Karlsruher Institut für Technologie

Vorlesung Softwaretechnik I
Übung 2

SWT I – Sommersemester 2019

Walter F. Tichy, Sebastian Weigelt, Tobias Hey, Martin Blersch

IPD Tichy, Fakultät für Informatik

Aufgabe 1
Szenario

Die Startseite der Applikation soll eine Übersicht über das Nutzerkonto und die zuletzt erstellten HDR-Bilder bereitstellen. Per Klick auf ein bereits vorhandenes HDR-Bild wird eine Vorschau erzeugt und die Schaltflächen „Teilen“ und „Schließen“ werden angezeigt. „Schließen“ soll die Vorschau schließen und über „Teilen“ soll es möglich sein das Bild auf soziale Netzwerke wie Facezine oder Instagrim hochzuladen. Über die Schaltfläche „Neues HDR-Bild“ auf der Startseite gelangt der Nutzer zu einer Übersicht der auf dem Mobiltelefon gespeicherten Bilder. Diese Vorschauansicht soll in Echtzeit bis zu einhundert (100) Bilder auf einem Mittelklasse-Smartphone anzeigen können. Der Nutzer wählt drei (3) Bilder einer Bildreihe und bestätigt seine Auswahl mit einem Klick auf „HDR-Bild“ erstellen. Die Eingabebilder werden anschließend auf den Pear-Corp-Zentralserver hochgeladen und das HDR-Bild erzeugt. Die Erzeugung des Ergebnisbildes (HDR) soll nicht länger als sieben (7) Sekunden in Anspruch nehmen. Außerdem sollen mindestens eintausend (1000) Nutzer gleichzeitig mit dem Server kommunizieren können. Ist der Vorgang abgeschlossen, wird dem Nutzer die Schaltfläche „HDR-Bild abholen“ angezeigt. Per Klick auf diese Schaltfläche wird dem Nutzer ein Vorschaubild und Kaufoptionen angezeigt. Der Nutzer kann entweder ein Einzelbild für 0,99€ kaufen oder ein Monatsabonnement über 6,99€ abschließen. Das Monatsabonnement erlaubt das unbegrenzte Erzeugen von HDR-Bildern. Hat der Nutzer eine Kaufoption gewählt, wird das Bild im Bilderordner des Mobiltelefons gespeichert. Besitzt der Nutzer bereits ein Monatsabonnement, soll keine Kaufoption sondern direkt eine Schaltfläche „HDR-Bild jetzt speichern“ angezeigt werden. Alle Bilder, sowohl die Eingabe- als auch die Ergebnisbilder, sollen dauerhaft auf dem Pear-Corp-Zentralserver zur Analyse der Nutzererfahrung gespeichert werden.

Aufgabe 1
Szenario

Die Startseite der Applikation soll eine Übersicht über das Nutzerkonto und die zuletzt erstellten HDR-Bilder bereitstellen. Per Klick auf ein bereits vorhandenes HDR-Bild wird eine Vorschau erzeugt und die Schaltflächen „Teilen“ und „Schließen“ werden angezeigt. „Schließen“ soll die Vorschau schließen und über „Teilen“ soll es möglich sein das Bild auf soziale Netzwerke wie Facezine oder Instagrim hochzuladen. Über die Schaltfläche „Neues HDR-Bild“ auf der Startseite gelangt der Nutzer zu einer Übersicht der auf dem Mobiltelefon gespeicherten Bilder. Diese Vorschauansicht soll in Echtzeit bis zu einhundert (100) Bilder auf einem Mittelklasse-Smartphone anzeigen können. Der Nutzer wählt drei (3) Bilder einer Bildreihe und bestätigt seine Auswahl mit einem Klick auf „HDR-Bild“ erstellen. Die Eingabebilder werden anschließend auf den Pear-Corp-Zentralserver hochgeladen und das HDR-Bild erzeugt. Die Erzeugung des Ergebnisbildes (HDR) soll nicht länger als sieben (7) Sekunden in Anspruch nehmen. Außerdem sollen mindestens eintausend (1000) Nutzer gleichzeitig mit dem Server kommunizieren können. Ist der Vorgang abgeschlossen, wird dem Nutzer die Schaltfläche „HDR-Bild abholen“ angezeigt. Per Klick auf diese Schaltfläche wird dem Nutzer ein Vorschaubild und Kaufoptionen angezeigt. Der Nutzer kann entweder ein Einzelbild für 0,99€ kaufen oder ein Monatsabonnement über 6,99€ abschließen. Das Monatsabonnement erlaubt das unbegrenzte Erzeugen von HDR-Bildern. Hat der Nutzer eine Kaufoption gewählt, wird das Bild im Bilderordner des Mobiltelefons gespeichert. Besitzt der Nutzer bereits ein Monatsabonnement, soll keine Kaufoption sondern direkt eine Schaltfläche „HDR-Bild jetzt speichern“ angezeigt werden. Alle Bilder, sowohl die Eingabe- als auch die Ergebnisbilder, sollen dauerhaft auf dem Pear-Corp-Zentralserver zur Analyse der Nutzererfahrung gespeichert werden.

Aufgabe 1
Gliederung

1. Zielbestimmung
2. Produkteinsatz
3. Funktionale Anforderungen
4. Produktdaten
5. Nichtfunktionale Anforderungen
6. Systemmodelle
   a) Szenarien
   b) Anwendungsfälle
7. Glossar (Begriffslexikon zur Beschreibung des Produktes)
8. Durchführbarkeitsanalyse (für Aufgabe 3)

Aufgabe 1 – 1. Zielbestimmung

- Die Firma Pear Corp. soll in die Lage versetzt werden
  - ihr Produkt iMage als Plattform für die Erzeugung und den Vertrieb von HDR-Bildern einzusetzen

Aufgabe 1 – 2. Produkteinsatz

- Das System soll als Klient-Dienstgeber-System implementiert werden.
  - Der Dienstgeber ist für die Berechnung der HDR-Bilder zuständig, soll als Web-Applikation zur Verfügung gestellt werden und auf dem Firmen-internen Servern der Pear Corp. laufen.
  - Die Klient-Anwendung soll als Smart-Phone-Applikation für unterschiedliche Betriebssysteme angeboten werden und lokal auf den Nutzer-Endgeräten laufen.

Aufgabe 1 – 3. Funktionale Anforderungen

- /FA10/ Übersicht anzeigen
  - /FA11/ HDR-Bild-Historie anzeigen
- /FA20/ Bild aus Bild-Historie auswählen
  - /FA21/ Schaltflächen „Teilen“ und „Schließen“ anzeigen
  - /FA22/ …
- /FA30/ Schaltfläche „Neues HDR-Bild“ anzeigen
- /FA40/ Auswahl der Schaltfläche „Neues HDR-Bild“ öffnet Übersicht verfügbarer Bilder
- /FA50/ Bildreihe auswählen
- /FA60/ HDR-Bild auf Pear-Corp.-Zentralserver erzeugen
- …

Aufgabe 1 – 4. Produktdaten

- /PD10/ Die erstellten HDR-Bilder müssen inklusive Meta-Daten in der Applikation gespeichert werden
- /PD20/ Vorschaubilder der HDR-Bilder müssen vorgehalten werden
- /PD30/ Die Abonnement-Informationen des Benutzerprofils müssen in der Applikation hinterlegt werden
- /PD40/ Das Standardverzeichnis für Bilder muss gespeichert werden
- …

Aufgabe 1 – 5. Nichtfunktionale Anforderungen

- /NF10/ Die Übersicht des Bilderordners (/FA40/) muss in Echtzeit bei bis zu 100 Bildern auf einem Mittelklasse-Smartphone erzeugt werden
- /NF20/ Das Erzeugen des HDR-Bildes (/FA60/) darf nicht länger als 7 Sekunden benötigen
- /NF30/ Die Kommunikationsschnittstelle des Pear-Corp.-Zentralservers muss mindestens 1000 gleichzeitige Verbindungen ermöglichen
- …

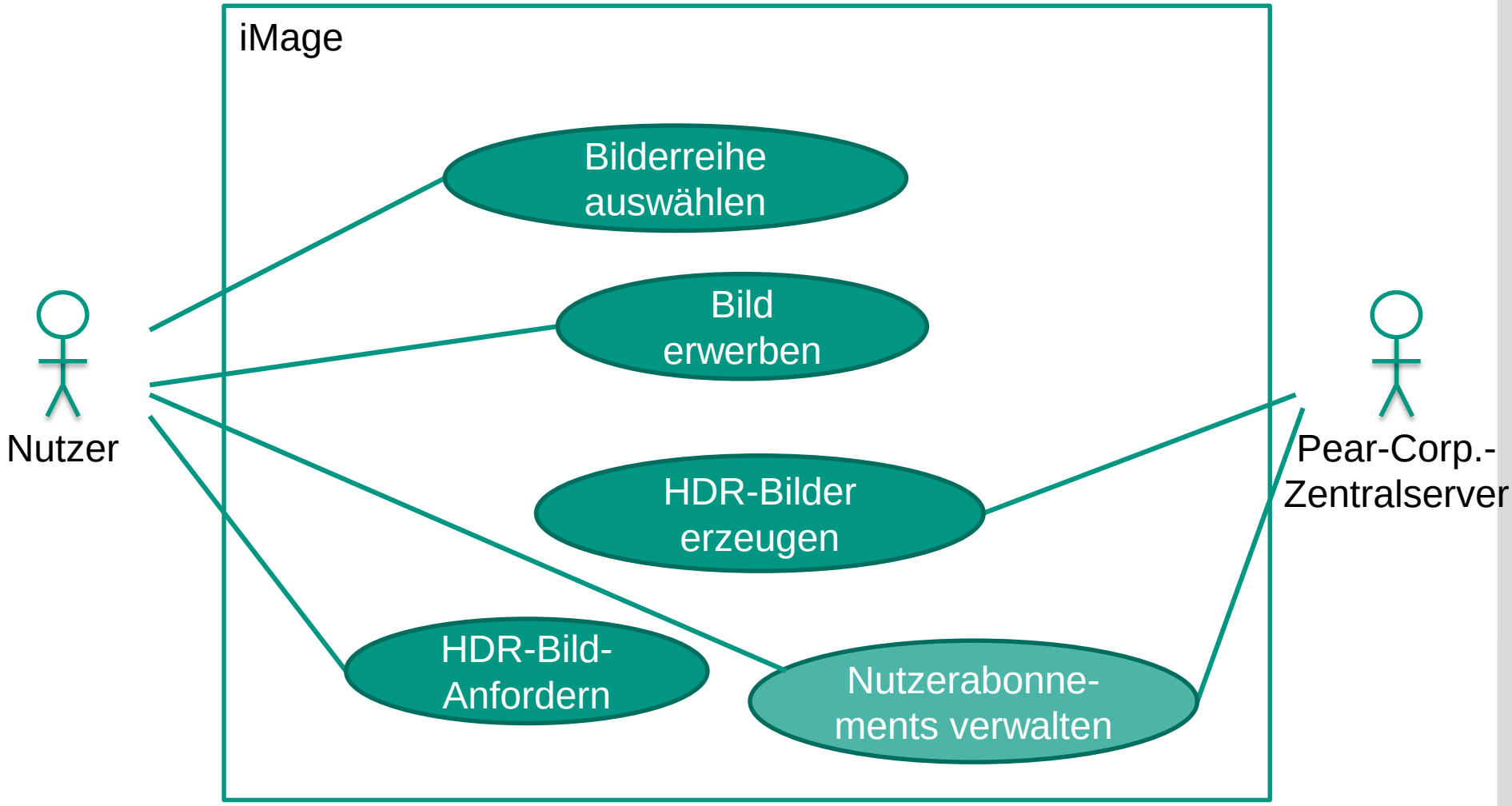
Aufgabe 1 – 6.a Szenarien

- Anfordern eines HDR-Bildes
  Der Nutzer ruft über die Schaltfläche „*Neues HDR-Bild*“ eine Auswahlübersicht der auf dem Gerät gefundenen Bilder auf. Dort wählt er drei Bilder einer Bildreihe und bestätigt seine Auswahl durch einen Klick auf „*HDR-Bild erstellen“.* Die Bildreihe wird an den Pear-Corp.-Zentralserver übermittelt.
  […]

Hier könnte noch beschrieben werden, was Nutzer-seitig nach der Berechnung des Bildes auf dem Server geschieht.

- Erzeugung eines HDR-Bildes
  Der Pear-Corp.-Zentralserver nimmt die Anfrage des Klients zur Berechnung eines HDR-Bildes entgegen. Aus der übermittelten Bildreihe wird mit HDrize ein HDR-Bild erzeugt. Die Bildreihe und das Ergebnisbild werden in der Datenbank abgelegt. Das HDR-Bild anschließend an den Klienten übermittelt.

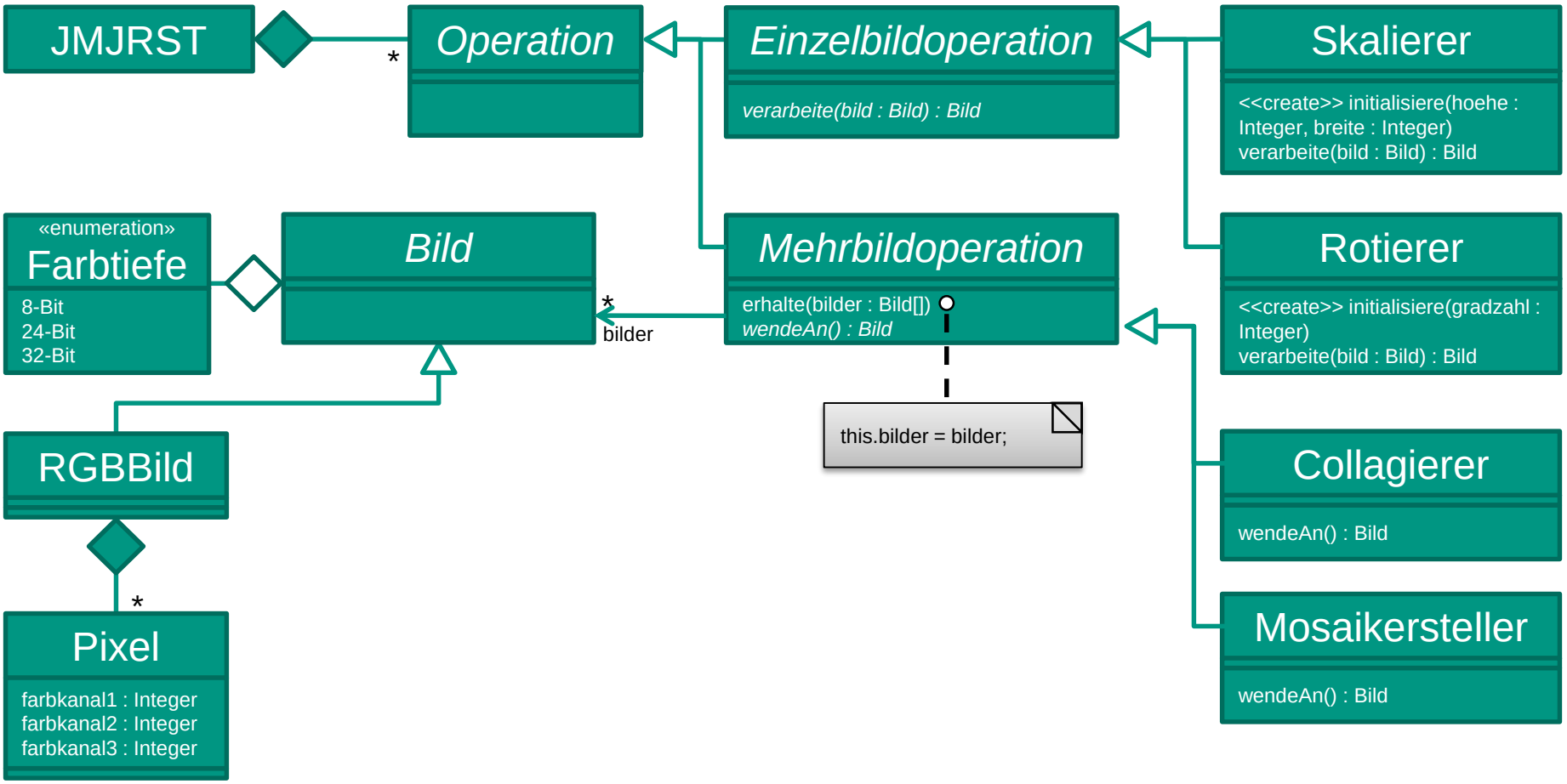
Aufgabe 1 – 6.b Anwendungsfalldiagramm



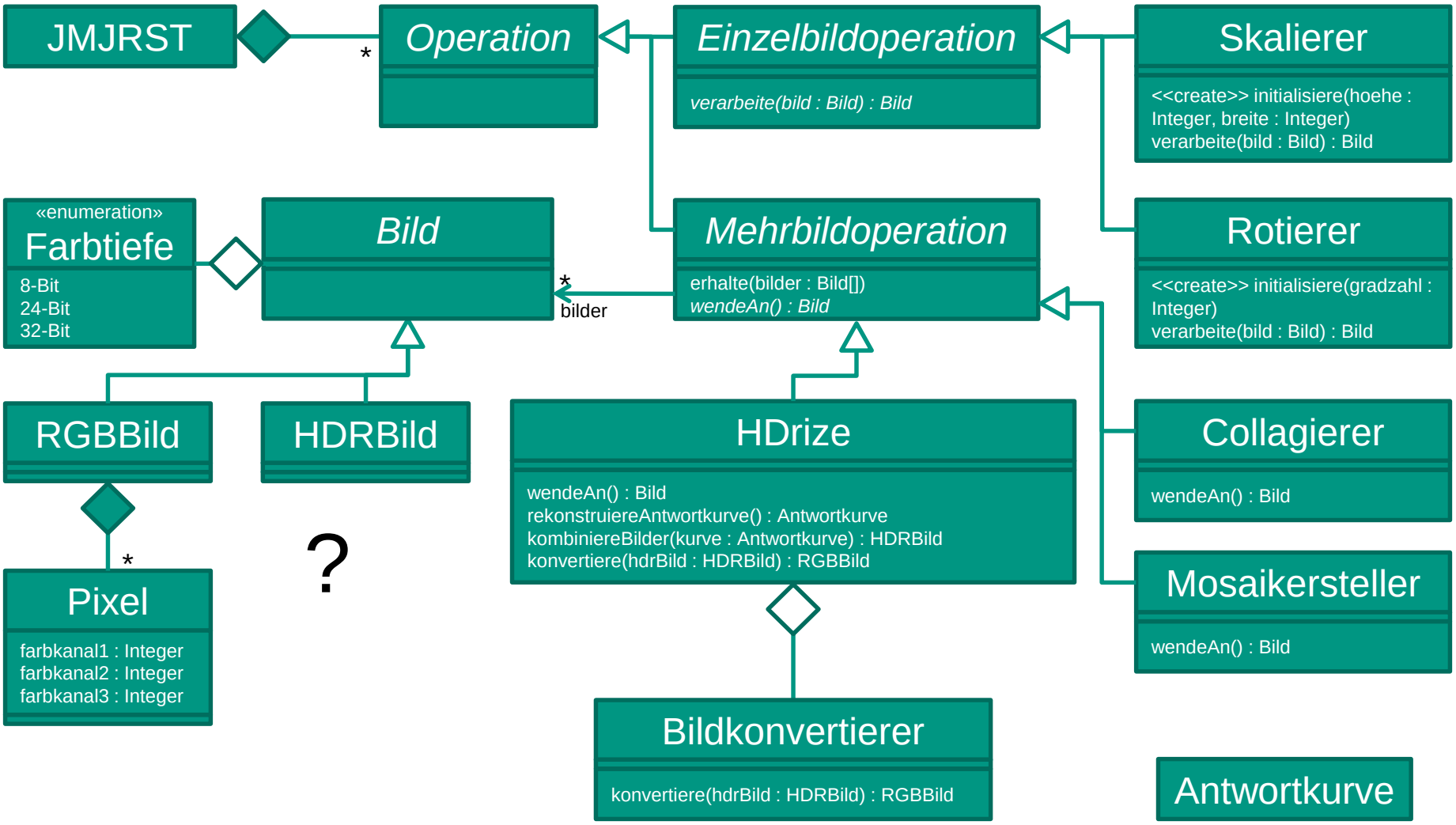
Aufgabe 1 – 7. Glossar

- HDR-Bild
  - Bild mit hohem Dynamikumfang.
- Bildreihe
  - Menge von Bildern mit gleichem Motiv aber unterschiedlichen Aufnahmeparametern, z.B. Belichtungszeit, Blende oder Sensor-Empfindlichkeit.
- Vorschaubild
  - HDR-Bild mit verringerter Auflösung zur Darstellung in Nutzer-Applikation.
- …

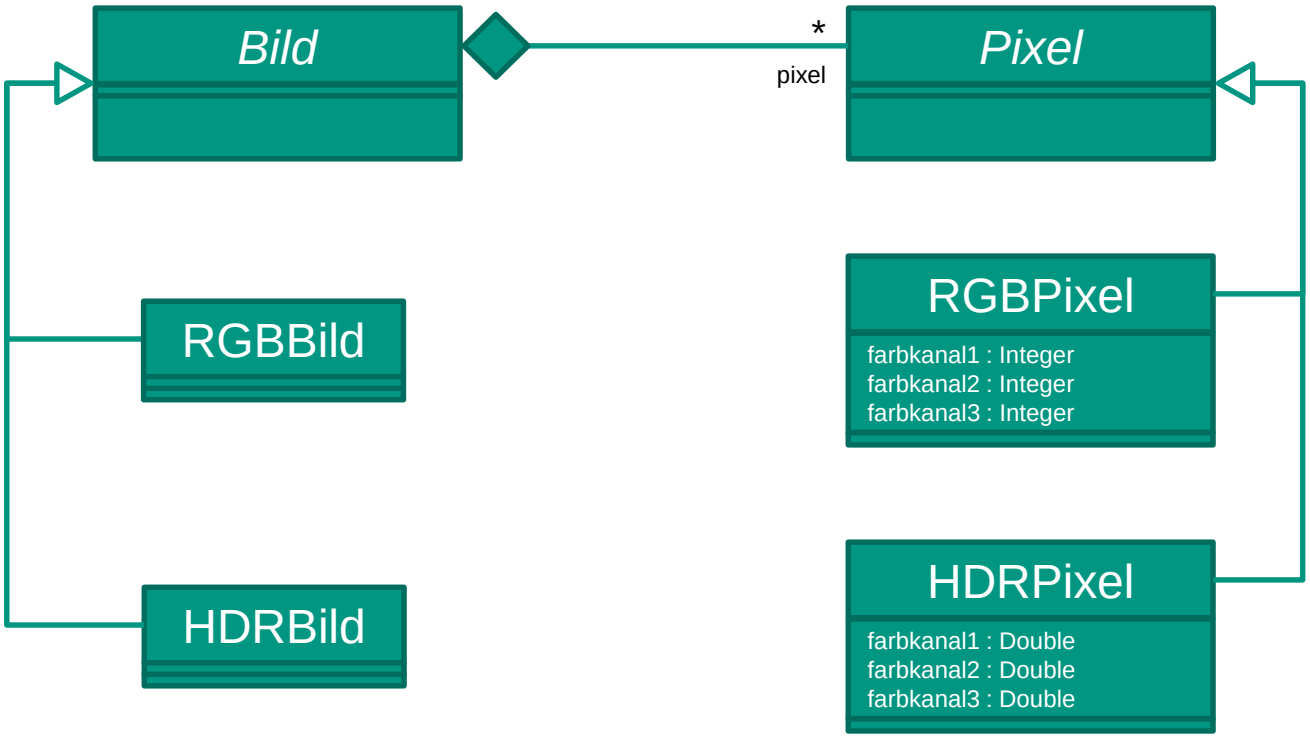
Aufgabe 2 – Klassendiagramm Teil 1



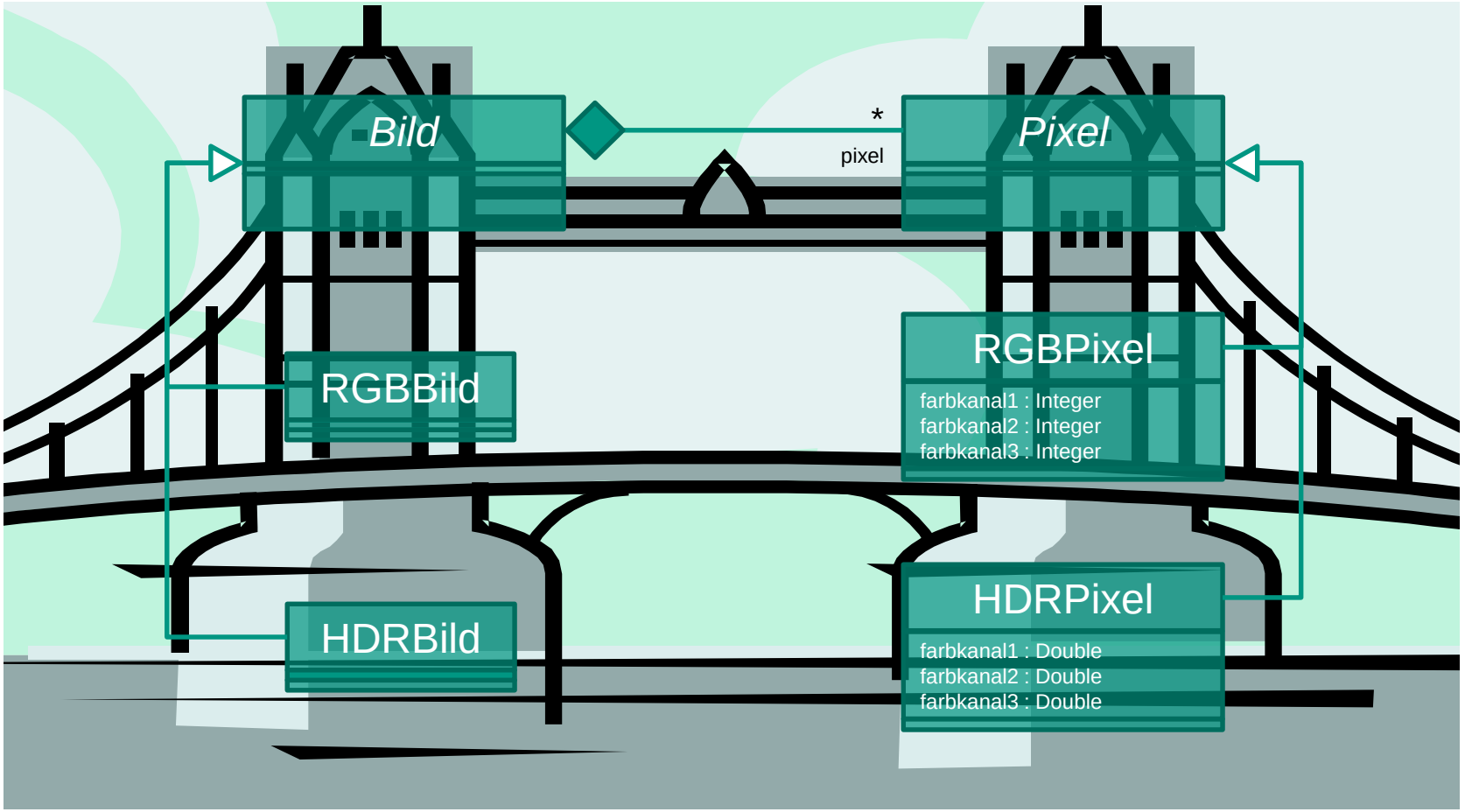
Aufgabe 2 – Klassendiagramm Teil 2



Aufgabe 2 – Klassendiagramm Teil 2



Aufgabe 2 – Klassendiagramm Teil 2



Aufgabe 3 – Durchführbarkeitsuntersuchung

- Fachliche Qualifikation vorhanden
  - M.Sc. mehrfach im Haus vorhanden
  - Technische Infrastruktur vorhanden oder leicht beschaffbar
- Alternative Lösungsvorschläge
  - Verwendung vorhandener Bildverarbeitungsalgorithmen möglich
- Prüfen der personellen Durchführbarkeit
  - Fachkräfte in andere Projekte eingebunden; Umpriorisierung/Neueinstellungen nötig
- Prüfen der Risiken
  - Kunden stark interessiert an kontrastreichen Bildern
  - Starke Konkurrenz durch bereits etablierte Bildverarbeitungsprodukte
- Prüfen der ökonomischen Durchführbarkeit
  - Aufwands- und Terminschätzung: Aufwand machbar bis zur Veröffentlichung
  - Kosten der Entwicklung (keine, da nur Studenten im Übungsbetrieb) vs. erwarteter Nachfrage
- Rechtliche Gesichtspunkte
  - Patente der Firma Abode prüfen und ggf. ankaufen

Aufgabe 4 – Matrix-Implementierung

```
public final class Matrix implements IMatrix {
  private final int rows;
  private final int cols;
  private final double[][] data;

  /**
   * Create a new matrix.
   *
   * @param mtx
   *          the original matrix
   */
  public Matrix(IMatrix mtx) {
    this(mtx.copy());
  }

 /**
   * Create a matrix (only zeros).
   *
   * @param rows
   *          the amount of rows
   * @param cols
   *          the amount of columns
   */
  public Matrix(int rows, int cols) {
    if (rows < 1 || cols < 1)
      throw new IllegalArgumentException("rows and cols have to be >= 1");
    this.rows = rows;
    this.cols = cols;
    this.data = new double[rows][cols];
  }
  //...
```

copy() auch Teil der Schnittstelle, Implementierung folgt später!

Fange ungültige Reihen- und Spaltenangaben ab!

Aufgabe 4 – Matrix-Implementierung (2)

```
public final class Matrix implements IMatrix {
  //...
 /**
   * Create a new matrix.
   *
   * @param mtx
   *          the original matrix mtx[Rows][Cols]
   */
  public Matrix(double[][] mtx) {
    this(mtx.length, mtx[0].length);
    for (int r = 0; r < this.rows; r++)
      for (int c = 0; c < this.cols; c++)
        this.set(r, c, mtx[r][c]);
  }

  @Override
  public double[][] copy() {
    double[][] copy = new double[this.rows][this.cols];
    for (int r = 0; r < this.rows; r++)
      for (int c = 0; c < this.cols; c++)
        copy[r][c] = this.data[r][c];
    return copy;
  }
  //...
```

Nutzt anderen Konstruktor und befüllt die Zellen

Aufgabe 4 – Matrix-Implementierung (3)

```
public final class Matrix implements IMatrix {
  //...
 @Override
  public int rows() {
    return this.rows;
  }

  @Override
  public int cols() {
    return this.cols;
  }

  @Override
  public void set(int r, int c, double v) {
    this.data[r][c] = v;

  }

  @Override
  public double get(int r, int c) {
    return this.data[r][c];
  }
}
```

Aufgabe 4 – Matrix-Berechnungen

```
public final class MatrixCalculator implements IMatrixCalculator<Matrix> {
 @Override
  public Matrix multiply(Matrix a, Matrix b) {
    if (a.cols() != b.rows())
      return null;

    Matrix res = new Matrix(a.rows(), b.cols());

    for (int c = 0; c < a.rows(); c++) {
      for (int d = 0; d < b.cols(); d++) {
        double sum = 0;
        for (int k = 0; k < b.rows(); k++) {
          sum = sum + a.get(c, k) * b.get(k, d);
        }

        res.set(c, d, sum);
      }
    }
    return res;
  }

  @Override
  public Matrix transpose(Matrix mtx) {
    Matrix res = new Matrix(mtx.cols(), mtx.rows());
    for (int r = 0; r < res.rows(); r++)
      for (int c = 0; c < res.cols(); c++)
        res.set(r, c, mtx.get(c, r));
    return res;
  }
  //...
}
```

Vorbedingung prüfen

Aufgabe 4 – Matrix-Berechnungen (2)

```
public class MatrixCalculator implements IMatrixCalculator<Matrix> {
  //...
  @Override
  public Matrix inverse(Matrix mtx) {
    // 0. Prüfe ob Matrix quadratisch
    // 1. Lege Hilfsmatrix der Größe n x 2n an (n = Spalten-/Reihenanzahl
          der Eingabematrix)
    // 2. Belege linke Hälfte der Hilfsmatrix mit der Eingabematrix
    // 3. Belege die Diagonale der rechten Hälfte der Hilfsmatrix mit 1
    // 4. Löse Gleichungssystem
    // 5. Lege Ergebnismatrix an und schreibe Ergebnis des Gleichungsystem
          in diese
    // 6. Berechne Produkt von Original- und Ergebnismatrix
    // 7. Wenn Produkt der Einheitsmatrix entspricht, gib Ergebnismatrix zurück
  }
}
```

Implementierung siehe Musterlösung

Aufgabe 4 – HDR-Bild zusammensetzen

```
public class HDRCombine implements IHDRCombine {
  @Override
  public float[] calculateWeights() {
    final int offset = 20;

    float[] weight = new float[256];
    weight[0] = 1f;

    for (int a = 1; a < offset; a++) {
      weight[a] = (float) (weight[a - 1] + (1.0 / offset));
    }
    for (int a = offset; a < (256 - offset); a++) {
      weight[a] = 2;
    }

    for (int a = 256 - offset; a < 256; a++) {
      weight[a] = weight[256 - a - 1];
    }

    return weight;
  }
  //...
}
```

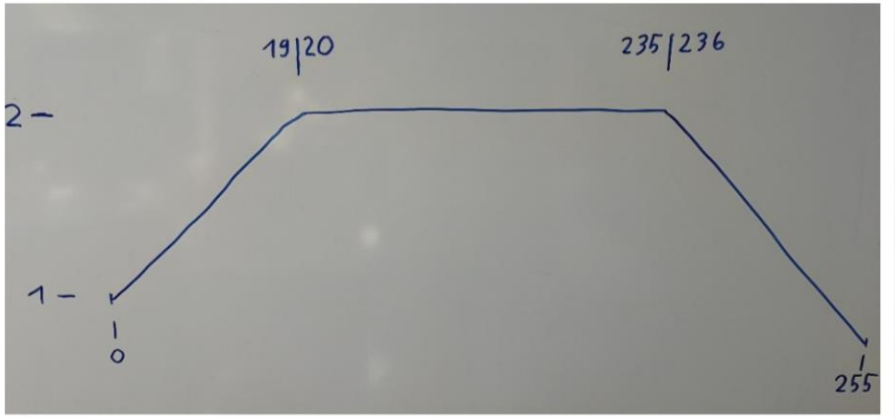
Setze Offset für Randglättung

erstes Gewicht = 1

Berechne erste 20 Werte

Mittlere Werte sind alle 2

Letzte 20 Werte wie erste 20



Aufgabe 4 – HDR-Bild zusammensetzen (2)

```
public class HDRCombine implements IHDRCombine {
  //...
@Override
  public HDRImage createHDR(ICameraCurve curve, EnhancedImage[] imageList) {
    int width = imageList[0].getWidth();
    int height = imageList[0].getHeight();

    float[][][] result = new float[][][] {
        new float[height][width],
        new float[height][width],
        new float[height][width]
    };

    float[] w = this.calculateWeights();
    //...
}
```

Selbiges für:
`float[][][] numerator` und
`float[][][] denominator`

Speicherplätze für:

$$hdr(x, y, kanal) = \frac{wert(x, y, kanal)}{norm(x, y, kanal)}$$

Aufgabe 4 – HDR-Bild zusammensetzen (3)

```
public class HDRCombine implements IHDRCombine {
  //...
@Override
  public HDRImage createHDR(ICameraCurve curve, EnhancedImage[] imageList) {
    //...
    for (EnhancedImage ei : imageList) {
      float exposure = ei.getExposureTime();

      for (int x = 0; x < width; x++) {
        for (int y = 0; y < height; y++) {
          int[] color = ei.getRGB(x, y);
          float[] resp = curve.getResponse(color);

          for (int c = 0; c < HDRCombine.CHANNELS; c++) {
            numerator[c][y][x] += w[color[c]] * resp[c] / exposure;
            denominator[c][y][x] += w[color[c]];
          }

        }
      }
    }
    //...
}
```

Berechnung pro Kanal (in diesem Fall 3)

Berechnung von:

$$wert(x, y, kanal) = \sum_{n=1}^{3} \frac{gewicht\left(rgb_{kanal}\left(x_{bild_n}, y_{bild_n}\right)\right) * kurve\left(rgb_{kanal}\left(x_{bild_n}, y_{bild_n}\right)\right)}{belichtungzeit_{bild_n}}$$

$$norm(x, y, kanal) = \sum_{n=1}^{3} gewicht\left(rgb_{kanal}\left(x_{bild_n}, y_{bild_n}\right)\right)$$

Aufgabe 4 – HDR-Bild zusammensetzen (3)

```
public class HDRCombine implements IHDRCombine {
  //...
@Override
  public HDRImage createHDR(ICameraCurve curve, EnhancedImage[] imageList) {
    //...
    for (int x = 0; x < width; x++) {
      for (int y = 0; y < height; y++) {
        for (int c = 0; c < HDRCombine.CHANNELS; c++) {
          result[c][y][x] = numerator[c][y][x] / denominator[c][y][x];
        }
      }
    }

    return new HDRImage(resu
  }
}
```

Berechnung von:

$$hdr(x, y, kanal) = \frac{wert(x, y, kanal)}{norm(x, y, kanal)}$$

Aufgabe 5 – commons-cli in Maven einbinden

```
[...]
<dependencies>
  <!-- Abhängigkeit für commons-cli -->
  <dependency>
    <groupId>commons-cli</groupId>
    <artifactId>commons-cli</artifactId>
    <!-- Version ist im uebungsparent vorgegeben
    <version>1.4</version> -->
  </dependency>
  [...]
</dependencies>
[...]
```

Junit analog einfügen:
GroupID: `org.junit`
ArtifactID: `junit`

Aufgabe 5 – Klassenpfad per Maven anpassen

```
[...]
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      [...]
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <classpathPrefix>lib/</classpathPrefix>
          </manifest>
        </a
      </con
[...]
```

Führt zu folgendem Zusatzeintrag in der Manifest-Datei:

```
Class-Path: lib/commons-cli/commons-cli/1.4/commons-cli-1.4.jar
lib/swt1/iMage.HDrize/0.0.1-SNAPSHOT/iMage.HDrize-0.0.1-
SNAPSHOT.jar …
```

Aufgabe 5 – Abhängigkeiten automatisch kopieren

```
[...]
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-dependency-plugin</artifactId>
  <executions>
    <execution>
      <id>copy-dependencies</id>
      <phase>package</phase>
      <goals>
        <goal>copy-dependencies</goal>
      </goals>
      <configuration>
        <!-- nächste Folie -->
      </configuration>
    </execution>
[...]
```

Aufgabe 5 – Abhängigkeiten automatisch kopieren

```
<!-- Bibliotheken nach target/lib kopieren -->
<outputDirectory>
  ${project.build.directory}/lib
</outputDirectory>

<!- Kopiere alles, was zum Übersetzen nötig war
    (aber nicht die Test-Bibliotheken) -->
<includeScope>
  runtime
</includeScope>
```

Sammelt die Bibliotheken ein:

```
projekt/target#$ ls lib/

commons-cli-1.4.jar
iMage.HDrize-0.0.1-SNAPSHOT.jar
iMage.Hdrize.base-0.0.1-SNAPSHOT.jar
```

Aufgabe 5 – Abhängigkeit zu iMage.HDrize

```
<dependencies>
        <!-- andere Abhängigkeiten -->
        <dependency>
                <groupId>swt1</groupId>
                <artifactId>iMage.HDrize</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </dependency>
</dependencies>
```

Wie in Aufgabe 4 definiert:

```
<groupId>swt1.ub2.a4</groupId>
<artifactId>133742</artifactId>
```

Aufgabe 5 – Kommandozeilenprogramm

```
public static void main(String[] args) {
    CommandLine cmd = null;
    try {
      cmd = App.doCommandLineParsing(args);
    } catch (ParseException e) {
      System.err.pri                                    ());
      System.exit(1);
    }

    double lambda = cmd.hasOption(App.CMD_OPTION_LAMBDA)
        ? Double.parseDouble(cmd.getOptionValue(App.CMD_OPTION_LAMBDA))
        : App.DEFAULT_LAMBDA;

    if (lambda <= 0 || lambda > 100) {
      System.err.println("Lambda is invalid: " + lambda);
      System.exit(1);
    }

    int samples = cmd.hasOption(App.CMD_OPTION_SAMPLES)
        ? Integer.parseInt(cmd.getOptionValue(App.CMD_OPTION_SAMPLES))
        : App.DEFAULT_SAMPLES;

    if (samples < 1 || samples > 1000) {
      System.err.println("Invalid number of samples: " + samples);
      System.exit(1);
    }
    //…
```

Lesen der Kommandozeilen-Parameter

Kurzschreibweise: if(a) { x = y; } else{ x = z; }

Wenn Lambda gegeben, dann lesen

Fehler: Lambda muss zwischen 0 und 100 sein

Wenn Stichprobenanzahl gegeben, dann lesen

Fehler: Stichprobenanzahl muss zwischen 1 und 1000 liegen

Aufgabe 5 – Kommandozeilenprogramm

```
public static void main(String[] args) {
    //…
    File[] input = null;
    try {
      input = App.processInputFiles(cmd.getOptionValue(App.CMD_OPTION_INPUT_IMAGES));
    } catch (IOException e) {
      System.err.println(e.getMessage());
      System.exit(1);
    }

    EnhancedImage[] images = App.toEnhancedImages(input);

    IHDrize<Matrix> hdrize = new HDrize();
    BufferedImage hdr = hdrize.createRGB(images, samples, lambda, new MatrixCalculator());
    if (hdr == null) {
      System.err.println("So   rror occured while creating hdr image");
      System.exit(1);
    }

    File output = null;
    try {
      output = App.ensureFile(cmd.getOptionValue(App.CMD_OPTION_OUTPUT_IMAGE), true);
      ImageIO.write(hdr, "png", output);
    } catch (IOException e) {
      System.err.println("Could not save image: " + e.getMessage());
      System.exit(1);
    }
}
```

Lesen der Eingabedateien (Hilfsmethode siehe nächste Folie)

Erzeuge EnhancedImages aus Eingabedateien

Erzeuge HDR-Bild mit HDrize

Hilfsmethode: *„Ensure that a file exists (or create if allowed by parameter).” (siehe nächste Folie)*

Ergebnis-Bild speichern

Aufgabe 5 – Kommandozeilenprogramm

```
private static File ensureFile(String path, boolean create) throws IOException {
   File file = new File(path);
   if (file.exists()) {
     return file;
   }
   if (create) {
     file.createNewFile();
     return file;
   }

   // File not available
   throw new IOException("The specified file does not exist: " + path);
 }
```

Wenn die Datei existiert: Gib sie zurück!

Andernfalls, wenn die Erzeugung erlaubt ist (z.B. für Ausgabedateien): Erzeuge eine neue Datei!

Aufgabe 5 – Kommandozeilenprogramm

```
private static File[] processInputFiles(String dir) throws
   File directory = App.ensureFile(dir, false);
   List<File> jpgs = new ArrayList<>();
   for (File file : directory.listFiles(f -> f.getName().endsWith(".jpg")))
     jpgs.add(file);

   if (jpgs.size() % 2 == 0 || jpgs.size() <= 1) {
     System.err.println("Found " + jpgs.size() + " files. This isn't an odd value.");
     System.exit(1);
   }
   File[] result = jpgs.toArray(File[]::new);
   for (File image : result) {
     String name = image.getName();
     name = name.substring(0, name.length() - ".jpg".length());
     if (name.length() < 3 || !result[0].getName().startsWith(name.substring(0, 3))) {
       System.err.println("Naming violation: " + image.getName() + " & "
           + result[0].getName());
       System.exit(1);
     }
   }

   return result;
 }
```

Lese alle *.jpg Dateien ein

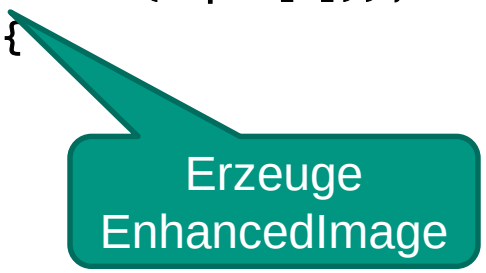
Überprüfe ungerade Anzahl an Bildern

Überprüfe Präfix

Aufgabe 5 – Kommandozeilenprogramm

```
private static EnhancedImage[] toEnhancedImages(File[] input) {
    EnhancedImage[] result = new EnhancedImage[input.length];
    for (int i = 0; i < result.length; i++) {
      try {
        result[i] = new EnhancedImage(new FileInputStream(input[i]));
      } catch (ImageReadException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
      }
    }

    return result;
  }
```



Aufgabe 6 – Schnittstelle für Filter in iMage

iMage soll in die Lage versetzt werden eine Menge von Filtern auf Bilder anzuwenden. Es sollen sowohl einzelne Bilder als auch Mengen von Bildern verarbeitet werden können. Filter werden auf genau ein Bild angewendet. Das Originalbild soll unverändert bleiben; stattdessen soll eine Kopie zurückgeliefert werden. Die Filter-Menge soll wiederverwendbar sein, d.h. dieselbe Filter-Menge soll nacheinander auf unterschiedliche Bilder oder Bild-Mengen angewendet werden können. Die Menge der Filter soll zwischen den Anwendungen verändert werden können.

Aufgabe 6 – Schnittstelle für Filter in iMage

iMage soll in die Lage versetzt werden eine Menge von Filtern auf Bilder anzuwenden. Es sollen sowohl einzelne Bilder als auch Mengen von Bildern verarbeitet werden können. Filter werden auf genau ein Bild angewendet. Das Originalbild soll unverändert bleiben; stattdessen soll eine Kopie zurückgeliefert werden. Die Filter-Menge soll wiederverwendbar sein, d.h. dieselbe Filter-Menge soll nacheinander auf unterschiedliche Bilder oder Bild-Mengen angewendet werden können. Die Menge der Filter soll zwischen den Anwendungen verändert werden können.

Alles wichtig! ☺

Aufgabe 6 – Schnittstelle für Filter in iMage

```
public interface IFilter {

        public BufferedImage apply(BufferedImage image);

}
```

JavaDoc: *„… creates a new image…“*

„Filter werden auf genau ein Bild angewendet.“

„Das Originalbild soll unverändert bleiben; stattdessen soll eine Kopie zurückgeliefert werden.“

Aufgabe 6 – Schnittstelle für Filter in iMage

```
public interface IPipeline {

  public void addFilter(IFilter filter);

  public void removeFilter(IFilter filter);

  public void setFilterList(List<IFilter> filterList);

  public BufferedImage apply(BufferedImage image);

  public Collection<BufferedImage> applyBatch(Collection<BufferedImage> image);

}
```

„Die Menge der Filter soll zwischen den Anwendungen verändert werden können.“

„iMage soll in die Lage versetzt werden eine Menge von Filtern auf Bilder anzuwenden. Es sollen sowohl einzelne Bilder als auch Mengen von Bildern verarbeitet werden können.“

„Die Filter-Menge soll wiederverwendbar sein […]“