Karlsruher Institut für Technologie

Prof. Dr.-Ing. Anne Koziolek
Institute of Information Security and Dependability (KASTEL)
https://sdq.kastel.kit.edu/wiki/Programmieren
programmieren-vorlesung@cs.kit.edu

Programmieren – Wintersemester 2025/26

Übungsblatt 4
Version 1.0

20 Punkte

Ausgabe:	17.12.2025, ca. 12:00 Uhr
Abgabestart:	07.01.2026, 12:00 Uhr
Abgabefrist:	15.01.2026, 06:00 Uhr

Plagiarismus

Es werden nur selbstständig angefertigte Lösungen 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 jederzeit (auch nachträglich) 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. Alles, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde, muss deutlich kenntlich gemacht werden. Beachten Sie darüber hinaus die Richtlinien der Fakultät zum Verwenden von Generativer KI [1].

Studierende, die den ordnungsgemäßen Ablauf einer Erfolgskontrolle stören, können von der Erbringung der Erfolgskontrolle ausgeschlossen werden. Ebenso stellt unter anderem die Weitergabe von Teilen von Testfällen oder Lösungen bereits eine Störung des ordnungsgemäßen Ablaufs dar. Auch diese Art von Störungen können ausdrücklich jederzeit zum Ausschluss der Erfolgskontrolle führen. Dies bedeutet ausdrücklich, dass auch nachträglich die Punktzahl reduziert werden kann.

Kommunikation und aktuelle Informationen

In unseren *FAQs*[2] finden Sie einen Überblick über häufig gestellte Fragen und die entsprechenden Antworten zum Modul „Programmieren“. Bitte lesen Sie diese sorgfältig durch, noch bevor Sie Fragen stellen, und überprüfen Sie diese regelmäßig und eigenverantwortlich auf Änderungen. Beachten Sie zudem die Hinweise im Wiki[3].

In den *ILIAS-Foren* oder auf *Artemis* veröffentlichen wir gelegentlich wichtige Neuigkeiten. Eventuelle Korrekturen von Aufgabenstellungen werden ebenso auf diesem Weg bekannt gemacht. Das aktive Beobachten der Foren wird daher vorausgesetzt.

[1] https://www.informatik.kit.edu/faq-wiki/doku.php?id=generative_ki
[2] https://sdq.kastel.kit.edu/wiki/Programmieren/FAQ
[3] https://sdq.kastel.kit.edu/programmieren/

Überprüfen Sie das Postfach Ihrer *KIT-Mailadresse* regelmäßig auf neue E-Mails. Sie erhalten unter anderem eine Zusammenfassung der Korrektur per E-Mail an diese Adresse. Alle Anmerkungen können Sie anschließend im Online-Einreichungssystem[4] einsehen.

Bearbeitungshinweise

Bitte beachten Sie, dass das erfolgreiche Bestehen der verpflichtenden Tests für eine erfolgreiche Abgabe von Übungsblatt 4 notwendig ist. Ihre Abgabe wird automatisch mit null Punkten bewertet, falls eine der nachfolgenden Regeln verletzt ist. Sie müssen zuerst die verpflichtenden Tests bestehen, bevor die anderen Tests ausgewertet werden können. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

- Achten Sie auf fehlerfrei kompilierenden Programmcode.
- Verwenden Sie ausschließlich *Java SE 21*.
- Sofern in einer Aufgabe nicht ausdrücklich anders angegeben, verwenden Sie keine Elemente der Java-Bibliotheken. Ausgenommen ist die Klasse `java.util.Scanner` und alle Elemente aus den folgenden Paketen: `java.lang`, `java.io`, `java.util`, und `java.util.regex`.
- Achten Sie darauf, nicht zu lange Zeilen, Methoden und Dateien zu erstellen. Sie müssen bei Ihren Lösungen eine maximale Zeilenbreite von 140 Zeichen einhalten.
- Halten Sie alle Whitespace-Regeln ein.
- Halten Sie alle Regeln zu Variablen-, Methoden- und Paketbenennung ein.
- Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute.
- Nutzen Sie nicht das `default`-Package.
- `System.exit()`, `Runtime.exit()` oder ähnliches dürfen nicht verwendet werden.
- Halten Sie die Regeln zur Javadoc-Dokumentation ein.
- Halten Sie auch alle anderen Checkstyle-Regeln ein.

Diese folgenden Bearbeitungshinweise sind relevant für die Bewertung Ihrer Abgabe. Dennoch wird Ihre Abgabe durch das Abgabesystem *nicht* automatisch mit null Punkten bewertet, falls eine der nachfolgenden Regeln verletzt ist. Orientieren Sie sich zudem an den Bewertungskriterien im Wiki.

- Fügen Sie außer Ihrem u-Kürzel keine weiteren persönlichen Daten zu Ihren Abgaben hinzu.
- Beachten Sie, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung als auch Funktionalität bewertet werden. Halten Sie die Hinweise zur Modellierung im 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.

[4] `https://artemis.cs.kit.edu/`

- Geben Sie im Javadoc-Autoren-Tag nur Ihr u-Kürzel an.
- Wählen Sie aussagekräftige Namen für alle Ihre Bezeichner.

Checkstyle

Das Online-Einreichungssystem überprüft Ihre Quelltexte während der Abgabe automatisiert auf die Einhaltung der Checkstyle-Regeln. Es gibt speziell markierte Regeln, bei denen das Online-Einreichungssystem die Abgabe mit null Punkten bewertet, da diese Regeln verpflichtend einzuhalten sind. Andere Regelverletzungen können zu Punktabzug führen. Sie können und sollten Ihre Quelltexte bereits während der Entwicklung auf die Regeleinhaltung überprüfen. Das Programmieren-Wiki beschreibt, wie Checkstyle verwendet werden kann.

Abgabehinweise

Die Abgabe im Online-Einreichungssystem wird am 07.01.2026, 12:00 Uhr, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Einreichungssystem bei der richtigen Aufgabe vor Ablauf der Abgabefrist am 15.01.2026, 06:00 Uhr, hochzuladen. Beginnen Sie frühzeitig mit dem Einreichen, um Ihre Lösung dahingehend zu testen, und verwenden Sie das Forum, um eventuelle Unklarheiten zu klären. Falls Sie mit Git abgeben, *muss immer* auf den `main`-Branch gepusht werden.

- Geben Sie online Ihre `*.java`-Dateien zur Aufgabe A in Einzelarbeit mit der entsprechenden Ordnerstruktur im zugehörigen Verzeichnis ab.
- Geben Sie online Ihre `*.java`-Dateien zur Aufgabe B in Einzelarbeit mit der entsprechenden Ordnerstruktur im zugehörigen Verzeichnis ab.

Wiederverwendung von Lösungen

Falls Sie für die Bearbeitung der Abschlussaufgaben oder Übungsblätter Beispiellösungen aus diesem Semester wiederverwenden, *müssen* Sie in die entsprechenden Klassen "Programmieren-Team" ins Autor-Tag eintragen. Dies ist nötig, um die Checkstyle-Kriterien zu erfüllen.

Aufgabe A: Postamt (12 Punkte)

In dieser Aufgabe soll ein System für ein Postamt implementiert werden, bei der sich verschiedene Benutzer registrieren können. Für die Registrierung ist der Personalausweis notwendig. Existierende Benutzer können nach erfolgreicher Anmeldung Postdienstleistungen wie z.B. das Abholen und Versenden von Paketen durchführen. Anmelden kann sich ein **Kunde** mit seinem persönlichen Benutzernamen und Passwort.

Ein **Postmitarbeiter** kann sich mit Hilfe seiner Personalnummer und seinem Passwort am Postamtsystem anmelden. Im angemeldeten Zustand kann er im Auftrag eines Kunden Post versenden und ihm seine hinterlegte Post aushändigen. Ebenso kann ein Postmitarbeiter einem Kunden weitere Auskünfte über die bereits getätigten Postdienstleistungen erteilen (vgl. `list-mail`-Befehl und `list-price`-Befehl).

Ein **Callcentermitarbeiter** kann sich ebenfalls mit Personalnummer und Passwort am Postamtsystem anmelden. Er hat wie ein Postmitarbeiter die Möglichkeit Kunden Auskünfte über ihr Konto am Postamt zu geben und das Passwort eines Kunden zurückzusetzen (vgl. `reset-pin`-Befehl).

Es gibt verschiedenen **Postdienstleistungen**, die ein Kunde versenden und empfangen kann. Die Postdienstleistungen haben unterschiedliche Preise. Briefe kosten 0.70 Geldeinheiten, Einwurf-Einschreiben liegen bei 1.20 Geldeinheiten und Einschreiben schlagen mit 2.00 Geldeinheiten zu Buche. Für Pakete gibt es unterschiedliche Größen mit unterschiedlichen Preisen. Ein PaketS kostet 5.00, das PaketM kostet 6.00 und das PaketL 7.00 Geldeinheiten. Dies bedeutet, dass es sich bei einer Postdienstleistung um ein Einwurf-Einschreiben, ein Einschreiben, einen Brief oder ein Paket (in der entsprechenden Größe) handeln kann.

Berücksichtigen Sie bei dieser Aufgabe Groß-/Kleinschreibung. Dies bedeutet, dass die Groß-/Kleinschreibung der Ausgabe mit der Eingabe übereinstimmt.

Geben Sie alle Preise auf zwei Nachkommastellen abgerundet aus.

Alle in dieser Aufgabe verwendeten Passwörter sollen (trotz Sicherheitsbedenken) im Klartext gespeichert werden.

A.1 Interaktive Benutzerschnittstelle

Nach dem Start nimmt Ihr Programm über die Konsole Befehle entgegen, die im Folgenden näher spezifiziert werden. Nach Abarbeitung eines Befehls wartet Ihr Programm auf weitere Befehle, bis das Programm irgendwann durch die Eingabe der Zeichenfolge `quit` beendet wird.

Gehen Sie davon aus, dass beim Start Ihres Programms weder Nutzer, noch Post im System des Postamtes registriert bzw. hinterlegt sind.

Fehlermeldungen

Achten Sie darauf, dass durch Ausführung der folgenden Befehle die Vorgaben in der Aufgabenstellung nicht verletzt werden und geben Sie in diesen Fällen eine aussagekräftige Fehlermeldung aus. Auch wenn die Benutzereingabe nicht dem vorgegebenen Format entspricht, ist eine Fehlermeldung auszugeben. Nach der Ausgabe einer Fehlermeldung soll das Programm wie erwartet fortfahren und wieder auf die nächste Eingabe warten. Jede Fehlermeldung muss mit `Error ,␣` beginnen und darf keine Zeilenumbrüche enthalten. Den weiteren Text der Fehlermeldung dürfen Sie frei wählen, er sollte jedoch sinnvoll sein.

Beispielinteraktionen

Die Zeilennummern und die Trennlinie sind kein Bestandteil der Benutzerschnittstelle, sie dienen lediglich zur Orientierung für die gegebenen Beispielinteraktionen. Die Eingabezeilen werden mit dem `>` (Größer-als-Zeichen) gefolgt von einem Leerzeichen eingeleitet, diese beiden Zeichen sind ebenfalls kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabezeilen. Beachten Sie, dass die Beispielabläufe der einzelnen Befehle unabhängig voneinander zu betrachten sind.

Platzhalter

Beachten Sie, dass bei der Beschreibung der Eingabe- und Ausgabe die Wörter zwischen spitzen Klammen (`<` und `>`) für Platzhalter stehen, welche bei der konkreten Ein- und Ausgabe durch Werte ersetzt werden. Diese eigentlichen Werte enthalten bei der Ein- und Ausgabe keine spitzen Klammern. Vergleichen Sie hierzu auch die jeweiligen Beispielabläufe.

A.2 Befehle

A.2.1 `add-customer`-Befehl

Der `add-customer`-Befehl fügt dem System einen neuen Kunden hinzu. Diese Operation kann nur durchgeführt werden, wenn kein Nutzer (Kunde, Postmitarbeiter oder Callcentermitarbeiter) mit Hilfe des `authenticate`-Befehls angemeldet ist.

A.2.1.1 Eingabe

`add-customer <Vorname>;<Nachname>;<Benutzername>;<Passwort>;<Perso>`

`<Vorname>` und `<Nachname>` beschreiben den Kunden und sind jeweils ein beliebiger `String` ohne Zeilenumbruch und ohne Semikolon. `<Benutzername>` ist ein `String` ohne Zeilenumbruch und ohne Semikolon mit mindestens 4 und höchstens 9 Stellen zur eindeutigen Identifizierung eines Kundens im System (vgl. eindeutige Identifizierung eines Postmitarbeiters und Callcentermitarbeiters durch `<Personalnummer>` bei `add-mailman`-Befehl und `add-agent`-Befehl). `<Passwort>` ist ein beliebiger `String` ohne Zeilenumbruch und ohne Semikolon mit mindestens 4 und höchstens 9 Stellen. `<Perso>` repräsentiert eine Personalausweisnummer und ist eine

Zeichenkette `String` ohne Zeilenumbruch und ohne Semikolon mit 9 Stellen zur eindeutigen Identifizierung eines Kunden, die Buchstaben und Zahlen enthalten kann. Der Benutzername und die Personalausweisnummer müssen aus Datenschutzgründen unterschiedlich sein.

A.2.1.2 Ausgabe Vorausgesetzt, dass ein neuer Kunde erfolgreich dem System hinzugefügt wurde, wird `OK` ausgegeben. Im Fehlerfall (z.B. wenn ein Kunde mit der angegebenen Personalausweisnummer existiert) wird eine Fehlermeldung beginnend mit `Error ,␣` ausgegeben.

A.2.2 `add-mailman`-Befehl

Der `add-mailman`-Befehl fügt dem System einen neuen Postmitarbeiter hinzu. Diese Operation kann nur durchgeführt werden, wenn kein Nutzer (Kunde, Postmitarbeiter oder Callcentermitarbeiter) mit Hilfe des `authenticate`-Befehls angemeldet ist.

A.2.2.1 Eingabe

`add-mailman <Vorname>;<Nachname>;<Personalnummer>;<Passwort>`

`<Vorname>` und `<Nachname>` beschreiben den Postmitarbeiter und sind jeweils ein beliebiger `String` ohne Zeilenumbruch und ohne Semikolon. `<Personalnummer>` ist eine positive `Integer`-Zahl zur eindeutigen Darstellung eines Postmitarbeiters. `<Passwort>` ist ein beliebiger `String` ohne Zeilenumbruch und ohne Semikolon mit mindestens 4 und höchstens 9 Stellen.

A.2.2.2 Ausgabe Vorausgesetzt, dass ein neuer Postmitarbeiter erfolgreich dem System hinzugefügt wurde, wird `OK` ausgegeben. Im Fehlerfall (z.B. wenn bereits ein Mitarbeiter mit der angegebenen Personalnummer existiert) wird eine Fehlermeldung beginnend mit `Error ,␣` ausgegeben.

A.2.3 `add-agent`-Befehl

Der `add-agent`-Befehl fügt dem System einen neuen Callcentermitarbeiter hinzu. Diese Operation kann nur durchgeführt werden, wenn kein Nutzer (Kunde, Postmitarbeiter oder Callcentermitarbeiter) mit Hilfe des `authenticate`-Befehls angemeldet ist.

A.2.3.1 Eingabe

`add-agent <Vorname>;<Nachname>;<Personalnummer>;<Passwort>`

`<Vorname>` und `<Nachname>` beschreiben den Callcentermitarbeiter und sind jeweils ein beliebiger `String` ohne Zeilenumbruch und ohne Semikolon. `<Personalnummer>` ist eine positive `Integer`-Zahl zur eindeutigen Darstellung eines Callcentermitarbeiter. `<Passwort>` ist ein beliebiger `String` ohne Zeilenumbruch und ohne Semikolon mit mindestens 4 und höchstens 9 Stellen.

A.2.3.2 Ausgabe Vorausgesetzt, dass ein neuer Callcentermitarbeiter erfolgreich dem System hinzugefügt wurde, wird `OK` ausgegeben. Im Fehlerfall (z.B. wenn bereits ein Mitarbeiter mit der angegebenen Personalnummer existiert) wird eine Fehlermeldung beginnend mit `Error ,␣` ausgegeben.

A.2.4 `authenticate`-Befehl

Der `authenticate`-Befehl authentifiziert einen Nutzer (Kunde, Postmitarbeiter oder Callcentermitarbeiter) am Postamtsystem und räumt ihm je nach hinterlegten Rechten bestimmte Möglichkeiten ein Befehle aufzurufen. Es kann immer nur ein Nutzer gleichzeitig im Postamtsystem angemeldet sein. Bevor sich ein neuer Nutzer erfolgreich mit dem `authenticate`-Befehl authentifiziert, muss sich der zuvor angemeldete Nutzer mittels `logout`-Befehl abmelden. Beispiel: Kunde A ist angemeldet. Möchte sich nun Kunde B oder ein Mitarbeiter anmelden, so muss sich Kunde A zuvor mittels `logout`-Befehl abmelden.

A.2.4.1 Eingabe `authenticate <Benutzername>;<Passwort>`

`<Benutzername>` ist ein `String`, der den persönlichen Nutzernamen (Benutzername eines Kunden oder Personalnummer eines Mitarbeiters) repräsentiert. `<Passwort>` ist ein `String`, der das persönliche Passwort des Nutzers repräsentiert.

A.2.4.2 Ausgabe Im Fall einer erfolgreichen Authentifizierung eines registrierten Nutzers, wird `OK` ausgegeben. Im Fehlerfall (z.B. wenn ein Benutzer bereits im System angemeldet ist.) wird eine Fehlermeldung beginnend mit `Error ,␣` ausgegeben. Es sollen auch der ungültige Benutzername und das ungültige Passwort in der Fehlermeldung beginnend mit `Error ,␣` vorhanden sein.

A.2.5 `logout`-Befehl

Der `logout`-Befehl meldet einen zuvor mit dem `authenticate`-Befehl authentifizieren Nutzer ab und entzieht ihm bis zu seiner erneuten Authentifizierung die Möglichkeiten entsprechende Befehle aufzurufen.

A.2.5.1 Eingabe `logout`

A.2.5.2 Ausgabe Falls ein zuvor erfolgreich authentifizierter Nutzer abgemeldet wurde, wird `OK` ausgegeben. Im Fehlerfall (z.B. wenn der abzumeldende Nutzer nicht vorher authentifiziert wurde) wird eine Fehlermeldung beginnend mit `Error ,␣` ausgegeben.

A.2.6 `send-mail`-Befehl

Der `send-mail`-Befehl ermöglicht einem Kunden oder einem Postmitarbeiter, der im Auftrag eines Kunden handelt, das Versenden einer Postdienstleistung von einem Sender an einen Empfänger. Diese Operation kann nur durchgeführt werden, wenn ein Kunde (Sender) oder ein Postmitarbeiter sich vorher mit Hilfe des `authenticate`-Befehls angemeldet und noch nicht abgemeldet hat.

Versendet ein angemeldeter Kunde an einen weitere registrierten Kunden Post, so lautet das Eingabeformat wie folgt:

A.2.6.1 Eingabe (Kunde) `send-mail <Postdienstleistung>;<Empfänger>`

`<Postdienstleistung>` ist ein String aus der Menge {`Brief`, `Einwurf-Einschreiben`, `Einschreiben`, `PaketS`, `PaketM`, `PaketL`} ohne Zeilenumbruch und ohne Semikolon.
`<Empfänger>` ist ein registrierter Kunde (vgl. `add-customer`-Befehl) im System, der in diesem Befehl durch seinen eindeutigen Benutzernamen `<Benutzername>` repräsentiert wird.

Versendet ein angemeldeter Postmitarbeiter im Auftrag eines Kunden Post, so wird das Eingabeformat um den zugehörigen `<Sender>` erweitert:

A.2.6.2 Eingabe (Postmitarbeiter)
`send-mail <Postdienstleistung>;<Empfänger>;<Sender>`

`<Sender>` ist ebenso ein registrierter Kunde (vgl. `add-customer`-Befehl) im System, der in diesem Befehl durch seinen eindeutigen Benutzernamen `<Benutzername>` repräsentiert wird.

A.2.6.3 Ausgabe (Kunde und Postmitarbeiter)
Im Erfolgsfall wird `OK` ausgegeben. Im Fehlerfall wird eine Fehlermeldung beginnend mit `Error,␣` ausgegeben.

A.2.7 `get-mail`-Befehl

Der `get-mail`-Befehl ermöglicht einem Kunden und einem Postmitarbeiter, der im Auftrag eines Kunden handelt, Post, die für den entsprechenden Kunden `<Empfänger>` im System hinterlegt ist, aus einer Station im Postamt zu entnehmen. Diese Operation kann nur durchgeführt werden, wenn sich ein Kunde oder Postmitarbeiter vorher mit Hilfe des `authenticate`-Befehls angemeldet und noch nicht abgemeldet hat.

A.2.7.1 Eingabe (Kunde) `get-mail`

A.2.7.2 Eingabe (Postmitarbeiter) `get-mail <Empfänger>`

A.2.7.3 Ausgabe (Kunde und Postmitarbeiter)

Vorausgesetzt, dass die Post, die für den Kunden hinterlegt ist, aus der Station des Postamtes erfolgreich durch den Kunden oder Postmitarbeiter entnommen wurde, wird `OK` ausgegeben. Falls keine Post für den registrierten Kunden hinterlegt war oder bei anderen Fehlern, wird eine Fehlermeldung beginnend mit `Error ,␣` ausgegeben.

A.2.8 `list-mail`-Befehl

Der `list-mail`-Befehl gibt eine sortierte Liste der Post aus, die für einen Kunden aktuell im Postamt hinterlegt und im System dokumentiert ist. Dieser Befehl ist sowohl von einem Kunden als auch von Postmitarbeitern und Callcentermitarbeitern, die beide im Auftrag des Kunden handeln, durchführbar. Das Eingabeformat lautet für einen Kunden wie folgt:

A.2.8.1 Eingabe (Kunde) `list-mail`

Bei einem registrierten und angemeldeten Post- und Callcentermitarbeiter lautet das Eingabeformat wie folgt:

A.2.8.2 Eingabe (Post- und Callcentermitarbeiter) `list-mail <Benutzername>`

`<Benutzername>` ist der eindeutige Benutzername eines bereits durch `add-customer` registrierten Kunden.

A.2.8.3 Ausgabe (Kunde, Post- und Callcentermitarbeiter) `<Postdienstleistung>;<Anzahl>`

Es wird eine Liste alphabetisch sortiert nach der `<Postdienstleistung>` ausgegeben.
`<Postdienstleistung>` ist ein String aus der Menge {`Brief`, `Einwurf-Einschreiben`, `Einschreiben`, `PaketS`, `PaketM`, `PaketL`} ohne Zeilenumbruch und ohne Semikolon.
`<Anzahl>` beschreibt die Anzahl der vorhandenen Postdienstleistung. Postdienstleistungen, bei denen die Anzahl `0` ist, werden nicht ausgegeben.
Die Ausgabe erfolgt je Postdienstleistung zeilenweise.

Wenn keine Post, für einen registrierten Kunden hinterlegt ist, wird nur `OK` ausgegeben. Im Fehlerfall (z.B. Kunde existiert nicht) wird eine Fehlermeldung beginnend mit `Error ,␣` ausgegeben.

A.2.9 `list-price`-Befehl

Der `list-price`-Befehl gibt den kumulierten Preis aus, den ein Kunde für die Nutzung eines Dienstleistungstypen bereits gezahlt hat. Diese Operation kann durchgeführt werden, wenn sich ein Kunde, der im System einen Benutzernamen `<Benutzername>` besitzt, vorher mit Hilfe des `authenticate`-Befehls angemeldet und noch nicht wieder abgemeldet hat. Für einen Kunden ist dieser Befehl parameterlos.

A.2.9.1 Eingabe (Kunde) `list-price`

Ebenso kann ein angemeldeter Post- und Callcentermitarbeiter im Auftrag eines Kunden diese Abfrage durchführen. Dazu ist die Angabe des Kunden mit `<Benutzername>` notwendig:

A.2.9.2 Eingabe (Post- und Callcentermitarbeiter) `list-price <Benutzername>`

A.2.9.3 Ausgabe (Kunde, Post- und Callcentermitarbeiter)

`<Postdienstleistung>;<Anzahl>;<Preis>`

Wenn ein Kunde bereits Postdienstleistungen über das Postamt versandt hat, wird eine Liste seiner versandten Post – alphabetisch sortiert nach der `<Postdienstleistung>` – ausgegeben.
`<Postdienstleistung>` ist ein String aus der Menge {`Brief`, `Einwurf-Einschreiben`, `Einschreiben`, `PaketS`, `PaketM`, `PaketL`} ohne Zeilenumbruch und ohne Semikolon.
`<Anzahl>` ist eine positive `Integer`-Zahl beginnend mit `1`, die die Anzahl der versandten Post je Dienstleistungstyp angibt.
`<Preis>` ist der kummulierte Preis, den ein Kunde für die Inanspruchnahme eines Postdienstleistungstyps bereits gezahlt hat.
Die Ausgabe erfolgt je Postdienstleistung zeilenweise.

Wenn der Kunde noch keine Post (für einen gegebenen Dienstleistungstypen) versendet hat, wird nur `OK` ausgegeben. Im Fehlerfall wird eine Fehlermeldung beginnend mit `Error,␣` ausgegeben.

A.2.10 `reset-pin`-Befehl

Mit Hilfe des `reset-pin`-Befehls kann ein Callcentermitarbeiter das Passwort für einen Kunden durch ein neues Passwort ersetzen. Diese Operation kann nur durchgeführt werden, wenn sich ein Callcentermitarbeiter vorher mit Hilfe des `authenticate`-Befehls angemeldet und noch nicht wieder abgemeldet hat.

A.2.10.1 Eingabe `reset-pin <Benutzername>;<Perso>;<Passwort>`

`<Benutzername>` ist der Benutzername eines bereits durch `add-customer` registrierten Kunden. `<Perso>` ist die Personalausweisnummer eines bereits durch `add-customer` registrierten Kunden. `<Passwort>` ist ein beliebiger `String` ohne Zeilenumbruch und ohne Semikolon mit mindestens 4 und höchstens 9 Stellen.

A.2.10.2 Ausgabe Vorausgesetzt, dass Benutzername und Personalausweisnummer zueinander passen und ein angemeldeter Callcentermitarbeiter das Passwort für einen registrierten Kunden ersetzt hat, wird `OK` ausgegeben.

Beispielinteraktion

```
1 > reset-pin Erika;234567890;123
2 Error, incorrect input format, password is too short.
3 > reset-pin Erika;234567890;pass2
4 OK
```

Der `quit`-Befehl

Der parameterlose `quit`-Befehl ermöglicht es, das Programm jederzeit vollständig zu beenden. Beachten Sie, dass hierfür keine Methoden wie `System.exit()` oder `Runtime.exit()` verwendet werden dürfen.

Eingabe `quit`

Ausgabe Im Erfolgsfall findet keine Ausgabe statt.

Beispielinteraktion

```
1  > quit quit
2  Error, incorrect input format, this command does not accept any parameters.
3  > quit
```

A.3 Beispielinteraktion

Beachten Sie im Folgenden, dass Eingabezeilen mit dem >-Zeichen eingeleitet werden, gefolgt von einem Leerzeichen. Diese beiden Zeichen sind ausdrücklich kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabe.

Beispielinteraktion

```
1  > add-customer Max;Mustermann;MaxM;pass1;123456789
2  OK
3  > add-customer Erika;Mustermann;Erika;pass2;234567890
4  OK
5  > add-mailman Heinrich;Hinz;1;pass1
6  OK
7  > add-agent Konrad;Kunz;2;pass2
8  OK
9  > authenticate Erika;pass2
10 OK
11 > send-mail Brief;MaxM
12 OK
13 > list-price
14 Brief;1;0.70
15 > logout
16 OK
17 > quit
```

Aufgabe B: Mau-Mau (8 Punkte)

In dieser Aufgabe sollen Sie eine Variante des beliebten Kartenspiels Mau-Mau für genau vier Spieler auf der Kommandozeile implementieren. Die Spieler haben jeweils das Ziel, ihre eigenen Karten so schnell wie möglich abzulegen. Gespielt wird dabei mit dem deutschem Blatt mit insgesamt 32 Karten.

B.1 Deutsches Blatt

Ein vollständiges deutsches Blatt besteht aus 32 verschiedenen Karten. Jede Karte wird durch zwei Parameter bestimmt: Einen Kartenwert und eine Farbe. Das deutsche Blatt kennt die Farben Eichel, Laub, Herz und Schellen. Die Kartenwerte sind sowohl Zahlen (7, 8, 9 und 10) als auch Figuren (Bube, Dame, König und Ass).

Zur Identifizierung wird jeder Karte eine eindeutige Kennung, d.h eine Kombination aus dem Zahlenwert oder dem Anfangsbuchstaben des Zeichens und dem Anfangsbuchstaben der Farbe zugeordnet: `7E`, `8E`, `9E`, `10E`, `BE`, `DE`, `KE`, `AE`, `7L`, `8L`, `9L`, `10L`, `BL`, `DL`, `KL`, `AL`, `7H`, `8H`, `9H`, `10H`, `BH`, `DH`, `KH`, `AH`, `7S`, `8S`, `9S`, `10S`, `BS`, `DS`, `KS` und `AS`.

B.2 Mischen

Das Mischen ist ein Verfahren, mit dem ein Blatt nach dem Zufallsprinzip gemischt wird, um ein Zufallselement im Kartenspiel zu schaffen. Dazu müssen die 32 Karten zunächst in genau der zuvor angegebenen Reihenfolge (siehe Abschnitt B.1) in einer `List`[5] (z.B. einer `ArrayList`[6]) gespeichert werden, sodass `7E` den Index 0 hat, `8E` den Index 1 und so weiter. Um nun diese Liste nach dem Zufallsprinzip unter Verwendung der angegebenen Zufallsquelle zu mischen, muss diese der `Collections::shuffle`-Methode[7] übergeben werden.

Zusätzlich muss der Methode noch eine Instanz der `Random`-Klasse[8] als Zufallsquelle übergeben werden. Diese Instanz wird verwendet, um einen Strom von Pseudo-Zufallszahlen zu erzeugen. Um das Spiel so einfach wie möglich testen zu können, muss diese übergebene Instanz selbst mit einem Startwert initialisiert werden. Verwendet man den gleichen Startwert in deterministischen Zufallsquellen, erhält man die gleiche Folge von Pseudo-Zufallszahlen: `Collections.shuffle(list, new Random(seed))`

[5] https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/List.html

[6] https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/ArrayList.html

[7] https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Collections.html#shuffle(java.util.List,java.util.Random)

[8] https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Random.html

B.3 Spielregeln

Zu Beginn erhält jeder Spieler fünf Karten auf die Hand. Dazu erhält der erste Spieler die ersten fünf Karten des zuvor gemischten Aufnahmestapels (siehe Abschnitt B.2 , beginnend mit der Karte an Index 0), der zweite Spieler die nächsten fünf Karten des Aufnahmestapels und so weiter. Die oberste Karte dieses Aufnahmestapels wird danach offen daneben gelegt und bildet den Ablagestapel. Die restlichen Karten bleiben verdeckt im Aufnahmestapel.

Der erste Spieler ist immer der Startspieler und beginnt das Spiel. Er versucht, eine seiner Karten offen abzulegen, wobei diese denselben Kartenwert oder dieselbe Farbe wie die oberste Karte des Ablagestapels haben muss. Beispielsweiße darf die `7H` also entweder auf eine andere Herz-Karte oder eine andere 7 gelegt werden.

Die vier Spieler spielen beginnend mit dem ersten Spieler abwechselnd nach der Reihe, sodass nach dem vierten Spieler wieder der erste Spieler an der Reihe ist. Wenn dies möglich ist, legt dabei jeder Spieler reihum eine seiner Karten offen auf den Ablagestapel. Kann ein Spieler keine Karte ablegen, so muss er eine Karte vom Aufnahmestapel ziehen und warten, bis er erneut an der Reihe ist.

Es gewinnt der Spieler, der zuerst alle seine Karten ablegen konnte. Sobald ein Spieler die letzte verbleibende Karte vom Aufnahmestapel zieht, endet das Spiel unentschieden. In beiden Fällen endet das Spiel automatisch auch für die anderen Spieler.

B.4 Interaktive Benutzerschnittstelle

Nach dem Start nimmt Ihr Spiel über die Konsole Eingaben entgegen, die im Folgenden näher spezifiziert werden. Nach Abarbeitung einer Eingabe wartet Ihr Spiel auf weitere Eingaben, bis das Spiel irgendwann durch die Eingabe der Zeichenfolge `quit` beendet wird. Wenn ein Spiel regulär endet, muss entweder ein neues Spiel gestartet (`start`) oder das Java-Programm vollständig beendet werden (`quit`). Ein neues Spiel kann immer gestartet werden, wobei dadurch das aktuelle Spiel, ohne zusätzliche Ausgaben, verworfen werden soll.

Die Eingabe und die Ausführung der Befehle dürfen nicht gegen zuvor definierte Spielregeln (siehe Abschnitt B.3) verstoßen. Bei einem Verstoß muss immer eine aussagekräftige Fehlermeldung ausgegeben werden, anschließend wartet das Spiel auf weitere Eingaben. Wenn für die Ausführung eines Befehls keine Ausgabe spezifiziert ist, erfolgt bei erfolgreicher Ausführung auch keine Ausgabe.

B.4.1 Platzhalter

Beachten Sie, dass bei der Beschreibung der Eingabe- und Ausgabeformate die Wörter zwischen spitzen Klammen (< und >) für Platzhalter stehen, welche bei der konkreten Ein- und Ausgabe durch Werte ersetzt werden. Die eigentlichen Werte enthalten bei der Ein- und Ausgabe keine spitzen Klammern.

<Seed> Startwert für das Mischen als `int`-Zahl.

<Ablagestapel> Kennung der obersten Karte des Stapels, siehe Abschnitt B.1.

<Aufnahmestapel> Ganzzahlige Anzahl von Karten auf dem Aufnahmestapel.

<Nummer> Nummer eines Spielers (1, 2, 3 oder 4).

`<Hand>` Kommagetrennte, nach Kennung lexikografisch sortierte Auflistung der Karten eines Spielers.

`<Karte>` Kennung einer Karte, siehe Abschnitt B.1.

B.4.2 Spiel starten

Der Befehl startet ein neues Mau-Mau-Spiel. Dazu wird das vollständige Blatt mit dem vorgegebenen Startwert `<seed>` vorher gemischt und jeweils fünf Karten an die vier Spieler verteilt. Danach wird der Spieler ausgegeben, welcher am Zug ist.

Eingabe `start␣<Seed>`

Ausgabe `Player␣1␣takes␣the␣turn.`

B.4.3 Stapel anzeigen

Der Befehl zeigt die Kennung der obersten Karte des Ablagestapels `<Ablagestapel>` und die Anzahl der Karten auf dem Aufnahmestapel `<Aufnahmestapel>` an.

Eingabe `show␣game`

Ausgabe `<Ablagestapel>␣/␣<Aufnahmestapel>`

B.4.4 Spieler anzeigen

Der Befehl zeigt die Hand `<Hand>` des Spielers mit der angegebenen Nummer `<Nummer>` an. Die einzelnen Karten der Hand werden lexikografisch nach ihrer Kennung sortiert und bei der Ausgabe durch ein Komma voneinander getrennt.

Eingabe `show␣<Nummer>`

Ausgabe `<Hand>`

B.4.5 Karte ablegen

Dieser Befehl legt die angegebene Karte `<Karte>` des angegebenen Spielers `<Nummer>` auf den Ablagestapel. Wenn ein Spieler alle seine Karten auf diese Weise ausspielen konnte, endet das Spiel und es wird eine Nachricht ausgegeben, andernfalls wird nichts ausgegeben und der nächste reguläre Spieler ist an der Reihe.

Eingabe `discard␣<Nummer>␣<Karte>`

Ausgabe `Game␣over:␣Player␣<Nummer>␣has␣won.`

B.4.6 Karte aufnehmen

Mit diesem Befehl nimmt der angegebene Spieler `<Nummer>` die oberste Karte vom Aufnahmestapel auf seine Hand. Wenn ein Spieler die letzte verbleibende Karte von diesem Aufnahmestapel zieht, endet das Spiel und es wird eine Nachricht ausgegeben, ansonsten wird nichts ausgegeben und das Spiel geht regulär weiter.

Eingabe `pick␣<Nummer>`

Ausgabe `Game␣over:␣Draw.`

B.4.7 Spiel beenden

Mit diesem Befehl wird das Java-Programm vollständig beendet. Hierfür dürfen keine Methoden wie `System.exit()`, `Runtime.exit()` oder vergleichbar verwendet werden.

Eingabe `quit`

B.5 Beispielinteraktion

Die Zeilennummern und die Trennlinie sind kein Bestandteil der Benutzerschnittstelle, sie dienen lediglich zur Orientierung für die gegebene Beispielinteraktion. Die Eingabezeilen werden mit einer rechten spitzen Klammer (>) gefolgt von einem Leerzeichen eingeleitet, diese beiden Zeichen sind ebenfalls kein Bestandteil des eingegebenen Befehls, sondern dienen der Unterscheidung zwischen Ein- und Ausgabezeilen.

Beispielinteraktion

```
1  > start 07101825
2  Player 1 takes the turn.
3  > show game
4  8L / 11
5  > discard 1 10L
6  > show 1
7  7L,AH,BE,BS
8  > discard 2 DL
9  > discard 3 8E
10 Error, 8E cannot be stacked on DL.
11 > pick 3
12 > show game
13 DL / 10
14 > quit
```