1 – 2 – 3 – vorbei…

DHBW Karlsruhe, Vorlesung Programmieren I+II, Probe-Programmentwurf

Juli 2021

Bearbeitungszeit: 120 Minuten

Aufgabe

Schreiben Sie eine Java-Anwendung `jBay`, die ein Auktionssystem implementiert!

Teilaufgabe a) **[30%]**

Entwickeln Sie die Klassen `Bieter`, `Ware`, `Gebot`, `Auktion` und `Auktionshaus`!

Ein `Bieter` hat einen Vornamen und einen Nachnamen. Die Methode `getFullName()` soll Vor- und Nachnamen getrennt durch ein Leerzeichen als `String` zurückgeben.

Eine `Ware` hat einen Titel und eine Beschreibung. Implementieren Sie einen Konstruktor `Ware(String titel, String beschreibung)`!

Ein `Gebot` besteht aus einem (Höchst-)Betrag (`double`), den ein `Bieter` in einer Auktion maximal zu zahlen bereit ist, und der Referenz auf diesen `Bieter`.

Eine `Auktion` hat Referenzen auf eine `Ware` und ein `Gebot` (das derzeitige Höchstgebot, anfangs `null`). Außerdem hat sie Attribute für den aktuellen Preis (`double`, anfangs `0.0`) und das Ende der `Auktion` (Typ `Calendar`, *s. Hinweis zu* `Calendar` *am Ende*).
(*Hinweis: Der aktuelle Preis ist im Allgemeinen niedriger als der im Höchstgebot enthaltene Höchstbetrag. Er erhöht sich mit jedem Gebot nach den unten stehenden Regeln.*)
`Auktion` beinhaltet eine statische Konstante `double increment=1.0`. Dabei handelt es sich um den Mindestpreis, den ein erstes Gebot erzielen muss. Auch jedes neue Gebot muss den aktuellen Preis um mindestens dieses `increment` übertreffen.
`Auktion` hat eine Methode `boolean gebotAbgeben(Gebot g)`, die ein neues Gebot entgegennimmt. Wird dieses Gebot `g` zum Höchstgebot, wird `true` zurückgegeben, ansonsten `false`. Für das neue `Gebot g` gelten folgende Regeln:

- Ist der in `g` gebotene Betrag niedriger als (aktueller Preis+`increment`), wird das Gebot abgelehnt.
- Ist `g` das erste Gebot dieser Auktion, wird der aktuelle Preis auf `increment` festgesetzt.*
- Ist das Gebot `g` vom selben Bieter wie das momentane Höchstgebot, wird nur eine Erhöhung des Gebots akzeptiert. Am aktuellen Preis ändert sich jedoch nichts.
- Ist der in `g` gebotene Betrag zwar höher oder gleich (aktueller Preis+`increment`), jedoch geringer oder gleich dem momentanen Höchstgebot, wird der aktuelle Preis auf das Minimum aus dem in `g` gebotenem Betrag+`increment` und momentanem Höchstgebot festgesetzt.
- Ist der in `g` gebotene Betrag höher oder gleich (aktueller Preis+`increment`) und auch höher als das momentane Höchstgebot, wird der aktuelle Preis auf das Minimum aus dem in `g` gebotenem Betrag und (momentanem („alten“) Höchstgebot+`increment`) festgesetzt.*

*) `g` *wird in diesen Fällen neues Höchstgebot.*

Implementieren Sie einen Konstruktor `Auktion(Ware ware, int dauer)`, wobei `dauer` die Dauer der Auktion in Minuten bedeutet (*s. Hinweis zu* `Calendar`).

Ein `Auktionshaus` bietet eine Methode `void addAuktion(Auktion a)`, um eine weitere Auktion hinzuzufügen, sowie eine Methode `void removeAuktion(Auktion a)`, um eine Auktion aus dem Auktionshaus zu entfernen. Die Methode `List<Auktion> getAuktionen()` liefert eine Liste der vorhandenen Auktionen. Es können beliebig viele Auktionen hinzugefügt werden.

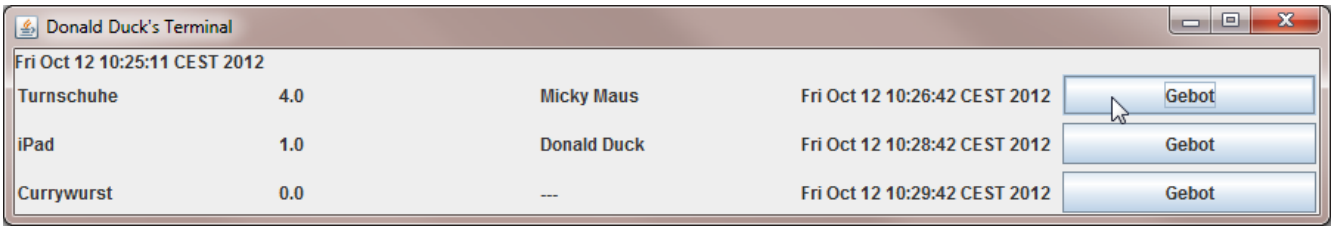
Teilaufgabe b) **[3%]**

Schreiben Sie die Klasse `jBay` mit der folgenden `main`-Methode (*Konstruktor-Aufrufe von* `Ware` *bzw.* `Auktion` *wie in Teilaufgabe a); die Zahlen 2, 4 und 5 bedeuten, dass die jeweilige Auktion in 2 (bzw. 4 bzw. 5) Minuten enden soll.*):

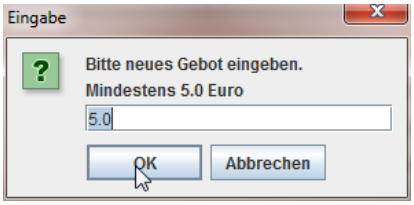
```
public static void main(String[] args) {
        Auktionshaus jbay = new Auktionshaus();
        jbay.addAuktion(new Auktion(
             new Ware("Turnschuhe","Tolle Turnschuhe, kaum getragen"),2));
        jbay.addAuktion(new Auktion(
             new Ware("iPad","Nagelneues iPad 3"),4));
        jbay.addAuktion(new Auktion(
             new Ware("Currywurst","Scharf, ohne Pommes"),5));
        // An dieser Stelle wird in Teilaufgabe c) erweitert
}
```

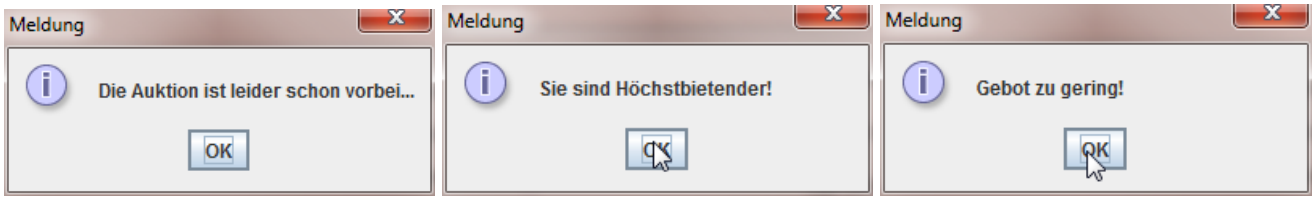
Teilaufgabe c) **[30%]**

Entwickeln Sie eine Klasse `BieterTerminal`, die jeweils eine Bieter-Sitzung implementiert!
Sie soll einen Konstruktor `BieterTerminal(Bieter bieter, Auktionshaus ah)` haben, der ein neues Terminal für einen bestimmten `Bieter` und ein bestimmtes `Auktionshaus` erzeugt.
Das `BieterTerminal`-Fenster soll im Titel den Namen des Bieters, oben die aktuelle Uhrzeit (zunächst zum Zeitpunkt des Öffnens des Fensters) sowie eine Liste der im `Auktionshaus` vorhandenen Auktionen (jeweils mit Titel der Ware, aktuellem Preis, momentan höchstbietendem Bieter („---“ falls noch kein Gebot vorliegt), Ende der Auktion und einem Button „Gebot“) anzeigen (*s. Screenshot*).



Beim Klick auf „Gebot“ soll für diese `Auktion` ein neues `Gebot` eingelesen werden (z.B. mit `JOptionPane.showInputDialog(...)`). Dabei soll (der aktuelle Preis+`increment`) als Betrag vorgeschlagen werden. Falls die `Auktion` schon beendet ist (Aktuelle Zeit>Ende der Auktion), soll stattdessen ein entsprechender Hinweis kommen (s.u.). Nach Übergabe des eingelesenen Betrages an die Methode `gebotAbgeben` der `Auktion`, soll der Nutzer in einer Box mitgeteilt bekommen, ob das Gebot zu gering war (s.u.) oder ob er nun Höchstbietender ist. Ungültige Eingaben sollen abgefangen und wie 0.0 behandelt werden.





Alle an das `Auktionshaus` angeschlossenen `BieterTerminals` sollen dann ihre Anzeige entsprechend anpassen, z.B. den neuen Höchstbietenden und den aktualisierten Preis anzeigen.

Erweitern Sie `Auktionshaus` um Methoden `void register(BieterTerminal bt)` und `void unregister(Bieterterminal bt)`, die ein `BieterTerminal` bei diesem `Auktionshaus` an- bzw. abmelden.

Implementieren Sie eine Methode `void updateTerminals()`, die den `BieterTerminal`s mitteilt, dass sie neu dargestellt werden müssen.

Erweitern Sie die main-Methode von `jBay` um die folgenden Zeilen:

```
BieterTerminal b1 = new BieterTerminal(new Bieter("Micky","Maus"),jbay);
BieterTerminal b2 = new BieterTerminal(new Bieter("Donald","Duck"),jbay);
```

Teilaufgabe d) **[7%]**

Loggen Sie alle Gebote (egal ob erfolgreich oder nicht) in einer Textdatei „`auktionen.txt`“ (im Projekt-Verzeichnis) mit. Neue Logs sollen dabei an die Datei angehängt werden, ohne dass die bisherigen Einträge entfernt werden. Ein Log-Eintrag soll etwa diese Form haben:

```
[Fre Oct 12 10:26:55 CEST 2012] Gebot von Donald Duck für iPad: 1.0 Euro.
```

Teilaufgabe e) **[5%]**

Zeigen Sie in jedem `BieterTerminal`, jeweils gesteuert durch einen Java-Thread, sekündlich die aktuelle Uhrzeit an.

Hinweis zu „Calendar“:

Mit Hilfe der Klasse `java.util.Calendar` lässt sich die aktuelle Uhrzeit abrufen oder eine beliebige Uhrzeit (jeweils inkl. Datum) darstellen.

Eine `Calendar`-Instanz mit der aktuellen Uhrzeit erzeugen:

```
java.util.Calendar zeit = java.util.Calendar.getInstance();
```

Darstellung dieser Zeit als String:

```
zeit.getTime().toString(); // z.B. "Fri Oct 12 10:26:42 CEST 2012"
```

Inhalt von `zeit` auf „in zwei Minuten ab jetzt“ festlegen:

```
int minuten = 2;
zeit.setTimeInMillis(System.currentTimeMillis() + 60000 * minuten);
```

(`System.currentTimeMillis()` liefert die Anzahl der gemäß Systemuhr seit Mitternacht des 1. Januar 1970 vergangenen Millisekunden als `long`, derzeit etwa 1.349 Billionen ms.)

Vergleich von `zeit` mit der aktuellen Uhrzeit:

```
if (zeit.after(java.util.Calendar.getInstance())) { … }
```

Analog dazu gibt es eine Methode `before(…)`

Weitere Hinweise zum Programm:

- In der grafischen Oberfläche und der Log-Datei muss die Anzeige von Geldbeträgen nicht auf zwei Nachkommastellen normiert werden!
- Zur Vereinfachung: Das Betätigen des Schließen-Buttons in einem der `BieterTerminal`s soll jeweils die ganze Anwendung (alle Fenster) schließen.
- Verwenden Sie sprechende (selbsterklärende) Namen für die Variablen etc.
- Die Verwendung von GUI-Buildern ist nicht erlaubt.

Hinweis zur Bewertung:

- 25% der Punkte werden nach Funktionstests Ihrer Lösung vergeben.
- 75% der Punkte werden entsprechend des in den Teilaufgaben angegebenen Schlüssels auf Basis des Quellcodes vergeben.