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 4

GUI-Programmierung, Zustandsdiagramm, Benutztrelation, Geheimnisprinzip und Entwurfsmuster

Ausgabe: Mittwoch, 5. Juni 2019
Abgabe: Mittwoch, 19. Juni 2019, 12:00 Uhr

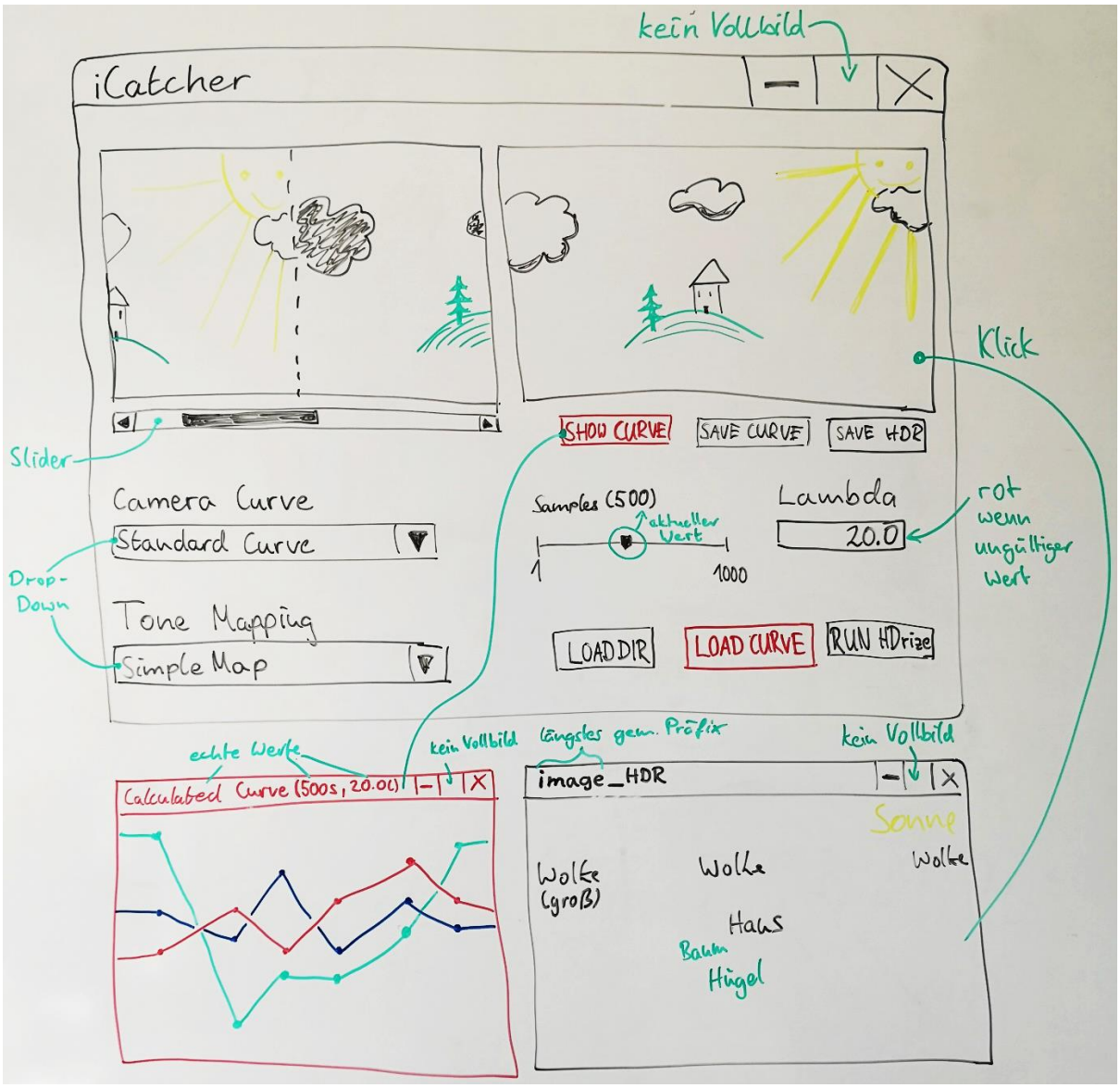
Geben Sie Ihre Lösung mit Deckblatt (mit Name, Matrikelnummer und der Nummer Ihres Tutoriums deutlich lesbar) ab, damit Ihr Tutor Korrekturhinweise und Ihre Punkte notieren kann. Werfen Sie es in die Holzkiste vor Raum 369 im Informatik-Hauptgebäude 50.34. Verwenden Sie ausschließlich das Deckblatt zur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

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

Aufgabe 1: Grafische Benutzeroberfläche für iMage (10 Punkte + 2 + 3 Bonuspunkte)

Ihre Projektleiterin hat bei der letzten Vorstandssitzung stolz die von Ihnen erstellten HDR-Bilder dem Vorstand präsentiert (und sich natürlich direkt eine Gehaltserhöhung abgeholt). Als sie jedoch die genaue Funktionsweise demonstrieren möchte, scheitert sie an der Benutzung der Kommandozeile. Schnell erklärt sie, dass ihre Mitarbeiter an dieser Stelle (wieder einmal) nicht mitgedacht hätten, sie hätte ja schon seit Monaten eine einfach zu benutzende Nutzerschnittstelle gefordert. Schnell überzeugt sie den Vorstand von der Implementierung einer graphischen Benutzeroberfläche (GBO, engl. GUI). Unter ihrer Anleitung skizziert der Praktikant ihre Idee an der Weißwandtafel (engl. Whiteboard). Vom Vorstand gefeiert, erhält sie sofort alle Freiheiten für die Umsetzung. Diese nutzt Ihre Projektleiterin, indem sie zunächst den Arbeitstitel „iCatcher“ vergibt und Sie anschließend mit der Umsetzung ihrer „höchst innovativen Idee“ beauftragt:

a) Implementieren Sie eine GBO für HDrize mithilfe der Java-Grafikbibliothek Swing entsprechend der Skizze. Erstellen Sie hierzu ein neues Maven-Modul namens `iMage.iCatcher` mit den üblichen Konfigurationseinstellungen und verwenden Sie Checkstyle, Git, JUnit usw. Die Klasse mit der `main`-Methode der GBO soll als Hauptklasse im JAR-Manifest eingetragen werden; sorgen Sie auch für die Einträge der benötigten Bibliotheken im Manifest und das Kopieren in das `target`-Verzeichnis wie in Aufgabe 3 von Aufgabenblatt 3. Tragen Sie HDrize als Abhängigkeit ein. Wenn Sie über keine funktionierende Lösung verfügen, können Sie auch die Musterlösung einbinden. (Diese hat die GroupID `swt1` und die ArtifactID `iMage.HDrize` sowie die Version `0.0.1-SNAPSHOT`.) Verwenden Sie keinen Code-Erzeuger für die GBO, schreiben Sie allen Code selbst. (Erstellen Sie z.B. das Layout mit einem LayoutManager.)



Ihre Projektleiterin erklärt zusätzlich zur Skizze, dass das Fenster „etwa“ 800x700 Pixel (Breite mal Höhe) groß sein soll. Oben links sollen die Original-Bilder (sobald geladen) angezeigt werden. Diese sollen nebeneinander gelegt werden (aufsteigend von links nach rechts). Mithilfe eines Schiebebalkens soll die Ansicht so verändert werden können, dass die Bildübersicht kontinuierlich verschoben werden kann. In der Skizze ist bspw. die rechte Hälfte von Bild 1 und die linke Hälfte von Bild 2 zu sehen. Die Bilder sollen so verkleinert werden, dass das erste Bild angezeigt wird, solange der Scroll-Balken am linken Anschlagspunkt verweilt (Standardposition). Oben rechts soll eine verkleinerte Vorschau des Ergebnis-HDR-Bild angezeigt werden (sobald berechnet). Für die Vorschau der Eingabebilder und der Ergebnisbilder stehen jeweils 350x250 Pixel zur Verfügung, die möglichst formatfüllend aber verzerrungsfrei ausgenutzt werden sollen.

Das Vorschaubild für das Ergebnis-HDR-Bild soll gleichzeitig als Knopf (engl. Button) dienen (d.h. es soll auf einem Knopf angezeigt werden). Wenn man auf den Knopf klickt, öffnet sich ein Dialogfenster (Pop-Up). In diesem Dialogfenster soll das erzeugte Bild in voller Größe angezeigt werden.

In einem Textfeld, das ausschließlich Fließkommazahlen zulässt (Double), soll der Lambda-Wert festgelegt werden. Wird im Textfeld etwas anderes eingetragen als ein gültiger Fließkommazahlwert, soll der Text rot dargestellt werden (Wertebereich von Lambda beachten).

Daneben soll mithilfe des Schiebebalken die Anzahl der Stichproben (Samples) zur Berechnung der Antwortkurve eingestellt werden. Der aktuelle Wert soll zudem in Klammern angezeigt werden (Standardwert beachten).
Zusätzlich soll über ein Auswahlmenü (ComboBox) ausgewählt werden, ob eine Standard-Kurve oder eine berechnete Kurve für die Erzeugung des HDR-Bildes verwendet wird. Wird die Standard-Kurve gewählt werden die Werte für Lambda und den Schwellwert bei der späteren Berechnung ignoriert.
Über ein weiteres Auswahlmenü soll die Art der Dynamikkompression (engl. Tone Mapping) gewählt werden (siehe Enumeration in `HDRImageIO`).
Mit einem Klick auf „LOAD DIR“ wird ein Dialogfenster zur Auswahl des Ordners, der die Eingabebilder enthält, geöffnet. Stellen Sie sicher, dass ausschließlich Ordner ausgewählt werden können. Anschließend sollen die geladenen Bilder im Vorschaufeld oben links angezeigt werden. Verwenden Sie als Filtermechanismen für die Bilder die gleichen wie in der Kommandozeilen-Implementierung (Übungsblatt 2, Aufgabe 5). Achten Sie darauf, hierfür sinnvolle Fehlerbehandlungen einzuführen.
Mit einem Klick auf „RUN HDrize“ soll die Berechnung des HDR-Bildes gestartet werden. Stellen Sie sicher, dass die Ausführung nur gestartet werden kann, wenn in allen Feldern der GUI gültige Werte eingetragen sind und eine Eingabebildreihe geladen wurde. Nach der Berechnung des HDR-Bildes soll dieses in der Vorschau oben rechts angezeigt werden.
Mit einem Klick auf „SAVE HDR“ soll ein Dialogfenster zum Speichern des Bildes angezeigt werden (es sollen ausschließlich Bilder vom Typ PNG (Dateiendung) zugelassen werden). Mit einem Klick auf „SAVE CURVE“ soll die erzeugte Antwortkurve gespeichert werden können (Dateiformat „.bin“).
Stellen Sie sicher, dass sowohl „SAVE HDR“ als auch „SAVE CURVE“ erst nach einer erfolgreichen Berechnung angeklickt werden können. Für „SAVE CURVE“ gilt zusätzlich, dass die Antwortkurve natürlich nur gespeichert werden kann, wenn sie berechnet wurde (Auswahlmenü „berechnete Kurve“).

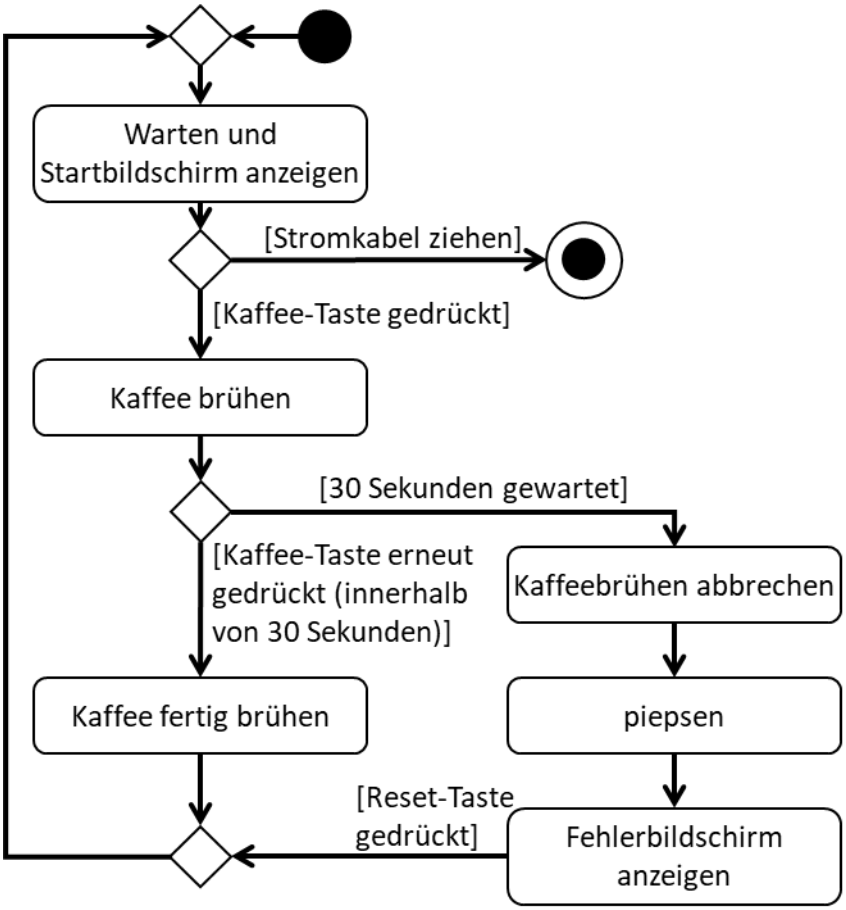
Hinweis: Die roten Bestandteile der Skizze sind nur für die Bonusaufgaben relevant.

b) Bonusaufgabe: Der hauptsächliche Beitrag zum Berechnungsaufwand der HDR-Bilder bildet die Rekonstruktion der Antwortkurve der Bilder. Aus diesem Grund soll die Benutzeroberfläche (und die nachgelagerte Ausführung) so angepasst werden, dass gespeicherte Antwortkurven wieder geladen (und angewendet) werden können. Fügen Sie hierzu einen neuen Knopf „LOAD CURVE“ zu Ihrer Benutzeroberfläche hinzu und platzieren Sie diesen, wie in der Skizze ersichtlich. Bei einem Klick auf diesen Knopf soll ein Dialogfenster zum Öffnen von Antwortkurven (Dateiendung „.bin“) angezeigt werden. Beachten Sie dabei folgendes: Wird eine Antwortkurve geladen, soll im Auswahlmenü für die Antwortkurve automatisch „Loaded Curve“ (muss ggf. zunächst hinzugefügt werden) gewählt werden. Wird stattdessen zuerst „Loaded Curve“ im Auswahlmenü gewählt (d.h. es wurde noch keine Kurve geladen), soll ebenfalls das Dialogfenster zum Öffnen einer Antwortkurve angezeigt werden.
c) Bonusaufgabe: Ihre Projektleiterin kann sich unter dieser obskuren „Antwortkurve“ einfach nichts vorstellen. Um sich unzählige Stunden an der Weißwandtafel zu ersparen, entschließen Sie sich, eine Visualisierung für die Antwortkurve zu erstellen. Fügen Sie hierzu zunächst einen neuen Knopf „SHOW CURVE“ hinzu. Bei einem Klick auf diesen Knopf soll die aktuell berechnete Antwortkurve angezeigt werden. Wurde noch keine Antwortkurve berechnet, soll man den Knopf auch nicht betätigen können. Beachten Sie: In dieser Aufgabe wird neben der Funktionalität und der Korrektheit auch die Schönheit Ihrer Darstellung bewertet.

Führen Sie die üblichen Plausibilitätstests vor der Abgabe durch.

Aufgabe 2: Diagramme überführen (5 Punkte)

Die beste Entwicklerin Ihres Teams meldet Ihnen, dass die Kaffeemaschine in der Kaffeeküche defekt ist. Allerdings verhält sich die Kaffeemaschine nur in bestimmten Situationen seltsam. Das Hauptproblem scheint wohl zu sein, dass man während des Brühvorgangs erneut die Kaffee-Taste betätigen muss, weil der Brühvorgang ansonsten nach 30 Sekunden abbricht. Die Entwicklerin, die den Kaffee dringend zum Arbeiten benötigt, hat Ihnen in ihrer Verzweiflung das Verhalten der Kaffeemaschine als UML-Aktivitätsdiagramm modelliert. Auch wenn Sie das etwas übertrieben finden, beschließen Sie, sich um das Problem zu kümmern. Um sich einen besseren Überblick über die Lage zu verschaffen, beschließen Sie, statt der Aktivitäten die durchlaufenen Zustände der Kaffeemaschine zu betrachten.



Überführen Sie das obenstehende UML-Aktivitätsdiagramm in ein UML-Zustandsdiagramm. Modellieren Sie die durchlaufenen Zustände der Küchenmaschine möglichst exakt. Ihr UML-Zustandsdiagramm soll mit einer möglichst kleinen Menge an Zuständen auskommen. Versehen Sie die Zustandsübergänge mit Ereignissen und Operationen und verwenden Sie ggf. Eintritts- und Austrittsaktionen sowie Wächterbedingungen.

Hinweis: Zeichnen Sie das Diagramm als Übung für die Klausur von Hand.

Aufgabe 3: Geheimnisprinzip (2 Bonuspunkte)

Die Pear Corp. hat zwei Werksstudenten (Herrn Knöpfle und Frau Spätzle) von der Schwäbischen Hochschule für Informations-Technologie eingestellt und Ihrer Abteilung zugewiesen. Beide sind zwar umgänglich, haben aber deutlich weniger Wissen von Softwaretechnik als Sie das (von KIT-Studenten) gewohnt sind. Bereits nach der ersten kleinen Programmieraufgabe bemerken Sie, dass Knöpfle und Spätzle offensichtlich noch nie etwas vom Geheimnisprinzip gehört haben. Da die Pear Corp. – trotz mehrerer Eingaben Ihrerseits – immer noch

kein internes Weiterbildungsprogramm ins Leben gerufen hat, entscheiden Sie sich, die Beiden persönlich anhand von Beispielen aus der Java-Standard-Bibliothek fortzubilden. Sie bereiten das Folgende vor:
Geben Sie für die untenstehenden Klassen bzw. Schnittstellen jeweils stichpunktartig an, welches Geheimnis verborgen wird bzw. verborgen werden soll. Beschreiben Sie ebenfalls stichpunktartig wie die Verbergung umgesetzt wird:

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

Aufgabe 4: Benutztrelation (5 Punkte)

Ihr Entwicklerteam verliert so langsam den Überblick über das iMage-Projekt. Insbesondere die Abhängigkeiten zwischen den einzelnen Modulen sorgen für viel Verwirrung. Da Sie als Teamchef für die Zufriedenheit Ihres Teams verantwortlich sind (und das meiste Entwurfswissen besitzen), beschließen Sie, die Abhängigkeiten zwischen den einzelnen Modulen übersichtlich mithilfe der Benutztrelation darzustellen.

a) Stellen Sie die Benutztrelation für alle Module von iMage graphisch dar. Erfassen Sie dabei die Maven-Module, die auf den Übungsblättern 1 bis 4 entstanden sind. Erfassen Sie nur die Abhängigkeiten in Form von «uses»-Pfeilen (und lassen Sie die Vererbungsbeziehung der pom.xml-Konfigurationen außen vor). Verwenden Sie für die Module im Diagramm die Modulnamen, nicht die Maven-Artefakt-Bezeichner.

Ihre Projektleiterin hat von Ihrem „Benutztrelations-Diagramm-Dings-Da“, wie sie es nennt, Wind bekommen und ist begeistert. Frei nach dem Motto „Keine gute Tat bleibt ungestraft“ verlangt Sie von Ihnen, Ihr Diagramm noch zu erweitern. „Wir müssen mal prüfen, welche Bibliotheken wir überhaupt so nutzen… nicht, dass wir da unnötig Lizenzgebühren zahlen irgendwo (oder zu wenige)…“, stellt Sie fest. Als Sie bemerken, dass Sie aus der Sache nicht mehr herauskommen, beginnen Sie Ihr Diagramm wie folgt zu erweitern:

b) Stellen Sie die Benutztrelation zwischen allen Modulen von iMage und den benötigten Abhängigkeiten graphisch dar. Betrachten Sie auch hier die Übungsblätter 1 bis 4 (inklusive Bonusaufgaben). Kennzeichnen Sie, welche Abhängigkeiten nur zum Testzeitpunkt bestehen (z.B. mit mit «uses (T)»-Pfeilen).; geben Sie die ArtfactID von externen Bibliotheken an. Beachten Sie, dass die Benutztrelation transitiv ist. Gehen Sie für die Betrachtung der Benutztrelation davon aus, dass die Programmieraufgaben vollständig bearbeitet wurden (verwenden Sie ggf. die Musterlösungen der vorangegangenen Übungsblätter).

Hinweis: Verwenden Sie unterschiedlich-farbige Stifte für die Bearbeitung der beiden Aufgabenteile a) und b), um Ihrem Tutor die Korrektur zu erleichtern.

Anmerkung: Die «uses (T)»-Pfeile sind nicht standardkonform – wir benutzen Sie, damit Sie alle Abhängigkeiten in einem Diagramm darstellen können. Alternativ können Sie auch zwei Diagramme zeichnen: eins für die übliche Benutztrelation und eins für die Benutztrelation zum Testzeitpunkt.

Aufgabe 5: Entwurfsmuster (5 Punkte)

Für HDrize sind zukünftig jede Menge Erweiterungen und Verbesserungen vorgesehen. Um für die Zukunft gerüstet zu sein, beschließen Sie zukünftig (noch) stärker auf Entwurfsmuster zu setzen. Allerdings ist Ihnen noch nicht ganz klar, an welchen Stellen überhaupt Bedarf besteht. Daher stellen Sie einen Ideen-Kasten auf und sammeln von Ihren Teammitgliedern anonyme Vorschläge für potentielle Stellen zur Erweiterung bzw. Verbesserung von HDrize. Nach einer Woche leeren Sie den Kasten und finden tatsächlich die drei folgenden sinnvollen Vorschläge:

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 Primitive-DenseStore (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."

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!"

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."

Geben Sie für jeden Vorschlag eines Team-Mitglieds an, welches Entwurfsmuster sich anbieten würde, um das Entwurfsproblem zu lösen. Zeichnen Sie außerdem jeweils das zugehörige UML-Klassendiagramm. Ihr Klassendiagramm soll auf die konkrete Situation angepasst sein. Es genügt also nicht, die Klassendiagramme aus der Vorlesung zu übernehmen. Kennzeichnen Sie in Ihren Klassendiagrammen, welcher Bestandteil Ihrer Modellierung (Klasse, Methode, Attribut usw.) welchem Bestandteil des jeweiligen Entwurfsmusters entspricht.

Hinweis: Zeichnen Sie die Diagramme als Übung für die Klausur von Hand.

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.