Karlsruher Institut für Technologie

PD Dr. rer. nat. Robert Heinrich
Institute of Information Security and Dependability (KASTEL)
https://sdq.kastel.kit.edu/wiki/Programmieren/
programmieren-vorlesung@cs.kit.edu

Programmieren – Wintersemester 2024

Übungsblatt 2
Version 1.0

20 Punkte

Ausgabe:	22.11.2023, ca. 12:00 Uhr
Abgabe:	29.11.2023, 12:00 Uhr
Abgabefrist:	07.12.2023, 06:00 Uhr

Geschlechtergerechte Sprache

Wenn das generische Maskulinum gewählt wurde, geschieht dies zur besseren Lesbarkeit und zum einfachen Verständnis der Aufgabenstellung. Sofern nicht anders angegeben, beziehen sich Angaben im Sinne der Gleichbehandlung auf Vertretende aller Geschlechter.

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 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.

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 zum Ausschluss der Erfolgskontrolle führen.

Kommunikation und aktuelle Informationen

In unseren *FAQs*[1] finden Sie einen Überblick über häufig gestellte Fragen und die entsprechenden Antworten zum Modul „Programmieren“. Bitte lesen Sie diese sorgfältig durch, noch bevor Sie

[1] https://sdq.kastel.kit.edu/wiki/Programmieren/FAQ

Fragen stellen, und überprüfen Sie diese regelmäßig und eigenverantwortlich auf Änderungen. Beachten Sie zudem die Hinweise im ILIAS-Wiki[2].

In den *ILIAS-Foren* und 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.

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[3] einsehen.

Bearbeitungshinweise

Bitte beachten Sie, dass das erfolgreiche Bestehen der verpflichtenden Tests für eine erfolgreiche Abgabe von Übungsblatt 2 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 17*.
- 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` und `java.io`.
- Achten Sie darauf, nicht zu lange Zeilen, Methoden und Dateien zu erstellen. Sie müssen bei Ihren Lösungen eine maximale Zeilenbreite von 120 Zeichen einhalten.
- Halten Sie alle Whitespace-Regeln ein.
- Halten Sie alle Regeln zu Variablen-, Methoden- und Paketbenennung 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 ILIAS-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 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.

[2] `https://ilias.studium.kit.edu/goto.php?target=wiki_2213632_Hauptseite`
[3] `https://artemis.praktomat.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 im ILIAS beschreibt, wie Checkstyle verwendet werden kann.

Abgabehinweise

Die Abgabe im Online-Einreichungssystem wird am 29.11.2023, 12:00 Uhr, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Einreichungssystem bei der richtigen Aufgabe vor Ablauf der Abgabefrist am 07.12.2023, 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.
- Geben Sie online Ihre `*.java`-Dateien zur Aufgabe C in Einzelarbeit mit der entsprechenden Ordnerstruktur im zugehörigen Verzeichnis ab.

Aufgabe A: String-Utility (6 Punkte)

Implementieren Sie die folgenden sechs öffentlichen Methoden in einer Utility-Klasse `StringUtils`. Diese Klasse darf keine `main`-Methode enthalten. Stellen Sie zudem sicher, dass alle Methoden statisch sind und die Klasse keine Attribute beinhaltet.

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 gegebene Zeichenkette umgekehrt. Diese Methode gibt die Zeichen in umgekehrter Reihenfolge wieder als eine neue Zeichenkette zurück.
2. Schreiben Sie eine Methode `boolean isPalindrome(String word)`, welche überprüft, ob die ihr gegebene 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 (beginnend bei 0) entfernt. Diese Methode gibt eine neue Zeichenkette ohne das Zeichen zurück.
4. Schreiben Sie eine Methode `boolean isAnagram(String word, String otherWord)`, welche überprüft, ob die beiden gegebenen Zeichenketten ein Anagramm voneinander sind. Als Anagramm wird eine Zeichenkette bezeichnet, die aus einer anderen Zeichenketten allein durch Umstellung der Zeichen gebildet ist.
5. Schreiben Sie eine Methode `String capitalize(String word)`, welche das erste Zeichen der ihr gegebenen Zeichenkette großschreibt. Diese Methode gibt eine neue Zeichenkette mit einem Zeichen in Großschreibung 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.

Hinweis: Die Verwendung von `char[]` ist nicht verboten, aber auch nicht zwingend benötigt.

Aufgabe B: Zig Zag (8 Punkte)

In dieser Aufgabe soll eine Zeichenkette in ein mehrzeiliges Zigzag-Muster überführt werden. Dazu werden die Buchstaben vertikal nach unten versetzt, sodass die Zeichen eine ZigZag-förmige Linie bilden (hier mit Tiefe 4, Länge 13):

```
1 | t    a     g
2 |  h  s z   a
3 |    i i  i z
4 |     s    g
```

Für ein beliebiges Wort der Länge n (in Zeichen) und einer maximalen Tiefe t (in Zeilen) kann die Anordnung der Buchstaben in einem ZigZag-Muster folgendermaßen beschrieben werden:

Das Wort wird in t Zeilen angeordnet. Jede Zeile enthält n Zeichen, welche entweder Zeichen der ursprünglichen Zeichenkette oder Leerzeichen sind. Die Buchstaben der ursprünglichen Zeichenkette werden so in die Zeilen platziert, dass aufeinander folgende Zeichen nie in der gleichen Zeile sind.

1. Das Muster beginnt oben links und setzt sich diagonal abwärts nach rechts fort, bis Tiefe t erreicht wird.
2. Nachdem die Zeile t erreicht wurde, ändert sich die Richtung, und das Muster setzt sich diagonal von unten links nach oben rechts fort, bis die erste Zeile erreicht wird.
3. Nun beginnt man wieder mit Schritt 1 und wiederholt dieses Muster, bis alle Zeichen der ursprünglichen Zeichenkette in das ZigZag-Muster platziert wurden.

Da es immer n Zeichen in einer Zeile gibt, werden die verbleibenden Plätze nach dem letzten Buchstaben des Wortes mit Leerzeichen aufgefüllt (wieder mit Tiefe 4, Länge 13):

```
1 | t␣␣␣␣␣a␣␣␣␣␣g
2 | ␣h␣␣␣s␣z␣␣␣a␣
3 | ␣␣i␣i␣␣␣i␣z␣␣
4 | ␣␣␣s␣␣␣␣␣g␣␣␣
```

Die genaue Anzahl der Zeilen und die Anzahl der Zeichen in jeder Zeile hängen von n und t ab. Hierbei gilt sowohl $1 < n < t$ als auch $1 < t < n$ als eine valide Eingabe.

Beachten Sie, dass das Muster je nach Wert von t variieren kann. Wenn t größer ist, werden mehr Zeilen im Muster angezeigt, und das Muster wird tiefer. Jedoch enthält es dann auch weniger Richtungswechsel. Wenn t kleiner ist, werden weniger Zeilen angezeigt, und das Muster enthält mehr Richtungswechsel. Hier das vorherige Beispiel, jedoch nun mit Tiefe 3:

```
1 | t   i   i   g
2 |  h s s z g a
3 |    i   a   z
```

B.1 Zig Zag API

Implementieren Sie eine Klasse, welche Methoden für das Erzeugen und Rückgängigmachen von beliebigen ZigZag-Mustern anbietet.

Hierbei soll eine Methode aus einer Zeichenkette mit Angabe einer Tiefe das ZigZag-Muster generieren und dieses als mehrzeilige Zeichenkette zurückgeben.

Eine weitere Methode soll eine mehrzeilige Zeichenkette annehmen, welche ein ZigZag-Muster enthält, und daraus die ursprüngliche Zeichenkette errechnen. Dieses wird dann als Zeichenkette zurückgegeben.

B.2 Hauptklasse

Erstellen Sie zudem eine Hauptklasse mit einer `main`-Methode, welche es erlaubt, die von Ihnen implementierte Funktionalität anzusprechen. Hierbei werden immer mindestens zwei Argumente übergeben.

Als erstes Argument wird immer eine Ganzzahl für die Tiefe übergeben. Hat die Tiefe einen Wert größer 0, dann wird als zweites Argument **eine** Zeichenkette übergeben. Für diese Zeichenkette soll das Zigzag-Muster der angegebenen Tiefe berechnet und auf der Konsole per Standardausgabe ausgegeben werden.

Hat die Tiefe den Wert 0, dann werden eine oder mehrere Zeichenketten übergeben. Diese Zeichenketten repräsentieren die einzelnen Zeilen eines ZigZag-Musters. Die erste Zeichenkette stellt dabei Zeile 1 dar, die zweite Zeile 2, und so weiter. Alle Zeichenketten haben hierbei die gleiche Länge. Für das in den Zeichenketten enthaltene ZigZag-Muster soll dann die ursprüngliche Zeichenkette berechnet und diese dann auf der Konsole per Standardausgabe ausgegeben werden.

Ist die Tiefe negativ, soll `"Invalid Depth: <t>"` ausgegeben werden, wobei `<t>` ein Platzhalter für den Wert der Tiefe ist. Sie können davon ausgehen, dass immer mindestens zwei Argumente übergeben werden. Zudem können Sie davon ausgehen, dass keine Zeichenketten leer (`""`) sind, und dass das erste Argument immer eine valide Ganzzahl ist.

B.3 Beispielinteraktion

Die Beispielabläufe zeigen Ausführungen des Programms. Die Zeilennummern und die Trennlinie sind kein Bestandteil der Benutzerschnittstelle, sie dienen lediglich zur Orientierung für die gegebene Beispielinteraktion. Die Zeichen `%>` (Prozent-Zeichen und Größer-Zeichen gefolgt von einem Leerzeichen) stellt die Kommandozeile dar. Der Name des Programms dient nur als Beispiel und ist nicht vorgeschrieben.

Beispielinteraktion

```
1 %> java 4 thisisazigzag
2 t     a     g
3  h   s z   a
4   i i   i z
5    s     g
```

Beispielinteraktion

```
%> java 3 thisisazigzag
t   i   i   g
 h s s z g a
  i   a   z
```

Beispielinteraktion

```
%> java 0 "t  i   i   g" " h s s z g a " " i  a   z  "
thisisazigzag
```

Beispielinteraktion

```
%> java 7 "Dies ist ein langer Satz..."
D                    .
 i        n l        z .
  e      i   a      t   .
   s    e    n    a
               g   S
     i t        e
      s          r
```

B.4 Hinweis zur Kommandozeileninteraktion

Bei dem Aufruf einer Java-Anwendung, die über die Kommandozeile mehrere Argumente entgegennimmt, ist es wichtig, sicherzustellen, dass Argumente mit Leerzeichen richtig behandelt werden. Die korrekte Behandlung kann je nach System und Kommandozeile variieren.

- In den meisten Fällen können Argumente, die Leerzeichen enthalten, durch die Verwendung von doppelten Anführungszeichen (") um das Argument herum behandelt werden. Zum Beispiel: `"Dies ist ein Argument mit Leerzeichen"`.
- Falls doppelte Anführungszeichen als Teil eines Arguments benötigt werden, müssen sie möglicherweise mit einem Escape-Zeichen (\) maskiert werden.
- In der Unix Kommandozeile können auch einzelne Anführungszeichen (') verwendet werden, welche mächtiger sind. Diese verhindern, dass Sonderzeichen interpretiert werden (z.B. $, !, \, *, ", usw., außer ').
- Einige Kommandozeilen erfordern das Verwenden von Escape-Zeichen (\) vor Leerzeichen, um sicherzustellen, dass sie als Teil des Arguments und nicht als Trennzeichen interpretiert werden. Zum Beispiel: `Dies\ ist\ ein\ Argument\ mit\ Leerzeichen`.

Aufgabe C: Kalender (6 Punkte)

In dieser Aufgabe sollen die grundlegenden Klassen für einen Kalender modelliert werden. Das primäre Ziel ist es, sich mit dem Entwurf und Erstellen von Klassen in Java vertraut zu machen. Darüber hinaus sollen Sie das Hinzufügen von Attributen zu Klassen üben und die Auswahl geeigneter Typen für diese Attribute erlernen. Es geht hierbei um die reine objektorientierte Modellierung der Klassen und ihrer Attribute. Das heißt, Ihre Klassen beinhalten noch keine Konstruktoren oder Methoden. Achten Sie auf den Einsatz von *sinnvollen* Datentypen und wählen Sie aussagekräftige *englischsprachige* Bezeichner.

Beachten Sie, dass in Java üblicherweise die elementaren Typen `boolean` für Aussagenlogik, `int` für Ganzzahlen und `double` für Fließkommazahlen sowie die Klasse `String` für Zeichenketten benutzt werden. Darüber hinaus ist das Konzept `enum` für Aufzählungen nützlich, für die bereits beim Schreiben des Java-Quelltext alle möglichen Werte bekannt sind.

Wochentag Als Wochentag wird einer der sieben Tage der Woche mit wiederkehrender Benennung bezeichnet: *Montag*, *Dienstag*, *Mittwoch*, *Donnerstag*, *Freitag*, *Samstag* und *Sonntag*.

Monat Als Monat wird einer der zwölf Monate eines Jahres mit wiederkehrender Benennung bezeichnet: *Januar*, *Februar*, *März*, *April*, *Mai*, *Juni*, *Juli*, *August*, *September*, *Oktober*, *November* und *Dezember*.

Uhrzeit Die Uhrzeit ist die Angabe eines genau bestimmten Momentes in einem zeitlichen Bezugssystem, ausgedrückt in *Stunden*, *Minuten* und *Sekunden* ab Mitternacht eines Tages. Wobei der Tag in 24 Stunden, die Stunde in 60 Minuten und die Minute in 60 Sekunden unterteilt ist. Sekunden werden hierbei nicht weiter in kleinere Einheiten zerteilt.

Kalenderdatum Das Kalenderdatum ist die Angabe eines Kalendertages im gregorianischen Kalender, ausgedrückt in einem *Jahr*, einem *Monat* im Jahr und einem *Tag* im Monat. Im Kalender sollen nur Jahre von einschließlich -999999 bis einschließlich +999999 betrachtetet werden. Ein Jahr wird in 12 Monate eingeteilt, welche eine Zeitspanne von entweder 28, 29, 30 oder 31 aufeinanderfolgenden Tagen haben.

Zeitpunkt Ein Zeitpunkt ist die Angabe eines Momentes, ausgedrückt in *Stunden*, *Minuten* und *Sekunden* eines Tages, welcher wiederum in einem Kalenderdatum, in einem *Jahr*, einem *Monat* im Jahr und einem *Tag* im Monat, ausgedrückt wird.

Zeitzone Eine Zeitzone wird durch diejenigen Zonen der Erde gebildet, in welchen die gleiche Uhrzeit und das Kalenderdatum gelten. Eine Zeitzone hat eine festen *Zeitabstand* in Stunden und einen *Identifikator* aus mehreren Zeichen des standardmäßigen Java-Zeichensatzes. Der Zeitabstand einer Zeitzone zu einer koordinierten Weltzeit beträgt zwischen minus zwölf und plus 14 Stunden.

Zonenzeit Die Zonenzeit ist der einheitliche *Zeitpunkt* in einer *Zeitzone*.