Übungsblatt

Entwurfsmuster, Refaktorisieren

Aufgabe 1: Jäger des verlorenen Entwurfsmusters (6 Punkte)

Sie sind inzwischen total angetan von Entwurfsmustern und suchen selbst überall nach Entwurfsmustern in Software-Projekten. Sie vermuten sogar, dass es einige in der Java-API geben muss. Schnell werden Sie fündig. Geben Sie an, welches Entwurfsmuster die folgenden Klassen/Schnittstellen der Java-API bilden:

a) Die Klasse `java.awt.Desktop.` (https://docs.oracle.com/en/java/javase/16/docs/api/java.desktop/java/awt/Desktop.html)

b) Die Methode **`contains(Object o)`** der Klasse **`java.util.AbstractCollection<E>`**. (https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/util/AbstractCollection.html)

c) Die Schnittstelle **`java.awt.event.ActionListener`** und die implementierende Klasse **`javax.swing.AbstractAction`** inklusive deren Unterklassen **`CloseAction`**, **`IconifyAction`** und **`RestoreAction`**. https://docs.oracle.com/en/java/javase/16/docs/api/java.desktop/javax/swing/AbstractAction.html
Hinweis: Gemeint ist nicht das Muster Beobachter!

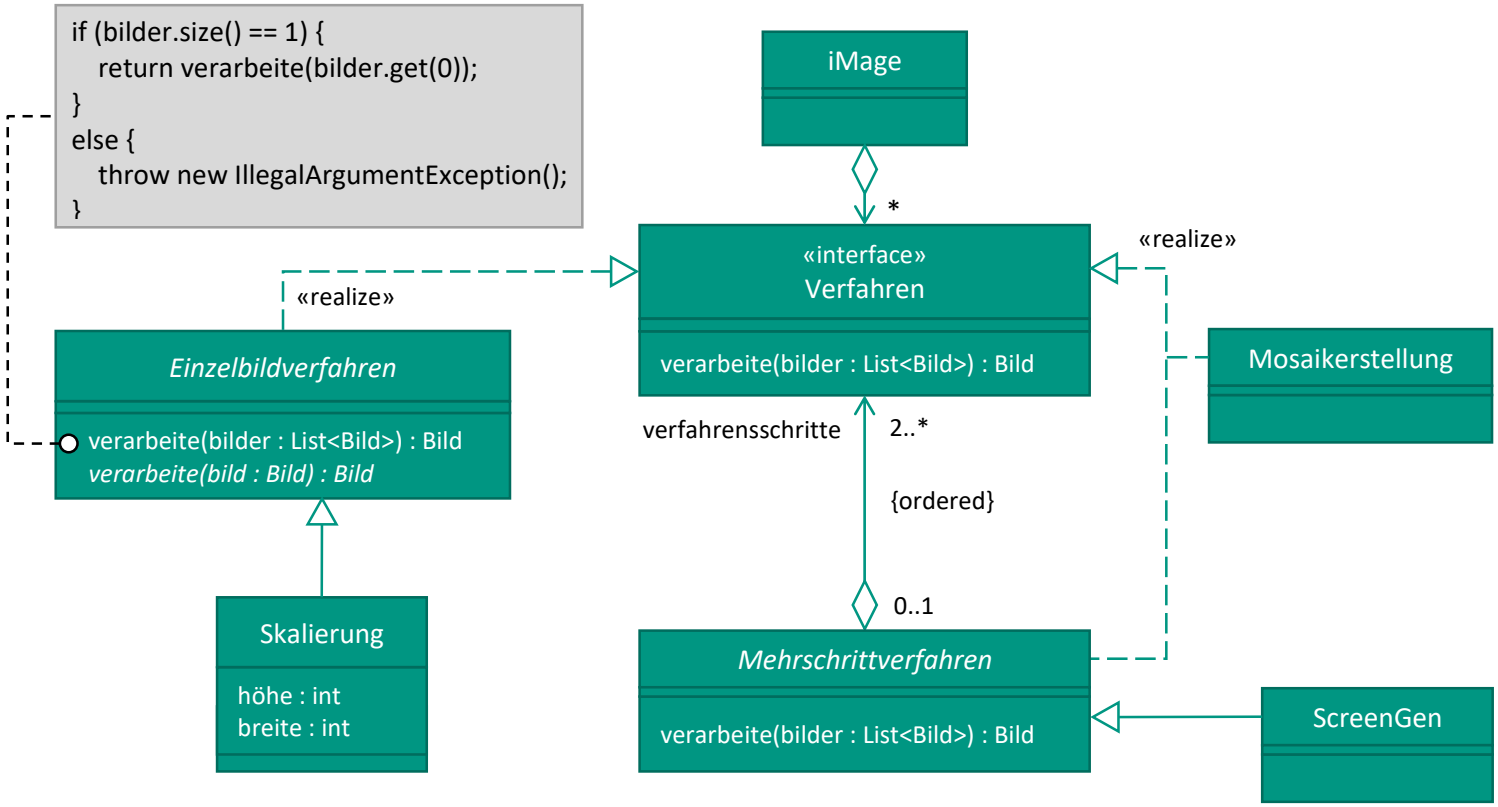
Zeichnen Sie zu jedem der gefundenen Entwurfsmuster ein UML-Klassendiagramm, welches den Ausschnitt aus der Java-API wiedergibt, der das gefundene Entwurfsmuster repräsentiert. Beschränken Sie sich in den UML-Klassendiagrammen dabei auf die für das Entwurfsmuster relevanten Methoden und Attribute. Geben Sie an, welche der Klassen, Methoden und Attribute in Ihrem UML-Klassendiagramm welchen Rollen in dem Entwurfsmuster entsprechen. Verwenden Sie dazu die Begriffe aus der Vorlesung. Begründen Sie jeweils, warum Sie dieses Muster erkannt haben.

Aufgabe 2: Entwurf (6 Punkte)

Während des Übergangs von der Definitions- zur Entwurfsphase realisiert die Projektleitung, dass der bisherige Entwurf von iMage mit einem fundamentalen Nachteil verbunden ist: Mehrbild- und Einzelbildverfahren verfügen über keine gemeinsame Schnittstelle, die eine Gleichbehandlung verschiedener Verfahren ermöglicht (siehe Musterlösung zu Übungsblatt 2, Aufgabe 3).

Um den „Karren aus dem Dreck zu ziehen“, erstellt Ihr Team einen Alternativentwurf (siehe Abbildung). In diesem wird die Operation `verarbeite(bilder : List<Bild>) : Bild` zum „kleinsten gemeinsamen Nenner“ der verschiedenen Verfahren bestimmt und in die Schnittstelle `Verfahren` hochgezogen (die Schnittstelle `Mehrbildverfahren` entfällt somit). Daraufhin wird das `Einzelbildverfahren` so angepasst, dass es sich gemäß der Schnittstelle von `Verfahren` verhält (unter der Voraussetzung, dass die Liste der übergebenen `bilder` aus einem einzigen Bild besteht).

Außerdem wird eine neue Klasse namens `Mehrschrittverfahren` eingeführt, um neben `ScreenGen` noch andere Verfahren zu unterstützen, die durch mehrere einzelne Verfahrensschritte implementiert werden. Jedes `Mehrschrittverfahren` enthält mindestens zwei beliebige andere `Verfahren`, die jeweils einen Verfahrensschritt darstellen und daher durch die Aggregation `verfahrensschritte` referenziert werden.



a) Welches Entwurfsmuster verwendet die Klasse `Einzelbildverfahren`, um sich gemäß der Schnittstelle von `Verfahren` zu verhalten? Nennen Sie das verwendete Muster und beschreiben Sie, zu welchem Zweck es in dem gegebenen Entwurf dient. Beschreiben Sie ebenfalls, inwiefern die Anwendung des Musters von seiner typischen Struktur auf den Vorlesungsfolien abweicht.

Bei genauerer Betrachtung des Entwurfs fällt Ihnen auf, dass zwei verschiedene Variationspunkte „durcheinandergewürfelt“ wurden: Einerseits variieren die Verfahren anhand der Tatsache, ob mehrere Bilder oder nur ein einziges Bild verarbeitet werden. Andererseits variieren die Verfahren auch je nachdem, ob ihre interne Implementierung aus einem einzigen oder aus mehreren Verfahrensschritten (wie z. B. `ScreenGen`) besteht.

b) Beschreiben Sie welches Problem besteht, wenn ein `Einzeldbildverfahren` durch mehrere Verfahrensschritte implementiert werden soll. Geben Sie an, welche Art von nichtfunktionaler Anforderung durch den gegebenen Entwurf negativ beeinträchtigt wird.

c) Wählen Sie ein geeignetes Entwurfsmuster, um die beiden Variationspunkte voneinander zu entkoppeln. Nennen Sie das von Ihnen verwendete Muster und bringen Sie es zur Anwendung, indem Sie einen verbesserten Entwurf in Form eines UML-Klassendiagramms angeben.

Hinweise:

- Führen Sie als Teil Ihres Entwurfs eine Klasse namens `VerfahrenImpl` ein, um zwischen einschrittigen und mehrschrittigen Verfahrensimplementierungen zu variieren.
- Integrieren Sie hierbei das Entwurfsmuster *Befehl*, so dass die Klasse `VerfahrenImpl` als *Befehl* fungiert. Dazu soll sie über eine Operation namens `führeDurch()` verfügen, die das jeweilige Verfahren durchführt.
- Bedenken Sie, dass ein mehrschrittiges Verfahren durch eine Abfolge von Befehlen implementiert werden muss.
- Beachten Sie außerdem, dass jeder *Befehl* über eine Menge von Bildern als *Empfänger* verfügen muss.

Aufgabe 3: Entwurfstheorie (3 Punkte)

Ihre Chefin „verdonnert“ Sie zur Vorbereitung eines Recruiting-Events der Pear Corp. an der *Schwäbischen Hochschule für Informations-Technologie*. „Öffentlichkeitsarbeit gehört zum Job eines Softwaretechnikers dazu – deal with it!“, entgegnet sie Ihnen forsch. Ihre erste Aufgabe besteht darin, eine Präsentationsfolie zu erstellen, auf der Sie den Studierenden der Hochschule die Unternehmensstruktur der Pear Corp. näherbringen. Diese Struktur stellt sich wie folgt dar: Die Pear Corp. setzt sich aus mehreren Abteilungen zusammen. Jede Abteilung besteht wiederum aus mehreren Teams. Mitarbeiter*innen sind jeweils Teil eines solchen Teams. Diese Struktur kann jederzeit durch das Hinzufügen oder Entfernen neuer Abteilungen, Teams, oder Mitarbeiter*innen angepasst werden. Die Kundschaft der Pear Corp. kann die interne Struktur getrost ignorieren, denn aus Kundensicht ist es unerheblich, ob ein Auftrag auf Unternehmensebene, auf Abteilungsebene, oder auf der Ebene von Teams oder Mitarbeiter*innen abgewickelt wird. Um den Studierenden die Unternehmensstruktur zu verdeutlichen, beschließen Sie, auf geeignete Entwurfsmuster zurückzugreifen.

a) Sie erinnern sich an das Entwurfsmuster *Kompositum*. Begründen Sie, warum sich dieses Muster zur Modellierung der Struktur der Pear Corp. eignet. Beziehen Sie sich in Ihrer Begründung auf die oben beschriebenen Merkmale der Unternehmensstruktur und geben Sie an, inwiefern das Muster diese Merkmale begünstigt.

Um Konflikte zwischen den Mitarbeiter*innen zu lösen, verfügt die Pear Corp. über ein „Streitschlichtungsprogramm“. Zu diesem Zweck stellt ein Subunternehmen mehrere Streitschlichter*innen zur Verfügung (d. h. die Streitschlichter*innen sind selbst nicht Teil der Unternehmensstruktur). Im Streitfall nehmen die Streitschlichter*innen den Kontakt zu den zwei zerstrittenen Mitarbeiter*innen auf, um den Streit aus der Welt zu schaffen. Die „Streithähne“ müssen dabei nicht direkt miteinander kommunizieren, denn jegliche Kommunikation wird ausschließlich über die Streitschlichtung abgewickelt.

b) Überlegen Sie sich ein geeignetes Entwurfsmuster, das sich zur Modellierung des Streitschlichtungsprogramms eignet. Beziehen Sie sich in Ihrer Begründung auf die oben beschriebenen Merkmale der Streitschlichtung und geben Sie an, inwiefern das Muster diese Merkmale begünstigt.

Aufgabe 4: Unternehmensstruktur der Pear Corp. (5 Punkte)

In dieser Aufgabe sollen Sie die Unternehmensstruktur der Pear Corp. (siehe Aufgabe 3) unter der Verwendung von Entwurfsmustern modellieren. Gegeben sei folgender Auszug aus einer CSV-Datei, die die Mitarbeiter*innen, Teams und Abteilungen auflistet.

Mitarbeiter*in	Team	Abteilung
Edelgard Schlosser	Gezwitscher	Social Media
Leo Kruse	Facezine	Social Media
Finn Messner	Facezine	Social Media
Reinhardt Garber	Analyse	Marketing
Sarah Haupt	Analyse	Marketing
Xaver Warner	Testen	Entwicklung
Beata Reis	Implementierung	Entwicklung
Rosa Loritz	Implementierung	Entwicklung
Ewald Beltz	Entwurf	Entwicklung

Sie können davon ausgehen, dass jedes Team nur zu einer Abteilung und jede*r Mitarbeiter*in nur zu einem Team gehört. Insgesamt gehört **jede** Abteilung zum Pear Corp. Unternehmen.

a) Erstellen Sie ein neues Modul namens `iMage.company` für die Abbildung der Unternehmensstruktur. Denken Sie an die üblichen benötigten Konfigurationsdateien (`src.xml`, usw.).
b) Verwenden Sie das Entwurfsmuster Kompositum, um die Struktur der Tabelle auf Klassen abzubilden. Überlegen Sie sich dazu, welche Klassen dabei ein Blatt darstellen und welche Klassen als Kompositum dienen.
c) Transformieren Sie die Daten aus der Tabelle in die in Aufgabenteil b) angelegte Struktur. Die Tabelle wird Ihnen als CSV-Datei zur Verfügung gestellt und kann variieren. Legen Sie daher eine Klasse an, die eine CSV-Datei entgegennimmt und diese in die Struktur aus Aufgabenteil b) überführt. Sie können zum Parsen Bibliotheken wie OpenCSV (groupId `com.opencsv` artifactId `opencsv`) verwenden.
   Schreiben Sie anschließend mindestens einen geeigneten Testfall, der Ihre Implementierung testet.
d) Implementieren Sie nun das Entwurfsmuster Besucher. Der Besucher soll die Klassen aus Aufgabenteil b) besuchen. Dadurch können neue Operationen auf der Daten-Struktur angelegt werden, ohne diese ändern zu müssen.
e) Implementieren Sie einen konkreten Besucher, der die Mitarbeiter der Pear Corp., welche unter der entsprechenden Stelle in der Hierarchie aufgehängt sind, zählt. Nutzen Sie dazu das Besucher-Entwurfsmuster und fügen Sie keine weitere Funktionalität den Daten-Klassen hinzu.
   Testen Sie auch hier Ihre Implementierung durch einen geeigneten Testfall.

Aufgabe 5: Refaktorisieren von ScreenGen (6 Punkte)

Nachdem Ihr Kollege bereits einen Stummel für die Aufwertung durch einen Text zur Verfügung gestellt hat (Übungsblatt 4, Aufgabe 4), stellen Sie fest, dass die Schnittstelle der ScreenGen-Bibliothek nur *einen* Aufwertungsprozess entgegennimmt. Der Vertrieb würde aber gerne auf den Kunden individuell zugeschnittene Pakete von Aufwertungsprozessen anbieten. Da es allerdings schon Kunden gibt die die bisherige Schnittstelle verwenden, können Sie diese nicht ohne Weiteres ändern.
Sie erinnern sich an die Entwurfsmuster der Softwaretechnik-1-Vorlesung und überlegen sich, wie man eine Aufwertung dennoch durch einen Text "dekorieren" kann.

a) Erweitern Sie Ihre Implementierung des Moduls `iMage.screengen` so, dass Aufwertungen (`ScreenImageEnhancement`) dekoriert werden können. Verwenden Sie hierzu das aus der Vorlesung bekannte Dekorierer-Entwurfsmuster.
b) Implementieren Sie nun den konkreten Dekorierer `TextDecorator`. Im Konstruktor soll unter anderem der Text als `String`, die Schriftart als `java.awt.Font` und die Position des Textes als `org.iMage.screengen.base.Position` übergeben werden. Nach der Anwendung des konkreten Dekorierers soll der übergebene Text in der Schriftart an der bestimmten Position auf dem bereits aufgewerteten Bild zu sehen sein.

Nachdem Ihre Projektleiterin die Änderungen abgesegnet hat, kommt sie mit einer weiteren Bitte auf Sie zu:
Neben dem naiven Überlagern zweier Bilder (Übungsblatt 2, Aufgabe 4), soll nun auch eine Variante des Porter/Duff-Algorithmus umgesetzt werden. Sie stellen fest, dass das Skelett beider Verfahren identisch ist. Nur die pixelweise Berechnung der Ausgabe-Farbe basierend auf der Hintergrund- und Vordergrund-Farbe variiert.

c) Überlegen Sie sich, durch welches Entwurfsmuster die erwähnten Redundanzen vermieden werden können. Setzen Sie anschließend das Entwurfsmuster um. Nach dem Refaktorisieren soll die Klasse `BackgroundEnhancement` weiterhin die bestehende Schnittstelle (`ScreenImageEnhancement`) und das naive Überlagern anbieten. Somit müssen Sie nicht ihre GBO aus Aufgabenblatt 4 oder die Testfälle anpassen.

d) Implementieren Sie nun die Variante "Source Over" des Porter/Duff-Algorithmus zum Überlagern zweier Bilder. Berechnen Sie dafür den normalisierten Alpha-Wert

$$\alpha_C = \alpha_A + (1 - \alpha_A) \cdot \alpha_B$$

A ist die normalisierte Quell-Farbe (source, hier: Vordergrund), *B* die normalisierte Ziel-Farbe (destination, hier: Hintergrund) und *C* die normalisierte Ergebnis-Farbe, die sich aus dem Überlagern von *A* und *B* ergibt. Sollte $\alpha_C = 0$ gelten, dann ist $C = B$. Ansonsten werden die Farb-Kanäle $\beta \in \{\text{Rot}, \text{Grün}, \text{Blau}\}$ für *C* wie folgt berechnet:

$$\beta_C = \frac{1}{\alpha_C} \cdot (\alpha_A \cdot \beta_A + (1 - \alpha_A) \cdot \alpha_B \cdot \beta_B)$$

Normalisieren bedeutet hier, dass die Farb-Kanäle von [0, 255] auf [0, 1] abgebildet werden. Beim Übergang von [0, 1] zu [0, 255] sollen die Komponenten abgerundet werden.

Hinweis: Beachten Sie bei der Benennung von Methoden und Klassen, dass die Bestandteile des Entwurfsmusters eindeutig zugeordnet werden können.

Hinweise zur digitalen Abgabe der Theorieaufgaben

Um Ihrem Tutor die Zuordnung zu erleichtern, versehen Sie Ihre handschriftlichen Abgaben mit Namen, Matrikelnummer und Aufgabennummer!

Digitalisierung und Komprimierung:

Damit Ihre handschriftlichen Lösungen entgegengenommen werden können, müssen Sie diese digitalisieren (sofern Sie diese nicht digital handschriftlich erstellt haben). 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 **12 Mb 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.**

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 16.

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.a1" für Aufgabe 1 des Vorbereitungsblatts oder „swt1.ub3.a2" für Aufgabe 2 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.