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_1096553&client_id=produktiv
Web: https://ps.ipd.kit.edu
LEZ: https://lez.ipd.kit.edu

SWT1: Übungsblatt 2

Anforderungsanalyse, UML, Mosaikbilder, Kommandozeile

Ausgabe: Mittwoch, 6. Mai 2020
Abgabe: Mittwoch, 20. Mai 2020, 12:00 Uhr

Abgabe der Theorieaufgaben

Aufgrund der aktuellen Covid-19-Situation finden die Abgaben der Theorieaufgaben vorerst nicht wie gewohnt über einen Kasten im Informatikgebäude statt. Stattdessen dient die LEZ zur Abgabe Ihrer Lösungen. Diese erreichen Sie unter https://lez.ipd.kit.edu.

Alle Theorie-Aufgaben inklusive aller Diagramme sind handschriftlich anzufertigen!
Ausgeschlossen sind auch digital (z.B. auf dem Tablet) angefertigte „handschriftliche" Lösungen!

Digitalisierung und Komprimierung

Damit Ihre handschriftlichen Lösungen entgegengenommen werden können, müssen Sie diese digitalisieren. Hierfür scannen Sie Ihre Lösungen bitte ein oder erstellen alternativ Fotos von diesen. Bitte nummerieren Sie die Blätter dazu und lösen maximal eine Aufgabe pro Seite. Also z.B. Aufgabe 1 a) bis c) auf Seite 1, Aufgabe 2 auf Seite 2 usw. Schreiben Sie nicht die Lösungen für Aufgabe 2 und 3 auf die gleiche Seite, da die LEZ eine Abgabe pro Aufgabe erwartet. Erstellen Sie die Bilder im JPEG-Format, um die Dateigröße später einfach reduzieren zu können.
Die LEZ nimmt Abgaben nur bis zu einer Größe von **15Mb pro Aufgabe** an. Nachdem Sie Ihre Lösungen digitalisiert haben, müssen Sie diese also ggf. komprimieren. Wir empfehlen hierzu die Anwendung JPEGOptimizer (https://github.com/collicalex/JPEGOptimizer/releases).

JPEG Optimizer

Wenn Sie die .jar Datei öffnen, finden Sie die Möglichkeit Quelle und Ziel Ihrer Dateien festzulegen. Wenn Sie Ihre Originale behalten wollen, wählen Sie 2 unterschiedliche Dateipfade. Ansonsten ist es auch möglich, Ihre Bilder zu überschreiben. In dem Feld „Max Diff:" empfehlen wir Werte zwischen 0.75% und 1.5%, diese haben sich als guter Kompromiss zwischen Größe und Qualität ergeben. Anschließend starten Sie den Komprimierungsvorgang mit einem Klick auf „Optimize" in der rechten oberen Ecke der GUI.

Abgabe bei der LEZ

Kontrollieren Sie Ihre Abgaben auf Lesbarkeit. Sollte Ihre Abgabe unlesbar sein, führt dies zu Punktabzug.
Erstellen Sie **ein .zip Archiv pro Aufgabe** und nennen dieses **Aufgabe_<Nummer>.zip.**
Das fertige .zip-Archiv laden Sie dann nach dem gleichen Schema wie bei den Praxisaufgaben zur LEZ hoch. **Sie haben zwei Wochen zur Bearbeitung jedes Übungsblattes Zeit, wir können nicht garantieren, dass die LEZ 5 Minuten vor Fristende standhält, wenn hunderte Studenten gleichzeitig hochladen.**

Aufgabe 1: Erstellung eines Lastenhefts (6 Punkte)

Die Pear Corp möchte iMage als Applikation zur Anwendung von Kunstfiltern vermarkten. Nutzer sollen Kunstfilter auf größere Mengen frei verfügbarer Bilder anwenden können. Um die Bilder zu beschaffen, plant die Geschäftsführung eine Applikation, die diese automatisch aus dem Internet lädt (eine Art *Web-Crawler*). Ihr Team wurde mit der Umsetzung dieser Applikation betraut. Hierzu hat Ihre Projektleiterin ein Szenario verfasst, das die gewünschte Funktionsweise der Applikation beschreibt:

> *Über die Startseite der Applikation sollen dem Nutzer verschiedene Vorgehen angeboten werden. Er kann zwischen einem reinen Suchen von Bildern, dem Anzeigen der letzten Bilder und einem Suchverfahren mit integrierter Komprimierung wählen können. Bei der Auswahl einer der beiden Suchvorgehen wird ein Auswahlfenster geöffnet, in dem verschiedene Optionen ausgewählt werden können. Zum einen soll die Anzahl, die Nutzungsrechte, das Dateiformat und die Größe der zu suchenden Bilder konfiguriert werden können. Zum anderen sollen die (Sub-)Domänen angegeben werden können in denen nach Bildern gesucht wird. Die Suche soll für eine Anzahl von fünfhundert (500) Bildern maximal zehn (10) Minuten benötigen und selbstständig nach einer Suchdauer von einer Stunde abbrechen.*
> *Das Anzeigen der letzten Bilder öffnet eine Übersicht über alle geladenen Bilder. Diese Übersicht soll die geladenen Bilder in einer Größe anzeigen, sodass das Motiv der Bilder erkennbar bleibt. Maximal sollen 50 Bilder gleichzeitig angezeigt werden. Der Nutzer kann in dieser Übersicht die Bilder nach Kriterien, wie mittlerer Farbwert, Name oder Herkunft filtern und sortieren. Außerdem werden dem Nutzer die Nutzungsrechte der Bilder über eine Farbskala kodiert angezeigt. Aus dieser Übersicht kann der Nutzer ausgewählte Bilder entweder unter Angabe eines Dateipfades lokal speichern oder diese auf den PearCorp-Zentralserver hochladen. Nach dem Hochladen wird dem Nutzer eine URL zur späteren Verwendung ausgegeben. Der Zugriff auf den Zentralserver soll von mindestens einhundert (100) Nutzern gleichzeitig erfolgen können und die Dauer des Hochladens der Bilder maximal linear mit der Anzahl der Bilder wachsen. Wählt der Nutzer die Suche mit integrierter Komprimierung, kann er zwischen verschiedenen Komprimierungsverfahren wählen und den Grad der Komprimierung einstellen. Der maximale Grad an Komprimierung soll dabei abhängig von der Bildgröße so berechnet werden, dass die Motive in 90 % der Bilder erkennbar bleiben. Die Suche mit Komprimierung soll maximal doppelt so lange benötigen wie die reine Suche (nach unkomprimierten Bildern).*

Aufgabe: 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 Ihr Lastenheft als .pdf in einem wie oben beschrieben benannten .zip Archiv ab. Dies ist (neben Aufgabe 2) die einzige Ausnahme zur Regelung der handschriftlichen Abgaben.

Aufgabe 2: Durchführbarkeitsuntersuchung (3 Bonuspunkte)

Da Ihre Projektleiterin auf die Beförderung angewiesen ist (die Anzahlung für ihren „Hightech-Bunker für schlechte Zeiten mit allem Schnickschnack“) und die Geschäftsführung ihr bereits im Nacken sitzt, möchte sie nicht leichtfertig Ihren Planungen, wie in Aufgabe 2 beschrieben, zustimmen. Der weite Weg bis ins mittlere Management hat ihr jedwede Fantasie für schnelle Umsetzungen geraubt und folglich zweifelt sie auch an der Umsetzbarkeit Ihres Planes (aus Aufgabe 1). Sie sind weiterhin von Ihrer beschriebenen Applikation überzeugt und bieten ihr an, eine kurze Durchführbarkeitsuntersuchung des Projektes anzufertigen.

Aufgabe: Wenden Sie für Ihre Durchführbarkeitsuntersuchung die sechs aus der Vorlesung bekannten Kriterien auf das Szenario aus Aufgabe 1 an. Setzen Sie die Kriterien in Bezug zum Szenario; Nennung von Stichworten alleine genügt nicht.

Verwenden Sie für die Durchführbarkeitsanalyse die Vorlage aus ILIAS (Lastenheft-Vorlage). Laden Sie die benötigten Dateien herunter und legen Sie dafür ein neues Verzeichnis `image.durchfuehrbarkeit` in Ihrem SWT1-Git-Depot an. Verfahren Sie mit der enthaltenen `.gitignore`-Datei ebenso, wie auf Übungsblatt 1 beschrieben.
Geben Sie Ihre Durchführbarkeitsanalyse als .pdf in einem wie oben beschrieben benannten .zip-Archiv ab. Dies ist (neben Aufgabe 1) die einzige Ausnahme zur Regelung der handschriftlichen Abgaben.

Aufgabe 3: Klassendiagramm (5 + 2 Punkte)

Die Abteilung für *Forschung, Entwicklung und Fortschritt* der Pear Corp. hat ein neues revolutionäres Bildformat erfunden, mit dem sich Bilder in komprimierter Form darstellen lassen. Der Vorstand ist dermaßen überzeugt von der Erfindung, dass er die Integration des Formats in das Projekt iMage in Auftrag gibt. Da Ihre Projektleiterin immer noch auf eine baldige Beförderung hofft, möchte sie Verzögerungen oder gar ein Scheitern der Integration mit allen Mitteln vermeiden. Aus diesem Grund soll die Implementierung des Formats zunächst in Form eines UML-Klassendiagramms entworfen werden.
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:

> *Ein Bild besteht aus einer Menge von zweidimensional angeordneten Bildpunkten, wobei maximal 65528 Punkte pro Dimension dargestellt werden können. Jeder Bildpunkt ist durch eine x- und y-Koordinate gekennzeichnet, die jeweils als natürliche Zahl angegeben werden. Zu Kompressionszwecken besteht ein Bild außerdem aus einer Menge von Blöcken, wobei jeder Block genau 8x8 Bildpunkte umfasst. Jeder Punkt gehört also zu genau einem Block. Die Farbe eines Bildpunktes wird durch einen Helligkeitswert sowie durch zwei Buntheitswerte beschrieben. Beides sind spezielle Formen eines Farbwertes. Jeder dieser Farbwerte wird durch eine Ganzzahl kodiert. Um Bilder komprimieren zu können, ist der Wertebereich dieser Zahl durch eine Farbtiefe beschränkt. Die Farbtiefe gibt dabei ganzzahlig den minimal sowie maximal möglichen Farbwert an. Jeder Farbwert bezieht sich individuell auf eine für ihn gültige Farbtiefe, wobei unterschiedliche Farbwerte durch dieselbe Tiefe beschränkt sein können. Die verschiedenen dabei eingesetzten Farbtiefen werden durch das Bild verwaltet.*

Nach Abschluss der Integration in iMage soll der Entwurf des neuartigen Bildformates um zwei verschiedene Modi für den Bildaufbau erweitert werden. Im sequenziellen Modus werden die Blöcke und die darin enthaltenen Bildpunkte vollständig der Reihe nach aufgebaut. Im progressiven Modus wird jeder Block zunächst nur auf Basis eines einzelnen Bildpunktes aufgebaut, während die einzelnen Blöcke im Anschluss schrittweise um die weiteren Bildpunkte ergänzt werden. Die Information über den zu verwendenden Modus soll dabei optional dem entsprechenden Bild direkt beigefügt sein.

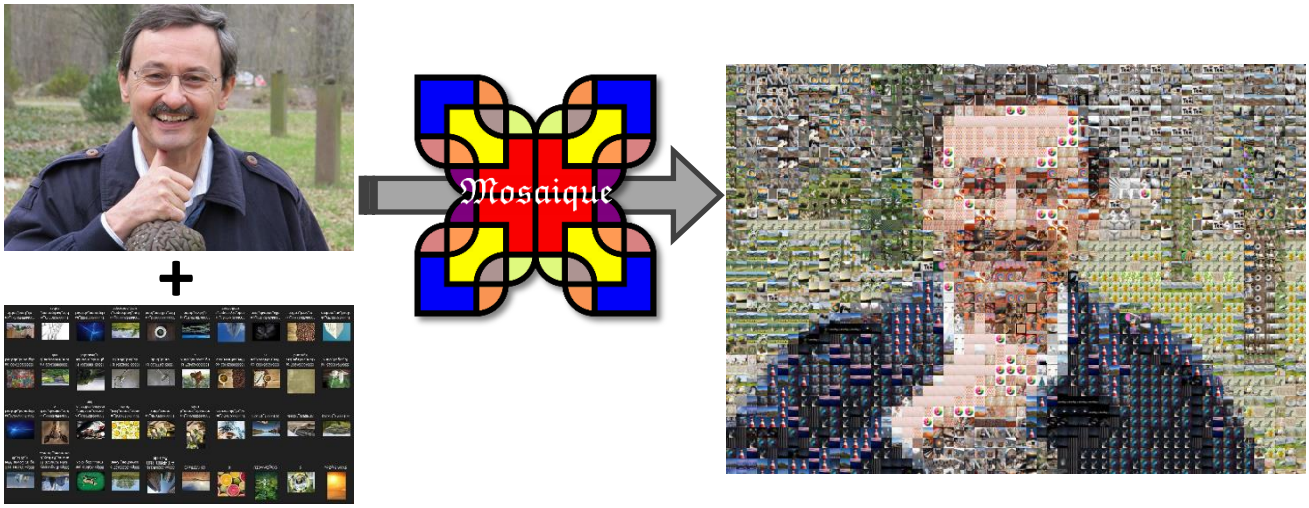
b) Erweitern Sie Ihr UML-Klassendiagramm gemäß der nachfolgenden Beschreibung:

> *Ein Bild verfügt optional über einen Modus, mit dessen Hilfe das entsprechende Bild aufgebaut wird. Zu diesem Zweck ist das aufzubauende Bild durch den Modus direkt zugreifbar, und muss daher nicht an die Operation (zum Bildaufbau) übergeben werden. Die Aufbauoperation liefert kein Ergebnis zurück. Der zu verwendende Modus ist entweder sequentiell oder progressiv und stellt eine entsprechende Umsetzung der Aufbauoperation zur Verfügung.*

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 4: Mosaique für iMage (9 Punkte)

Die Pear Corp. möchte nun auch endlich Geld mit iMage verdienen. Als ersten Kunstfilter schlägt die Vorstandsvorsitzende nach ihren Eindrücken aus ihrem letzten Italienurlaub eine Art „Mosaikifizierung“ von Bildern vor. Ihre Abteilung wird mit der Umsetzung betraut und Ihre Projektleiterin bestimmt kurzerhand Sie zum Leitenden. Die Aufgabe Ihres Teams ist es, (zunächst herauszufinden, was sich die Vorstandsvorsitzende unter „Mosaikifizierung“ vorstellt, und dann) eine Bibliothek zu schreiben, welche die Funktionalität implementiert. Sie nennen die neue Bibliothek Mosaique und machen sich an die Arbeit. Die Mosaique-Bibliothek soll eine Sammlung von Bildern entgegennehmen und ein übergebenes Bild aus Mosaikteilen bestehend aus Bildern der Sammlung zusammensetzen. Das Ergebnisbild sieht in etwa wie folgt aus:



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 und beachten Sie die JavaDoc-Dokumentation!

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

Das von Ihnen zusammengestellte Team hat bereits eine grobe Struktur für Mosaique definiert. Einige Details der Implementierung konnte Ihr Team jedoch nicht umsetzen und es ist an Ihnen als Leitenden 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.mosaique.base` und wird per Abhängigkeit in den Projekten genutzt (GroupID swt1, ArtifactID iMage.mosaique.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 ein .zip-Archiv mit Bildern, die Sie für die Mosaikteile benutzen können. Außerdem finden Sie dort ebenfalls ein Eingabe- und Ausgabebild. Diese können Sie zur Orientierung verwenden.

Das Modul `iMage.mosaique` innerhalb des Projekts `iMage` enthält ausschließlich die Logik für die Erzeugung der Mosaikbilder (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):

Die Mosaikerzeugung soll zunächst mit rechteckigen Mosaikteilen arbeiten. Hierzu haben Ihre Kollegen bereits den Klassenrumpf `RectangleShape` angelegt. Um die durch die Schnittstelle `IMosaiqueShape` geforderten Funktionalitäten anzubieten, vervollständigen Sie die Klasse um folgende Funktionalitäten:

a) Implementieren Sie den Konstruktor `RectangleShape(BufferedArtImage image, int w, int h)`, sodass das übergebene Bild auf die geforderte Breite (w) und Höhe (h) skaliert wird (Sie können hierzu die Methoden aus der Klasse `iMage.mosaique.base.ImageUtils` verwenden). Speichern Sie das skalierte Bild geeignet als Attribut von `RectangleShape`.
b) Implementieren Sie die Methode `getAverageColor()`, welche den durchschnittlichen Farbwert des skalierten Bildes im `TYPE_INT_ARGB` -Format zurückgibt. Der durchschnittliche Farbwert

eines (skalierten) Bildes setzt sich aus einem durchschnittlichen Farbwert pro Farbkanal (Alpha, Rot, Grün, Blau) über alle Pixel des skalierten Bildes zusammen.

c) Implementieren Sie die Methode `drawMe(BufferedArtImage targetRect)`, sodass diese das übergebene `BufferedArtImage` mit dem Inhalt des skalierten Bildes befüllen. Dies bedeutet, dass nach Anwendung von `drawMe` die Pixel des `targetRect` denen des skalierten Bildes entsprechen. Sollte das übergebene Bild kleiner sein als das skalierte Bild füllen Sie das übergebene Bild soweit mit Pixeln des skalierten Bildes auf wie möglich (rechter/unterer Teil des skalierten Bildes wird abgeschnitten) .

d) Implementieren Sie die weiteren `getter`-Methoden entsprechend der Schnittstellenbeschreibung.

Hinweis: Versuchen Sie unnötiges Neuberechnen von Informationen zu vermeiden.

Hinweis: Thumbnails sind `BufferedImages`, welche der Größe der Mosaikteile entsprechen und denselben Inhalt besitzen. Sie werden für einen späteren Teil der Implementierung benötigt.

Die Auswahl der passenden Mosaikteile für einen Bildbereich übernimmt in Mosaique der `RectangleArtist`. Dieser kennt die Mosaikteile für eine vorgegebene Größe (von Mosaikteilen) und bietet die Funktionalität das passendste Teil auszuwählen. Implementieren Sie hierzu die folgenden Funktionalitäten in der Klasse `RectangleArtist`:

e) Implementieren Sie zunächst den Konstruktor `RectangleArtist(Collection<BufferedArtImage> images, int tileWidth, int tileHeight)`. Überlegen Sie sich eine geeignete Form, wie Sie die Menge der Mosaikteile speichern.

f) Implementieren Sie die Methode `getTileForRegion(BufferedArtImage region)`, welche das passendste Mosaikteil für die übergebene Region zurückgibt. Hierzu muss der Durchschnittsfarbwert der übergebenen Bildregion berechnet werden und aus allen dem `RectangleArtist` bekannten Mosaikteilen das Teil ausgewählt werden, welches den geringsten Farbwertabstand zu dem durchschnittlichen Farbwert der Region aufweist. Abschließend soll das jeweilige Mosaikteil als `BufferedArtImage` zurückgegeben werden. Als Metrik für den Farbwertabstand soll der euklidische Abstand verwendet werden. Fassen Sie hierzu die Farbwerte als vierdimensionale Vektoren (für die vier Farbkanäle) auf.

g) Implementieren Sie auch hier die weiteren `getter`-Methoden entsprechend der Schnittstellenbeschreibung.

h) Schreiben Sie für `getTileForRegion` mindestens zwei sinnvolle Testfälle.

Hinweis: Sollten mehrere Mosaikteile den gleichen Farbwertsabstand aufweisen wählen Sie zufällig eines davon aus.

Hinweis: Versuchen Sie unnötiges Neuberechnen von Informationen zu vermeiden.

Das eigentliche Mosaikbild wird abschließend in der Klasse `MosaiqueEasel` zusammengesetzt. Auch hier haben Ihre Kollegen bereits einen Klassenrumpf zur Verfügung gestellt.

i) Implementieren Sie die Methode `createMosaique(BufferedImage input, IMosaiqueArtist artist)`, welche aus einem übergebenen Eingabebild ein Mosaikbild erstellt. Nutzen Sie zur Kachelung des Bildes den übergebenen `IMosaiqueArtist`. Das Bild soll aus Mosaikteilen zusammengesetzt werden, welche die Höhe und Breite besitzen mit der der übergebene Artist umgehen

kann. Die Mosaikteile sollen ohne Überlappung direkt aneinandergelegt werden. Sollte die Höhe/Breite des Bildes nicht ganzzahlig durch die entsprechende Breite/Höhe der Mosaikteile geteilt werden können, soll die jeweils letzte Kachel pro Zeile/Spalte (der Rest) abgeschnitten werden. Im folgenden Bild ist dieses Verhalten verdeutlicht (Kacheln werden entlang der roten Linien abgeschnitten).

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.
2. 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 vorgegebenen vergleichen müssen.
3. Achten Sie auf die JavaDoc-Kommentare in den vorgegebenen Klassen bzw. Schnittstellen. Diese machen weitere Vorgaben bzw. geben weitere Hinweise zur Implementierung.
4. Behandeln Sie „normale, zu erwartende“ Fehler und Ausnahmen.
5. Packen Sie keine größeren Bilddateien in Ihre Abgabe! Falls Sie Bilder für Ihre Testfälle benötigen, erstellen Sie diese entweder programmatisch, oder nutzen sie minimale Bilder (Gesamtgröße aller Testbilder sollte 1 MB **nicht** überschreiten).
6. Führen Sie die üblichen Plausibilitätstests vor der Abgabe durch.

Aufgabe 5: Kommandozeilenprogramm für Mosaique (3 Punkte)

Implementieren Sie ein Kommandozeilenprogramm im Modul `iMage.mosaique-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.mosaique` 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 Kommandozeilenprogramms soll folgendes gelten: Beim Programmaufruf können folgende Parameter angegeben werden:

- i: Pfad zum Eingabebild (obligatorisch)
- t: Pfad zum Ordner mit den Mosaikbildern (obligatorisch)
- o: Pfad für die Ausgabe-Datei (obligatorisch)
- w: Breite der Mosaikteile (Integer, Wertebereich: (0, Breite des Eingabebildes]) (optional: Standardwert: 10% (abgerundet) der Breite des Eingabebildes)
- h: Höhe der Mosaikteile (Integer, Wertebereich: (0, Höhe des Eingabebildes]) (optional: Standardwert: 10% (abgerundet) der Höhe des Eingabebildes)

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

1. Einlesen des Eingabebildes (im JPEG/PNG-Format)
2. Einlesen aller Bild-Dateien (JPEG/PNG-Format) im Ordner der Mosaikbilder und folgender Überprüfung, ob mindestens 10 Bilder enthalten sind
3. Prüfen, ob die Werte für Breite und Höhe gültig sind
4. Erstellen eines `RectangleArtist` und des `MosaiqueEasel` mit den entsprechenden Argumenten
5. Mosaique mit den entsprechenden Argumenten ausführen (`createMosaique(…)`-Methode)
6. Ergebnis-Mosaik-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.

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 14.

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.