**Aufgabe: Kunden-Bestand durchsuchen**

Ein Kundenbestand sei wie folgt definiert:

Name, alphabetisch, 30  
Vorname, alphabetisch, 30  
Straße, alphabetisch, 50  
Haus-Nr., alphanumerisch, 8  
PLZ, alphanumerisch, 10  
Ort, alphanumerisch, 50  
Telefon-Nr., alphanumerisch, 50  
Mobil-Nr., alphanumerisch, 50  
E-Mail-Adresse, alphanumerisch, 50

1. Bitte erstellen Sie zunächst einen Test-Bestand in Form einer einfachen Textdatei, bei der die einzelnen Datenfelder durch Semikolons getrennt sind. In jeder Zeile befindet sich ei Datensatz. Z.B.:

Mouse;Mickey;Erpelweg;3;25435;Entenhausen;(0355) 4711; (0163) 52 52 52;mickey.mouse@netenhausen.com

Donald;Duck;Gansgasse;42;25356;Entenhausen;(0355) 1234; (0162) 12 12 12;donald.duck@duckor.co.en

Die Datei soll mindestens 10 Datensätze enthalten.

1. Modellieren Sie einen Kunden gemäß der oben angegeben Beschreibung als Java-Klasse.
2. Erstellen Sie ein Java-Programm, welches den Test-Bestand aus der Textdatei einliest und je Datensatz ein neues Java-programm der Klasse Kunde erzeugt. Legen Sie die Java-Objekte in einem Array mit dem Namen „kundenliste“ ab.
3. Erweitern Sie das Java-Programm um eine Listen-Ausgabe aller im Array gespeicherten Kunden-Objekte.

**Aufgabe: Kunden-Bestand sortieren**

Nutzen Sie das Ergebnis der letzten Aufgabe „Kunden-Bestand einlesen“.

1. Erweitern Sie das Programm durch eine Sortier-Methode „selectionSort( Kunde [] kundenliste)“

Diese Methode soll die Kundenliste nach dem Nachnamen in aufsteigender Reihenfolge sortieren. Als Sortier-Methode soll der Algorithmus „Sortieren durch direktes Auswählen“ verwendet werden (siehe auch deutsche Wikipedia „selectionSort“)

1. Welche Änderungen müssen Sie durchführen, damit die Sortierung grundsätzlich nach Nachnamen, aber innerhalb von Gruppen gleicher Nachnamen zusätzlich nach Vornamen sortiert sein sollen?

Implementieren Sie die zusätzliche notwendige Logik.

**Aufgabe: Kunden-Bestand durchsuchen**

Nutzen Sie das Ergebnis der letzten Aufgabe „Kunden-Bestand sortieren“.

Sortierte Datenbestände werden nicht nur für eine geordnete List-Ausgabe benötigt, sondern auch, um in großen Datenmengen effizient nach vorgegebenen Einträgen zu suchen.

1. Erweitern Sie das Programm mit einer Methode „searchSequential (Kunde [] kundenListe, String name)“.

Diese Methode soll den ersten Kunden in der Kundenliste zurückgeben, der dem gesuchten Namen entspricht. Benutzen Sie hierzu eine Schleife, die die Kundenliste vom ersten Element an, über das zweite, usw. durchsucht.

1. Erweitern Sie das Programm mit einer Methode „searchBinary ( Kunden [] kundenListe, String name)“.

Auch diese Methode soll den ersten Kunden in der Kundenliste zurückgeben, der dem gesuchten Namen entspricht. Benutzen Sie hierzu die sogenannte „binäre Suche“:

Die Liste wird zunächst in zwei etwa gleich große Teile geteilt. Danach wird festgestellt, ob sich das gesuchte Element im ersten oder im zweiten Teil befindet.

Befindet sich das gesuchte Element im ersten Teil, wird dieser wiederum in zwei Teile aufgeteilt. Befindet es sich im zweiten Teil, ist die Vorgehensweise entsprechend.

Dies passiert so lange, bis nur noch ein Element übrigbleibt.

Hinweis: Der Algorithmus kann rekursiv implementiert werden.

Sie auch auf der deutschen Wikipedia Seite: „Binäre Suche“

1. Vergleichen Sie die beiden Such-Möglichkeiten aus 1. und 2. hinsichtlich ihrer Effizienz. Welche Suchmöglichkeiten benötigt weniger Operationen? Welche weniger Laufzeit?

**Aufgabe 2 (Punktezahl 10) (Objektorientierter Entwurf)**

Ein Lebensmittelladen verkauft verschiedene Früchte und Gemüse, und zwar Äpfel (Granny, Smith, Golden Delicious und Boskop), Birnen (Helenen und Williams), Kohl (Blumenkohl und Rotkohl) und Salat (Kopfsalat und Feldsalat).

* Alle Frischwaren haben einen Sortennamen, ein Verpackungsdatum und einen Preis.
* Beim Gemüse wird der Name des Lieferanten und das Herkunftsland verzeichnet.
* Bei Früchten gibt es verschiedene Güteklassen (erster oder zweiter Klasse, Fallobst …).
* Der Feldsalat hat ein Verfallsdatum (letzter Verkaufstag).

1. Entwerfen Sie eine sinnvolle Klassenhierarchie, indem Sie das Klassendiagramm zeichnen.

1. Geben Sie die entsprechenden Klassendeklarationen in Java an. Ein Datenzugriff von außen soll nur über eine Funktion print\_Etikette möglich sein.
2. Die Lieferung des Großhändlers besteht aus einer menge von Kisten mit Äpfeln, Birnen, Kohl und Salat (nur eine Sorte pro Kiste). Es soll für jede Kiste ein entsprechendes Etikett gedruckt werden können. Geben Sie eine geeignete Klassendeklaration für Lieferung an und implementieren Sie die Funktion print\_Kisten\_Etiketten.

**Aufgabe 3 (Punktzahl 10) Objektorientierter Entwurf**

Ein Musikladen verkauft Blasinstrumente und Seiteninstrumente.

* Alle Instrumente haben einen Namen, einen Hersteller und einen Preis.
* Bei Seiteninstrumente wird die Sorte und die Anzahl der Seiten vermerkt.
* Bei Blasinstrumenten wird das Material vermerkt, aus dem sie hergestellt sind (Holz, Metall, …).

1. Entwerfen Sie eine sinnvolle Klassenhierarchie, indem Sie das Klassendiagramm zeichnen.
2. Geben Sie die entsprechenden Klassendeklarationen in Java an. In jeder Klasse soll ein Datenzugriff von außen nur über eine Funktion print\_Info möglich sein.
3. Implementieren Sie die Funktion print\_Info für jede Klasse.