Chương 4

Ghi/đọc dữ liệu của ứng dụng Java ra file

7.0 Dẫn nhập
7.1 Tổng quát về đời sống của dữ liệu của ứng dụng Java
7.2 Ghi dữ liệu ra file ở dạng nhị phân
7.3 Đọc dữ liệu từ file ở dạng nhị phân
7.4 Ghi dữ liệu ra file ở dạng text
7.5 Đọc dữ liệu từ file text
7.6 Thí dụ về đọc/ghi dữ liệu cổ điển
7.7 Ghi/Đọc hệ thống đối tượng ra/vào file
7.8 Thí dụ về đọc/ghi hệ thống đối tượng
7.9 Kết chương

4.0 Dẫn nhập

- Chương này giới thiệu các đối tượng phục vụ ghi/đọc dữ liệu ra/vào file dạng nhị phân hay dạng văn bản cùng các tác vụ ghi/đọc dữ liệu cổ điển ra/vào file.
- Chương này cũng giới thiệu các đối tượng phục vụ ghi/đọc hệ thống đối tượng ra/vào file cùng các tác vụ ghi/đọc hệ thống đối tượng có mối quan hệ tham khảo phức tạp ra/vào file.

4.1 Tổng quát về đời sống của dữ liệu ⊂ ứng dụng Java

- Khi chương trình bắt đầu chạy, nó sẽ tạo ra dữ liệu, xử lý dữ liệu cho đến khi hoàn thành nhiệm vụ và dừng chương trình.
- Khi chương trình kết thúc, thường các dữ liệu của chương trình sẽ bị mất. Do đó, trong lúc chương trình chạy, khi cần thiết, ta sẽ ghi các dữ liệu ra file để lưu giữ lâu dài. Sau này khi cần dùng lại, ta sẽ đọc dữ liệu từ file vào các biến chương trình để xử lý tiếp.

4.1 Tổng quát về đời sống của dữ liệu ⊂ ứng dụng Java

- Java cung cấp nhiều class đối tượng khác nhau để phục vụ việc ghi/đọc dữ liệu theo nhiều cơ chế khác nhau :
  1. Nhập/xuất theo luồng các byte thô (Byte Streams).
  2. Nhập/xuất theo luồng các ký tự thô (Character Streams).
  3. Nhập/xuất theo luồng có đệm (Buffered Streams) hầu giảm thiểu số lần truy xuất I/O thực sự.
  4. Scanning và Formatting hỗ trợ việc nhập/xuất văn bản có định dạng.
  5. Nhập/xuất với Console và các luồng chuẩn (Standard Streams).
  6. Nhập/xuất các biến dữ liệu cơ bản theo luồng Data Streams.
  7. Nhập/xuất các đối tượng theo luồng Object Streams.

4.2 Ghi/đọc dữ liệu ra file theo luồng byte thô

- Cho phép chương trình ghi/đọc tuần tự từng byte dữ liệu từ đầu đến cuối, hệ thống chưa biết nghĩa của từng byte và chưa xử lý gì trong việc phục vụ.
- Class phục vụ việc đọc luồng byte phải là class con của class InputStream, tác vụ cơ bản của class này là read() sẽ đọc 1 byte hiện hành trong luồng byte vào chương trình.
- Tương tự, class phục vụ việc ghi luồng byte phải là con của class OutputStream, tác vụ cơ bản của class này là write() sẽ ghi 1 byte của chương trình ra vị trí hiện hành trong luồng byte.
- Thí dụ 2 class FileInputStream và FileOutputStream phục vụ đọc/ghi luồng byte thô ra file.
- Các class phục vụ đọc/ghi luồng byte thô được đặt trong package java.io.

4.2 Ghi/đọc dữ liệu ra file theo luồng byte thô

- Sau đây là mã nguồn của chương trình Java thực hiện việc copy nội dung file in.txt thành file out.txt :

```
//khai báo dùng các đối tượng truy xuất file trong package java.io
import java.io.*;
//định nghĩa class chương trình
public class CopyBytes {
    //định nghĩa điểm nhập chương trình
    public static void main(String[] args) throws IOException {
        //định nghĩa các biến đối tượng đọc/ghi luồng byte
        FileInputStream in = null;
        FileOutputStream out = null;
        //tiếp tục ở slide kế
```

4.2 Ghi/đọc dữ liệu ra file theo luồng byte thô

```
        try {
            //tạo các đối tượng đọc/ghi luồng byte
            in = new FileInputStream("d:\\in.bin");
            out = new FileOutputStream("d:\\out.bin");
            int c;
            //lặp đọc/ghi từng byte
            while ((c = in.read()) != -1) out.write(c);
            //nếu có file nhập và chưa đóng thì đóng file lại
            if (in != null) in.close();
            //nếu có file xuất và chưa đóng thì đóng file lại
            if (out != null) out.close();
        } catch (Exception e) {
        }
    } //hết hàm main
} //hết class chương trình
```

4.2 Ghi/đọc dữ liệu ra file theo luồng byte thô

- Lưu ý là tác vụ read() chỉ đọc 1 byte từng thời điểm, nhưng kiểu dữ liệu của giá trị trả về là int, nếu đọc được byte dữ liệu, giá trị của byte trả về nằm trong phạm vi 0-255, nếu thất bại tác vụ trả về mã lỗi là -1.
- Tương tự tác vụ write() chỉ ghi từng thời điểm 1 byte, nhưng kiểu tham số là int.
- Trước khi chương trình kết thúc, ta phải đóng các file được truy xuất trước đó lại. Ta gọi tác vụ close() để thực hiện đóng file. Để việc đóng file không bị lỗi, ta phải kiểm tra xem có đối tượng file thực sự không ?

4.3 Ghi/đọc dữ liệu ra file theo luồng ký tự thô

- Cho phép chương trình ghi/đọc tuần tự từng ký tự dữ liệu từ đầu đến cuối. Bên trong chương trình Java, mỗi ký tự được miêu tả theo mã Unicode, khi xuất ra luồng ký tự, máy tự động giải mã thành dạng ký tự được qui định bởi locale hiện hành : có thể là mã ANSI 1 byte (mặc định), mã UTF-8, mã Unicode 2 byte…
- Class phục vụ việc đọc luồng ký tự phải là class con của class Reader, tác vụ cơ bản của class này là read() sẽ đọc 1 ký tự hiện hành trong luồng ký tự vào chương trình.
- Tương tự, class phục vụ việc ghi luồng ký tự phải là con của class Writer, tác vụ cơ bản của class này là write() sẽ ghi 1 ký tự của chương trình ra vị trí hiện hành trong luồng ký tự.
- Thí dụ 2 class FileReader và FileWriter phục vụ đọc/ghi luồng ký tự thô ra file.

4.3 Ghi/đọc dữ liệu ra file theo luồng ký tự thô

- Các class phục vụ đọc/ghi luồng ký tự thô được đặt trong package java.io.
- Sau đây là mã nguồn của chương trình Java thực hiện việc copy nội dung file in.txt thành file out.txt :

```
//khai báo dùng các đối tượng truy xuất file trong package java.io
import java.io.*;
//định nghĩa class chương trình
public class CopyCharacters {
    //định nghĩa điểm nhập chương trình
    public static void main(String[] args) throws IOException {
        //định nghĩa các biến đối tượng đọc/ghi luồng ký tự
        FileReader in = null;
        FileWriter out = null;
        //tiếp tục ở slide kế
```

4.3 Ghi/đọc dữ liệu ra file theo luồng ký tự thô

```
        try {
            //tạo các đối tượng đọc/ghi luồng ký tự
            in = new FileReader("d:\\in.txt");
            out = new FileWriter("d:\\out.txt");
            int c;
            //lặp đọc/ghi từng ký tự
            while ((c = in.read()) != -1) out.write(c);
            //nếu có file nhập và chưa đóng thì đóng file lại
            if (in != null) in.close();
            //nếu có file xuất và chưa đóng thì đóng file lại
            if (out != null) out.close();
        } catch (Exception e) {
        }
    } //hết hàm main
} //hết class chương trình
```

4.3 Ghi/đọc dữ liệu ra file theo luồng ký tự thô

- Lưu ý là tác vụ read() chỉ đọc 1 ký tự từng thời điểm, nhưng kiểu dữ liệu của giá trị trả về là int, nếu đọc được ký tự dữ liệu, mã Unicode (thường là 2 byte) của ký tự được trả về, nếu thất bại tác vụ trả về mã lỗi là -1.
- Tương tự tác vụ write() chỉ ghi từng thời điểm 1 ký tự, nhưng kiểu tham số là int.
- Trước khi chương trình kết thúc, ta phải đóng các file được truy xuất trước đó lại. Ta gọi tác vụ close() để thực hiện đóng file. Để việc đóng file không bị lỗi, ta phải kiểm tra xem có đối tượng file thực sự không ?

4.4 Ghi/đọc dữ liệu ra file theo luồng đệm (Buffered)

- Việc đọc/ghi từng byte hay từng ký tự dùng luồng byte hay luồng ký tự là hoàn toàn thô, chưa được đệm, nghĩa là mỗi lần chương trình gọi tác vụ read/write, máy sẽ truy xuất file trực tiếp trên đĩa để đọc/ghi liền dữ liệu.
- Hoạt động I/O thường chậm hơn nhiều so với việc CPU thực hiện lệnh, để cải tiến độ hiệu quả của việc đọc/ghi dữ liệu, Java đã định nghĩa các class phục vụ đọc/ghi byte/ký tự theo cơ chế đệm như sau : mỗi lần chương trình gọi hàm read() để đọc 1 byte/ký tự, máy sẽ truy xuất file trên đĩa và đọc 1 chuỗi byte với độ lớn xác định vào vùng nhớ RAM đệm để phục vụ nhanh cho các tác vụ read() kế tiếp của chương trình. Tương tự, mỗi lần chương trình gọi hàm write() để ghi 1 byte/ký tự, máy chỉ chứa byte/ký tự này vào vùng nhớ RAM đệm chứ không ghi ngay ra file, chỉ khi bộ đệm đầy (hay do yêu cầu cụ thể) máy mới ghi vùng RAM đệm ra file.

4.4 Ghi/đọc dữ liệu ra file theo luồng đệm (Buffered)

- Các class phục vụ việc đọc/ghi dùng bộ đệm có tiếp đầu ngữ "Buffered", cụ thể 2 class đọc/ghi luồng byte có đệm là BufferedInputStream và BufferedOutputStream, 2 class đọc/ghi luồng ký tự có đệm là BufferedReader và BufferedWriter.
- Theo góc nhìn của người lập trình, việc dùng các class đọc/ghi dữ liệu có đệm cũng giống y như dùng các class đọc/ghi dữ liệu không đệm. Ta chỉ cần lưu ý 1 số khác biệt :
  - class phục vụ ghi (output) dữ liệu đệm có thêm tác vụ flush() để người lập trình có thể yêu cầu hệ thống ghi tường minh bộ nhớ RAM đệm ra file tại những thời điểm đặc biệt để đảm bảo mọi dữ liệu mà chương trình đã yêu cầu ghi trước đó đều thực sự được ghi ra file.

4.4 Ghi/đọc dữ liệu ra file theo luồng đệm (Buffered)

- class BufferedReader có thêm tác vụ readLine() để đọc 1 dòng văn bản (được kết thúc bởi ký tự CR và/hoặc LF)
- class BufferedWriter có thêm tác vụ write(String) để ghi 1 chuỗi văn bản, tác vụ newLine() để ghi thêm ký tự kết thúc dòng (CR và/hoặc LF) hầu tạo 1 dòng văn bản.

❑ Thí dụ, ta hãy dùng 2 class BufferedReader và BufferedWriter để viết lại chương trình copy file văn bản trong mục 4.3.

4.4 Ghi/đọc dữ liệu ra file theo luồng đệm (Buffered)

- Sau đây là mã nguồn của chương trình Java thực hiện việc copy nội dung file in.txt thành file out.txt :

```
//khai báo dùng các đối tượng truy xuất file trong package java.io
import java.io.*;
//định nghĩa class chương trình
public class CopyLines {
    //định nghĩa điểm nhập chương trình
    public static void main(String[] args) throws IOException {
        //định nghĩa các biến đối tượng đọc/ghi luồng ký tự đệm
        BufferedReader in = null;
        BufferedWriter out = null;
        //tiếp tục ở slide kế
```

4.4 Ghi/đọc dữ liệu ra file theo luồng đệm (Buffered)

```
        try {
            //tạo các đối tượng đọc/ghi luồng ký tự đệm
            in = new BufferedReader(new FileReader("c:\\in.txt"));
            out = new BufferedWriter(new FileWriter("c:\\out.txt"));
            String i;
            //lặp đọc/ghi từng ký tự
            while ((i = in.readLine()) != null) {
                out.write(i); out.newLine();
            }
            if (in != null) in.close(); //đóng file nhập lại
            if (out != null) out.close(); //đóng file xuất lại
        } catch (Exception e) {
        }
    } //hết hàm main
} //hết class chương trình
```

4.5 Ghi/đọc dữ liệu cơ bản ra/vào file nhị phân

- Trong các mục 4.2, 4.3, 4.4, ta đã giới thiệu qui trình đọc/ghi từng byte hay từng ký tự theo cơ chế đệm hay không đệm. Tuy nhiên dữ liệu của phần mềm thường rất đa dạng về kiểu dữ liệu như : số nguyên, số thực, chuỗi văn bản, trị luận lý, ngày tháng,… Trong mục này, ta sẽ biết ý tưởng và qui trình đọc/ghi dữ liệu thuộc các kiểu cơ bản ra/vào file theo đúng định dạng nhị phân bên trong chương trình, chứ không bị giải mã/mã hóa trong việc ghi/đọc.
- Class phục vụ việc đọc dữ liệu thuộc các kiểu cơ bản ở dạng nhị phân phải là class hiện thực interface DataInput, các tác vụ cơ bản của class này là readChar(), readInt(), readDouble(), readUTF()...
- Tương tự, class phục vụ việc ghi dữ liệu thuộc các kiểu cơ bản ở dạng nhị phân phải là class hiện thực interface DataOutput, tác vụ cơ bản của class này là writeChar(), writeInt(), writeDouble(), writeChars(), writeUTF()...

4.5 Ghi/đọc dữ liệu cơ bản ra/vào file nhị phân

- 2 class phục vụ đọc/ghi dữ liệu nhị phân được dùng phổ biến nhất là DataInputStream và DataOutputStream. 2 class này cũng nằm trong package java.io
- Qui trình điển hình để ghi dữ liệu trong chương trình ra file ở dạng nhị phân (không giãi mã dữ liệu) :

```
//1. tạo đối tượng quản lý file xuất
FileOutputStream fout = new FileOutputStream(path);
//2. tạo đối tượng quản lý file xuất có đệm (nếu cần hiệu quả)
BufferedOutputStream bouts = new BufferedOutputStream(fout);
//3. tạo đối tượng ghi file nhị phân
DataOutputStream out = new DataOutputStream(bouts);
```

4.5 Ghi/đọc dữ liệu cơ bản ra/vào file nhị phân

```
//4. xử lý dữ liệu theo yêu cầu chương trình
int i = -15;
double d = -1.5;
String s = "Nguyễn Văn Hiệp";
Boolean b = true;
//5. ghi dữ liệu ra file
out.writeBoolean(b);      //ghi trị luận lý
out.writeInt(i);          //ghi trị nguyên 32 bit
out.writeDouble(d);       //ghi trị thực 64 bit
out.writeUTF(s);          //ghi chuỗi theo cách mã hóa UTF-8
//6. đóng các đối tượng được dùng lại
out.close();
bouts.close();
fout.close();
```

4.5 Ghi/đọc dữ liệu cơ bản ra/vào file nhị phân

- Qui trình điển hình để đọc dữ liệu từ file nhị phân vào các biến chương trình (không mã hóa dữ liệu) :

```
//1. tạo đối tượng quản lý file nhập
FileInputStream fin = new FileInputStream(path);
//2. tạo đối tượng quản lý file nhập có đệm (nếu cần hiệu quả)
BufferedInputStream bins = new BufferedInputStream(fin);
//3. tạo đối tượng đọc file nhị phân
DataInputStream in = new DataInputStream(bins);
```

4.5 Ghi/đọc dữ liệu cơ bản ra/vào file nhị phân

```
//4. định nghĩa các biến theo yêu cầu chương trình
int i;
double d;
String s;
Boolean b;
//5. đọc dữ liệu từ file vào các biến
b= in.readBoolean();     //đọc trị luận lý
i = in.readInt();        //đọc trị nguyên 32 bit
d = in.readDouble();     //đọc trị thực 64 bit
s = in.readUTF();        //đọc chuỗi UTF-8
//6. đóng các đối tượng được dùng lại
in.close();
bins.close();
fin.close();
```

4.6 Ghi/đọc dữ liệu cơ bản ra/vào file text

- Mặc dù việc ghi/đọc dữ liệu ra file ở dạng nhị phân (y như trong máy) là rất đơn giản, hiệu quả (khỏi phải thực hiện mã hóa/giải mã dữ liệu). Tuy nhiên, file nhị phân cũng có 1 số nhược điểm :
  - người dùng khó xem, khó kiểm tra nội dung của file.
  - người dùng khó tạo dữ liệu dưới dạng nhị phân để chương trình đọc vào xử lý.
- Trong trường hợp cần nhập nhiều thông tin cho chương trình, ta không thể dùng các đối tượng giao diện như textbox, listbox. Trong trường hợp này, ta sẽ dùng trình soạn thảo văn bản để soạn dữ liệu dưới dạng văn bản hầu xem/kiểm tra/sửa chữa dễ dàng. File văn bản chứa dữ liệu là danh sách gồm nhiều chuỗi, mỗi chuỗi miêu tả 1 dữ liệu (luận lý, số nguyên, số thực, chuỗi,...), các chuỗi sẽ được ngăn cách nhau bởi 1 hay nhiều dấu ngăn. Dấu ngăn thường dùng là ký tự gióng cột TAB, ký tự xuống hàng.

4.6 Ghi/đọc dữ liệu cơ bản ra/vào file text

- Class phục vụ việc đọc các chuỗi và mã hóa thành dữ liệu thuộc các kiểu cơ bản có tên là Scanner thuộc package java.util, các tác vụ cơ bản của class này là nextBoolean(), nextChar(), nextInt(), nextDouble(), nextLine()...
- Tương tự, class phục vụ việc ghi dữ liệu thuộc các kiểu cơ bản ra dạng chuỗi phải là class hiện thực interface PrintWriter, tác vụ cơ bản của class này là print() sẽ giải mã từ 0 tới n dữ liệu thuộc các kiểu cơ bản và xuất ra dạng chuỗi tương đương.

4.6 Ghi/đọc dữ liệu cơ bản ra/vào file text

- Qui trình điển hình để ghi dữ liệu trong chương trình ra file ở text (có kèm giải mã dữ liệu) :

```
//1. tạo đối tượng quản lý file xuất
FileOutputStream fout = new FileOutputStream(path);
//2. tạo đối tượng quản lý file xuất có đệm (nếu cần hiệu quả)
BufferedOutputStream bouts = new BufferedOutputStream(fout);
//3. tạo đối tượng ghi file text
PrintWriter out = new PrintWriter(bouts);
```

4.6 Ghi/đọc dữ liệu cơ bản ra/vào file text

```
//4. xử lý dữ liệu theo yêu cầu chương trình
int i = -15;
double d = -1.5;
String s = "Nguyễn Văn Hiệp";
Boolean b = true;
//5. giải mã và ghi dữ liệu ra file
out.print(b); out.print(", ");     //xuất trị luận lý và dấu ngăn
out.print (i); out.print(", ");    //xuất số nguyên và dấu ngăn
out.print(d); out.println(", ");   //xuất số thực và dấu ngăn rồi xuống hàng
out.println(s);                    //xuất chuỗi rồi xuống hàng
//6. đóng các đối tượng được dùng lại
out.close();
bouts.close();
fout.close();
```

4.6 Ghi/đọc dữ liệu cơ bản ra/vào file text

- Qui trình điển hình để đọc chuỗi từ file text, mã hóa và chứa vào các biến chương trình :

```
//1. tạo đối tượng quản lý file nhập ký tự
FileReader fin = new FileReader(path);
//2. tạo đối tượng quản lý file nhập ký tự có đệm (nếu cần hiệu quả)
BufferedReader bins = new BufferedReader(fin);
//3. tạo đối tượng đọc file text và mã hóa thành các dữ liệu cơ bản
Scanner in = new Scanner(bins);
//4. khai báo dấu ngăn các thành phần text trong file
in.useDelimiter(",\\s*");
```

4.6 Ghi/đọc dữ liệu cơ bản ra/vào file text

```
//4. định nghĩa các biến theo yêu cầu chương trình
int i;
double d;
String s;
Boolean b;
//5. đọc dữ liệu từ file vào các biến
b = in.nextBoolean();    //đọc trị luận lý
i = in.nextInt();        //đọc trị nguyên 32 bit
d = in.nextDouble();     //đọc trị thực 64 bit
s = in.nextLine();       //đọc chuỗi UTF-8
//6. đóng các đối tượng được dùng lại
in.close();
bins.close();
fin.close();
```

4.7 Ghi/Đọc hệ thống đối tượng ra/vào file

- Đọc/ghi dữ liệu trên các biến thuộc kiểu giá trị (int, double, char,..) rất dễ vì nội dung của các biến này không chứa tham khảo đến các thành phần khác. Ngược lại, việc đọc/ghi nội dung của 1 đối tượng thường rất khó khăn vì đối tượng có thể chứa nhiều tham khảo đến các đối tượng khác và các đối tượng có thể tham khảo vòng lẫn nhau. Để hỗ trợ việc đọc/ghi nội dung của đối tượng, Java đề nghị kỹ thuật "Serialization".
- Để định nghĩa 1 class "serializable", ta chỉ cần dùng mệnh đề implements để khai báo class hiện thực interface Serializable.

```
class MyClass implements Serializable {
   ...
}
```

4.7 Ghi/Đọc hệ thống đối tượng ra/vào file

- Để ghi 1 đối tượng (và toàn bộ các đối tượng mà nó phụ thuộc) ở dạng nhị phân, ta viết template như sau :

```
//1. Định nghĩa đối tượng FileOutputStream file; miêu tả file chứa thông tin.
FileOutputStream file = new FileOutputStream("c:\userInfo.ser");
//2. Định nghĩa đối tượng ObjectOutputStream out; phục vụ việc ghi đối
    tượng.
ObjectOutputStream out = new ObjectOutputStream(file);
//3. Gọi tác vụ out.writeObject() khi cần ghi đối tượng.
out.writeObject(user);
//4. Gọi tác vụ out.flush() để bắt hệ thống lưu vật lý dữ liệu lên đĩa.
out.flush();
//5. Gọi tác vụ out.close() để đóng đối tượng out.
out.close(); file.close();
```

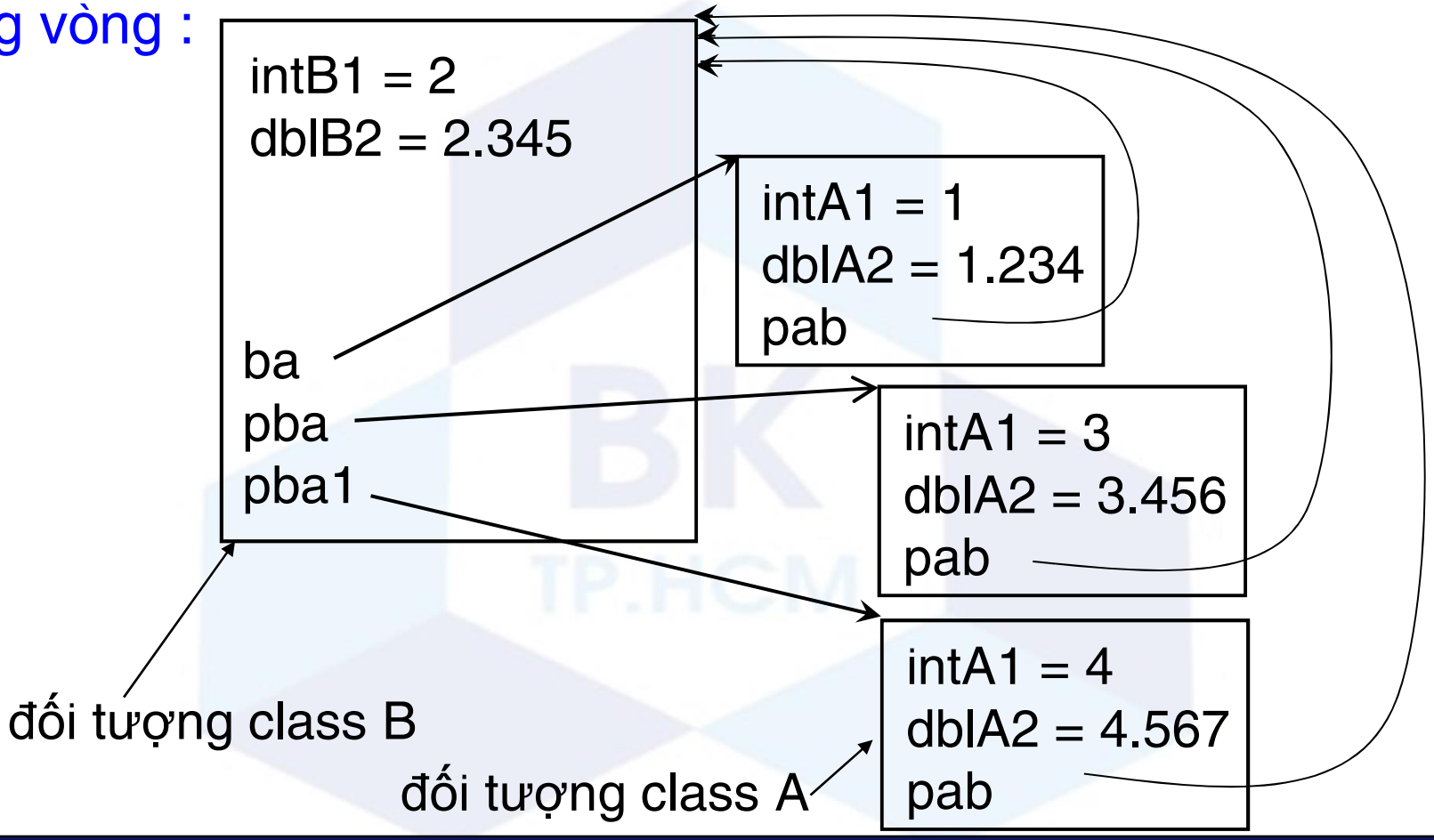
4.7 Ghi/Đọc hệ thống đối tượng ra/vào file

- Để đọc lại 1 đối tượng (và toàn bộ các đối tượng mà nó phụ thuộc) ở dạng nhị phân, ta viết template như sau :

```
//1. Định nghĩa đối tượng FileInputStream file; miêu tả file chứa thông tin.
FileInputStream file = new FileInputStream("c:\userInfo.ser");
//2. Định nghĩa đối tượng ObjectOutputStream out; phục vụ việc ghi đối
    tượng.
ObjectInputStream in = new ObjectInputStream(file);
//3. Gọi tác vụ out.readObject() khi cần ghi đối tượng.
obj = in.readObject();
//4. Gọi tác vụ in.close() để đóng đối tượng in.
in.close(); file.close();
```

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

- Giả sử ta có hệ thống các đối tượng với trạng thái và mối quan hệ giữa chúng cụ thể như sau. Lưu ý chúng có mối quan hệ bao gộp dạng vòng :



4.8 Thí dụ về đọc/ghi hệ thống đối tượng

- Giả sử biến b đang tham khảo tới đối tượng class B. Hãy viết chương trình ghi hệ thống đối tượng này lên file để khi cần, đọc lại vào bộ nhớ hầu xử lý tiếp.
- Qui trình xây dựng ứng dụng giải quyết yêu cầu trên như sau :

1. Chạy NetBean 7.0.1, nếu cửa sổ Project có hiển thị các Project cũ hãy đóng chúng lại.
2. Chọn menu File.New Project để máy hiển thị cửa sổ "New Project", chọn mục "Java" trong Listbox Categories, chọn mục "Java Application" trong Listbox Projects rồi click button Next để hiển thị cửa sổ "New Application".

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

3. Xác định thư mục chứa Project ở textbox "Project Location", nhập "RWObject" vào textbox "Project Name", click button Finish để máy tạo thực sự Project. Cửa sổ soạn mã nguồn của class chương trình có tên là RWObject hiển thị.
4. Viết code cho hàm main của class chương trình như sau :

```
public static void main(String[] args) {
    Create_SaveObject();
};
```

5. Viết thêm 2 tác vụ Create_SaveObject() và ReadObject() như sau:

```
public static void Create_SaveObject() {
    //khởi tạo đối tượng b theo lược đồ trên
   B b = new B();
   b.init(2,2.345);
```

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
    b.Setba(1,1.234,b);
    b.Setpba(3,3.1416,b);
    b.Setpba1(4,4.567,b);
     //ghi đối tượng b
    try {
        FileOutputStream file = new FileOutputStream("d:\\persist.bin");
        ObjectOutputStream out = new ObjectOutputStream(file);
        out.writeObject (b);
        out.flush();
        out.close();
        file.close();
    } catch ( java.io.IOException IOE) {
        System.out.print("IOException");
    }
}
```

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
public static void ReadObject() {
      try {
        FileInputStream file = new FileInputStream("d:\\persist.bin");
        ObjectInputStream input = new ObjectInputStream(file);
        B b = (B) input.readObject ();
        input.close();
        file.close();
      } catch ( java.io.IOException IOE) {
        System.out.print("IOException");
      }
      catch (ClassNotFoundException cnfe) {
        System.out.print("ClassNotFoundException");
      }
   }
```

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

6. Dời chuột về đầu file mã nguồn, thêm lệnh import sau vào ngay dưới lệnh package :

```
import java.io.*;
```

7. dời chuột về mục rwobject trong cây Project, ấn kép chuột vào nó để hiển thị menu lệnh, chọn option New.Java Class để hiển thị cửa sổ "New Java Class". Nhập tên “A” vào textbox "Class name' rồi click button Finish để máy tạo thực sự class mới theo yêu cầu. Cửa sổ soạn mã nguồn của class được hiển thị, hãy soạn nội dung của class như sau :

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
public class A implements Serializable {
    private int intA1;
    private double dblA2;
    private B pab;
    public A() {}
    public void init(int a1, double a2, B p) {
        this.intA1 = a1;
        this.dblA2 = a2;
        this.pab = p;
    }
}
```

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

6. Dời chuột về đầu file mã nguồn, thêm lệnh import sau vào ngay dưới lệnh package :

```
import java.io.*;
```

7. dời chuột về mục rwobject trong cây Project, ấn kép chuột vào nó để hiển thị menu lệnh, chọn option New.Java Class để hiển thị cửa sổ "New Java Class". Nhập tên “B” vào textbox "Class name' rồi click button Finish để máy tạo thực sự class mới theo yêu cầu. Cửa sổ soạn mã nguồn của class được hiển thị, hãy soạn nội dung của class như sau :

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
public class B implements Serializable {
    private   int intB1;
    private   double dblB2;
    private   A ba;
    private  A pba;
    private  A pba1;
    public B() {
    }
    public void init(int b1, double b2) {
       this.intB1 = b1;
       this.dblB2 = b2;
       ba = new A();
       pba = new A();
       pba1 = new A();
    }
```

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
    public void Setba (int a1, double a2, B b) {
       this.ba.init (a1,a2,b);
    }

    public void Setpba (int a1, double a2, B b) {
       this.pba.init (a1,a2,b);
    }

    public void Setpba1 (int a1, double a2, B b) {
       this.pba1.init (a1,a2,b);
    }
}
```

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

10. Dời chuột về đầu file mã nguồn, thêm lệnh import sau vào ngay dưới lệnh package :

    ```
    import java.io.*;
    ```

11. Chọn menu File.Save All để lưu nội dung hiện hành của các file của Project lên đĩa.
12. Chọn menu Run.Run Main Project để dịch và chạy thử ứng dụng. Nếu bạn nhập đúng các đoạn code trên thí ứng dụng sẽ chạy tốt, nó tự tạo hệ thống đối tượng y như slide bài giảng và ghi hệ thống file này lên file d:\persist.bin.
13. Mở lại file RWObject.java, hiệu chỉnh lại thân của hàm main thành :

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

```
public static void main(String[] args) {
    ReadObject();
}
```

14. Dời chuột đến lệnh B b = (B) input.readObject (); trong tác vụ ReadObject(), thiết lập cursor chuột ở lệnh này, chọn menu Debug.Run to cursor để chạy ứng dụng, ứng dụng sẽ dừng ở lệnh do cursor qui định.
15. Dời chuột vào cửa sổ Variable, ấn phải chuột vào hàng "Enter new watch", chọn chức năng "New watch", nhập b vào TextBox rồi click button OK để tạo mục hiển thị nội dung chi tiết của biến này. Lúc này, máy báo chưa có biến này vì hàng lệnh định nghĩa và khởi tạo giá trị biến b chưa được thực thi.

4.8 Thí dụ về đọc/ghi hệ thống đối tượng

16. Ấn phím F8 để máy thị hành lệnh B b = (B) input.readObject (); và dừng lại. Khảo sát lại nội dung hiện hành của biến b, lúc này nội dung trong biến b được thiết lập lại y như trước đây khi nó được ghi ra file bởi phần mềm ghi đối tượng.
17. Xem lại mã nguồn các file đã viết để củng cố kiến thức về lập trình đọc/ghi đối tượng trong Java.

4.9 Kết chương

- Chương này đã giới thiệu các đối tượng phục vụ ghi/đọc dữ liệu ra/vào file dạng nhị phân hay dạng văn bản cùng các tác vụ ghi/đọc dữ liệu cổ điển ra/vào file.
- Chương này cũng đã giới thiệu các đối tượng phục vụ ghi/đọc hệ thống đối tượng ra/vào file cùng các tác vụ ghi/đọc hệ thống đối tượng có mối quan hệ tham khảo phức tạp ra/vào file.