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 2024/25

Übungsblatt 3 Version 1.0	20 Punkte	Ausgabe:	27.11.2024, ca. 12:00 Uhr
		Abgabestart:	04.12.2024, 12:00 Uhr
		Abgabefrist:	12.12.2024, 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 3 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`, `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.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 04.12.2024, 12:00 Uhr, freigeschaltet. Achten Sie unbedingt darauf, Ihre Dateien im Einreichungssystem bei der richtigen Aufgabe vor Ablauf der Abgabefrist am 12.12.2024, 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: Dots-and-Boxes (12 Punkte)

In dieser Aufgabe sollen das Spiel Dots-and-Boxes[5] implementieren.

A.1 Grundlagen

Dots-and-Boxes ist ein einfaches Spiel, in dem 2 Personen abwechselnd Linien zwischen Punkten auf einem quadratischen Spielfeld ziehen. Wenn eine Person die vierte Linie um eine Box herum zieht, bekommt er einen Punkt und darf erneut eine Linie ziehen. Das Spiel endet wenn alle Linien gezogen worden sind worauf die Person mit den meisten Punkten gewinnt. Die Spielfelder können eine beliebige Größe zwischen $2x2$ und $9x9$ haben.

Es gibt drei verschiedene Startbelegungen für das Spiel, welche in Abbildung A.1 dargestellt sind:

- Ein amerikanisches Spiel beginnt mit einem komplett leeren Spielfeld
- Ein isländisches Spiel beginnt mit Linien am linken und unteren Rands des Spielfels
- Ein schwedisches Spiel beginnt mit Linien an allen Rändern des Spielfelds

Abbildung A.2 zeigt nun einen Bespielablauf auf einem $2x2$ Feld mit amerikanischer Startbelegung. Person A beginnt und gewinnt nach dem 12. Zug mit 3 zu 1 Punkten.

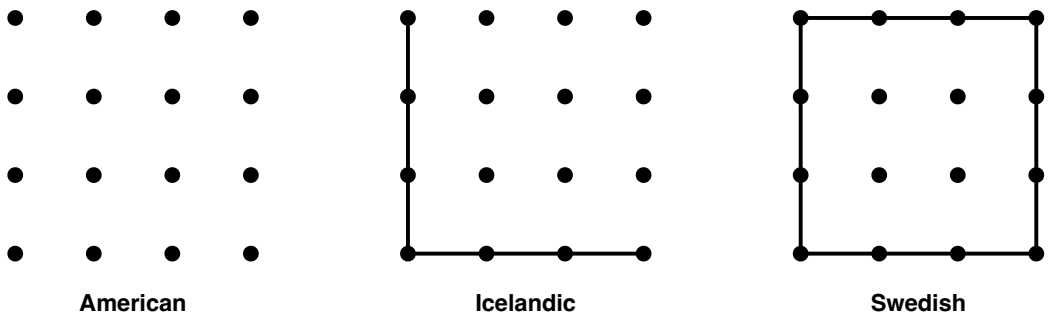

Abbildung A.1: Verschiedene Startbelegungen für Dots-and-Boxes

[5] `https://en.wikipedia.org/wiki/Dots_and_boxes`

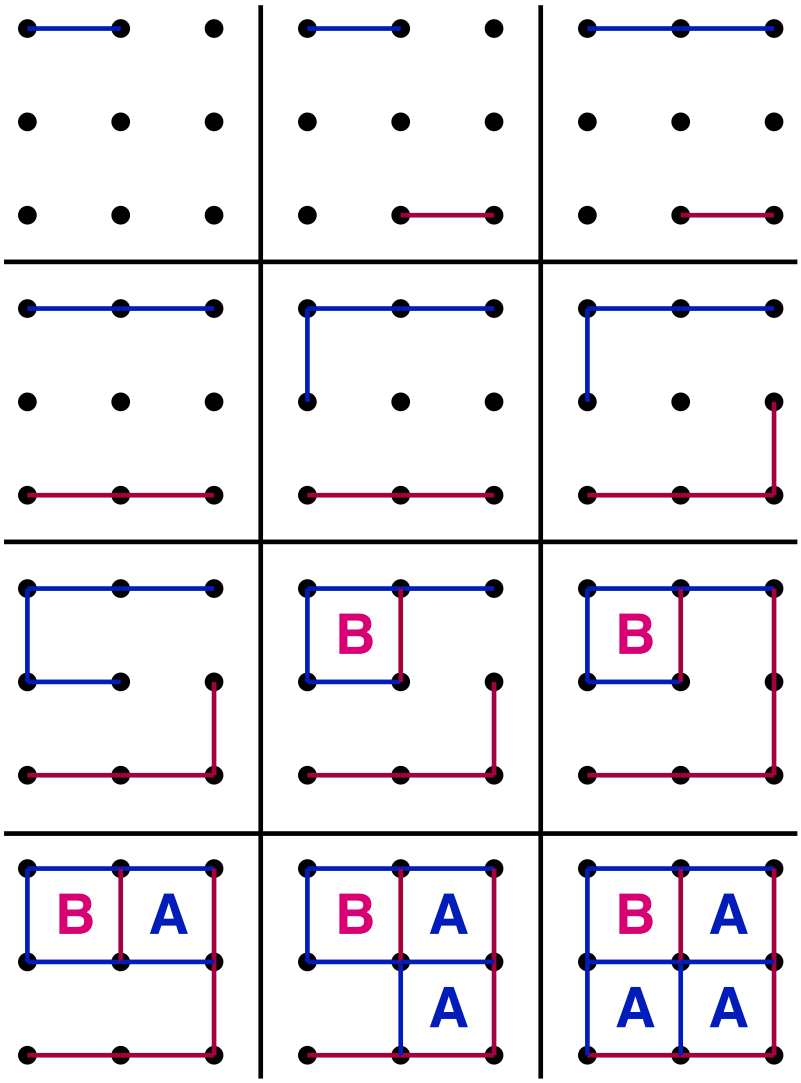

Abbildung A.2: Beispielablauf wobei Person A beginnt und gewinnt

A.2 Funktionalität

Im Folgenden wird die Funktionalität der Implementierung beschrieben. Es wird der Spielablauf erklärt und zudem wird erläutert, wie Personen mit dem Programm interagieren.

A.2.1 Ausgabe des Spielfeldes

Die Ausgabe zeigt ein Raster aus Punkten, Linien und Kästchen, um den Spielfeldstatus darzustellen. Die Punkte werden als Plus-Zeichen `+` an den Ecken der Kästchen angezeigt, während horizontale und vertikale Linien durch Bindestriche `-` und senkrechte Striche `|` die Verbindungen markieren, die von den Personen gezogen wurden. Spalten sind mit Buchstaben und Zeilen mit Zahlen beschriftet, um die Orientierung zu erleichtern. Vollständig umschlossene Kästchen enthalten das Symbol der Person (z.B. A), der das Kästchen für sich beansprucht hat.

Im Folgenden sehen sie eine Beispielausgabe für ein 2*x*2 Feld. Achten sie auf die korrekte Anzahl von Leerzeichen, welche im Beispiel extra markiert worden sind.

```
1 ␣␣␣a␣␣␣b
2 ␣+␣-␣+␣-␣+
3 1|␣A␣|
4 ␣+␣-␣+␣␣␣+
5 2|␣␣␣␣␣␣␣|
6 ␣+␣␣␣+␣-␣+
```

A.2.2 Spielstart

Das Spiel wird über eine `main`-Methode gestartet. Über die Kommandozeilenargumente werden die Größe des Spielfeld (Zahl zwischen und `2` und `9`) und die Startbelegung(`american`, `icelandic`, oder `swedish`) festgelegt.

Ein Beispielaufruf ist `java DotsAndBoxes 2 american`

Bevor der erste Spielzug beginnt, wird das Spielfeld ausgeben. Das Programm nimmt per Kommandozeileneingabe abwechselnd die Züge der zwei Personen entgegen. Es beginnt immer zuerst die Person `A` und danach folgt die Person `B`. Nach jedem Zug wird das Spielfeld ausgegeben.

A.2.3 Spielzug

Ein Spielzug enthält die folgenden Schritte (in dieser Reihenfolge):

1. Zu Beginn eines Spielzugs wird angegeben welche Person am Zug ist z.B.: (`"Player A:"`).
2. Danach wird der Zug der Person eingelesen. Die Person gibt über die Kommandozeile eine Kombination aus Buchstabe und Zahl, die das Feld referenziert (z.B. `a1`), darauf folgt ein Leerzeichen und darauf folgt `up`, `right`, `down`, oder `left`.
3. Ist diese Linie bereits gezogen oder die Eingabe außerhalb des Feldes oder ungültig, so wird erneut Schritt 1 und 2 des Spielzuges ausgeführt.
4. Zuletzt wird das geänderte Spielfeld ausgegeben.
5. Wenn eine Person die vierte Linie um ein Feld gezogen hat, wird dieses mit seiner Kennung markiert und sie ist erneut am Zug.
6. Sollte eine Person das Kommando `quit` eingeben, so wird das Programm vorzeitig beendet.

A.2.4 Spielende

Wenn alle Linie gezogen wurden, endet das Spiel. Die Person mit den meisten Kästchen gewinnt. Auf der Kommandozeile wird `Player A wins!` bzw. `Player B wins!` ausgegeben.

A.2.5 Beispielablauf

Im Folgenden sehen sie einen Beispielauflauf des Programmes. Die Zeilennummern und die Trennlinie sind kein Bestandteil der Benutzerschnittstelle, sie dienen lediglich zur Orientierung für die gegebene Beispielinteraktion. Nutzereingaben sind mit dem Zeichen `>` gekennzeichnet.

Beispielinteraktion

```
%> java
   DotsAndBoxes
   2 american
   a   b
 +   +   +
1
 +   +   +
2
 +   +   +
Player A:
> a1 up
   a   b
 + - +   +
1
 +   +   +
2
 +   +   +
Player B:
> b2 down
   a   b
 + - +   +
1
 +   +   +
2
 +   + - +
Player A:
> b1 up
   a   b
 + - + - +
1
 +   +   +
2
 +   + - +
Player B:
> a2 down
   a   b
```

Beispielinteraktion

```
 + - + - +
1
 +   +   +
2
 + - + - +
Player A:
> a1 left
   a   b
 + - + - +
1|
 +   +   +
2
 + - + - +
Player B:
> b2 right
   a   b
 + - + - +
1|
 +   +   +
2       |
 + - + - +
Player A:
> a1 down
   a   b
 + - + - +
1|
 + - +   +
2       |
 + - + - +
Player B:
> a1 right
   a   b
 + - + - +
1| B |
 + - +   +
```

Beispielinteraktion

```
2       |
 + - + - +
Player B:
> b1 right
   a   b
 + - + - +
1| B |   |
 + - +   +
2       |
 + - + - +
Player A:
> b1 down
   a   b
 + - + - +
1| B | A |
 + - + - +
2       |
 + - + - +
Player A:
> b2 left
   a   b
 + - + - +
1| B | A |
 + - + - +
2   | A |
 + - + - +
Player A:
> a2 left
   a   b
 + - + - +
1| B | A |
 + - + - +
2| A | A |
 + - + - +
Player A wins!
```

Aufgabe B: Adventskalender (8 Punkte)

In dieser Aufgabe entwerfen Sie eine Datenstruktur für einen Adventskalender. Der Adventskalender enthält Süßigkeiten, die jeweils durch die gegebene Klasse `Candy` repräsentiert werden. Jede Süßigkeit ist hierbei hinter einer Tür versteckt, die einem Tag im Dezember zugeordnet ist (zum Beispiel vom 01.12 bis zum 24.12 bei 24 Süßigkeiten). Eine Tür kann nur dann geöffnet werden, wenn das aktuelle Datum des entsprechenden Tags schon erreicht wurde. So können zum Beispiel am 4. Dezember nur die ersten vier Türen geöffnet werden. Jede Tür kann hierbei nur einmal geöffnet werden. Der aktuelle Tag wird mit den Süßigkeiten in der Datenstruktur intern gespeichert.

In diesem Übungsblatt sollen Datentypen aus dem Paket `java.util` verwendet werden. Die Methoden der Datenstruktur verwenden Listen als Rückgabewerte und als Parameter.
Hinweis: In Java können Sie eine Liste von Süßigkeiten wie folgt erzeugen:
`List<Candy> sweets = new ArrayList<>();`
Diese Liste ist initial leer und auf dem Listenobjekt können dann verschiedene Methoden aufgerufen werden. Informieren Sie sich dazu im Javadoc der Java-17-API [6].

Implementieren Sie die folgenden Methoden der Klasse `AdventCalendar`:

- `public AdventCalendar(List<Candy> candies)` ist ein Konstruktor und erstellt einen Kalender mit den in `candies` übergebenen Süßigkeiten. Sie können hierbei davon ausgehen, dass die Liste nicht null oder leer ist und eine bestimmte Anzahl an Süßigkeiten enthält, die nicht null sind. Die n-te Süßigkeit in der Liste ist hierbei dem n-ten Tag zugeordnet. Je nach Größe der Liste wird der *Maximaltag* bestimmt und der Adventskalender hat dementsprechend viele Türen (wichtig: die Größe ist *nicht* auf 24 beschränkt). Nach Aufruf des Konstruktors ist der aktuelle Tag der Tag *vor* dem ersten Dezember.
- `public int getDay()` gibt den aktuellen Tag als Ganzzahl zwischen 0 und dem Maximaltag zurück. Der Tag vor dem ersten Dezember wird durch den Wert 0 repräsentiert. Der erste Tag an dem eine Türe geöffnet werden kann wird also durch den Wert 1 repräsentiert.
- `public boolean nextDay()` erhöht den intern gespeicherten Tag. Damit wird gesteuert, welche Tür geöffnet werden kann. Wurde der Tag erhöht, wird `true` zurückgegeben. Der Tag kann nur bis zum Maximaltag erhöht werden. Danach wird beim Aufruf der Methode `false` zurückgegeben.
- `public boolean nextDays(int days)` erhöht den intern gespeicherten Tag um mehrere Tage. Das Parameter `days` bestimmt hierbei die Anzahl der erhöhten Tage. Der Tag kann nur bis zum Maximaltag erhöht werden. Würde die Erhöhung über den Maximaltag hinaus gehen, werden keine Tage erhöht und `false` wird zurückgegeben. Ist `days` kleiner oder gleich 0, wird direkt `false` zurückgegeben. Bei Erfolg wird `true` zurückgegeben.
- `public boolean isDoorOpen(int number)` überprüft, ob die Tür mit der passenden Nummer bereits geöffnet wurde. Ist dies der Fall, wird `true` zurückgegeben. Ansonsten wird `false` zurückgegeben. Ist die Nummer echt kleiner eins oder echt größer dem Maximaltag, wird `false` zurückgegeben.

[6] `https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html`

- `public Candy openDoor(int number)` öffnet die Tür mit der passenden Nummer und gibt die Süßigkeit zurück. Wurde die Tür bereits geöffnet, wird `null` zurückgegeben. Ist die Nummer echt größer als der aktuelle Tag, wird `null` zurückgegeben. Ist die Nummer echt kleiner eins oder echt größer dem Maximaltag, wird `null` zurückgegeben.
- `public List<Candy> openDoors(List<Integer> numbers)` öffnet alle Türen deren Nummer in der Liste enthalten ist und gibt die Süßigkeiten in gleicher Reihenfolge als Liste zurück. Wurde eine Tür bereits geöffnet, wird die Tür übersprungen, also kein Element in die Liste hinzugefügt. Ist eine Nummer kleiner als 1 oder größer als der Maximaltag, wird die Nummer auch übersprungen.
- `public int numberOfUnopenedDoors()` gibt die Anzahl der bisher noch nicht geöffneten Türen zurück, die zum aktuellen Tag geöffnet werden könnten. Damit ist der Rückgabewert immer zwischen 0 und der Ausgangsgröße des Kalenders. Ist zum Beispiel der Tag der 9. Dezember und es wurden bisher drei Türen geöffnet, wird 6 zurückgegeben.
- `public void reset()` setzt den Kalender zum Ursprungszustand zurück. Der interne Tag ist nun wieder der Tag *vor* dem ersten Dezember. Alle Türen sind wieder geschlossen und die Süßigkeiten hinter den Türen entsprechen dem Zustand nach dem Aufruf des Konstruktors.
- `public boolean resetLastDoors(int doors)` setzt die letzten Türen zurück indem die Süßigkeiten zurückgelegt und die Türen geschlossen werden. Anschließend wird `true` zurückgegeben. Wenn mehr Türen zurückgesetzt werden sollen als Türen geöffnet wurden, macht die Methode nichts und `false` zurück.
- `public String toString()` gibt den Adventskalender in Textform zurück. Eine bereits geöffnete Tür wird durch einer öffnenden eckigen Klammer gefolgt von *drei Leerzeichen* und abschließend einer schließenden eckigen Klammer dargestellt. Eine ungeöffnete Tür mit der Süßigkeiten mit dem Namen `"Schokolade"` und der Häufigkeit 2 wird als `[Schokolade:2]` repräsentiert, also eine öffnende eckige Klammer gefolgt von dem Namen der Süßigkeit, dann das Zeichen `":"`, dann die Häufigkeit abschließend eine schließende eckige Klammer. Die Türen werden in der Reihe der Tage konkateniert ausgegeben. Mit Ausnahme der letzten Tür wird nach jeweils vier Türen ein Zeilenumbruch angehängt. Ansonsten werden *keine* Trennzeichen verwendet. Ein Kalender mit 25 Tagen, der nur mit der Süßigkeit `"S"` (Häufigkeit 1) gefüllt wurde und bei dem Tür 1 bis 6 bereits geöffnet wurden, wird wie folgt repräsentiert:

```
[   ][   ][   ][   ]
[   ][   ][S:1][S:1]
[S:1][S:1][S:1][S:1]
[S:1][S:1][S:1][S:1]
[S:1][S:1][S:1][S:1]
[S:1][S:1][S:1][S:1]
[S:1]
```

Hinweise: In dieser Aufgabe sollen Sie sinnvolle Javadoc Kommentare für die Klasse und alle öffentlichen Methoden schreiben. Implementieren Sie keine weiteren öffentlichen Methoden und keine öffentlichen Attribute. Erstellen Sie keine weiteren Klassen. Private Attribute und private Methoden sind erlaubt. Verändern Sie nicht die gegebene Klasse mit dem Namen `Candy`.