Cấu trúc dữ liệu cơ bản

Câu 1.

a. Chạy chương trình sau đây (Chương 29, Java How to Program 6th Edition), chú ý xem dữ liệu thuộc nhiều kiểu được chèn vào trong StringBuffer theo định dạng nào.

```
1  // Fig. 29.14: StringBufferInsert.java
2  // StringBuffer methods insert, delete and deleteCharAt.
3
4  public class StringBufferInsert
5  {
6     public static void main( String args[] )
7     {
8        Object objectRef = "hello";
9        String string = "goodbye";
10       char charArray[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
11       boolean booleanValue = true;
12       char characterValue = 'K';
13       int integerValue = 7;
14       long longValue = 10000000;
15       float floatValue = 2.5f; // f suffix indicates that 2.5 is a float
16       double doubleValue = 33.333;
17
18       StringBuffer buffer = new StringBuffer();
19
20        buffer.insert( 0, objectRef );
21        buffer.insert( 0, "  " ); // each of these contains two spaces
22        buffer.insert( 0, string );
23        buffer.insert( 0, "  " );
24        buffer.insert( 0, charArray );
25        buffer.insert( 0, "  " );
26        buffer.insert( 0, charArray, 3, 3 );
27        buffer.insert( 0, "  " );
28        buffer.insert( 0, booleanValue );
29        buffer.insert( 0, "  " );
30        buffer.insert( 0, characterValue );
31        buffer.insert( 0, "  " );
32        buffer.insert( 0, integerValue );
33        buffer.insert( 0, "  " );
34        buffer.insert( 0, longValue );
35        buffer.insert( 0, "  " );
36        buffer.insert( 0, floatValue );
37        buffer.insert( 0, "  " );
38        buffer.insert( 0, doubleValue );
39
40        System.out.printf(
41           "buffer after inserts:\n%s\n\n", buffer.toString() );
42
43        buffer.deleteCharAt( 10 ); // delete 5 in 2.5
44        buffer.delete( 2, 6 ); // delete .333 in 33.333
45
46        System.out.printf(
47           "buffer after deletes:\n%s\n", buffer.toString() );
48     } // end main
49 } // end class StringBufferInsert
```

b. Viết thêm các dòng lệnh thích hợp để:
tìm và xóa hết tất cả các kí tự không phải là chữ cái hoặc chữ số ra khỏi buffer.
c. Tạo một StringBuffer có nội dung tùy ý (có thể gán thẳng trong chương trình thay vì nhập từ bàn phím) rồi nhân đôi nó theo thứ tự đảo ngược để tạo xâu đối gương. Ví dụ: biến "hi there!" thành "hi there!!ereht ih". Sử dụng hai cách: (1) dùng phương thức reverse() có sẵn, (2) dùng vòng lặp tự viết.

Câu 2.
Một trò chơi như sau: một nhóm người đứng thành vòng tròn. Một người hô "CHO". Người đứng liền bên phải hô "MÀY". Người tiếp theo bên phải hô "CHẾT" và ra khỏi vòng tròn. Người tiếp theo bên phải hô "CHO",…trò chơi tiếp diễn cho đến khi vòng tròn chỉ còn một người. Hãy dùng một danh sách thuộc kiểu List<String> để mô phỏng trò chơi này, dùng Iterator khi muốn duyệt quanh vòng tròn, và phương thức remove() khi loại một người chơi ra khỏi vòng. Danh sách này lưu tên của các người chơi theo thứ tự ngược chiều kim đồng hồ. Người đứng đầu danh sách ban đầu sẽ là người hô "CHO" đầu tiên.
Tại mỗi vòng chơi, in danh sách người chơi còn trong vòng để thấy hoạt động của trò chơi. Ví dụ: thông tin thể hiện là tên những người chơi còn đứng trong vòng tròn theo thứ tự ngược chiều kim đồng hồ.
Ví dụ: 7 người chơi {a b c d e f g}
Start: a b c d e f g
1: a b d e f g
2: a b d e g
3: a d e g
4: a d e
5: a d
6: d

a. viết chương trình mô phỏng trên với danh sách người chơi được hard code trong chương trình (viết cố định trong mã chương trình mà không phải nhập)
b. sửa để nhập danh sách người chơi từ bàn phím, tên mỗi người chơi là một xâu kí tự a..z.

File và dòng dữ liệu

Câu 3 *(Exercise 14.8 Java How to Program 6th)***.**

Khi xử lý dữ liệu thương mại, người ta thường cần đến vài file trong mỗi hệ thống. Chẳng hạn trong một hệ thống quản lý tài khoản (account) khách hàng, thường có một tệp master chứa thông tin chi tiết về từng khách hàng như tên, địa chỉ, số điện thoại, số dư nợ của tài khoản (outstanding balance), ngưỡng nợ (credit limit), các hệ số giảm giá (discount terms), hợp đồng, và có thể cả một lịch sử tóm tắt về các giao dịch mua hàng (purchase) và thanh toán (payment) gần đây.

Khi một giao dịch xảy ra (nghĩa là khi hàng được bán và khi tiền thanh toán được gửi đến), thông tin về chúng được nhập vào một file. Định kì với mỗi doanh nghiệp (với một số công ty là hàng tháng, một số khác là hàng tuần, và một số khác là mỗi ngày), file lưu các giao dịch (gọi là "trans.txt") được dùng để cập nhật file master (gọi là "oldmast.txt") về lịch sử giao dịch mua

hàng và thanh toán của mỗi tài khoản. Trong mỗi lần cập nhật, file master được ghi thành file có tên "newmast.txt", file này sau đó sẽ được dùng cho lần cập nhật định kì tiếp theo.

Các chương trình so khớp file phải xử lý một số vấn đề mà không gặp phải trong các chương trình đơn file. Ví dụ: nếu một khách hàng có mặt trong file master nhưng gần đây không thực hiện giao dịch nào, trong file giao dịch sẽ không có thông tin về họ. Ngược lại, một khách hàng mới thực hiện một số giao dịch (do đó có mặt trong file giao dịch) nhưng thông tin về người này chưa kịp có mặt trong file master do là khách hàng mới và công ty chưa kịp tạo bản ghi master về người này.

Viết một chương trình so khớp thông tin tài khoản. Sử dụng số hiệu tài khoản (account number) tại mỗi file làm trường khóa cho bản ghi để so khớp thông tin. Giả thiết rằng mỗi file là một file text tuần tự với các bản ghi được lưu theo thứ tự tăng dần của số hiệu tài khoản.

a. Viết class `TRansactionRecord`. Các đối tượng thuộc lớp này chứa một số hiệu tài khoản và giá trị giao dịch (amount for the transaction). Viết các phương thức để đọc và sửa các giá trị này.
b. Sửa class `AccountRecord` trong Hình. 14.6 để có thêm phương thức `combine`, phương thức này lấy một đối tượng `transactionRecord` object và tính lại số dư nợ của đối tượng `AccountRecord` theo giá trị của đối tượng `transactionRecord`.

```
 1  // Fig. 14.6: AccountRecord.java
 2  // A class that represents one record of information.
 3  package com.deitel.jhtp6.ch14; // packaged for reuse
 4
 5  public class AccountRecord
 6  {
 7     private int account;
 8     private String firstName;
 9     private String lastName;
10     private double balance;
11
12     // no-argument constructor calls other constructor with default values
13     public AccountRecord()
14     {
15        this ( 0, "", "", 0.0 ); // call four-argument constructor
16     } // end no-argument AccountRecord constructor
17
18     // initialize a record
19     public AccountRecord( int acct, String first, String last, double bal
)
20     {
21        setAccount( acct );
22        setFirstName( first );
23        setLastName( last );
24        setBalance( bal );
25     } // end four-argument AccountRecord constructor
26
27     // set account number
28     public void setAccount( int acct )
29     {
30        account = acct;
31     } // end method setAccount
```

```
32
33     // get account number
34     public int getAccount()
35     {
36        return account;
37     } // end method getAccount
38
39     // set first name
40     public void setFirstName( String first )
41     {
42        firstName = first;
43     } // end method setFirstName
44
45     // get first name
46     public String getFirstName()
47     {
48        return firstName;
49     } // end method getFirstName
50
51     // set last name
52     public void setLastName( String last )
53     {
54        lastName = last;
55     } // end method setLastName
56
57     // get last name
58     public String getLastName()
59     {
60        return lastName;
61     } // end method getLastName
62
63     // set balance
64     public void setBalance( double bal )
65     {
66        balance = bal;
67     } // end method setBalance
68
69     // get balance
70     public double getBalance()
71     {
72        return balance;
73     } // end method getBalance
74  } // end class AccountRecord
```

c. Viết một chương trình tạo dữ liệu test. Sử dụng dữ liệu mẫu tại Hình 14.39 và Hình 14.40. Chạy chương trình để tạo các file `TRans.txt` và `oldmast.txt`, chương trình so khớp thông tin tài khoản sẽ dùng đến những file này.

Figure 14.39. Sample data for master file.

Master file Account number	Name	Balance
100	Alan Jones	348.17

Figure 14.39. Sample data for master file.

Master file Account number	Name	Balance
300	Mary Smith	27.19
500	Sam Sharp	0.00
700	Suzy Green	14.22

Figure 14.40. Sample data for transaction file.

Transaction file Account number	Transaction amount
100	27.14
300	62.11
400	100.56
900	82.17

d. Viết class `FileMatch` cung cấp chức năng so khớp file. Lớp này cần chứa các phương thức đọc `oldmast.txt` và `TRans.txt`. Khi gặp hai bản ghi về cùng một tài khoản (các bản ghi có cùng số hiệu tài khoản ở cả file master và file giao dịch), cộng số tiền bằng đô la trong bản ghi giao dịch vào số dư (balance) hiện tại tại bản ghi master, và ghi bản ghi đó vào `"newmast.txt"`. (Giả thiết rằng trong file giao dịch, các giao dịch mua có giá trị dương, còn giao dịch thanh toán có giá trị âm.) Khi gặp một bản ghi master của một tải khoản cụ thể nhưng không có bản ghi giao dịch tương ứng, chỉ cần ghi nguyên trạng bản ghi master đó vào trong `"newmast.txt"`. Khi gặp một bản ghi giao dịch mà không có bản ghi master tương ứng, ghi ra file nhật trình (log file) thông báo `"Unmatched transaction record for account number..."` (điền số hiệu tài khoản từ số hiệu ghi trong bản ghi giao dịch). File nhật trình là file text có tên `"log.txt"`.