Lehrstuhl für Angewandte Informatik I
Prof. Dr. Bernhard Westfechtel
Sandra Greiner

Weihnachtsblatt

Tannenbaum – Arrays und Zufallszahlen

Implementieren Sie eine Klasse, deren Methoden einen Weihnachtsbaum zu einer vorgegebenen Zeilenhöhe konstruieren und auf der Standardausgabe ausgeben. Der Weihnachtsbaum soll in Form eines Dreiecks abgespeichert werden, d.h. in einem zweidimensionalen Array von `char`-Werten **mit unterschiedlich langen Zeilen**. Ein Beispiel eines solchen Arrays finden Sie im Handbuch der Java-Programmierung von Guido Krüger im Abschnitt 4.4.3 zum Thema mehrdimensionale Arrays.[1]

Der Weihnachtsbaum soll als Spitze ein 'T' besitzen, die Ränder werden mit Kugeln 'o' belegt, ansonsten kommt immer nach zwei „normalen“ Reihen ('*') eine Reihe Lametta ('w') von der Spitze des Baums beginnend. Die Ausgabe des Weihnachtsbaums soll formatiert erfolgen. Mit der zusätzlichen Ausgabe von Leerzeichen an geeigneten Stellen erreichen Sie, dass Ihre Ausgabe der unten abgebildeten Darstellung entspricht. (Im zweidimensionalen Array selbst sollen keine Leerzeichen abgespeichert werden.)

```
Bitte Höhe des Tannenbaums festlegen:
8
             T
           o * o
         o * * * o
       o w w w w w o
     o * * * * * * * * o
   o * * * * * * * * * * o
 o w w w w w w w w w w w o
o * * * * * * * * * * * * * o
```

(a) Schreiben Sie dafür eine Methode `erstelleTannenbaum()`, die die Höhe übergeben bekommt. Dann legt es das benötigte 2-dimensionale Array an und füllt es mit den entsprechenden Einträgen. Rückgabewert ist der Baum in Form des 2-dimensionalen `char`-Arrays.

(b) Die Ausgabe des Baumes soll in einer separaten Methode `druckeTannenbaum()` erfolgen, der das 2-dimensionale Array übergeben wird.

(c) Schreiben Sie eine `main`-Methode, die die Höhe des Baumes von der Standardeingabe einliest und einen entsprechenden Baum unter Nutzung der beiden oben beschriebenen Methoden auf der Standardausgabe ausgibt.

Mögliche Erweiterungen:

- Implementieren Sie eine Methode, die das Tannenbaumarray übergeben bekommt und nun per Zufall Kerzen reihenweise verteilt.
  - Bestimmen Sie zunächst per Zufall, wie viele Kerzen in einer Reihe platziert werden sollen. Es können maximal alle Nadeln('*') oder Lamettaelemente('w') ersetzt werden.

[1] Ein Link auf die Online-Ausgabe des Buches ist unter http://www.javabuch.de/ zu finden.

– Bestimmen Sie dann per Zufall, an welcher Position die Kerze gesetzt werden soll.

Achten Sie dabei darauf, dass die Kugeln am Rand und die Spitze der Tanne dabei bestehen bleiben. Daher muss für jede Zeile zunächst entschieden werden, wie viele Kerzen maximal darin Platz haben.

- Geben Sie mehrere Tannenbäume (ggf.) von unterschiedlicher Größe aus.
- Im e-Learning findet sich ein Projekt mit dessen Hilfe man **bunte** Weihnachtsbäume zeichnen kann. Lesen Sie sich in die Doku ein

Plätzchen – Aufzählungstypen, Arrays und Klassen mit Struktur, s. auch Blatt 9

Wir erweitern unsere Plätzchen-Aufgabe um die folgende Bestandteile: überprüfen Sie anschließend mit der `Main`-Klasse, die im Elearning bereitgestellt wird.

(a) Die `PlätzchenDose` soll die folgenden weiteren Methoden erhalten:

- `verringere(...)`, der eine Anzahl an Plätzchen übergeben wird, die aus der Dose entnommen wird. Falls mehr entnommen werden soll als vorhanden, muss kein Fehler ausgegeben werden. Der Bestand ist dann lediglich 0.
- `toString()`. Diese Methode wird geerbt von der Klasse `Object` und ist für jede beliebige Klasse vorhanden. Sie gibt **immer** einen String zurück.

  Wir überschreiben (**Redefinieren**) sie, indem wir darin unser gewünschtes Verhalten implementieren. Sie soll einen formatierten String zurückgegeben, der die Eigenschaften der Klasse nennt, also *welche* Sorte von Plätzchen darin enthalten ist, wie viele sich gerade darin befinden und insgesamt Platz haben.

(b) Implementieren Sie eine Klasse `Mensch`. Jeder Mensch kann eine Reihe von Plätzchendosen besitzen. Diese merkt er sich in einem Array mit Platz für fünf verschiedene Dosen.
Ein Mensch verändert den Bestand der Plätzchen, wenn er Plätzchen backt und nascht. Dafür und zum Füllen des Plätzchenarrays benötigen Sie die folgenden Methoden:

- `fuegeEin(...)`, die eine Plätzchendose übergeben bekommt. Diese wird an der nächsten freien Stelle im Array eingefügt. Sollte das Array schon voll sein, wird ein doppelt so großes Array erzeugt und die schon vorhanden Dosen darin gespeichert.
- `backe(...)`, die den Plätzchentyp und die Anzahl, wie viel gebacken wurde, übergeben bekommt. Sie sucht im bestehenden Plätzchenvorrat eine Dose, die den gleichen Typ beinhaltet und füllt sie. Wenn keine passende Dose vorhanden ist oder die Dose dabei voll wird, wird eine neue Plätzchendose angelegt (und mit dem Rest befüllt). Der maximale Inhalt der neuen Dose soll dabei so gewählt werden, dass noch Platz für weitere Plätzchen ist.
- `nasche(...)`, die den Plätzchentyp und die Anzahl übergeben bekommt. Suchen Sie im Array nach einer passenden Dose. Falls danach die Dose leer ist, wird sie aus dem Array entfernt. Achten Sie darauf, dass Sie eventuelle Lücken schließen, wenn Sie den Vorrat in der Mitte des Arrays löschen.
- `toString(...)`, die den gesamten Plätzchenvorrat des Menschen als formatierten String zurückgibt unter Nutzung der `toString()`-Methode der `PlätzchenDose`.

(3+13 = 16 Punkte)

OnlineShop – Klassen und Aufzählungstypen

Wer von Ihnen hat alle Geschenke im Laden gekauft? Mindestens eins wurde mit Sicherheit online bestellt, denn Online-Shops werden immer beliebter.
Erstellen Sie die folgenden typheterogenen Datenstrukturen als Java-Klassen. Als Vorlage bekommen Sie eine Klasse `Main`, die eine main-Methode enthält, die einen Beispielablauf im Onlineshop simuliert. Wählen Sie für alle Komponenten geeignete Datentypen, sodass sie kompatibel mit der vorgegebenen main-Methode sind.

(a) Definieren Sie zunächst die folgenden Datentypen mit geeigneten Konstruktoren und achten Sie auf die Sichtbarkeiten der jeweiligen Felder. Für Felder, deren Wert nicht mehr von außen geändert werden soll, und deren Wert Sie aber in einer anderen Klasse verwenden wollen, schreiben Sie eine `gib...(...)` Methode, z.B. `gibArtikel(...)`, die den Artikel eines Kauf-Objekts zurückgibt:

- Einen Aufzählungstypen `Kategorie` für die Waren des Shops. Mögliche Kategorien sind Haushalt, Kleidung, Elektronik und Multimedia.
- Eine Klasse `Artikel`, der sich aus Name, Beschreibung, Preis und Kategorie zusammensetzt. Sie soll eine Methode `drucke(...)` enthalten, die die Daten des Artikels auf der Konsole ausgibt.
- Eine Klasse `Datum`, das Tag, Monat und Jahr speichert.
- Eine Klasse `Benutzer`, von dem Name, Vorname und die 100 zuletzt gekauften Artikel gespeichert werden.
- Eine Klasse `Kauf`. Instanzen dieser Klasse merken sich einen gekauften Artikel, den Benutzer, die erworbene Stückzahl und das Kaufdatum.

(b) Erweitern Sie die Klasse `Benutzer` um eine Methode `fuegeHinzu(...)`, die den zuletzt erworbenen Artikel übergeben bekommt und an leerer Stelle einfügt. Wenn das Artikel-Array voll ist, werden die am längsten enthaltenen Artikel überschrieben.

(c) Außerdem sind die letzten zwei erworbenen Artikel interessant, um Empfehlungen an andere Benutzer des Onlineshops auszugeben. Schreiben Sie dafür eine Methode `letzteZwei()`, die ein Array mit den 2 zuletzt gekauften Artikeln des Benutzers zurückgibt.

(d) Schreiben Sie nun eine Klasse `OnlineShop`, die 1000 Benutzer speichert und sich sich aus folgenden Objektmethoden zusammensetzt:

- `registriereBenutzer(...)`, die einen Benutzer übergeben bekommt und gemäß der Einfügereihenfolge speichert.
- `kaufeArtikel(...)`: Bekommt einen Artikel, den Benutzer, eine Stückzahl, sowie das Kaufdatum übergeben und erstellt damit ein neues `Kauf`-objekt, das zurückgegeben wird.
- Eine Methode `schreibeEmpfehlung(...)`, die ein Kaufobjekt übergeben bekommt und anhand dessen eine Empfehlung für den Benutzer ausgibt. Sie funktioniert nach dem Prinzip, „Kunden die diese Ware gekauft haben, haben auch ... gekauft“. Entsprechend wird sie am Ende von `kaufeArtikel()` aufgerufen. Die Methode geht so vor:
  Sie prüft, ob ein anderer Benutzer denselben Artikel gekauft hat. Falls dies der Fall ist, werden die letzten zwei erstandenen Artikel ermittelt (`letzteZwei()`) und als

Vorschlag für einen weiteren Kauf ausgegeben (`drucke()`). Danach ist die Methode zu Ende. Falls nicht genügend Artikel vorhanden sind oder keiner der anderen Benutzer diese Artikel gewählt hat, geben Sie eine standardmäßige Empfehlung aus.