Karlsruher Institut für Technologie

Fakultät für Informatik
Institut für Programmstrukturen und Datenorganisation (IPD)

Lehrstuhl Prof. Walter F. Tichy

E-Mail: swt1@ipd.kit.edu

Ilias: https://ilias.studium.kit.edu/goto.php?target=crs_945596&client_id=produktiv

Web: http://ps.ipd.kit.edu

LEZ: https://lez.ipd.kit.edu

SWT1: Übungsblatt 2

Anforderungsanalyse, UML, HDR, Kommandozeile, Schnittstellen

Ausgabe: Mittwoch, 8. Mai 2019
Abgabe: Mittwoch, 22. Mai 2019, 12:00 Uhr

Geben Sie Ihre Lösung mit Deckblatt (mit Name, Matrikelnummer und der Nummer Ihres Tutoriums deutlich lesbar) ab, damit Ihr Tutor Korrekturhinweise und Ihre Punkte notieren kann. Werfen Sie es in die Holzkiste vor Raum 369 im Informatik-Hauptgebäude 50.34. Verwenden Sie ausschließlich das Deckblatt zur SWT1 aus ILIAS.

Alle Theorie-Aufgaben inklusive aller Diagramme sind handschriftlich anzufertigen!
Ausgeschlossen sind auch kopierte oder ausgedruckte „handschriftliche“ Lösungen!

Aufgabe 1: Erstellung eines Lastenhefts (6 Punkte)

Die Pear Corp möchte iMage als App zur Erzeugung von HDR-Bildern vermarkten. Hierzu hat Ihre Projektleiterin ein Szenario verfasst, das die gewünschte Funktionsweise von iMage beschreibt:

> *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.*

Erstellen Sie ein Lastenheft und geben Sie mindestens 5 funktionale Anforderungen, 3 Produktdaten und 3 nichtfunktionale Anforderungen an. Das Lastenheft soll mindestens ein Anwendungsfalldiagramm enthalten. (Schreiben Sie zu jedem Kapitel der Lastenheftvorlage etwas.)
Das Szenario ist nicht vollständig und dadurch teilweise unpräzise. Es soll von Ihnen plausibel vervollständigt werden. Ihre Aufgabe ist es, Ihrer Projektleiterin das fertige Dokument als Entscheidungsgrundlage zu übergeben. Es werden in dieser Aufgabe daher nicht nur Punkte für die Anforderungen des Lastenhefts vergeben, sondern auch für die Plausibilität und die äußere Form des Dokuments. Das Dokument soll 5 Seiten (zzgl. Abbildungen und Glossar) nicht überschreiten.
Verwenden Sie für das Lastenheft die Vorlage aus ILIAS. Laden Sie die benötigten Dateien herunter und legen Sie dafür ein neues Verzeichnis `image.lastenheft` in Ihrem SWT1-Git-Depot an. Verfahren Sie mit der enthaltenen `.gitignore`-Datei ebenso, wie auf Übungsblatt 1 beschrieben.
Geben Sie einen Ausdruck Ihres Lastenheftes ab. (Drucken Sie wenn möglich beidseitig aus!) Dies ist die einzige Ausnahme zur Regelung der handschriftlichen Abgaben.

Aufgabe 2: Klassendiagramm (4 + 3 Punkte)

Ihre Projektleiterin wurde vom Vorstand der Pear Corp. mit der Umsetzung des Projektes iMage beauftragt. Da ihre anstehende Beförderung vom Erfolg des Projektes abhängt, möchte sie, dass zunächst der aktuelle Zustand des Systems als UML-Klassendiagramm festgehalten wird. Zur Erinnerung: iMage basiert auf JMJRST (siehe Übungsblatt 1).
Beachten Sie: Bitte verwenden Sie zur Bearbeitung der Aufgabenteile a) und b) unterschiedlich-farbige Stifte (z.B. blau für Teil a) und schwarz für Teil b)). Verwenden Sie keine roten Stifte.

a) Erstellen Sie basierend auf der folgenden Beschreibung ein UML-Klassendiagramm:

> *JMJRST enthält verschiedene Operationen. JMJRST unterscheidet bei Operationen zwischen Einzelbildoperationen und Mehrbildoperationen. Bei Einzelbildoperationen wird eine Verarbeitung auf einem Bild durchgeführt und das resultierende Bild zurückgegeben. Bereits vorhanden sind die Einzelbildoperationen „Rotieren“ und „Größe Ändern“, welche das Bild um eine gegebene Gradzahl drehen bzw. auf gegebene Höhe und Breite skalieren. In vorangehenden Erweiterungen von JMJRST durch die Pear Corp. wurden bereits die Mehrbildoperationen „Collage erstellen“ und „Mosaik erstellen“ umgesetzt. Mehrbildoperationen erhalten eine Menge von Bildern zur Verarbeitung. Auf diesen wenden sie ihre Operation an und geben das Ergebnis als neues Bild zurück. JMJRST verwendet RGB-Bilder, welche pro Pixel drei ganzzahlige Farbkanäle besitzen. Außerdem können Bilder in unterschiedlichen Farbtiefen vorliegen (8-Bit, 24-Bit, 32-Bit).*

Nun soll iMage so erweitert werden, dass dem Grundsystem JMJRST eine neue Mehrbildoperation HDrize hinzugefügt wird, die aus Bildern unterschiedlicher Belichtungszeiten ein Hochkontrastbild (HDR-Bild) zusammensetzt.

b) Erweitern Sie Ihr UML-Klassendiagramm gemäß der nachfolgenden Beschreibung:

> *HDrize ist eine Mehrbildoperation, welche mithilfe der folgenden drei Schritte ein HDR-Bild erstellt. HDrize rekonstruiert die Antwortkurven der Kamera aus den Eingabebildern. Mithilfe der Antwortkurven werden die Eingabebilder zu einem HDR-Bild kombiniert. Außerdem kann das*

kombinierte HDR-Bild durch eine Abbildung des Farbraumes in ein RGB-Bild konvertiert werden. Für den letzten Schritt benutzt HDrize einen Bildkonvertierer. Der Bildkonvertierer ist in der Lage ein HDR-Bild in ein RGB-Bild umzuwandeln. HDR-Bilder unterscheiden sich von RGB-Bildern dadurch, dass sie anstatt ganzzahliger Farbkanäle, Farbkanäle mit Gleitkommazahlen besitzen.

Verwenden Sie in Ihrer Modellierung UML-Elemente des Klassendiagramms; z.B. Klassen, Schnittstellen, Vererbungs- und Implementierungsbeziehungen, Assoziationen samt Multiplizitäten, Aggregationen und Kompositionen. Verzichten Sie auf OCL. Reichern Sie die Klassen um Attribute und Methodennamen an, soweit sie für die Modellierung der Beschreibung gebraucht werden.
Zeichnen Sie das Diagramm als Übung für die Klausur von Hand.

Aufgabe 3: Durchführbarkeitsuntersuchung (3 Bonuspunkte)

Da Ihre Projektleiterin auf die Beförderung angewiesen ist (die Anzahlung für ihre Villa an der Côte d'Azur ist demnächst fällig), möchte Sie nicht leichtfertig Ihren Erweiterungen, wie in Aufgabe 2 beschrieben, zustimmen. Der weite Weg bis ins mittlere Management hat ihr jedwede Fantasie für Neuerung geraubt und folglich zweifelt sie an der Umsetzbarkeit. Sie sind weiterhin von HDrize überzeugt und bieten ihr an, eine kurze Durchführbarkeitsuntersuchung des Projektes anzufertigen.
Wenden Sie für Ihre Durchführbarkeitsuntersuchung die sechs aus der Vorlesung bekannten Kriterien auf das Szenario aus Aufgabe 2 an. Setzen Sie die Kriterien in Bezug zum Szenario; Nennung von Stichworten alleine genügt nicht.
Fügen Sie die Durchführbarkeitsuntersuchung als zusätzlichen Abschnitt an das Lastenheft von Aufgabe 1 an. (Wenn Sie Aufgabe 1 nicht bearbeiten möchten, geben Sie eine ausgedruckte Version der Durchführbarkeitsuntersuchung ab.)

Aufgabe 4: HDrize für iMage (7 Punkte)

Ihre umfangreiche und detaillierte Durchführbarkeitsuntersuchung hat Ihre Projektleiterin und auch den Vorstand überzeugt; HDrize für iMage soll tatsächlich umgesetzt werden. „Da Ihnen so viel an dem Projekt liegt, möchten Sie sich doch bestimmt persönlich um dessen Umsetzung kümmern?", fragt Sie Ihre Projektleiterin. Sie erkennen sofort, dass es sich um eine Suggestivfrage handelt und beginnen zähneknirschend mit der Zusammenstellung eines Teams für die Implementierung. Uneigennützig ernennen Sie sich selbst zum Teamleiter. Die Aufgabe Ihres Teams ist es, eine Bibliothek zu schreiben, welche die Funktionalität von HDrize implementiert. Die HDrize-Bibliothek soll eine Bildreihe entgegennehmen und daraus ein HDR-Bild erzeugen. Das Ergebnisbild sieht in etwa wie folgt aus:

Sie dürfen, müssen aber nicht, Teile Ihres Klassendiagramms aus Aufgabe 2 verwenden. Beachten Sie: Aufgrund der Vorgaben für die Aufgaben 4 und 5 (siehe unten) ergeben sich zwingend Abweichungen vom Klassendiagramm in Aufgabe 2. Verwenden Sie keine (Grafik-)Bibliotheken, sondern implementieren Sie alle benötigten Operatoren selbst. Beachten Sie für die Bearbeitung von Aufgabe 4 (und für Aufgabe 5) folgendes:

- Führen Sie Versionskontrolle, benutzen Sie Maven und CheckStyle, wie Sie es auf Übungsblatt 1 erlernt haben. Dazu gehören sinnvolle Einbuchungen mit entsprechenden Nachrichten.
- Verzichten Sie auf eine graphische Oberfläche. Sie müssen lediglich von der Kommandozeile die Argumente übernehmen (siehe Aufgabe 5).
- Verwenden Sie die Klassen und Schnittstellen aus dem ILIAS, welche die Struktur vorgeben. Vervollständigen Sie die Klassenrümpfe. Ändern Sie keine Namen der vorgegebenen Schnittstellen und Klassen!

In dieser und der folgenden Aufgabe werden die einzelnen Programmteile modularisiert: Die Erzeugung der HDR-Bilder wird in einer Bibliothek (d.h. Modul 1) implementiert, die vom Kommandozeilenwerkzeug (d.h. Modul 2) verwendet wird. (Später wird HDrize in die grafische Oberfläche von iMage integriert werden.)

Das von Ihnen zusammengestellte Team hat bereits einen Großteil von HDrize implementiert. So wurde unter anderem der von HDrize ausgeführte Algorithmus (siehe Aufgabe 2) umgesetzt. Einige Details der Implementierung konnte Ihr Team jedoch nicht umsetzen und es ist an Ihnen als Teamleiter diese Lücken zu füllen. Da bei der Pear Corp. nur mit größter Sorgfalt gearbeitet wird, hat Ihr Team nicht versäumt Schnittstellen zu definieren und Klassenrümpfe anzulegen.

Die Basisimplementierung und Schnittstellen liegen im Maven-Modul `iMage.HDrize.base` und wird per Abhängigkeit in den Projekten genutzt (GroupID swt1, ArtifactID iMage.HDrize.base, Version 0.0.1-SNAPSHOT).

In ILIAS liegt eine Vorlage für die beiden Module mit den zu vervollständigenden Klassenrümpfen bereit, die Sie herunterladen und entpacken können. Integrieren Sie den Inhalt des ZIP-Archivs in Ihr Git-Depot. Vergessen Sie nicht, die beiden Module auch in Ihre oberste pom.xml (Projekt iMage) einzutragen.

Ebenfalls in ILIAS finden Sie eine Beispielbildreihe und ein daraus von HDrize erzeugtes HDR-Bild. Beides können Sie zur Orientierung und zur Überprüfung Ihrer Ergebnisse verwenden.

Das Modul `iMage.HDrize` innerhalb des Projekts `iMage` enthält ausschließlich die Logik für die Erzeugung der HDR-Bilder (es enthält keine Funktionalität zur Ausführung auf der Kommandozeile). Führen Sie folgende Schritte zur Vervollständigung der Implementierung des Moduls durch (Tipp: Lesen Sie zunächst alle Teilaufgaben, dies erspart Ihnen höchstwahrscheinlich Arbeit):

Für die Erzeugung des HDR-Bildes muss die Antwortkurve rekonstruiert werden. Hierzu müssen unterschiedliche Matrixoperationen auf den Bildmatrizen durchgeführt werden:

a) Implementieren Sie zunächst die Matrixrepräsentation der Bilder (Klasse `Matrix`). Diese dient ausschließlich zur Datenhaltung und soll beliebige Matrizen speichern können.
b) Implementieren Sie in der Klasse `MatrixCalculator` die üblichen Matrixoperationen:
    - Multiplikation

- Invertierung
- Transponierung

Mithilfe der Antwortkurve und der Eingabebildreihe kann das Ergebnisbild erzeugt werden. Implementieren Sie hierzu die entsprechenden Methoden in der Klasse HDRCombine:

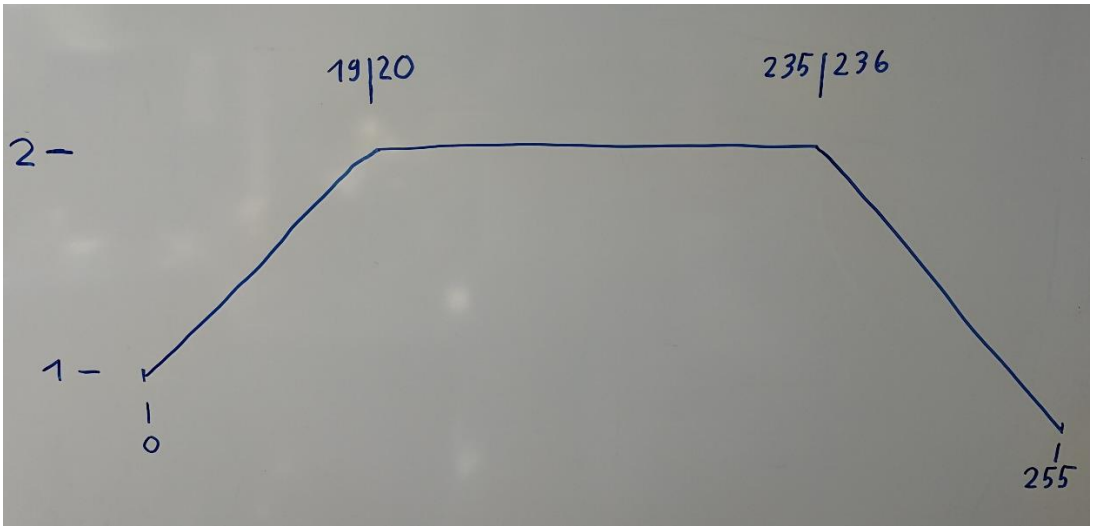
c) Implementieren Sie zunächst die Methode `calculateWeights()`. Diese berechnet eine verteilte Gewichtung für die Farbkanäle. Die Gewichtungskurve für die Werte 0 bis 255 berechnet sich wie folgt:

- Die ersten 20 Werte werden durch folgende rekursive Folge beschrieben:

  $a_0 = 1$

  $a_{n+1} = a_n + \frac{1}{20}\,, \qquad n \in \mathbb{N}_0 \mid n \geq 0, n < 20$
- Die letzten 20 Werte werden analog (achsensymmetrisch zur y-Achse) berechnet.
- Die restlichen Werte sind 2.

Ihre Projektleiterin hat Ihnen als Gedankenstütze zusätzlich folgende Skizze für die Gewichtungskurve an der Weißwandtafel hinterlassen:



d) Implementieren Sie die Methode `createHDR(…)`, welche das eigentliche HDR-Bild erzeugt. Hierzu werden aus den Eingabebildern die Werte für die Farbkanäle des HDR-Bildes anhand der folgenden Formeln berechnet:

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

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

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

Die Funktion *gewicht()* repräsentiert das Ergebnis der `calculateWeights()`-Methode für den jeweiligen Farbwert und die Funktion *kurve()* den Wert der Kamerakurve für den entsprechenden Farbwert. $\left(x_{bild_n}, y_{bild_n}\right)$ repräsentiert den Pixel an der Stelle (x,y) des n-ten Eingabebildes und $rgb_{kanal}()$ gibt den Farbwert des jeweiligen Farbkanals des Pixels zurück.

Hinweis: Durch geschickten Entwurf der Schleifen lässt sich Ausführungszeit sparen. Versuchen Sie außerdem unnötiges Neuberechnen von Informationen zu vermeiden.

Hinweis: Führen Sie die Berechnungen in c) und d) mit Fließkommazahlen durch.

e) Schreiben Sie für jede implementierte Methode mindestens einen sinnvollen Testfall.

Weitere Hinweise:

1. Achten Sie auf guten Stil. Das bedeutet auch (aber nicht ausschließlich), dass Sie Unschönheiten in Ihrem Quelltext mit Hilfe von CheckStyle beseitigen. Kommentieren Sie Ihren Quelltext. Schreiben Sie Unit-Tests für Ihre Implementierung. Es genügt die zu implementierenden Schnittstellen zu testen. Das bedeutet insbesondere, dass Sie Ihr Ergebnisbild nicht mit dem in ILIAS vorgegeben vergleichen müssen.
2. Achten Sie auf die JavaDoc-Kommentare in den vorgegebenen Klassen bzw. Schnittstellen. Diese machen weitere Vorgaben bzw. geben weitere Hinweise zur Implementierung.
3. Das Ergebnisbild soll so groß wie die Bilder der Eingabebildreihe sein; d.h. Sie müssen das Quellbild nicht verkleinern.
4. Behandeln Sie „normale, zu erwartende“ Fehler und Ausnahmen.
5. Führen Sie die üblichen Plausibilitätstests vor der Abgabe durch.

Aufgabe 5: Kommandozeilenprogramm für HDrize (3 Punkte)

Implementieren Sie ein Kommandozeilenprogramm im Modul `iMage.HDrize-cli`, das den Aufruf ihres Programms über die Kommandozeile steuert. Verwenden Sie die Vorlage aus dem ILIAS, welche die Struktur der Klasse (samt einiger Funktionalität) vorgibt; ändern Sie die Kommandozeilenschnittstelle nicht.

Passen Sie die vorgegebene pom.xml so an, dass als Abhängigkeit Ihre Version von `iMage.hdrize` verwendet wird. (Andernfalls wird automatisch eine Dummy-Implementierung verwendet. Sie können Aufgabe 5 somit auch bearbeiten, wenn Sie Aufgabe 4 nicht bearbeitet haben.)

Für Ihre Implementierung des Kommandozeilenprogramm soll folgendes gelten: Beim Programmaufruf können folgende Parameter angegeben werden:

- i: Pfad zum Ordner mit der Eingabebildreihe (obligatorisch)
- o: Pfad für die Ausgabe-Datei (obligatorisch)
- l: Lambda-Wert (Double, Wertebereich: (0, 100]) für den Algorithmus (optional: Standardwert 30)
- s: Anzahl der verwendeten Stichproben (Integer, Wertebereich: [1, 1000]) zur Erzeugung der Antwortkurve (optional: Standardwert 142)

Nach dem Einlesen der Kommandozeilenparameter soll das Programm folgende Schritte durchführen:

1. Einlesen aller Bild-Dateien (im JPEG-Format) aus dem Eingabeordner und folgende Prüfungen:
   - Die Anzahl der eingelesenen Bilder ist ungerade
   - Alle Bild-Dateien haben einen gemeinsamen Präfix, der mindestens 3 Zeichen lang ist
2. Prüfen, ob die Werte für das Lambda und die Stichprobenanzahl gültig sind
3. HDrize mit den entsprechenden Argumenten ausführen (`createRGB(…)`-Methode)
4. Ergebnis-HDR-Bild als PNG-Bild schreiben

Wichtig: Schlägt eine der oben genannten Prüfungen fehl, soll das Kommandozeilenprogramm abgebrochen und eine entsprechende Fehlermeldung ausgegeben werden.

Damit Sie das Kommandozeilenprogramm „einfacher“ aufrufen können, sorgen Sie dafür, dass Maven den Klassenpfad (`Class-Path`) mit den Bibliotheken versorgt (Einstellung `addClasspath` für das maven-jar-plugin). Mithilfe des maven-dependency-plugin können Sie Maven die benötigten Bibliotheken in das Verzeichnis `target` kopieren lassen. (Dadurch ersparen Sie sich das Angeben der benötigten Bibliotheken beim Aufruf auf der Kommandozeile.)

Führen Sie die üblichen Plausibilitätstests vor der Abgabe durch.

Aufgabe 6: Schnittstelle für Filter in iMage (2 Punkte)

Der Vorstand ist begeistert von HDrize. „Endlich erkennt man wie schick meine Villa bei Nacht aussieht.“, stellt auch Ihre Projektleiterin freudestrahlend fest. Da die Pear Corp. eine „progressive Innovationskultur“ verfolgt wird aber bereits jetzt über mögliche neue Funktionen diskutiert. Ihre Projektleiterin möchte gerne als nächstes verschiedene Filter (Graustufenfilter, Pinselfilter usw.) zu iMage hinzufügen. Durch Ihren Erfolg mit HDrize werden Sie von ihr zum Junior-Software-Architekten befördert und sogleich mit dem Entwurf geeigneter Schnittstellen beauftragt. (Diese Schnittstellen sollen später von Ihrem Entwickler-Team implementiert werden.) Ihre Projektleiterin gibt Ihnen folgende Beschreibung für die gewünschte Funktionalität:

> *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.*

Erstellen Sie aus der Beschreibung die nötigen Schnittstellen für eine Verwendung von Filtern in iMage.

a) Erstellen Sie ein neues Maven-Modul `iMage.filter` und geben Sie als Elter-Projekt wie gewohnt iMage an.
b) Definieren Sie die benötigten Schnittstellen. Sie können beliebig viele (sinnvolle) Schnittstellen anlegen. Achten Sie darauf, Ihre Schnittstellen mit JavaDoc-Kommentaren zu versehen.

Hinweis: Schreiben Sie ausschließlich die Schnittstellen. Sie müssen keine konkrete Implementierung angeben!

Geben Sie Ihre Implementierung wie gewohnt in der LEZ ab.
Führen Sie die üblichen Plausibilitätstests vor der Abgabe durch.

Hinweis zu den Werkzeugen

Git ist das in den Übungsaufgaben und in den Tutorien verwendete Werkzeug zur Versionskontrolle. Eclipse ist die in der Vorlesung und in den Tutorien verwendete Programmierumgebung. Sie können auch eine andere Programmierumgebung einsetzen, wenn Sie damit ausreichend vertraut sind und die gestellten Aufgaben genauso bearbeiten können. Es werden keine Aufgaben ausgegeben, deren Lösungen Eclipse-spezifische spezielle Fähigkeiten benötigen. Sie müssen „nur" in einer modernen Entwicklungsumgebung programmieren und Versionskontrolle einsetzen können.
Verwenden Sie bitte für alle Aufgaben Java 11.

Hinweis zu den Programmieraufgaben

Sofern nicht anders angegeben, sind die Aufgaben mit Java zu lösen. Die Einbuchungen und ihre Kommentare in Git werden bewertet (Existenz, sinnvolle Einbuchungshäufigkeit, -zeitpunkte und -dateien, sprechende Kommentare).
Konfigurieren Sie Maven mittels der Datei `pom.xml` **immer** wie folgt, um konsistente Projekte zu erhalten:

1. `ArtifactID` ist Ihre Matrikelnummer.
2. `GroupID` ist „swt1.ub<Übungsblatt-Nr.>.a<Aufgaben-Nr.>", also z. B. „swt1.ub0.a2" für Aufgabe 2 des Vorbereitungsblatts oder „swt1.ub3.a1" für Aufgabe 1 des dritten Übungsblattes.
3. Ändern Sie keine Einstellungen, die im XML-Dokument mit Kommentaren von uns versehen wurden (bspw. den Bereich DependencyManagement). Änderungen an diesen Bereichen können die automatische Analyse Ihrer Programme verhindern – in so einem Fall müssen Sie mit Punktabzug rechnen.
4. Wenn Sie Quelltexte abgeben, erzeugen Sie die Abgabedatei immer mit dem Befehl `mvn package`; laden Sie nicht die Binärfassung des Projekts hoch (*.jar), sondern die ZIP-Datei mit den Programmquellen. **Lösungen ohne Quelltext können nicht bewertet werden!**

Hinweis zur Lösungseinzugszentrale (LEZ)

Die Abgabe erfolgt per Lösungseinzugszentrale (LEZ). Sie können sich bei der LEZ mit Ihrem SCC-Benutzerkonto anmelden (https://lez.ipd.kit.edu/). Sie können pro Aufgabe eine einzelne Datei, ggf. auch ein gepacktes Archiv im ZIP-Format, abgeben. Soweit nicht anders angegeben, ist für die Programmieraufgaben hier immer `mvn package` auszuführen und die resultierende ZIP-Datei mit den Quelltextdateien Ihrer Lösung zu übertragen.
Achten Sie darauf, dass Sie eine signierte E-Mail der LEZ erhalten, in welcher eine Prüfsumme Ihrer abgegebenen Lösung enthalten ist. Diese dient bei technischen Problemen als Nachweis der Abgabe.

Hinweis zur Einzelbewertung

Die Lösungen aller Übungsblätter sind Einzelleistungen. Plagiieren von Lösungen führt zum Abzug von 25 Punkten bei allen am Plagiat Beteiligten.
Beispiel: Piggeldy lässt Frederick Aufgabe 2 abschreiben, ansonsten bearbeiten sie Übungsblatt 1 aber getrennt. Beide erhalten nun 25 Minuspunkte sowie keine Punkte für die übrigen Aufgaben von Blatt 1.

Hinweis zu Aktualisierungen des Übungsblatts

Verfolgen Sie Nachrichten in ILIAS und prüfen Sie es regelmäßig auf Aktualisierungen und Korrekturen des aktuellen Übungsblatts.