Karlsruher Institut für Technologie

Prof. Dr.-Ing. Anne Koziolek
Architecture-driven Requirements Engineering (ARE)
https://sdqweb.ipd.kit.edu/wiki/Programmieren
programmieren-vorlesung@ipd.kit.edu

Programmieren – Sommersemester 2020

Übungsblatt 1 Version 1.1	20 Punkte	Ausgabe:	28.04.2020, ca. 13:00 Uhr
		Praktomat:	05.05.2020, 13:00 Uhr
		Abgabefrist:	13.05.2020, 06:00 Uhr

Allgemeine Hinweise

Bei organisatorischen Fragen zu ihrem Studiengang und zu speziellen Prüfungssituationen wenden Sie sich bitte an den entsprechenden Studiengangservice. Für die Fakultät für Informatik ist dies beispielsweise der ISS[1]. Verbindliche Informationen zu den Prüfungsmodalitäten für Programmieren im Rahmen Ihres Studiums erhalten Sie in den jeweils zutreffenden Modulhandbüchern, der Studienordnung und per Nachfrage beim Studiengangservice.

Hinweise zum Übungsbetrieb

Organisatorisches

- Im Semesterverlauf werden **5 Übungsblätter** ausgegeben. Diese erscheinen im Abstand von zwei Wochen. Die Bearbeitungszeit beträgt jeweils zwei Wochen.
- Jedes Übungsblatt hat einen Umfang von 20 Punkten. Insgesamt können 100 Punkte durch die Bearbeitung der Übungsblätter erreicht werden. Sie bestehen den Übungsschein durch das Erhalten von mehr als 50% der maximal zu erreichenden Punktzahl aus den Übungsblättern und mehr als 75% der maximal zu erreichenden Punktzahl aus der Präsenzübung.
- Des Weiteren ist zum Erhalt eines Übungsscheins eine Anmeldung im Campus Management Portal[2] erforderlich.
- Der Übungsschein ist unbenotet.
- Abgegebene Übungsblätter werden durch einen Tutor korrigiert und bewertet.
- In diesem Semester finden **Präsenztutorien** statt. Alle organisatorischen Informationen werden im ILIAS bekanntgegeben.

[1] https://www.informatik.kit.edu/iss.php
[2] https://campus.studium.kit.edu/

- Der Erhalt des (unbenoteten) Übungsscheins ist Voraussetzung für die Teilnahme an den beiden (benoteten) Abschlussaufgaben.
- Die Prüfung Programmieren ist eine **Orientierungsprüfung**, das heißt es gelten besondere Prüfungsmodalitäten. Insbesondere müssen die beiden Abschlussaufgaben vor Ende des dritten Fachsemesters bestanden werden.
- Studierende, die den ordnungsgemäßen Ablauf einer Erfolgskontrolle stören, können von der Fortsetzung der Erfolgskontrolle ausgeschlossen werden.

Abgabesystem

- Die Übungsblätter werden über das **Praktomat-System**[3] abgegeben. Abgaben auf anderem Weg werden grundsätzlich abgelehnt.
- Die **Freischaltung der Abgabemöglichkeit** im Praktomat erfolgt jeweils eine Woche nach Ausgabe eines Übungsblattes bzw. eine Woche vor Ablauf der Abgabefrist. Eine frühere Abgabe ist nicht möglich.
- Der Praktomat erzwingt eine Mindestqualität der Abgabe, indem Abgaben abgelehnt werden, die bestimmte **Qualitätskriterien** verletzen. Planen Sie daher bei jeder Abgabe genügend Zeit ein, mindestens aber einen Tag Puffer. Die Qualitätskriterien werden jeweils im Abschnitt Bearbeitungshinweise beschrieben. Mit fortschreitendem Semester werden die Qualitätskriterien zunehmend strikt.
- Der Praktomat ist nur innerhalb des KIT-Netzes erreichbar. Benutzen Sie, wenn Sie Zuhause arbeiten, den VPN-Client des KIT[4].
- Um für die Abgabe der Übungsblätter freigeschaltet zu werden, müssen Sie rechtzeitig die Einverständniserklärung abgeben.
- Für die Teilnahme an dem Übungsbetrieb und/oder Abschlussaufgaben muss für dieses Semester erneut eine Einverständniserklärung abgegeben werden.

Kommunikation und aktuelle Informationen

- Fragen zu Vorlesungsinhalten und Übungsblättern stellen Sie bitte ausschließlich in den ILIAS-Foren. So profitieren auch Ihre Kommilitone davon. E-Mails mit inhaltlichen Fragen werden aus genau diesem Grund nicht beantwortet, auch nicht von Tutoren.
- In den ILIAS-Foren veröffentlichen wir gelegentlich wichtige Neuigkeiten. Eventuelle Korrekturen von Aufgabenstellungen werden auf diesem Weg bekannt gemacht. Das Beobachten der Neuigkeiten wird daher vorausgesetzt.
- Überprüfen Sie das Postfach Ihrer **KIT-Mailadresse** regelmäßig auf neue E-Mails. Sie erhalten eine Zusammenfassung der Korrektur per E-Mail an diese Adresse. Alle Anmerkungen des Tutors können Sie anschließend im Praktomaten online einsehen.

[3] `https://praktomat.cs.kit.edu/`
[4] `http://www.scc.kit.edu/dienste/vpn.php`

Plagiat

Es werden nur selbstständig angefertigt Abgaben akzeptiert. Das Einreichen fremder Lösungen, seien es auch nur teilweise Lösungen von Dritten, aus Büchern, dem Internet oder anderen Quellen, ist ein Täuschungsversuch und führt zur Bewertung „nicht bestanden“. Ausdrücklich ausgenommen hiervon sind Quelltextschnipsel von den Vorlesungsfolien und aus den Lösungsvorschlägen des Übungsbetriebes in diesem Semester. Alle benutzten Hilfsmittel müssen vollständig und genau angegeben werden und alles, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, muss deutlich kenntlich gemacht werden. Auch die Beihilfe, wie die Weitergabe der eigenen Lösung oder von Teilen davon, wird als Täuschungsversuch gewertet. Für weitere Ausführungen sei auf die Einverständniserklärung (Disclaimer) verwiesen.

Termine und Anmeldefristen

- **Praktomat**: Eine Anmeldung zu Tutorien ist im Sommersemester **nicht vorgesehen**. Es genügt, sich im Praktomat anzumelden, um einen Tutor zugewiesen zu bekommen. Die erstmalige Anmeldung im Praktomat hat bis **29.04.2020, 12:00 Uhr** zu erfolgen.
- **Übungsschein**: Eine Anmeldung ist zwischen **22.04.2020 – 21.05.2020, jeweils 12.00 Uhr** über das Campus Management Portal möglich.
- **Abschlussaufgaben**: Eine Anmeldung ist zwischen **20.07.2020 – 27.07.2020, jeweils 12.00 Uhr** über das Campus Management Portal möglich.
- **Disclaimer[5]:** Abgabe erfolgt E-Mail bis zum **29.04.2020, 12:00 Uhr**.
  - Digital Unterschreiben oder ausdrucken, anlog unterschreiben und einscannen oder abfotografieren.
  - Abgabe per E-Mail an `programmieren-vorlesung@ipd.kit.edu` mit Betreff: **Disclaimer SoSe2020**.

Abgabehinweise

Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe dieses Blattes notwendig ist. Der Praktomat wird Ihre Abgabe zurückweisen, falls eine der nachfolgenden Regeln verletzt ist. Eine zurückgewiesene Abgabe wird automatisch mit null Punkten bewertet. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

- Achten Sie auf fehlerfrei kompilierenden Programmcode.
- Verwenden Sie keine Elemente der Java-Bibliotheken, ausgenommen Elemente der Pakete `java.lang`.

[5] `https://sdqweb.ipd.kit.edu/disclaimer/`

Bearbeitungshinweise

Diese Bearbeitungshinweise sind relevant für die Bewertung Ihrer Abgabe, jedoch wird der Praktomat Ihre Abgabe **nicht** zurückweisen, falls eine der nachfolgenden Regeln verletzt ist.

- Beachten Sie, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung als auch Funktionalität bewertet werden. Halten Sie die Hinweise zur Modellierung im ILIAS-Wiki ein.
- Programmcode muss in englischer Sprache verfasst sein.
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich.
- Die Kommentare sollen einheitlich in englischer oder deutscher Sprache verfasst werden.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Montag, den 5. Mai 2020 um 13:00 Uhr**, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Praktomat bei der richtigen Aufgabe vor Ablauf der Abgabefrist hochzuladen.

- Geben Sie Ihre Klassen zu Aufgabe A als `*.java`-Dateien ab.
- Geben Sie Ihre Klassen zu Aufgabe B als `*.java`-Dateien ab.
- Geben Sie Ihre Klassen zu Aufgabe C als `*.java`-Dateien ab.

Aufgabe A: String-Utility (6 Punkte)

Implementieren Sie die folgenden sechs öffentlichen Methoden in einer Utility-Klasse `StringUtility`. Diese Klasse muss in dem Packet `edu.kit.informatik` liegen und darf keine `main`-Methode enthalten.

Hierbei sind alle diese Methoden statische und die Klasse beinhaltet keine Attribute. Sie könne davon ausgehen, dass die als Methodenparameter übergebenen Zeichenketten immer mindestens ein gültiges Zeichen enthalten. Beachten Sie bei diesen Zeichenketten sowohl die Groß- und Kleinschreibung, als auch Sonder- und Zahlzeichen. Verwenden Sie selber für diese Aufgabe **keine** weiteren Klassen direkt aus der Java-API, mit Ausnahme von `String`.

1. Schreiben Sie eine Methode `String reverse(String word)`, welche die ihr gegebenen Zeichenkette Umgekehrt. Diese Methode gibt die Zeichen in umgekehrter Reihenfolge wieder als eine neue Zeichenkette zurück.
2. Schreiben Sie eine Methode `boolean checkPalindrome(String word)`, welche überprüft, ob die ihr gegebenen Zeichenkette ein Palindrom ist. Ein Palindrom ist ein Wort, das vorwärts und rückwärts gelesen identisch ist.
3. Schreiben Sie eine Methode `String removeCharacter(String word, int index)`, welche aus der ihr gegebenen Zeichenkette ein einzelnes Zeichen an dem ihr gegeben Index entfernt. Diese Methode gibt eine neue Zeichenkette ohne das Zeichen zurück.
4. Schreiben Sie eine Methode `boolean checkAnagram(String word1, String word2)`, welche überprüft, ob die ihr beiden gegebenen Zeichenketten ein Anagramm voneinander sind. Als Anagramm wird eine Zeichenketten bezeichnet, die aus einer anderen Zeichenketten allein durch Umstellung der Zeichen gebildet ist.
5. Schreiben Sie eine Methode `String lowercase(String word)`, welche den ersten Buchstaben der ihr gegebenen Zeichenkette klein schreibt, falls dies ein Buchstabe an der ersten Stelle hat. Diese Methode gibt eine neue Zeichenkette mit einem Kleinbuchstabe an erster Stelle zurück.
6. Schreiben Sie eine Methode `int countCharacter(String word, char character)`, welche für ihr gegebenen Zeichenkette die Anzahl des Auftretens des ihr gegeben einzelnen Zeichens bestimmt. Diese Methode gibt die Anzahl des Zeichens in der Zeichenkette zurück.

Aufgabe B: Modellierung einer Ampelanlage (6 Punkte)

In dieser Aufgabe sollen Sie eine sehr stark vereinfachte Ampelanlage modellieren.

- Eine *Ampel* hat immer genau zwei *Licht*er, ein rotes und ein grünes.
- Ein Licht ist entweder an oder aus.
- Der initiale Zustand einer Ampel ist: rotes Licht an, grünes Licht aus.
- Jede *Kreuzung* hat eine Ampelanlage, bestehend aus 4 Ampeln, die über Zahlen 0 bis 3 identifiziert werden.

Fügen Sie der Klasse für das Licht einen Konstruktor mit einem Parameter hinzu, der den initialen Zustand (an oder aus) auf den übergebenen Wert setzt. Fügen Sie weiter eine Methode hinzu, die „den Lichtschalter drückt“, d.h. das Licht ausschaltet, falls es an ist, bzw. es anschaltet, falls es aus ist (verwenden Sie hierzu **keine** Kontrollstrukturen). Stellen Sie ebenso eine Methode zur Verfügung, die zurückgibt, ob das Licht momentan an ist.

Fügen Sie der Klasse für die Ampel einen Konstruktor hinzu, der den initialen Zustand der Ampel wie vorgegeben festlegt. Implementieren Sie eine Methode, um die Ampel von rot auf grün bzw. umgekehrt umzuschalten. Des Weiteren fügen Sie der Klasse eine Methode hinzu, die überprüft, ob die Ampel im Moment grün zeigt.

Die Klasse für die Kreuzung erhält einen Konstruktor, der die Ampelanlage erstellt und initialisiert. Implementieren Sie diesen Konstruktor so, dass er den oben genannten Anforderungen genügt. Zusätzlich sollen in diesem Konstruktor für den initialen Zustand der Kreuzung die ersten beiden Ampeln von Rot auf Grün umgeschalteten werden.

Fügen Sie der Klasse eine Methode hinzu, welche die Ampeln umschaltet. D.h. die aktuell grünen Ampeln sollen rot werden, dafür sollen die roten Ampeln grün werden. Achten Sie darauf, dass bei mehrmaligem Aufruf dieser Methode auch alle Ampeln einmal grün werden. Auch hier kommen Sie **ohne** Kontrollstrukturen aus. Als letztes entwerfen Sie eine Methode, welche prüft, ob es sicher (grüne Ampel) ist, die Ampel (über ihre Nummer identifiziert) zu überqueren.

Greifen Sie innerhalb einer Klasse ausschließlich auf deren **eigene** Attribute zu, jedoch niemals auf Attribute von Objekten anderer Klassen. Nutzen Sie stattdessen die Methoden, um den Zustand „fremder“ Objekte zu ändern.

Aufgabe C: Haareschneiden (8 Punkte)

Sie stehen in einer langen Schlange, um einen Haarschnitt in einem Friseursalon zu bekommen. Der Salon hat B Friseure im Dienst, welche von 1 bis B durchnummeriert sind. Der k-te Friseur braucht immer genau M_k Minuten, um einem Kunden die Haare zu schneiden. Ein Friseur kann gleichzeitig nur einem Kunden die Haare schneiden, sobald ein Friseur die Haare des Kunden fertig geschnitten hat, schneidet er sofort die Haare des nächsten Kunden.

Solange der Friseursalon geöffnet ist, geht der Kunde, welcher an der Spitze der Schlange steht, immer zum Friseur mit der niedrigsten Nummer, der verfügbar ist. Wenn kein Friseur verfügbar ist, wartet dieser Kunde, bis mindestens einer verfügbar ist.

Sie sind die N-te Person in der Warteschlange, und der Friseursalon hat gerade geöffnet. Welcher Friseur wird Ihnen die Haare schneiden?

Um diese Frage beantworten zu könne, nimmt Ihr Programm mehre natürliche Zahlen als Kommandozeilenparameter entgegen und gibt das Ergebnis wieder auf die Kommandozeile aus. Sie können davon ausgehen, dass die eingegebenen Parameter immer dem hier vorgegebenen Format entsprechen. Setzen Sie nur die in der Aufgabenstellung angegebenen Informationen um.

Die erste Zahl ist die Anzahl der Friseure B ($1 \leq B \leq 10^3$) und die zweite Zahl ist Ihr Platz in der Warteschlange N ($1 \leq N \leq 10^9$). Der Kunde am Anfang der Schlange hat den Platz mit der Nummer 1, der nächste ist Nummer 2 und so weiter. Die nachfolgenden Zahlen sind $M_1, M_2, \ldots, M_B$ ($1 \leq M_k \leq 10^5$).

Sobald das Ergebnis dieser Eingabe feststeht, wird die Nummer des Friseurs, der Ihnen die Haare schneiden wird, über den standardmäßigen Ausgabe-Stream auf die Kommandozeile ausgegeben. Danach beendet sich Ihr Programm regulär.

Beispielablauf

```
1 > java Haircut 2 4 10 5
2 1
3 > java Haircut 3 12 7 7 7
4 3
5 > java Haircut 3 8 4 2 1
6 1
```