Testat 1

Hinweise zur Bearbeitung und Abgabe

- Dieses Testat in Form einer Online-Hausarbeit ist eine Prüfungsleistung und muss daher **komplett selbständig** bearbeitet werden. Wir werden Plagiatsprüfungen durchführen. Bestätigte Plagiate führen zu Nicht-Bestehen des Testats.
- Für das Testat gilt eine "**Open-Book**"-Richtline, das heißt Kursmaterialien sowie Onlineressourcen wie die Java-Dokumentation oder Bücher dürfen zur Bearbeitung herangezogen werden.
- Die Abgabe erfolgt **bis zum 23.12.2020 um 23:59 Uhr auf GATE**[1] in der Veranstaltung "Einführung in die Programmierung (Testate)"
- Mit diesem Testat können Sie 20 Punkte erreichen. Dies entspricht 20% der gesamten Prüfungspunkte. Weitere Punkte können durch ein weiteres Testat im Januar und die Online-Klausur im Februar erreicht werden.
- Bis zur Abgabefrist am 23.12.2020 dürfen im Vorlesungs-Chat **keine inhaltlichen Fragen zum Testat** gestellt werden! Bitte wenden Sie sich nur in Ausnahmefällen an Fiona Draxler oder Dennis Dietz.
- Die zu bearbeitenden Aufgaben sind in zwei Teile gegliedert. Teil 1 beinhaltet alle Aufgaben, die umgesetzt werden **müssen**, damit man in diesem Prüfungsteil genug Punkte für die Bestehensgrenze der Gesamtprüfung sammelt (10 Punkte). Teil 2 beinhaltet alle weitere Aufgaben, für deren Umsetzung Sie zusätzliche Punkte (10 Punkte + 3 Bonuspunkte) erhalten können.
- Beachten Sie bei der Implementierungen folgende Richtlinien:
  - Ihr Programm muss kompilieren und darf keine Packagedeklarationen enthalten. Packageimporte sind möglich.
  - Verwenden Sie Kommentare zur Dokumentation in sinnvoller Weise. Beschreiben Sie etwa kurz, wie eine Methode zur allgemeinen Problemlösung beiträgt.
  - Der Programmcode muss sinnvoll geklammert und eingerückt werden. Benennungen sollten den Java-Namenskonvention (s. Folien 03-04) folgen und sollten möglichst selbsterklärend sein.
  - Achten Sie auf sinnvolle Fehlerbehandlung, etwa bei ungültigen Eingaben.
  - Wichtig sind die korrekte Verwendung von Arrays und die sinnvolle Verwendung von Prozeduren.

Viel Erfolg bei der Bearbeitung!

[1] `https://gate.ifi.lmu.de`

Einführung in das Thema

Sie haben durch die Vorlesung und Übung nun erste Grundlagen der Java-Programmierung gesammelt. Im ersten Testat möchten wir Ihnen die Möglichkeit bieten, Ihr erlerntes Wissen unter Beweis zu stellen und in einem praktischen Kontext zu testen. Hierfür haben wir die Entwicklung einer To-Do-Liste vorgesehen. Diese To-Do-Liste soll eine Reihe von zu erledigenden Aufgaben (To-Dos) verwalten. Dabei sollen mindestens die grundlegenden Funktionen

a) Einträge anzeigen

b) Eintrag hinzufügen

c) Eintrag löschen

implementiert werden und diese sollen über die Konsole ausführbar sein. Das Konzept kann aber mit verschiedenen Features erweitert werden.

Teil 1 – Minimale Funktionalität (10 Punkte)

Vorbereitung

Erstellen Sie eine Klasse `ToDo`. Legen Sie ein statisches Stringarray namens `todos` mit einer Kapazität von `10` an und der Sichtbarkeit **`public`** an, in dem Sie später Ihre Aufgaben speichern.

Hinzufügen von Aufgaben

Implementieren Sie eine Methode

```
public static boolean neueAufgabe(String titel) {...}
```

Diese Methode soll die neue Aufgabe in Ihrem wie oben angelegten Array speichern. Das neue Element soll an der nächsten "freien", also mit **`null`** belegten Position gespeichert werden. Das Einfügen soll nicht möglich sein, wenn es bereits eine Aufgabe mit dem gleichen Titel gibt oder wenn das Array bereits voll ist. Geben Sie einen Boolean-Wert zurück, der angibt, ob eine neue Aufgabe eingefügt werden konnte.

Anzeigen von Aufgaben

Implementieren Sie eine Methode

```
public static String listeAlleAufgaben() {...}
```

Diese Methode soll `Aufgaben auf der Liste:\n` gefolgt von allen gespeicherten Aufgaben als String zurückgeben. Formatieren Sie den String so, dass jede Aufgabe im String in einer eigenen Zeile steht. Leere Elemente sollen nicht ausgegeben werden.

Löschen von Aufgaben

Implementieren Sie eine Methode

```
public static boolean loescheAufgabe(String titel) {...}
```

Diese Methode soll die Aufgabe mit dem gegebenen Titel `titel` aus Ihrem Array entfernen, indem Sie die Stelle auf **`null`** setzen. Alle nachfolgenden Aufgaben sollen um eine Stelle nach vorne verschoben werden. Ist der Titel nicht in der Liste vorhanden, soll nichts passieren. Geben Sie einen Boolean-Wert zurück, der angibt, ob die Löschung vorgenommen werden konnte.

Interaktion über die Konsole

Implementieren Sie nun in der Mainmethode der Klasse `ToDo` ein Kommandozeileninterface, das die Interaktion mit den oben beschriebenen Methoden ermöglicht. Beim Starten des Programms soll zunächst ein Begrüßungstext auf der Konsole angezeigt werden:

```
Willkommen in Ihrer To-Do-Liste, was möchten Sie tun?
```

Dann sollen die folgenden Interaktionsmöglichkeiten aufgelistet werden:

```
[1] To-Dos anzeigen
[2] Eintrag hinzufügen
[3] Eintrag löschen
[4] Programm beenden
```

Nach Eingabe einer der Zahlen 1, 2, 3 oder 4 soll die entsprechende Aktion ausgeführt werden. Die genaue Interaktion ist in Abbildung 1 beschrieben. Prinzipiell besteht die Interaktion aus einem Wechsel von Konsolenausgaben, Nutzereingaben und im Hintergrund ausgeführten Methoden. Sobald eine Aktion beendet wurde, sollen wieder die Interaktionsmöglichkeiten aufgelistet werden.
Für Konsoleneingaben können Sie `Scanner` verwenden, eine Klasse, die Ihnen dabei hilft, Texteingaben zu verarbeiten. Dazu importieren Sie am Dateianfang mittels **`import`** `java.util.Scanner;` diese Klasse. Initialisieren Sie den Scanner über

```
Scanner scanner = new Scanner(System.in);
scanner.useDelimiter("\n"); // für Eingaben aus mehreren Wörtern
```

Damit geben Sie dem Programm an, dass ein Scanner namens `scanner` vorbereitet werden soll, der Eingaben von der Eingabekonsole (`System.in`) erwartet. Anschließend soll der Scanner eine ganze Zahl oder einen String einlesen:

```
int auswahl = scanner.nextInt();
String neuerTitel = scanner.next();
```

Sie können den Scanner mit `scanner.close()` schließen und das gesamte Programm mit dem Aufruf von `System.exit(0);` beenden. Andere Lösungen zur Eingabe außer Scanner sind zulässig.

Teil 2 – Mögliche Erweiterungen

Sobald Sie die Funktionalität aus Teil 1 korrekt implementiert haben, haben Sie 50% der Punkte erreicht. Nun können Sie mit den Aufgaben aus Teil 2 beginnen. Je mehr dieser Aufgaben Sie korrekt lösen, desto mehr Punkte erhalten Sie. Die Erweiterungen sind zunehmend komplex; man muss nicht alle umsetzen. Die Konsoleninteraktion aus Teil 1 muss zunächst nicht auf die neuen Funktionalitäten erweitert werden.

Bearbeiten eines Eintrags (1,5 Punkte)

Implementieren Sie eine Methode

```
public static boolean bearbeiteAufgabe(String alterTitel,
                                       String neuerTitel) {...}
```

Diese Methode soll die Aufgabe mit dem gegebenen Titel mit einem neuen Titel ersetzen. Geben Sie einen Boolean-Wert zurück, der angibt, ob die Bearbeitung durchgeführt werden konnte.

Hinzufügen eines Fälligkeitsdatums (4 Punkte)

a) Erweitern Sie Ihre Liste an Aufgaben mit einem Fälligkeitsdatum. Das Datum kann als String im Format "yyyy-MM-DD", also beispielsweise "2020-12-16" oder über die Klasse `LocalDate`[2] aus der Bibliothek `java.time` implementiert werden. Speichern Sie die Fälligkeitsdaten in einem zweiten Array oder einer anderen sinnvollen Variante wie mit Records oder einem zweidimensionalen Array. Nennen Sie in ersterem Fall Ihr neues Array `daten` und achten Sie darauf, beim Hinzufügen oder Löschen einer Aufgabe die Arrays zu synchronisieren! Initialisieren Sie die Fälligkeit jeder vorhandenen und neuen Aufgabe mit dem 23.12.2020.

b) Implementieren Sie nun eine Methode, mit der das Fälligkeitsdatum einer Aufgabe gesetzt werden kann:

```
public static boolean setzeDatum(String titel, int jahr,
                                 int monat, int tag) {...}
```

Wenn Sie das Datum als String speichern, können Sie die Parameter für Monat und Tag mit der Methode `String.format("%02d", zahl)` mit führenden Nullen versehen, damit der Datumsstring dem oben beschriebenen Format entspricht. Geben Sie einen Boolean-Wert zurück, der angibt, ob das Datum korrekt eingetragen werden konnte.

c) Implementieren Sie eine Methode

```
public static String listeAlleAufgabenMitDatum() {...}
```

Diese Methode soll die Ausgabe aus `listeAlleAufgaben` so erweitern, dass nun auch das Fälligkeitsdatum angezeigt wird:

```
Aufgaben auf der Liste:
Einkaufen, fällig am 2020-12-23
Putzen, fällig am 2020-12-23
```

[2] `https://www.w3schools.com/java/java_date.asp`

Sortierung (4 Punkte)

a) Implementieren Sie Methoden zum Sortieren Ihrer Einträge alphabetisch oder nach Fälligkeitsdatum (frühestes Datum als erstes Element):

```
public static String nachTitelSortieren(boolean aufsteigend) {...}
public static String nachDatumSortieren(boolean aufsteigend) {...}
```

Den Wert des Parameters `aufsteigend` können Sie zunächst auf **`true`** setzen und bei der Implementierung ignorieren.

b) Ändern Sie Ihre Sortiermethoden so ab, dass eine aufsteigende (`aufsteigend = true`) oder absteigende (`aufsteigend = false`) Reihenfolge gewählt werden kann.

Implementieren Sie die Sortierung selbst, d. h. verwenden Sie keine Helfermethoden wie `Arrays.sort()`! Die Wahl des Sortier-Algorithmus bleibt Ihnen überlassen. Mit `string1.compareTo(string2)` kann die alphabetische Reihenfolge von zwei Strings ermittelt werden. Die Originalarrays `todos` und ggf. `daten` sollen nicht bearbeitet werden. Formatieren Sie den Rückgabestring nach dem gleichen Prinzip wie in der Methode `listeAlleAufgabenMitDatum()`.

Erweiterte Interaktion über die Konsole (0,5 Punkte)

Erweitern Sie die Konsoleninteraktion aus Teil 1 für Ihre zusätzlich implementieren Methoden nach dem Schema aus Abbildung 2. Geben Sie zusätzlich `Die Aufgabe <Titel> wurde nicht gefunden.` aus, wenn eine nicht vorhande Aufgabe gewählt wird und `Ungültige Eingabe.` für alle anderen ungültigen Eingaben.

Persistente Speicherung in einer Datei (3 Bonuspunkte)

Diese Erweiterung ermöglicht es, weitere Punkte über die max. 20 Punkte des Testats hinaus zu erwerben. Sie wird nur empfohlen, wenn Sie sich mit der Lösung aller anderen Teilaufgaben sicher sind.

Sinnvollerweise sollte eine Aufgabenliste nach dem Beenden eines Programms nicht verschwinden, sondern ist bei der nächsten Ausführung immer noch vorhanden. Dies kann zum Beispiel über eine Datei gelöst werden, in der alle Einträge abgespeichert und später wieder ausgelesen werden. Recherchieren Sie, wie in Java Dateien gelesen und geschrieben werden können. Kapitel 18 aus "Java ist auch eine Insel"[3] bietet zum Beispiel eine Einführung in dieses Thema. Legen Sie dann eine Datei mit Ihrer Matrikelnummer + ".txt" als Dateinamen an und verwenden Sie diese, um beim Starten Ihres Programms die dort gespeicherte Liste an Aufgaben einzulesen. Schreiben Sie außerdem zu einem passenden Zeitpunkt erfolgte Änderungen in die Datei. Sie können dabei die Datei überschreiben oder editieren. Falls Sie Titel und Fälligkeitsdatum jeweils in eine Zeile schreiben, erhalten Sie mit `zeile.split(": ")` ein Stringarray, das alle durch Semicolon und Leerzeichen getrennten Teilstrings von `zeile` enthält.

[3] `http://openbook.rheinwerk-verlag.de/javainsel/18_001.html`

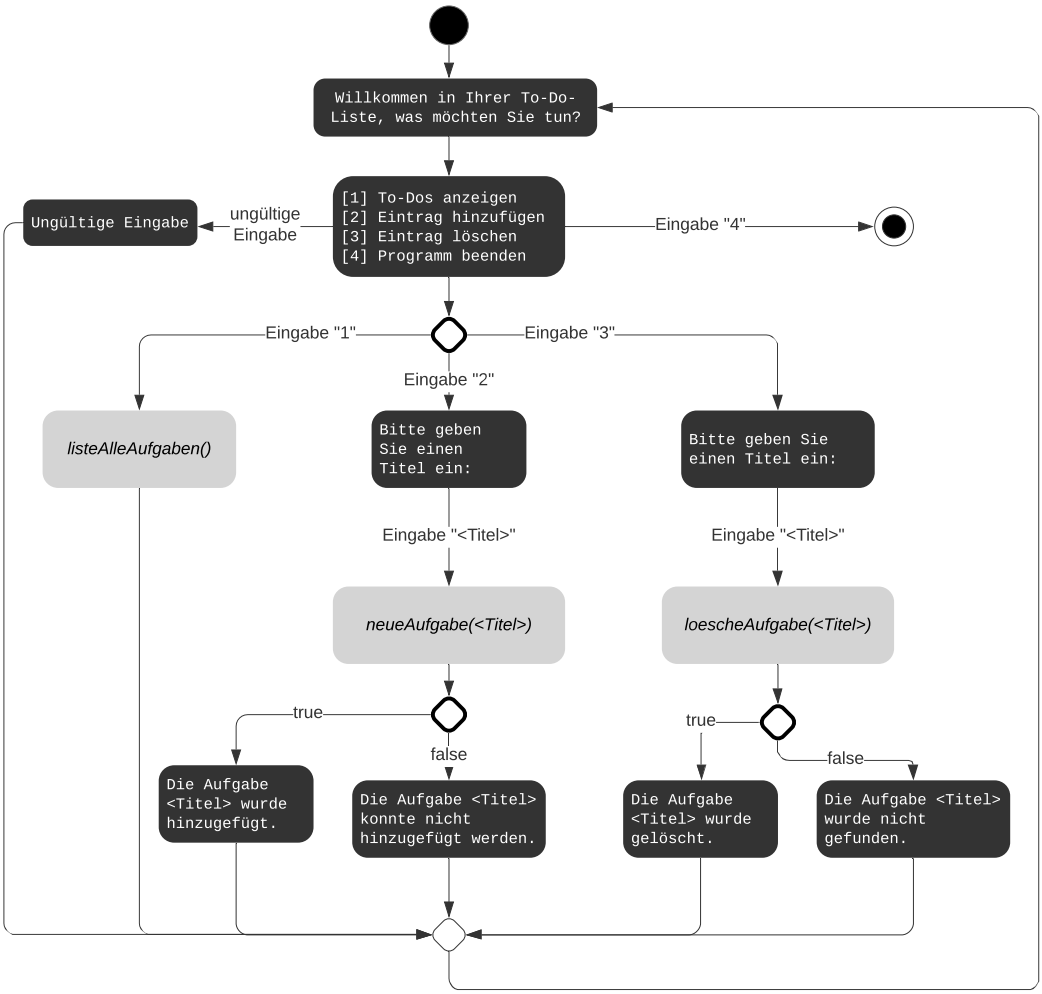

Abbildung 1: Aktivitätsdiagramm für die Konsoleninteraktion aus Teil 1. Schwarze Boxen markieren Konsolenausgaben, graue Boxen markieren ausgeführte Methoden und Rauten markieren Verzweigungen je nach Eingabe oder Ergebnis.

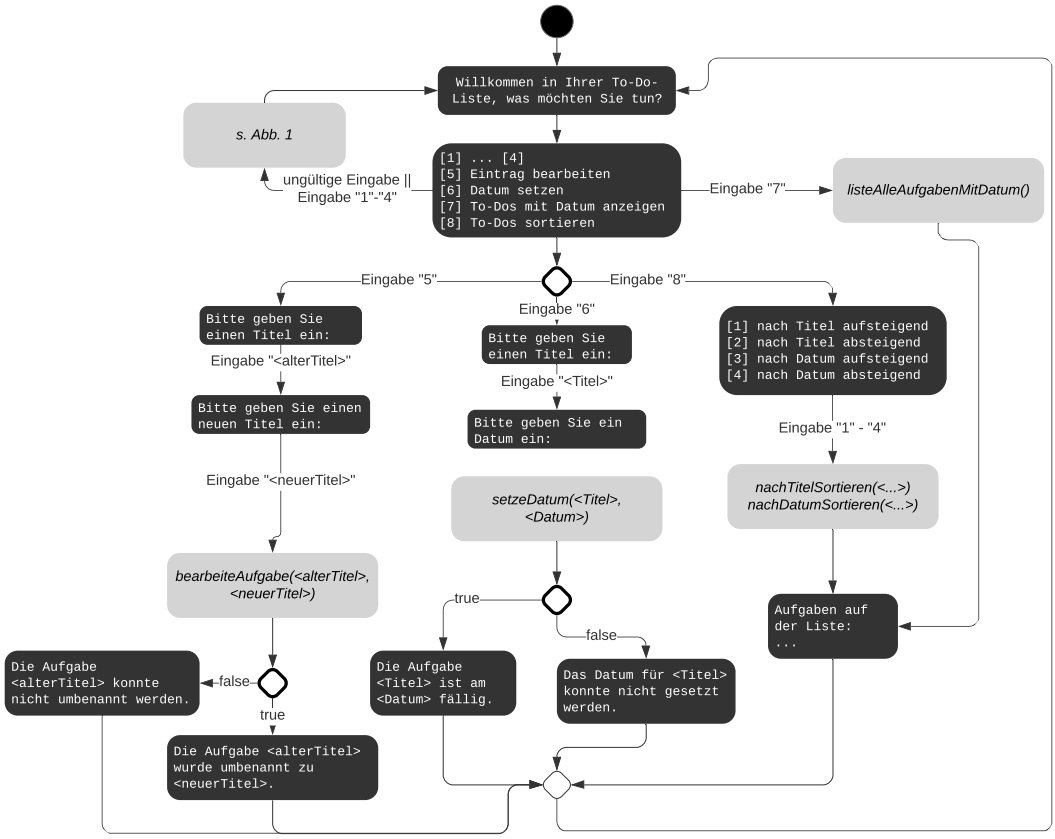

Abbildung 2: Aktivitätsdiagramm für die Konsoleninteraktion aus Teil 2. Fehlerbehandlung ist nicht im Detail beschrieben.