

Assignment 2

Abgabe bis zum 16. April 2023 um 23:59 Uhr

Geben Sie Ihr Assignment bis zum 16. April 2023 um 23:59 Uhr auf folgender Webseite ab:

<https://assignments.hci.uni-hannover.de>

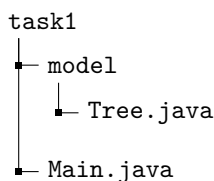
Ihre Abgabe muss aus einer einzelnen zip-Datei bestehen, die den Quellcode und ggf. ein PDF-Dokument bei Freitextaufgaben enthält. Beachten Sie, dass Dateien mit Umlauten im Dateinamen nicht hochgeladen werden können. Entfernen Sie die daher vor der Abgabe die Umlaute aus dem Dateinamen. Überprüfen Sie nach Ihrer Abgabe, ob der Upload erfolgreich war. Bei technischen Problemen, wenden Sie sich an Kevin Schumann oder Dennis Stanke. Es wird eine Plagiatsüberprüfung durchgeführt. Gefundene Plagiate führen zum Ausschluss von der Veranstaltung. In dieser Veranstaltung wird ausschließlich die Java/JavaFX Version 17 verwendet. Code und Kompilate anderer Versionen sind nicht zulässig. Ihre Abgaben sollen die vorgegebenen Dateinamen verwenden. Wiederholter Verstoß gegen diese Regel kann zu Punktabzügen führen.

Aufgabe 1: Binärer Suchbaum

Innerhalb dieser Aufgabe werden Sie eine Klasse, die einen binären Suchbaum repräsentiert, implementieren. Auch werden dafür typische Methoden realisiert.

Ein binärer Suchbaum ist ein Binärbaum, bei dem der linke Teilbaum stets kleinere Werte beinhaltet als im aktuellen Knoten. Größere Werte befindet sich im rechten Teilbaum. Des Weiteren sind beide Teilbäume ebenfalls binäre Suchbäume. **Jede Methode soll durch zwei Testfälle innerhalb der Main-Methode in der Main-Klasse getestet werden.**

a) Erstellen Sie zunächst die Klasse `Tree` in dem unten gezeigten Package. Ein `Tree` besitzt einen `Integer-Wert value` und sowohl einen `linken` als auch `rechten Teilbaum (Tree)` als `Attribut`. Implementieren Sie einen `Konstruktor`, der einen `Integer-Wert` unter `value` speichert. Setzen Sie weiter `Getter-` für alle Attribute und `Setter-Methoden` für den linken und rechten Teilbaum um.



b) Zur Ausgabe auf der Kommandozeile soll die Funktion `public String str()` realisiert werden. `str()` generiert den String *in-order*. Beim In-Order-Durchlauf wird erst der linke Teilbaum durchlaufen. Anschließend folgt die Wurzel und schließlich der rechte Teilbaum. Somit sind die Zahlen in dem resultierende String bei einem binären Suchbaum sortiert. (Siehe Figure 1 für ein Beispiel.

Strings können wie folgt konkateniert werden:

```
1 String a = "Hello";
2 String b = "World";
3
4 // result entspricht "Hello World"
5 String result = a + " " + b;
```

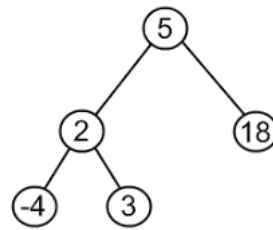


Figure 1: Binärbaum: Der In-Order-Durchlauf ist: -4 2 3 5 18

c) Implementieren Sie die Methode `public void insertValue(int value)`. Diese Methode nimmt den Integer-Wert *value* entgegen und fügt diesen in den binären Suchbaum ein, sodass die genannten Eigenschaften weiterhin erfüllt sind. Ist der Wert bereits vorhanden, soll kein neuer Knoten hinzugefügt werden.

d) Implementieren Sie die Methode `public boolean contains(int value)`. Falls der binäre Suchbaum den Wert *value* enthält, gibt die Methode *true* zurück anderenfalls *false*.

Aufgabe 2: Marketplace

In dieser Aufgaben sollen Sie einen Marketplace (bspw. das schwarze Brett auf StudIP) erstellen. Dies geschieht inkrementell über mehrere Übungsblätter. Es werden dabei neue Vorlesungskonzepte eingearbeitet.

a) Machen Sie sich mit der Namenskonvention für Java-Packages vertraut. Legen Sie anschließend die Packagestruktur für die URL <https://www.uni-hannover.de> und dem Packagenamen *task2* an. In den folgenden Teilaufgaben werden Sie darin weitere Subpackages anlegen.

b) Erstellen Sie die Klasse *User* im Subpackage *auth*. Die User-Klasse speichert:

- Benutzername : String
- Passwort : String
- Items : Item[]

Erstellen Sie sinnvollen Getter- und Setter-Methoden, sowie einen Konstruktor, der einen User mit einem Namen und Passwort initialisiert. Weiter sollen User in der Lage sein, Items (siehe Teilaufgabe b) zu entfernen oder hinzuzufügen. Implementieren Sie hierfür die Methoden `public boolean removeItem(Item item)` und `public boolean addItem(Item item)`. Beide Methoden geben nur dann *true* zurück falls das Hinzufügen oder Entfernen möglich ist.

c) Nun implementieren Sie die Klasse *Item* im Subpackage *offerings*. Die Klasse soll folgende Attribute haben:

- Name : String
- Verkaufspreis : Float
- Verkäufer : User

- Beschreibung : String

Implementieren Sie für alle Attribute Getter- und Setter-Methoden als auch einen Konstruktor, der das Item mit allen Attributen initialisiert.

d) Erstellen Sie die Klasse `Marketplace` im Package `task2`. Diese Klasse speichert ausschließlich bis zu 10 User, deren Benutzernamen einzigartig sind. Das Hinzufügen von Nutzern wird durch die Methoden `public boolean ↪ addUser(User user)` ermöglicht.

Implementieren Sie die Methode `public String str()` innerhalb der `Marketplace` Klasse. Diese Funktion gibt alle Items aller User in Form eines Strings zurück. Items sollen in der Form [Item: Tasche; Preis: 10.99; Verkäufer: Max Mustermann; Beschreibung: Eine Tasche] dargestellt werden.

Erstellen Sie innerhalb der Main-Methode einen `Marketplace` und fügen diesem schrittweise einen User und ein Item hinzu, welches Sie anschließend wieder entfernen. Geben Sie nach jedem Schritt den Zustand des `Marketplaces` mittels `public String str()` aus.

Aufgabe 3: Debugging

In diesen Aufgaben werden Sie einen fehlerhaften Java Codeabschnitte bekommen. Diese Fehler können **syntaktisch** oder **semantisch** sein. Jedoch beschreiben Kommentare beschreiben immer das **korrekte** Verhalten des Programms. Sie dürfen den Code kompilieren und ausführen um Fehler zu finden. Arbeiten Sie in den Template-Dateien in der `Debug02.zip`.

a) Der Code enthält 6 Fehler. Kompilieren und führen Sie den Code aus. Suchen Sie anhand der Fehlermeldungen des Compilers oder der Laufzeitfehlermeldungen Fehler im Code und korrigieren Sie diese. Beschreiben Sie am Ende der jeweiligen Zeile, was Sie korrigiert haben.

Erstellen Sie während der Fehlerkorrektur einen Blockkommentar unter dem Code. Schreiben Sie die Zeile(n) des Fehlers auf und beschreiben Sie den Fehler. Kopieren sie die Fehlermeldung, sofern es eine gab, anhand welcher Sie den Fehler erkannt haben. Die Dokumentation soll wie in folgendem Beispiel aussehen:

```
1 public class Debug { //K: class falsch geschrieben (ckass)
2
3 ...
4
5 /*
6 ...
7 Zeile 1: class Keyword falsch geschrieben (ckass):
8 Fehlermeldung:
9 *****
10 Debug.java:1: error: class, interface, or enum expected
11 public ckass Debug {
12     ^
13 *****
14 Der Compiler erwartet eines der drei oben angegebenen Keywords, hat aber nur das falsch
15     ↪ geschriebene ckass bekommen.
16 ...
17 */
```