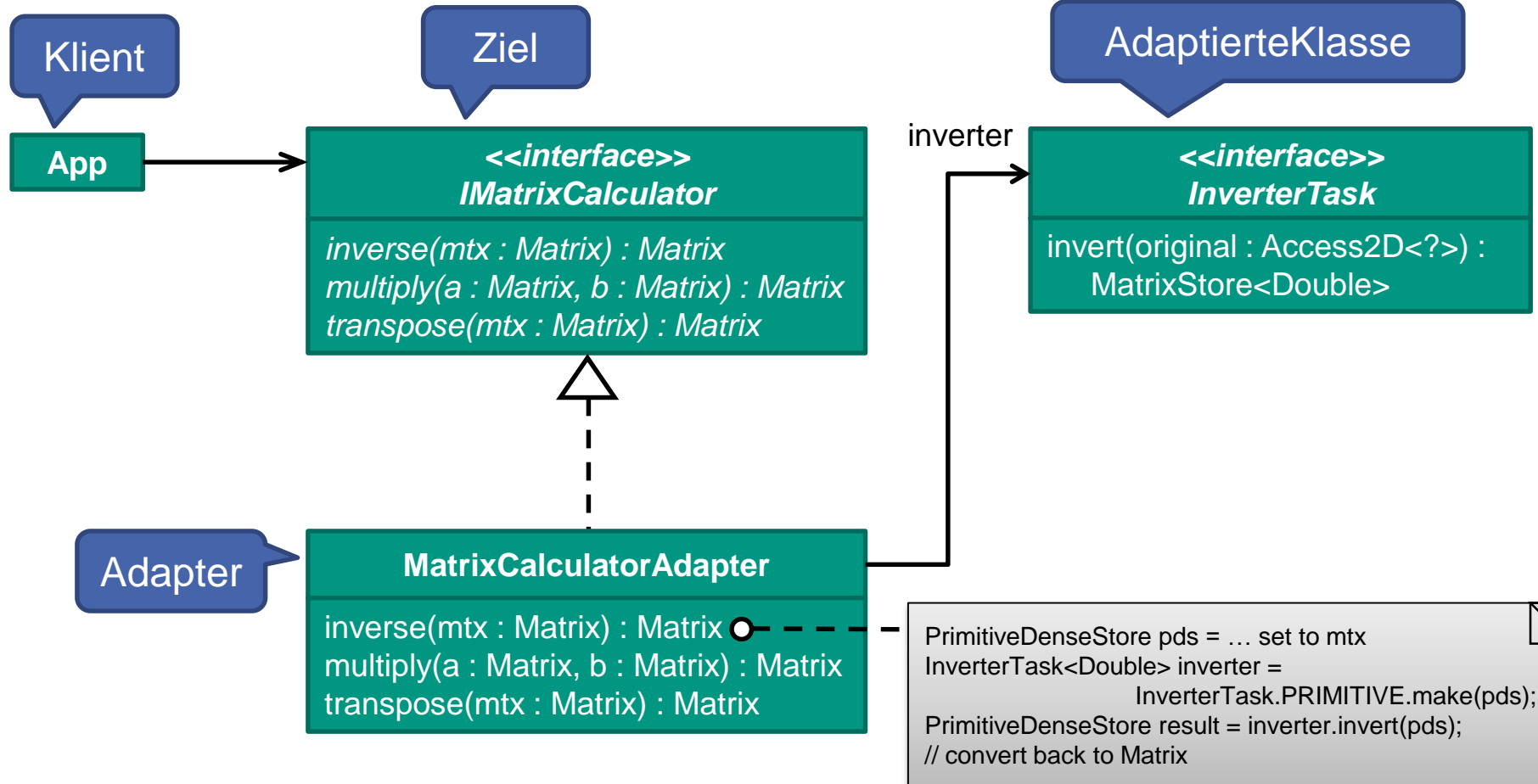
Aufgabe 5

Entwurfsmuster – A: Umsetzung



Aufgabe 5

Entwurfsmuster – A: Umsetzung



Aufgabe 5

Entwurfsmuster – B

„Die von oben haben ja gesagt, wir sollen uns HDRCombine nochmal vornehmen. Ich hab mir überlegt, es sollte eine Version geben in der die Gewichte schon vorberechnet sind, `calculateWeights()` also ein fertiges Array zurückgibt. Das jetzige soll aber auch erhalten bleiben. Außerdem sollte es in einer anderen Version möglich sein, direkt nach dem Zusammensetzen auch noch eine Methode zum Weichzeichnen aufzurufen. Ansonsten, finde ich, sollte sich an der Berechnung der Zusammensetzung der Bilder in `createHDR()` nichts ändern. Never touch a running system!“

Aufgabe 5

Entwurfsmuster – B: Auflösung

„Die von oben haben ja gesagt, wir sollen uns HDRCombine nochmal vornehmen. Ich hab mir überlegt, es sollte eine Version geben in der die Gewichte schon vorberechnet sind, `calculateWeights()` also ein fertiges Array zurückgibt. Das jetzige soll aber auch erhalten bleiben. Außerdem sollte es in einer anderen Version möglich sein, direkt nach dem Zusammensetzen auch noch eine Methode zum Weichzeichnen aufzurufen. Ansonsten, finde ich, sollte sich an der Berechnung der Zusammensetzung der Bilder in `createHDR()` nichts ändern. Never touch a running system!“

**Strategie oder
Schablonenmethode!**

Aufgabe 5

Entwurfsmuster – B: Umsetzung (Schablonenm.)

Schablonenmethode

AbstractHDRCombine

```
+ createHDR(curve : ICameraCurve,  
            imageList : EnhancedImage[]) : HDRImage  
- combine()  
# calculateWeights()  
# blur(img : HDRImage) :HDRImage
```

```
calculateWeights();  
combine();  
blur(...);
```

Einschubmethode

HDRCombineWithWeights

```
# calculateWeights()  
# blur(img : HDRImage) :HDRImage
```

```
return img;
```

HDRCombineWithBlur

```
# calculateWeights()  
# blur(img : HDRImage) :HDRImage
```

Aufgabe 5

Entwurfsmuster – C

„Der Vorstand hätte ja gerne, dass wir demnächst noch Filter (Sepia, Graustufen, Weichzeichner usw.) über die HDR-Bilder laufen lassen. Das geht einfach, mit einer Pipeline z.B. Aber wir müssen dann irgendwas mit unserer HDRImage-Klasse machen. Je nachdem, welche Filter kombiniert wurden entstehen da ja unterschiedliche Bilder bzw. Bild-Klassen. Für jede mögliche Kombination eine Unterklasse von HDRImage bilden skaliert nicht. Wäre doch toll, wenn man das stattdessen dynamisch zusammensetzen könnte (je nach Filterkombination). Zum Beispiel könnte die toString()-Methode die Namen der ausgeführten Filter auflisten.“

Aufgabe 5

Entwurfsmuster – C: Auflösung

„Der Vorstand hätte ja gerne, dass wir demnächst noch Filter (Sepia, Graustufen, Weichzeichner usw.) über die HDR-Bilder laufen lassen. Das geht einfach, mit einer Pipeline z.B. Aber wir müssen dann irgendwas mit unserer HDRImage-Klasse machen. Je nachdem, welche Filter kombiniert wurden entstehen da ja unterschiedliche Bilder bzw. Bild-Klassen. Für jede mögliche Kombination eine Unterklasse von HDRImage bilden skaliert nicht. Wäre doch toll, wenn man das stattdessen dynamisch zusammensetzen könnte (je nach Filterkombination). Zum Beispiel könnte die toString()-Methode die Namen der ausgeführten Filter auflisten.“



Dekorierer!

Aufgabe 5

Entwurfsmuster – C: Umsetzung

