

# SE2, Aufgabenblatt 1

Modul: Softwareentwicklung II – Sommersemester 2021

**Eclipse, Projekt Mediathek, Interfaces**

Moodle-URL: ..... <https://lernen.min.uni-hamburg.de/course/view.php?id=1088>

Übungswochenbeginn: ..... 08.04.2021

## Kernbegriffe

Eine *integrierte Entwicklungsumgebung* (engl.: integrated development environment, kurz IDE) ist beim Programmieren neben der Programmiersprache ein wichtiges Werkzeug. Sie stellt u.a. Editoren für verschiedene Quelltexte (z.B. Textdateien, XML-Dateien, Quellcode, ...) zur Verfügung. Sie lässt Quellcode durch den Compiler übersetzen und bietet Unterstützung beim Debuggen von Programmen. Sie integriert viele der zentralen Tätigkeiten bei der Softwareentwicklung unter einer gemeinsamen Oberfläche.

In den Quelltextkonventionen unterscheiden wir zwei wichtige Arten von Kommentaren: *Implementationskommentare* und *Schnittstellenkommentare*. Während Implementationskommentare sich an Wartungsprogrammierer richten, die den Code einer Klasse verstehen und ändern müssen, dienen Schnittstellenkommentare dazu, die Schnittstelle einer Klasse verstehen und nutzen zu können. Die Sprache Java unterstützt die Unterscheidung, indem sie unterschiedliche Kommentar-Arten anbietet. Implementationskommentare werden mit den Zeichen `/**` eingeleitet, sie sind nur im Quelltext für den Programmierer sichtbar. Für Schnittstellenkommentare werden so genannte *Javadoc-Kommentare* verwendet, aus denen eine API-Dokumentation generiert werden kann. Hierfür steht das konfigurierbare Werkzeug *javadoc* zur Verfügung, das in viele IDEs integriert ist. BlueJ z.B. benutzt es, um die Schnittstellen-Sicht einer Klasse im Editor zu erzeugen. Javadoc wertet auch die sogenannten *Javadoc-Tags* innerhalb dieser Kommentare aus, das sind vordefinierte Schlüsselwörter für bestimmte Aspekte der Dokumentation. Beispiele sind die Tags `@param`, `@return` und `@author`. Die komplette Dokumentation der Java-API ist aus javadoc-Kommentaren generiert.

## Checkliste Übungsschein

Um die erste Woche abzuschließen müsst ihr

### Außerhalb der Übungszeit

- ☐ Scheinkriterien lesen

Moodle-Tests **mit voller Punktzahl bis Mittwoch, den 14.04.2021, 15:45 abgeben:**

- ☐ Test Rahmenbedingungen
- ☐ Test Kernbegriffe
- ☐ Test SE1-Wissen
- ☐ Test Hello World

### Während der Übung

- ☐ Anwesenheit selbst eingetragen
- ☐ Präsenzübung 1.1 Abgabe bekommen
- ☐ Präsenzübung 1.2 Abgabe bekommen
- ☐ Präsenzübung 1.3 Abgabe bekommen





## Aufgabe 1.0 Scheinkriterien lesen

Um sicher zu gehen, dass jeder weiß, wie die Scheinkriterien aussehen, sollt ihr sie euch einmal durchlesen. Ihr findet die Scheinkriterien oben im Moodle. Ihr müsst **selber** darauf achten, dass ihr diese erfüllt, um den Übungsschein zu erhalten. Ihr dürft zwar unabhängig davon die Klausur mitschreiben, allerdings benötigt ihr den Übungsschein, um das Modul zu bestehen. Falls ihr den Übungsschein nicht erhaltet, müsst ihr nächstes Jahr noch einmal an der Übung teilnehmen.

## Aufgabe 1.1 Wechsel der Entwicklungsumgebung


BlueJ ist gut für den Einstieg in die objektorientierte Programmierung geeignet, für die Arbeit an größeren Projekten jedoch nicht. Deshalb steigen wir an dieser Stelle auf die Entwicklungsumgebung Eclipse um. Eclipse ist frei verfügbar und wird auch im professionellen Kontext häufig eingesetzt. Ihr solltet bereits Eclipse und Java installiert haben. Schaut ansonsten unter "Werkzeuge und Hilfsmittel" im Moodle nach.

In dieser Aufgabe geht es darum, Eclipse kennen zu lernen. Es macht nichts, wenn ihr den Quelltext nicht komplett versteht. Für den ersten Umgang mit Eclipse werden wir das Projekt *Mediathek* nutzen. Wenn ihr Probleme bei dieser Aufgabe habt, fragt lieber einen Betreuenden statt zu viel Zeit zu investieren!

-  **1.1.1** Startet Eclipse, indem ihr im Windows-Startmenü „eclipse“ eingibt und dann „Eclipse“ auswählt. Beim ersten Starten erhaltet ihr einen Dialog, in welchem ihr den so genannten Workspace angebt. Dies ist ein Ordner, in dem Eclipse alle Projekte ablegt und verwaltet. Sucht euch selber einen geeigneten Ordner aus. Falls dieser Dialog nicht erscheint, kann der Workspace über den Menüpunkt *File>Switch Workspace>Other...* geändert werden.
-  **1.1.2** Für einen reibungslosen Austausch von Daten über Plattformgrenzen hinweg sollte die Codierung für Textdateien auf UTF-8 gestellt werden. Geht dazu auf *Window>Preferences>General>Workspace* und stellt dort das *Text file encoding* auf *Other: UTF-8* ein.  
  
Ladet anschließend die Datei *Mediathek\_Vorlage\_Blatt01-03.zip* aus dem SE2-Moodle auf euren Rechner. In diesem Archiv befindet sich ein Eclipse-Projekt, das wir nun importieren. Wählt dazu in Eclipse *File>Import...* und dann unter *General Existing Projects into Workspace* und gebt im folgenden Schritt für **Select archive file** die Datei *Mediathek\_Vorlage\_Blatt01-03.zip* an.  
  
Wechselt mit *Window>Open Perspective>Java* in die so genannte *Java-Perspektive*. Wenn auf der linken Seite kein *Package Explorer* geöffnet sein sollte, öffnet ihn mit *Window>Show View>Package Explorer*. Klappt das Mediathek-Projekt auf und schaut euch an, wie Eclipse Klassen und deren Quelltexte darstellt (per Doppelklick auf eine Klasse wird ein Editor geöffnet).
-  **1.1.3** Macht euch klar, was die verschiedenen Fenster der Eclipse Java-Perspektive anzeigen (wenn nicht alles gleich verständlich ist - nicht so schlimm). Findet heraus, was die *Outline-View* ist, und erklärt es eurem Betreuer/eurer Betreuerin. Welche Bedeutung haben Farbe und Form? Warum beginnen einige Einträge mit einem Unterstrich?
-  **1.1.4** Eclipse kann euch beim Einhalten der Quelltextkonventionen helfen. Ladet aus dem Moodle die Datei *SE2Format.xml* herunter. Wählt dann *Preferences>Java>Code Style>Formatter*, drückt auf *Import...* und wählt die heruntergeladene XML-Datei. Bestätigt zuletzt mit OK. Unter *Preferences>Java>Editor>Save Actions* müsst ihr den Haken bei *Perform the selected actions on save* und dann bei *Format source code* setzen.

## Aufgabe 1.2 Mediathek kennen lernen

Mit dieser Aufgabe lernt ihr die Mediathek genauer kennen. Es handelt sich dabei um ein Entwicklungsprojekt, das uns während der gesamten Laborphase 1 begleiten wird. Die Mediathek ist eine Software für Mediathekare. Ein Mediathekar bedient über einen Touchscreen die Programmoberfläche, leiht Medien an Kunden aus und nimmt diese wieder zurück. Die aktuelle Version der Mediathek enthält noch nicht die gesamte benötigte Funktionalität. In den nächsten Wochen werdet ihr die Software nach und nach ausbauen. Die grafische Benutzungsoberfläche wird gestellt. Ihr seid für die Umsetzung der benötigten Funktionalität verantwortlich.

-  **1.2.1** In BlueJ lassen sich Exemplare interaktiv erzeugen und beliebige Methoden an diesen Exemplaren aufrufen. Das ist eine besondere Eigenschaft von BlueJ, auf die wir nun verzichten müssen. Üblicherweise werden Java-Programme über die Methode `public static void main(String[] args)` gestartet. Wählt im Package Explorer die Klasse *StartUpMediathek.Blatt.01.03* aus und startet die Methode *main*, indem ihr *Run As>Java Application* aus dem Kontextmenü der Klasse auswählt. Spielt **etwas** mit der Oberfläche herum. Welche Funktionalität ist bereits

implementiert? Findest du grundlegende Funktionalitäten, die nicht korrekt funktionieren? Was wird noch nicht unterstützt, sollte aber in keiner Mediathek fehlen?

- 1.2.2 Uns interessiert, wie die Test-Situation in dem Projekt aussieht. Führt dafür einmal alle Tests des Projektes durch indem ihr mit rechtsklick auf das Java-Projekt klickt und dann *Run as ...* → *JUnit Test* → auswählt. Wie viele Testklassen wurden ausgeführt?

- Wie viele Test-Klassen gibt es?

- 1.2.3 **Zeichnet ein UML-Klassendiagramm**, ausgehend vom Interface *VerleihService*. Eine Vorlage für Lucidchart findet ihr hier:

<https://lucid.app/lucidchart/invitations/accept/892c77b4-179e-478c-af3a-4982092d4917>

Öffnet den Link und speichert euch das Diagramm unter *Datei* > *Duplizieren*, um es bearbeiten zu können. Um die UML-Symbole zur Auswahl zu haben, klickt während das Diagramm geöffnet ist links oben auf *Formen* und setzt dann den Haken bei UML. Diese werdet ihr aber erst auf den kommenden Zetteln brauchen.

Über den Button oben rechts könnt ihr das Diagramm auch zum Bearbeiten für euren Partner frei geben. Speichert euch die Datei zum Weiterarbeiten für **spätere Zettel**! Einer von euch sollte sich das Dokument duplizieren, damit ihr beide eine Version habt. So stellt ihr sicher, dass ihr bei einem Partnertausch nicht am selben Dokument weiter arbeitet.

Im Moodle befindet sich bei Bedarf eine Zusammenfassung der UML Pfeile als Wiederholung von SE1.

### Aufgabe 1.3 Aufgabenteilung über Interfaces kennen lernen

Das GUI-Team arbeitet parallel zum SE2-Team. Damit die beiden Teams möglichst unabhängig voneinander arbeiten können, haben sie sich auf das Interface *VerleihService* geeinigt, über das die notwendigen Informationen ausgetauscht werden.

- 1.3.1 Das GUI-Team wollte nicht auf die Implementation des SE2-Teams warten und hat deshalb schon einmal einen so genannten *Dummy* geschrieben, der das Interface *VerleihService* implementiert. Untersucht die Klasse *DummyVerleihService*. Welchem Zweck dient sie? Implementiert sie das Interface so, wie die Autoren von *VerleihService* es sich gedacht haben? Welche Grenzen hat die Implementation? **Schreibt als Schnittstellenkommentar zu jeder Methode, was sie im Gegensatz zur Beschreibung aus dem Interface macht.**

- 1.3.2 Zum Glück gibt es mittlerweile eine Implementation des SE2-Teams, die besser funktioniert: *VerleihServiceImpl*. Wir wollen nun den *DummyVerleihService* ersetzen. Hierzu müssen wir erstmal herausfinden, an welcher Stelle ein Objekt dieser Klasse erzeugt wird. Klickt im Editor der Klasse *DummyVerleihService* mit rechts auf den Konstruktornamen und wählt *References* → *Project*. Es öffnet sich eine View, in der angezeigt wird, wo der Konstruktor überall aufgerufen wird. Dies ist in unserem Programm nur eine Stelle. Mit einem Doppelklick könnt ihr nun dorthin springen und stattdessen den Konstruktor des *VerleihServiceImpl* einsetzen.

- 1.3.3 Falls ihr eurem Partner das Java Projekt zur Verfügung stellen wollt, müsst ihr das Projekt exportieren. Das geht in Eclipse mit dem Menüpunkt *File* → *Export*. Dort wählt ihr *General* → *Archive File*. Auf späteren Zetteln werdet ihr so auch einige Aufgaben abgeben.

Falls noch nicht geschehen, stellt auch sicher, dass beide das UML in Lucidchart abgespeichert habe. Geht ggf. auf *Duplizieren* und speichert es euch ab.