SCHOOL-SCOUT.DE

Unterrichtsmaterialien in digitaler und in gedruckter Form

Auszug aus:

Klausur Informatik zum Thema Datenstrukturen

Das komplette Material finden Sie hier:

School-Scout.de

Klausur zur Informatik in der Qualifikationsphase

Kurzvorstellung des Materials	Beim vorliegenden Material handelt es sich um eine Klausur zur Informatik in der Qualifikationsphase der gymnasialen Oberstufe NRW zum Thema Datenstrukturen. Sie kam in der Praxis bereits zum Einsatz. Beim Erstellen der Klausur wurde sehr darauf geachtet, alle Aufgabenstellungen in einen Sachzusammenhang zu stellen, wie dies auch im Zentralabitur NRW vorgesehen ist. Sie eignet sich daher sehr, die Schülerinnen und Schüler auf die zentralen Abiturprüfungen vorzubereiten. Die Klausur orientiert sich inhaltlich an dem Arbeitsbuch zu Datenstrukturen, kann aber auch unabhängig von diesem eingesetzt werden.
Inhaltliche Schwerpunkte	Datenstrukturen Queue und Stack
Aufgabenarten	Anwendung von Datenstrukturen, Wahl geeigneter Datenstrukturen, Modellierung einer Situation mithilfe von Datenstrukturen, Implementierung einzelner Methoden, Interpretation einer Implementierung
Inhaltliche Voraussetzungen	• Grundlagen der objektorientierten Modellierung • Datenstrukturen Array, Queue und Stack • Explizites Casten von Objekten
Dauer	2 Unterrichtsstunden
Schlüsselwörter	Klausur, Informatik, Qualifikationsphase, Java, Datenstrukturen, Queue, Stack, Warteschlange, Stapel

Aufgabe 2 (43 Punkte)**.** Es soll das Lager eines Schuhgeschäftes modelliert werden. Zur Vereinfachung nehmen wir an, dass es in diesem nur zwei Stapel von Schuhkartons gibt. Auf dem einen Stapel werden Kartons mit braunen Schuhen, auf dem anderen Kartons mit schwarzen Schuhen gestapelt. Dabei wird die folgende Klasse **`Schuhkarton`** verwendet, deren Implementierung in der Anlage zu finden ist.

Schuhkarton

`-groesse: double`
`-preis: double`
`-Farbe: String`

`+Schuhkarton(pGroesse:double,pPreis:double,`
`pFarbe:String)`
`+getGroesse(): double`
`+getPreis(): double`
`+getFarbe(): String`
`+gibInfo()`

Die Schuhkartons sollen auf den beiden Stapeln jeweils immer der Größe nach sortiert gestapelt werden. Dabei sollen die Größen von oben nach unten innerhalb der Stapel zunehmen, d.h. ganz unten sind die jeweils größten Schuhe.

(a) Unter Verwendung der Datenstruktur Stack wurde die Klasse **`Schuhlager`** implementiert. Diese Implementierung ist ebenfalls in der Anlage zu finden. Sie soll die oben beschriebene Situation modellieren.
Erläutere Vor- und Nachteile der Datenstruktur Stack in dieser Situation. (7 Punkte)

(b) An verschiedenen Stellen der Implementierung der Klasse **`Schuhlager`**, wie etwa in Zeile 51, findet sich die Anweisung `(Schuhkarton)`.
Erkläre die Bedeutung dieser Anweisung und warum sie an dieser Stelle der Implementierung nötig ist. (6 Punkte)

(c) In einer Testklasse werden die folgenden Anweisungen gegeben.

```
1 Schuhlager meinLager = new Schuhlager();
2
3 meinLager.nimmAuf(new Schuhkarton(42, 89.95, "schwarz"));
4 meinLager.nimmAuf(new Schuhkarton(44, 109.95, "schwarz"));
5
6 meinLager.gibInfo();
```

Ermittle, zu welchen Ausgaben in der Konsole diese Anweisungen führen. (10 Punkte)

(d) *Erläutere im Detail die Implementierung der Methode* ***`sortiereInStapelEin(Stack pStapel, Schuhkarton pNeuer)`*** *der Klasse* **`Schuhlager`**. (10 Punkte)

(e) Erscheint eine neue Kollektion von Schuhen, so werden alle Schuhe, die bisher im Lager sind, auf einen einzigen Stapel sortiert. Ihre Farbe wird also nicht mehr berücksichtigt, jedoch wird weiterhin darauf geachtet, dass sie der Größe nach sortiert sind. Die Klasse **`Schuhlager`** soll daher eine neue Methode `sortiereAufHilfsstapel()` erhalten, die alle Schuhkartons der beiden Stapel `brauneSchuhe` und `schwarzeSchuhe` auf den Stapel `Hilfsstapel` legt. Die Schuhkartons sollen dabei in absteigender Reihenfolge sortiert sein, d.h. die kleinsten Schuhe sind anschließend ganz unten. Die beiden Stapel `brauneSchuhe` und `schwarzeSchuhe` sollen nach Ausführung der Methode leer sein.
Modelliere detailliert ein mögliches Vorgehen einer solchen Methode. Eine Implementierung ist nicht nötig, darf aber zur Verdeutlichung (auch auszugsweise) angegeben werden. Achte bei der Darstellung genau darauf, ob und wann ein Schuhkarton aus einem Stapel entfernt oder auch auf einen gelegt wird. (10 Punkte)

```
20    private void sortiereInStapelEin(Stack pStapel, Schuhkarton pNeuer){
21       while (pStapel.isEmpty()==false && ((Schuhkarton)pStapel.top()).
             getGroesse() < pNeuer.getGroesse()){
22          hilfsstapel.push(pStapel.top());
23          pStapel.pop();
24       }
25
26       pStapel.push(pNeuer);
27
28       while (hilfsstapel.isEmpty()==false){
29          pStapel.push(hilfsstapel.top());
30          hilfsstapel.pop();
31       }
32    }
33
34    public void gibInfo(){
35       if (schwarzeSchuhe.isEmpty()==false){
36          System.out.println("Schwarze Schuhe:");
37          gibStapelAus(schwarzeSchuhe);
38       } else{
39          System.out.println("Es gibt keine schwarzen Schuhe.");
40       }
41       if (brauneSchuhe.isEmpty() == false){
42          System.out.println("Braune Schuhe:");
43          gibStapelAus(brauneSchuhe);
44       } else{
45          System.out.println("Es gibt keine braunen Schuhe.");
46       }
47    }
48
49    private void gibStapelAus(Stack pStapel){
50       while (pStapel.isEmpty() == false){
51          ((Schuhkarton)pStapel.top()).gibInfo();
52          System.out.println();
53          hilfsstapel.push(pStapel.top());
54          pStapel.pop();
55       }
56       while (hilfsstapel.isEmpty() == false){
57          pStapel.push(hilfsstapel.top());
58          hilfsstapel.pop();
59       }
60    }
61 }
```

gelegt. Eine while-Schleife wird danach solange durchlaufen, wie der Hilfsstapel nicht leer ist. Bei jedem Durchlauf wird der oberste Karton aus dem Hilfsstapel auf den angegebenen Stapel gelegt und dann aus dem Hilfsstapel entfernt.

(e) Ein mögliches Vorgehen:

- Solange nicht beide Stapel leer sind, wiederhole die folgenden Schritte, wobei stets genau einer dieser drei Fälle zu behandeln ist (d.h. es wird „else if“ verwendet):
  - Falls `brauneSchuhe` leer ist, lege den obersten Schuhkarton aus `schwarzeSchuhe` auf `hilfsstapel` und entferne ihn dann aus `schwarzeSchuhe`.
  - Falls `brauneSchuhe` nicht leer ist, aber `schwarzeSchuhe` leer ist, lege den obersten Karton aus `brauneSchuhe` auf `hilfsstapel` und entferne ihn dann aus `brauneSchuhe`.
  - Falls keiner der beiden Stapel leer ist, vergleiche die Größen des obersten Kartons der braunen Schuhe und des obersten Kartons der schwarzen Schuhe. Lege den Karton mit der kleineren Größe auf `hilfsstapel` und entferne ihn aus dem ursprünglichen Stapel. (Sollten beide gleich groß sein, kann standardmäßig z.B. immer der schwarze umgelegt werden).

SCHOOL-SCOUT.DE

Unterrichtsmaterialien in digitaler und in gedruckter Form

Auszug aus:

Klausur Informatik zum Thema Datenstrukturen

Das komplette Material finden Sie hier:

School-Scout.de