Universität Stuttgart

Software Quality and Architecture Group

Übungsblatt 07

Programmierung und Softwareentwicklung (WS 2024/2025)

Abgabe: Fr. 06.12.2024, 23:59 Uhr — Besprechung: KW 49

- Bitte lösen Sie die Übungsaufgabe in **Gruppen von 2 Studierenden**.
- Dieses Übungsblatt besteht aus zwei Teilen (A, B). Teil A ist in der Präsenzübung zu lösen. Teil B ist in Heimarbeit (Gruppe von 2 Studenten) zu lösen und rechtzeitig abzugeben. Die Abgabe erfolgt über ILIAS.
- Geben Sie `.java`-Dateien nur im UTF-8 Encoding ab. Ändern Sie das Textdateiencoding auf UTF-8 ab, bevor Sie die Unterlagen herunterladen. Abhängig von Ihrem Betriebssystem müssen Sie möglicherweise auch nichts tun.
- Geben Sie zu Beginn der Dateien Ihre Namen (Vor- und Nachname), die Matrikelnummern und die E-Mail-Adressen an. Nutzen Sie bei Java-Dateien die korrekte JavaDoc-Syntax.
- Benennen Sie die Dateien nach dem folgenden Schema:
  1. **PSE**[**ÜB-Nr**]-[**Nachnamen der Teammitglieder**]-[**Nachname des Tutors**]**.pdf**
     Beispiel: PSE07-StießSpethKrieger-Becker.pdf
  2. [**Klassenname**]**.java**: Alle von Ihnen bearbeiteten Java-Dateien, die Lösungen für die Aufgaben enthalten.
- Missachtung der formalen Kriterien kann dazu führen, dass einzelne Aufgaben oder die gesamte Abgabe mit 0 Punkten bewertet werden.

Lernziel: Auf diesem Übungsblatt werden Sie Ihr Wissen über Sichtbarkeiten, Typen und Variablen, Kontrollflussstrukturen und eigene Klassen vertiefen. Des weiteren werden Sie den Umgang mit primitiven und unveränderlichen Datentypen und das Erstellen von Komplexen Algorithmen weiter üben.

Punkte: Dieses Übungsblatt enthält zwei Teile. In Teil B können Sie bis zu 50 Punkte und 2 Bonuspunkte erzielen. Zum Bestehen des Blatts benötigen Sie mindestens 25.0 Punkte.

Style: Bitte halten Sie die in der Vorlesung vorgestellten Style-Regeln ein. Dazu gehören auch JavaDoc sowie Vor- und Nachbedingungen. Der Style Ihrer Implementierung wird mit bis zu 50% bewertet.

Vorbereitung: Bitte erledigen Sie die folgenden Schritte **vor** der Präsenzübung.

- Importieren Sie das zu diesem Übungsblatt gehörende Maven Projekt in Ihre IDE. Sie finden das Maven Projekt in unserem git-Repository: `https://github.com/SQA-PSE-WS-2024-2025/exercise-sheet-07`
- Stellen Sie sicher, dass Sie Übungsblatt 06 absolviert haben, sowie alle Software installiert und funktionsfähig ist (IDE (Eclipse, IntelliJ, VSCode,...) und Java 21).

Unterlagen:

- Git-Repositories: `https://github.com/SQA-PSE-WS-2024-2025/`
- Dokumentation des Hamstersimulators: `https://tinyurl.com/5yx654w8`

Scheinkriterien: Durch die Teilnahme am Übungsbetrieb können Sie sich für die Teilnahme an der Klausur qualifizieren:

- Bestehen von min. 80% aller Übungsblätter.
- Ein Übungsblatt gilt als bestanden, wenn 50% der Punkte des abgegebenen Heimarbeitsteils erreicht wurden.
- Aktive Teilnahme an min. 80% der Übungen.

Viel Erfolg!

1 Teil A - Präsenzaufgaben

Aufgabe 1 Mutable und Immutable

Ziel dieser Aufgabe ist es, Ihnen die Konzepte von unveränderlichen Objekten näherzubringen. Dazu sollen Sie eine Klasse für unveränderliche Studierendenobjekte implementieren: Attribute der Objekte dieser Klasse sollen durch den Konstruktor gesetzt werden und anschließend nicht mehr verändert werden können.

Hinweis: Im Abschnitt *Anhang - Anleitungen* finden Sie Anleitungen und Hinweise zum Erstellen von eigenen Klassen und zum Generieren von Konstruktoren und Getter- und Setter-Operationen, auf die Sie in den folgenden Teilaufgaben zurückgreifen können.

(a) Was sind Vorteile des Konzepts von unveränderlichen Objekten?

(b) Erstellen Sie im Paket `de.unistuttgart.iste.sqa.pse.sheet07.presence.immutable` eine neue Klasse `MyImmutableStudent`. Die Klasse soll unveränderlich sein. Fügen Sie der Klasse die folgenden Elemente hinzu:

- ein Attribut[1] vom Typ `String`, dass den Namen repräsentiert.
- ein Attribut vom Typ `long`, dass die Matrikelnummer repräsentiert.
- einen Konstruktor und Getter-Operationen für alle Attribute.

Hinweis: Sie finden Regeln für unveränderliche Klassen auf der Folie „Regeln für den Code unveränderlicher Klassen“ im Foliensatz zu Typen und Variablen.

(c) Öffnen Sie die Klasse `Main`. Die Klasse hat eine Klassenoperation `main`. Erstellen Sie in der `main` Operation ein Objekt der Klasse `MyImmutableStudent` und geben Sie Namen und Matrikelnummer des Studierenden (mittels der Getter-Operationen) auf der Konsole aus. Führen Sie die `main`-Operation aus. Versuchen Sie, die Attribute des Studierendenobjekts zu ändern. Ist dies möglich?

(d) **Herausforderung** Passen Sie die `main`-Operation so an, dass für die Ausgabe von Name und Matrikelnummer die `format` Klassenoperation der Stringklasse verwendet wird. Die Dokumentation der `format`-Operation finden Sie unter: `https://tinyurl.com/formatJava21`. Da die Dokumentation etwas unübersichtlich ist, können Sie zusätzlich das Beispiel in Listing 1 betrachten.

```
1 System.out.printf("Beispiel:\n %d, %s", 42, "value");
2 //Konsolenausgabe: Beispiel: (Zeilenumbruch) 42, value
```

Listing 1: format Beispiel.

Aufgabe 2 Mutable-Objekt innerhalb von Immutable-Objekt

In dieser Aufgabe erweitern Sie die Klasse `MyImmutableStudent` um einen Geburtsort in Form einer Objektvariable vom Typ `Address`. Die Klasse `Address` wurde bereits implementiert, Sie finden sie im Repository. Die Implementierung von `Address` darf in dieser Aufgabe nicht verändert werden.

(a) Erläutern und begründen Sie, warum die Klasse `Address` nicht unveränderlich ist. Geben Sie im Code ein Beispiel an, das demonstriert, wie Attribute von `Address` geändert werden können.

(b) Fügen Sie ein neues Attribut vom Typ `Address` zur Klasse `MyImmutableStudent` hinzu. Das Attribut repräsentiert den Geburtsort des Studierenden. Erweitern Sie den Konstruktor um einen weiteren Parameter für die Adresse des Geburtsorts. Passen Sie auch die `main`-Operation an.

(c) Diskutieren Sie mit Ihrem Teampartner, ob und warum Ihre Klasse noch oder nicht mehr unveränderlich ist. Halten Sie Ihre Ergebnisse schriftlich fest.

(d) Falls die Objekte Ihrer Klasse nicht mehr unveränderlich sind, passen Sie Ihren Code so an, dass die Objekte der Klasse wieder unveränderlich werden. Bearbeiten Sie hierfür ausschließlich die Klasse `MyImmutableStudent`.

(e) **Herausforderung:** Erstellen Sie einen Record-Typ für Studiengänge mit dem Bezeichner `CourseOfStudy`, der einen String für den Namen des Studiengangs besitzt.

[1] Erinnerung: die Begriffe *Attribut* und *Objektvariable* sind synonym, vgl. VL 06 „06-Referenzen und Objekterzeugung“.

(f) **Herausforderung:** Fügen Sie in der Klasse `MyImmutableStudent` ein Attribut vom Typ `CourseOfStudy` hinzu, das den Studiengang eines Studierenden repräsentiert. Erweitern Sie den Konstruktor um einen zusätzlichen Parameter für den Studiengang des Studierenden. Passen Sie auch die `main`-Operation an. Geben Sie anschließend den Namen des Studiengangs auf der Konsole aus.

Aufgabe 3 Frühjahrsputz I

In dieser Aufgabe beschäftigen Sie sich mit funktionaler Dekomposition. Außerdem erstellen Sie Ihre erste eigene Klasse. Formulieren Sie, wo sinnvoll, JavaDoc inklusive Vor- und Nachbedingungen.

Paule befindet sich in seiner Höhle aus Abbildung 1. Überall liegen Körner verstreut.

Paule steht zu Beginn des Spiels links oben in der Höhle. Dieses Feld ist definiert als das erste Feld in der ersten Zeile. Nach rechts folgen dann die Felder zwei bis sechs und nach unten die Zeilen zwei und drei.

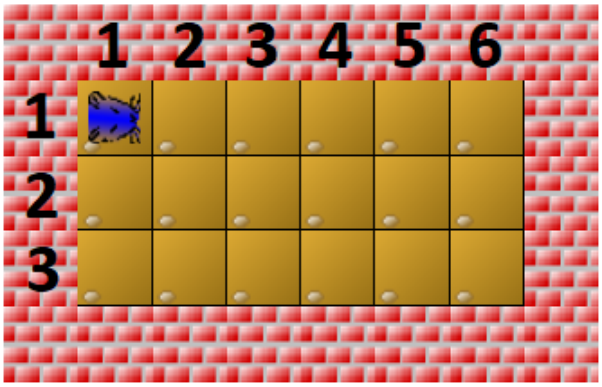

Abbildung 1: Paule in seiner Höhle.

Ziel dieser Aufgabe ist, dass Paule einen Frühjahrsputz macht. Dieser soll wie folgt ablaufen:

Als erstes werden in der ersten Zeile alle Körner auf ungeraden Feldern aufgesammelt. Danach läuft Paule in die zweite Zeile und sammelt dort alle Körner auf geraden Feldern auf. Danach läuft Paule in die dritte Zeile und sammelt dort nochmal alle Körner auf ungeraden Feldern auf. Zuletzt legt Paule alle eingesammelten Körner auf dem letzten Feld der dritten Zeile ab.

(a) Zerlegen Sie den oben beschriebenen Frühjahrsputz in wiederverwendbare Teilschritte. Beschreiben Sie jeden Teilschritt, z.B. mit Pseudocode. Halten Sie die Beschreibungen schriftlich fest.

(b) Legen Sie ein neues Paket `de.unistuttgart.iste.sqa.pse.sheet07.presence.cleaning` an. Erstellen Sie in diesem Paket die Klasse `SpringCleaning`. Fügen Sie der Klasse die folgenden Elemente hinzu:

- ein Attribut vom Typ `Hamster` mit dem Bezeichner `cleaningHamster`.
- einen Konstruktor mit der Signatur `public SpringCleaning(final Hamster cleaningHamster)`.

Hinweis: Beachten Sie die Hinweise zum Erstellen von eigenen Paketen und Klassen im Abschnitt *Anhang - Anleitungen*.

(c) Implementieren Sie den Konstruktor aus Teilaufgabe (b) so, dass er der Dokumentation aus Listing 2 entspricht. Überprüfen Sie die Vorbedingungen mit defensiver Programmierung.

```
1 /**
2  * Create and initialize a new spring cleaning object
3  * with the given parameters.
4  *
5  *  @param cleaningHamster the hamster that will clean.
6  *                         Must not be {@code null}.
7  */
```

Listing 2: Dokumentation des Konstruktors der Klasse `SpringCleaning`.

(d) Implementieren Sie in Teilaufgabe (a) identifizierten Teilschritte des Frühjahrsputz. Erstellen Sie für jeden Teilschritt eine neue Operation in der Klasse `SpringCleaning`.
Hinweis: Nutzen Sie den Modulo-Operator (%, Division mit Rest), um zu bestimmen, ob sich Paule auf einem geraden oder einem ungeraden Feld befindet.

(e) Implementieren Sie eine Operation `cleanCave`, in der der putzende Hamster die drei Zeilen der Höhle wie oben beschrieben putzt, also in der ersten Zeile alle Körner auf ungeraden Feldern, in der zweiten Zeile alle Körner auf den geraden Feldern und in der dritten Zeile wieder alle Körner auf ungeraden Feldern. Verwenden Sie die in Teilaufgabe (d) implementierten Operationen.

(f) Öffnen Sie die Klasse `SpringCleaningHamsterGame` und implementieren Sie in der Operation `run` die folgende Schritte. Führen Sie das Spiel danach aus.

- Erzeugen Sie eine Instanz der Klasse `SpringCleaning`. Übergeben Sie `paule` als Argument.
- Rufen Sie die Operation `cleanCave` auf.

Aufgabe 4 Frühjahrsputz II

In dieser Aufgabe verwenden Sie Aufzählungsdatentypen, um Paule und anderen Hamstern dabei zu helfen, ihre Höhle auf unterschiedliche Arten zu putzen. Öffnen Sie dazu die in Aufgabe 3 erstellte Klasse `SpringCleaning`.

(a) Fügen Sie der Klasse die folgenden Elemente hinzu:

- einen Aufzählungsdatentypen (`enum`) mit dem Bezeichner `CleaningMode` und den Konstanten `CLEAN_ODD` und `CLEAN_EVEN`.
- ein Attribut vom Typ `CleaningMode`. Wählen Sie einen passenden Bezeichner.

(b) Erweitern Sie den Konstruktor der Klasse um einen zweiten Parameter `cleaningMode` vom Typ `CleaningMode`. Verwenden Sie den Parameter, um das Attribut vom Typ `CleaningMode` aus der vorherigen Teilaufgabe zu initialisieren.

(c) Überarbeiten Sie die Operation `cleanCave` so, dass der putzende Hamster abhängig vom Wert des `CleaningMode`-Attributs in seiner Höhle entweder in allen Zeilen immer nur die Körner auf ungeraden Feldern (`CLEAN_ODD`) aufputzt, oder in allen Zeilen immer nur die Körner auf gerade Feldern (`CLEAN_EVEN`) aufputzt. Verwenden Sie dazu eine `switch`-Anweisung (siehe Abschnitt „Kammartige Verzweigungen: Java Switch am Beispiel“ und „Javas switch Instruktion“ im Foliensatz zu Kontrollflussstrukturen).

(d) Öffnen Sie die Klasse `SpringCleaningHamsterGame` und „reparieren“ Sie den Konstruktoraufruf in der Operation `run`. Verwenden Sie `SpringCleaning.CleaningMode.CLEAN_ODD` als Argument. Führen Sie das Spiel aus.

(e) Bleiben Sie in der Klasse `SpringCleaningHamsterGame` und erweitern Sie die Operation `run` wie folgt. Führen Sie das Spiel danach aus.

- Erstellen Sie einen weiteren Hamster. Platzieren Sie den neuen Hamster auf dem selben Feld wie Paule, also auf dem ersten Feld in der ersten Zeile. Lassen Sie den neuen Hamster nach Osten blicken.
- Erstellen Sie eine weitere Instanz der Klasse `SpringCleaning`. Übergeben Sie den neuen Hamster und `SpringCleaning.CleaningMode.CLEAN_EVEN` als Argumente.
- Rufen Sie die Operation `cleanCave` für die neue `SpringCleaning`-Instanz auf, damit der neue Hamster beim Putzen hilft.

2 Teil B - Heimarbeit

Aufgabe 1 Primitive Datentypen und Klassen (PDF)

In dieser Aufgabe beschäftigen Sie sich mit primitiven Datentypen und Klassen.

(a) *(3 Punkte)* Worin unterscheiden sich primitive Variablen und Referenzvariablen? Was sind Wrapper-Klassen?

(b) *(9 Punkte)* Betrachten Sie die nachfolgenden Ausdrücke. Gehen Sie davon aus, dass `game` ein Objekt der Klasse `HamsterGame` und `paule` ein Objekt der Klasse `Hamster` ist. Wir setzen voraus, dass sich `paule` in der Mitte eines 3×3-Territoriums befindet, nach Osten schaut und keine Körner im Mund hat. Auf den Feldern des Territoriums befindet sich jeweils ein Korn (vgl. Abbildung 2).

Geben Sie für jeden der folgenden Ausdrücke den Datentyp und den Wert an. Geben Sie bei Objekten auch die Werte aller Attribute für das in Abbildung 2 gegebene Szenario an.

```
1  paule
2  Long.valueOf(24756)
3  paule.frontIsClear() || paule.grainAvaiable()
4  5 - 3.5f
5  false != !false
6  "The size of the territory is " + 3 + 'x' + 3 + '!'
7  paule.getLocation()
8  paule.getLocation().getColumn() +
       paule.getLocation().getRow()
9  Integer.MAX_VALUE
```

(c) *(6 Punkte)* Der Codeschnipsel in Listing 3 ist fehlerhaft. In `numberOfColumns` sollte der Wert 3 gespeichert werden, da das Territorium drei Spalten besitzt. Erläutern Sie, welche Typen von `getTerritory()` und `getColumnCount()` zurückgegeben werden und weshalb der Fehler auftritt. Klassifizieren Sie den Fehler als lexikalisch, syntaktisch oder statisch bzw. dynamisch sematisch. Was muss verändert werden, damit die Spaltenzahl des Territoriums in `numberOfColumns` gespeichert wird? Gehen Sie auch hier davon aus, dass auf `game` und `paule` entsprechende Operationen der Klasse `HamsterGame` bzw. `Hamster` ausgeführt werden können.

```
1  int numberOfColumns =
       paule.getTerritory().getColumnCount();
```

Listing 3: Fehlerhafter Codeschnipsel.

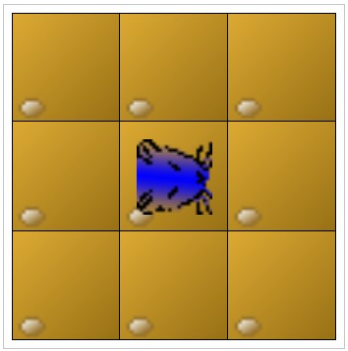

Abbildung 2: Paule auf dem kleinen Spielfeld

Aufgabe 2 Behandlung von Ausnahmen (PDF)

(a) *(3 Punkte)* Informieren Sie sich, warum Ausnahmen in Programmen benötigt werden und welche Möglichkeiten Sie in Java haben, mit Ausnahmen umzugehen. Nennen Sie außerdem vier relevante Java-Schlüsselwörter.

(b) *(3 Punkte)* In Java wird zwischen zwei Arten von Ausnahmen unterschieden: *checked Exceptions* und *unchecked Exceptions* (auch RuntimeExceptions). Recherchieren Sie, was der Unterschied zwischen den beiden ist.

Aufgabe 3 Exceptions (PDF, `.java`)

In dieser Aufgabe vertiefen Sie Ihre Kenntnis über die Behandlung von Ausnahmefällen.

(a) *(2 Punkte)* Stellen Sie sich ein beliebiges Territorium vor, in dem ein Hamster lebt. Es können unerwartete Probleme auftreten, wenn z. B. der Hamster vor einer Wand steht, diese Wand anschaut und sich um ein Feld nach vorne bewegen soll. Um solche Problemfälle zu behandeln, existieren auch im Hamstersimulator Exceptions. Nenne Sie zwei Runtime-Exceptions des Hamstersimulator.

(b) *(5 Punkte)* In der Winterzeit ist Paule faul geworden. Implementieren Sie die Operation `tryToMove` der Klasse `LazyHamsterGame`. In dieser Operation bewegt sich Paule nur noch mit einer Wahrscheinlichkeit von 75 Prozent nach vorne, ansonsten bleibt er sitzen und es wird eine `TooLazyException` geworfen.[2]

Hinweis: Nutzen Sie `Math.random()` um Paules Gemütslage zu modellieren.

(c) *(4 Punkte)* Paule hat zusätzlich Klaustrophobie bekommen und fürchtet sich, dass er in einem Feld eingesperrt ist. Implementieren Sie die Operation `isCaged` der Klasse `LazyHamsterGame`. Wenn Paules Feld auf allen vier Seiten an Wände grenzt, gibt die Operation `true` zurück, sonst `false`.

Hinweis: Hierbei handelt es sich um eine Abfrage. Achten Sie also darauf, dass sich der Hamster nach der Ausführung der Operation wieder in seinem Ausgangszustand befindet. Veränderung der Simulatorlogs dürfen Sie an dieser Stelle vernachlässigen.

(d) *(2 Punkte)* Handelt es sich bei der Operation `isCaged`, so wie Sie sie in der vorherigen Teilaufgabe implementiert haben, um eine stilistisch korrekte Abfrage? Begründen Sie Ihre Antwort.

Hinweis: Veränderungen des Simulatorlogs dürfen Sie and dieser Stelle vernachlässigen.

(e) *(7 Punkte)* Implementieren Sie die Operation `moveMultipleSteps` der Klasse `LazyHamsterGame`. Die Operation hat eine positive ganze Zahl n als Parameter und lässt Paule n Schritte laufen. Verwenden Sie zum Laufen die Operation `tryToMove`. Wenn das Feld vor Paule nicht frei ist, soll er sich nach links drehen. Wenn keines der an Paules Feld anliegenden Felder frei ist, wird eine `NoWayToGoException` geworfen. Wenn er nicht vorankommt, weil er zu faul ist, soll er angefeuert werden (Hamster feuert man an, indem man mittels ihrer `write`-Operation einen motivierenden Text ausgibt). So oder so soll er nach Ausführung dieser Operation tatsächlich genau n Schritte gemacht haben (Drehungen und Anfeuerungen zählen nicht als Schritte). Gehen Sie davon aus, dass an Paules Feld immer mindestens ein weiteres freies Feld angrenzt.

Führen Sie das Spiel aus.

Hinweis: Da Sie die zufallsbasierte Operation `tryToMove` zum Bewegen des Hamster nutzen, dürfen Sie in dieser Teilaufgabe ausnahmsweise auf Schleifenvarianten und -invarianten verzichten.

Hinweis: Wenn Sie das Spiel `LazyHamsterGameApp` starten, befindet sich Paule in dem in Abbildung 3 gezeigten Territorium. Wenn Sie das Spiel `CagedHamsterGameApp` starten, befindet sich Paule auf einem Territorium, in dem er von allen vier Seiten von Wänden umgeben ist. Starten Sie das erste Spiel, um das „normale“ Verhalten der Operation `moveMultipleSteps` zu beobachten. Starten Sie das zweite Spiel, um einen eingesperrten Hamster zu beobachten.

(f) *(2 Bonuspunkte)* Erläutern Sie, warum die zufallsbasierte `tryToMove`-Operation dazu führt, dass Sie in der Teilaufgabe (e) keine Schleifenvariante angeben können.

Aufgabe 4 IDE Tools (PDF)

Bei dieser Aufgabe geht es darum, Ihre verwendete IDE (IntelliJ IDEA, Eclipse oder VS Code) näher kennenzulernen.

Geben Sie für alle in den Teilaufgaben (a) bis (f) genannten IDE Features an,

1. Welche IDE Sie verwenden.
2. Wo das Feature zu finden ist und wie man es verwendet. Geben Sie, falls das Feature nicht automatisch aktiviert ist, außerdem an, wo und wie Sie das Feature aktivieren können.

[2] Diese recht unnatürliche Verwendung von Exceptions dient dazu, in übersichtlichem Rahmen das Arbeiten damit zu üben – Sie dürfen sich gerne auch eine Übermittlung der Anweisungen über eine instabile Netzwerkverbindung oder Ähnliches vorstellen; ein fauler Hamster ist aber schneller programmiert.

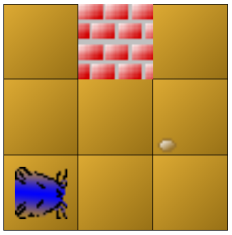

Abbildung 3: Ein Beispiel für `moveNSteps`: Für $n = 5$ würde Paule ganz nach rechts laufen, sich nach links drehen, nach oben laufen, wenden und noch ein Feld nach unten laufen, säße also am Ende beim Korn, egal wie oft er sich auf dem Weg dorthin bei `tryToMove` gesträubt hat.

Beispiel „Suche innerhalb einer Datei“: Verwendung `ctrl+f` oder Edit → Find/Replace, Feature immer verfügbar, keine zusätzliche Aktivierung nötig.

3. Was das Feature bewirkt.
4. Welchen Vorteil dieses Feature bietet.

Achten Sie insbesondere bei den Angaben zu 3. und 4. auf ausführliche Antworten. „Um das Entwickeln zu erleichtern“ ist beispielsweise nicht ausreichend.

Achtung: Antworten auf diese Fragen finden Sie nicht in den Vorlesungsfolien! Die Verwendung der Dokumentation Ihrer IDE und/oder das Verwenden von Suchmaschinen wird dringendst empfohlen.

(a) *(1 Punkt)* Autovervollständigung (Autocomplete)

(b) *(1 Punkt)* Automatische Imports (Autoimport)

(c) *(1 Punkt)* In Projekt suchen
(*Hinweis*: hiermit ist nicht die Suche innerhalb *einer einzelnen* Datei gemeint, sondern eine Suchfunktion, die ein ganzes Projekt auf einmal durchsuchen kann.)

(d) *(1 Punkt)* Spezifische Fehler und Warnungen im Code hervorheben und anzeigen lassen

(e) *(1 Punkt)* Methode extrahieren (extract method) (Teil von refactoring)

(f) *(1 Punkt)* Umbenennen von Bezeichnern (z.B. Methoden- oder Klassennamen)

3 Anhang - Anleitungen

Beim Bearbeiten des Übungsblattes können Sie auf die folgenden Anleitungen zurückgreifen.

IntelliJ

Getter- und Setter-Operationen generieren

Klicken Sie auf Code, Generate, Getter and Setter (alternativ kann man auch nur Getter bzw. Setter wählen). Markieren Sie die Variablen, von denen Sie eine Getter- und/oder Setter-Operation erzeugen lassen möchten (mehrere, indem Sie Strg/Ctrl beim Auswählen gedrückt halten) und bestätigen Sie mit OK.

Konstruktoren generieren

Klicken Sie auf Code, Generate, Constructor. Markieren Sie die Variablen, die Sie als Parameter im Konstruktor haben möchten (mehrere, indem Sie Strg/Ctrl beim Auswählen gedrückt halten) und bestätigen Sie mit OK.

Pakete erstellen

Sie erstellen ein neues Paket, indem Sie mit einem Rechtsklick auf einer beliebigen Datei → New → Package wählen. Hiernach geben Sie den gewünschten Paketnamen ein und bestätigen mit `Enter`.

Klassen erstellen

Sie erstellen eine neue Klasse, indem Sie mit einem Rechtsklick auf das Paket machen, in dem Sie die Klasse erstellen wollen, und dann New → Java Class wählen. Geben Sie den gewünschten Namen ein und bestätigen mit `Enter`.

Eclipse

Getter- und Setter-Operationen generieren

Klicken Sie auf Source, Generate Getters and Setters ... . Markieren Sie die Variablen, von denen Sie eine Getter- und/oder Setter-Operation erzeugen lassen möchten, mit einem Haken und bestätigen Sie mit Generate.

Konstruktoren generieren

Klicken Sie auf Source, Generate Constructor using Fields ... . Markieren Sie die Variablen, die Sie als Parameter im Konstruktor haben möchten, mit einem Haken und bestätigen Sie mit Generate.

Pakete erstellen

Machen Sie einen Rechtsklick auf eine beliebige Datei in Ihrem aktuellen Projekt und wählen Sie → New → Package. Im erscheinenden Fenster tragen Sie unter „Name“ den gewünschten Namen des Pakets ein. Drücken Sie dann Finish.

Klassen erstellen

Machen Sie einen Rechtsklick auf eine beliebige Datei in Ihrem aktuellen Projekt und wählen Sie → New → Class. Im erscheinenden Fenster tragen Sie unter „Name“ den gewünschten Namen der Klasse ein. Wählen Sie unter „Package“ das gewünschten Paket. Drücken Sie dann Finish.

VSCode

Getter- und Setter-Operationen generieren

Machen Sie einen Rechtsklick. Klicken Sie auf Source Action..., Generate Getters.... Markieren Sie die Variablen, von denen Sie eine Getter-Operation erzeugen lassen möchten, mit einem Haken und bestätigen Sie mit OK.

Konstruktoren generieren

Machen Sie einen Rechtsklick. Klicken Sie auf Source Action..., Generate Constructors.... Markieren Sie die Variablen, die Sie als Parameter im Konstruktor haben möchten und bestätigen Sie mit OK.

Pakete erstellen

Machen Sie einen Rechtsklick auf eine beliebige Datei in Ihrem aktuellen Projekt und wählen Sie → New Java Package.... Geben Sie den gewünschten Namen des Pakets ein und bestätigen Sie mit `Enter`.

Klassen erstellen

Machen Sie einen Rechtsklick auf das Packet, in dem sie die neue Klasse erzeugen wollen und wählen Sie → New Java File → Class... aus. Geben Sie den gewünschten Bezeichner der Klasse ein und bestätigen Sie mit `Enter`.