

LUỒNG I/O

Mục tiêu của môn học

Kết thúc chương, bạn có thể :

- Đề cập đến các khái niệm về luồng
- Mô tả các lớp InputStream và OutputStream
- Mô tả I/O mảng Byte
- Thực hiện các tác vụ đệm I/O và lọc
- Dùng lớp RandomAccessFile.
- Mô tả các tác vụ chuỗi I/O và ký tự
- Dùng lớp PrintWriter

9.1 Giới thiệu

Trong buổi học trước, chúng ta đã học về các dòng Synchronized. ngăn các dòng xảy ra việc chia sẻ (dùng chung) các đối tượng một cách đồng thời. Toàn bộ tiến trình này được quản lý bởi cơ chế đợi thông báo (wait-notify). Phương thức wait() báo cho dòng gọi từ bỏ monitor và nhập vào trạng thái ngủ cho đến khi các dòng khác nhập vào cùng monitor và gọi phương thức notify(). Phương thức notify() và notifyAll() tạo ra dòng thông báo cho các dòng khác gọi phương thức wait() của cùng đối tượng. Trong bài học trước, chúng ta cũng học về các điều kiện bế tắc là gì và cách tránh chúng.

Chương này giới thiệu khái niệm về luồng. Chúng ta cũng thảo luận các lớp khác nhau trong gói java.io trợ giúp các tác vụ nhập xuất.

9.2 Các luồng

Theo thuật ngữ chung, luồng là một dòng lưu chuyển. trong thuật ngữ về kỹ thuật luồng là một lộ trình mà dữ liệu được truyền trong một chương trình. Một ứng dụng về các luồng mà ta đã quen thuộc đó là luồng nhập System.in .

Luồng là những dàn ống (pipelines) để gửi và nhận thông tin trong các chương trình java. Khi một luồng dữ liệu được gửi hoặc nhận, ta tham chiếu nó như đang “ghi” và “đọc” một luồng theo thứ tự nêu trên. Khi một luồng được đọc hay ghi, các dòng khác bị phong tỏa. Nếu có một lỗi xảy ra khi đọc hay ghi luồng, một IOException được kích hoạt. Do vậy, các câu lệnh luồng phải bao gồm khối try-catch.

Lớp ‘java.lang.System’ định nghĩa các luồng nhập và xuất chuẩn. chúng là các lớp chính của các luồng byte mà java cung cấp. Chúng ta cũng đã sử dụng các luồng xuất để xuất dữ liệu và hiển thị kết quả trên màn hình. Luồng I/O bao gồm:

:

- Lớp System.out: Luồng xuất chuẩn dùng để hiển thị kết quả trên màn hình.
- Lớp System.in: Luồng nhập chuẩn thường đến từ bàn phím và được dùng để đọc các ký tự dữ liệu.
- Lớp System.err: Đây là luồng lỗi chuẩn.

Các lớp ‘InputStream’ và ‘OutputStream’ cung cấp nhiều khả năng I/O khác nhau. Cả hai lớp này có các lớp con để thực hiện I/O thông qua các vùng đệm bộ nhớ, các tập tin và ống dẫn. Các lớp con của lớp InputStream thực hiện đầu vào, trong khi các lớp con của lớp OutputStream thực hiện kết xuất.

9.3 Gói java.io

Các luồng hệ thống rất có ích. Tuy nhiên, chúng không đủ mạnh để dùng khi ứng phó với I/O thực tế. Gói java.io phải được nhập khẩu vì mục đích này. Chúng ta sẽ thảo luận tìm hiểu về các lớp thuộc gói java.io.

9.3.1 lớp InputStream

Lớp InputStream là một lớp trừu tượng. Nó định nghĩa cách nhận dữ liệu. Điểm quan trọng không nằm ở chỗ dữ liệu đến từ đâu, mà là nó có thể truy cập. Lớp InputStream cung cấp một số phương pháp để đọc và dùng các luồng dữ liệu để làm đầu vào. Các phương thức này giúp ta tạo, đọc và xử lý các luồng đầu vào. Các phương thức được liệt kê trong bảng 9.1

Tên phương thức	Mô tả
read()	Đọc các byte dữ liệu từ một luồng. Nếu như không dữ liệu nào là hợp lệ, nó khóa phương thức. Khi một phương thức được khóa, các dòng thực hiện được chờ cho đến khi dữ liệu hợp lệ.
read (byte [])	Trả về byte được ‘đọc’ hay ‘-1’, nếu như kết thúc của một luồng đã đến. Nó kích hoạt IOException nếu lỗi xảy ra.
read (byte [], int, int)	Nó cũng đọc vào mảng byte. Nó trả về số byte thực sự được đọc. Khi kết thúc của một luồng đã đến, nó kích hoạt IOException nếu lỗi xảy ra.
available()	Phương pháp này trả về số lượng byte có thể được đọc mà không bị phong tỏa. Nó trả về số byte hợp lệ. Nó không phải là phương thức hợp lệ đáng tin cậy để thực hiện tiến trình xử lý đầu vào.
close()	Phương thức này đóng luồng. Nó dùng để phóng thích mọi tài nguyên kết hợp với luồng. Luôn luôn đóng luồng để chắc chắn rằng luồng xử lý được kết thúc. Nó kích hoạt IOException nếu lỗi xảy ra.
mark()	Đánh dấu vị trí hiện tại của luồng.
markSupporte()	Trả về giá trị boolean nêu rõ luồng có hỗ trợ các khả năng mark và reset hay không. Nó trả về đúng nếu luồng hỗ trợ nó bằng không là sai.
reset()	Phương thức này định vị lại luồng theo vị

	trí được đánh dấu chót. Nó kích hoạt IOException nếu lỗi xảy ra.
skip()	Phương thức này bỏ qua 'n' byte đầu vào. 'n' chỉ định số byte được bỏ qua. Nó kích hoạt IOException nếu lỗi xảy ra. Phương thức này sử dụng để di chuyển tới vị trí đặc biệt bên trong luồng đầu vào.

Table 9.1 InputStream Class Methods

9.3.2 Lớp OutputStream

Lớp OutputStream cũng là lớp trừu tượng. Nó định nghĩa cách ghi các kết xuất đến luồng. Nó cung cấp tập các phương thức trợ giúp tạo ra, ghi và xử lý kết xuất các luồng. Các phương thức bao gồm:

Tên phương thức	Mô tả
write(int)	Phương thức này ghi một byte
write(byte[])	Phương thức này phong toả cho đến khi một byte được ghi. luồng chờ cho đến khi tác vụ ghi hoàn tất. Nó kích hoạt IOException nếu lỗi xảy ra.
write(byte[],int,int)	Phương thức này cũng ghi mảng các byte. Lớp OutputStream định nghĩa ba dạng quá tải của phương thức này để cho phép phương thức write() ghi một byte riêng lẻ, mảng các byte, hay một đoạn của một mảng.
flush()	Phương thức này xả sạch luồng. đệm dữ liệu được ghi ra luồng kết xuất. Nó kích hoạt IOException nếu lỗi xảy ra.
close()	Phương thức đóng luồng. Nó được dùng để giải phóng mọi tài nguyên kết hợp với luồng. Nó kích hoạt IOException nếu lỗi xảy ra.

Bảng 9.2 Các phương thức lớp OutputStream

9.3.3 Nhập và xuất mảng byte

Các lớp 'ByteArrayInputStream' và 'ByteArrayOutputStream' sử dụng các đệm bộ nhớ. Không cần thiết phải dùng chúng với nhau.

➤ Lớp ByteArrayInputStream

Lớp này tạo luồng đầu vào từ bộ nhớ đệm. Nó là mảng các byte. Lớp này không hỗ trợ các phương thức mới. Ngược lại nó chạy đè các phương thức của lớp InputStream như 'read()', 'skip()', 'available()' và 'reset()'.

➤ Lớp ByteArrayOutputStream

Lớp này tạo ra luồng kết suất trên một mảng các byte. Nó cũng cung cấp các khả năng bổ sung để mảng kết suất tăng trưởng nhằm mục đích chứa chỗ cho mảng được ghi. Lớp này cũng cung cấp các phương thức 'toArray()' và 'toString()'. Chúng được dùng để chuyển đổi luồng thành một mảng byte hay đối tượng chuỗi.

Lớp `ByteArrayOutputStream` cũng cung cấp hai phương thức thiết lập. Một chấp nhận một đối số số nguyên dùng để ấn định mảng byte kết xuất theo một kích cỡ ban đầu. và thứ hai không chấp nhận đối số nào, và thiết lập đệm kết xuất với kích thước mặc định. lớp này cung cấp vài phương thức bổ sung, không được khai báo trong `OutputStream`:

- `reset()`

Thiết lập lại kết xuất vùng đệm nhằm cho phép tiến trình ghi khởi động lại tại đầu vùng đệm.

- `size()`

Trả về số byte hiện tại đã được ghi tới vùng đệm.

- `writeto()`

Ghi nội dung của vùng đệm kết xuất ra luồng xuất đã chỉ định. Để thực hiện, nó chấp nhận một đối tượng của lớp `OutputStream` làm đối số.

Chương trình 9.1 sử dụng lớp '`ByteArrayInputStream`' và '`ByteArrayOutputStream`' để nhập và xuất:

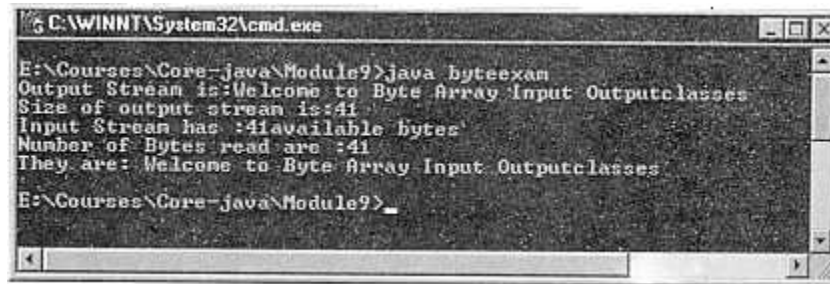
Program 9.1

```
import java.lang.System;
import java.io.*;
public class byteexam
{
    public static void main(String args[]) throws IOException
    {
        ByteArrayOutputStream os =new ByteArrayOutputStream();
        String s ="Welcome to Byte Array Input Outputclasses";
        for(int i=0; i<s.length( );i++)
            os. write (s.charAt(i) );
        System.out.println("Output Stream is:" + os);
        System.out.println("Size of output stream is:"+ os.size());
        ByteArrayInputStream in;
        in = new ByteArrayInputStream(os.toByteArray());
        int ib = in.available();

        System.out.println("Input Stream has : " + ib + "available bytes");
        byte ibuf[] = new byte[ib];
        int byrd = in.read(ibuf, 0, ib);

        System.out.println("Number of Bytes read are : " + byrd);
        System.out.println("They are: " + new String(ibuf));
    }
}
```

Hình 9.1 Xuất hiện kết xuất của chương trình:



```
C:\WINNT\System32\cmd.exe
E:\Courses\Core-java\Module9>java byteexam
Output Stream is:Welcome to Byte Array Input Outputclasses
Size of output stream is:41
Input Stream has :41available bytes
Number of Bytes read are :41
They are: Welcome to Byte Array Input Outputclasses
E:\Courses\Core-java\Module9>
```

Hình 9.1: sử dụng 1 sử dụng lớp ‘`ByteArrayInputStream`’ và ‘`ByteArrayOutputStream`’ cho nhập và xuất.

9.3.4 Nhập và xuất tập tin

Java hỗ trợ các tác vụ nhập và xuất tập tin với sự trợ giúp các lớp sau đây:

- `File`
- `FileDescriptor`
- `FileInputStream`
- `FileOutputStream`

Java cũng hỗ trợ truy cập nhập và xuất ngẫu nhiên hoặc trực tiếp bằng các lớp ‘`File`’, ‘`FileDescriptor`’, và ‘`RandomAccessFile`’.

➤ Lớp `File`

Lớp này được sử dụng để truy cập các đối tượng tập tin và thư mục. Các tập tin đặt tên theo qui ước đặt tên tập tin của hệ điều hành chủ. Các qui ước này được gói riêng bằng các hằng lớp `File`. Lớp này cung cấp các thiết lập các tập tin và các thư mục. Các thiết lập chấp nhận các đường dẫn tập tin tuyệt đối lẫn tương đối cùng các tập tin và thư mục. Tất cả các tác vụ thư mục và tập tin chung được thực hiện thông qua các phương thức truy cập của lớp *File*.

Các phương thức:

- Cho phép bạn tạo, xóa, đổi tên các file.
- Cung cấp khả năng truy cập tên đường dẫn tập tin.
- Xác định đối tượng có phải tập tin hay thư mục không.
- Kiểm tra sự cho phép truy cập đọc và ghi.

Giống như các phương thức truy cập, các phương thức thư mục cũng cho phép tạo, xóa, đặt tên lại và liệt kê các thư mục. Các phương pháp này cho phép các cây thư mục đang chéo bằng cách cung cấp khả năng truy cập các thư mục cha và thư mục anh em.

➤ Lớp `FileDescriptor`

Lớp này cung cấp khả năng truy cập các mô tả tập tin mà hệ điều hành duy trì khi các tập tin và thư mục đang được truy cập. Lớp này không cung cấp tầm nhìn đối với thông tin cụ thể do hệ điều hành duy trì. Nó cung cấp chỉ một phương thức có tên ‘`valid()`’, giúp xác định một đối tượng mô tả tập tin hiện có hợp lệ hay không.

➤ Lớp FileInputStream

Lớp này cho phép đọc đầu vào từ một tập tin dưới dạng một luồng. Các đối tượng của lớp này được tạo ra nhờ dùng một tập tin String, File, hoặc một đối tượng FileDescriptor làm một đối số. Lớp này chồng lên các phương thức của lớp InputStream. Nó cũng cung cấp các phương thức 'finalize()' và 'getFD()'.

Phương thức 'finalize()' được dùng để đóng luồng khi đang được bộ gôm rác Java xử lý. Phương thức 'getFD()' trả về đối tượng FileDescriptor biểu thị sự kết nối đến tập tin thực tế trong hệ tập tin đang được 'FileInputStream' sử dụng.

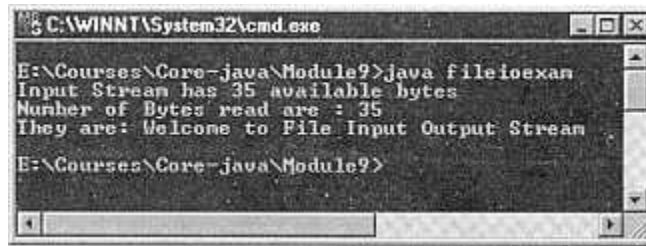
➤ Lớp FileOutputStream

Lớp này cho phép ghi kết xuất ra một luồng tập tin. Các đối tượng của lớp này cũng tạo ra sử dụng các đối tượng chuỗi tên tập tin, tập tin, FileDescriptor làm tham số. Lớp này chồng lên phương thức của lớp OutputStream và cung cấp phương thức 'finalize()' và 'getFD()'.

Chương trình 9.2

```
import java.io.FileOutputStream;
import java.io.FileInputStream;
import java.io.File;
import java.io.IOException;
public class fileioexam
{
    public static void main(String args[ ]) throws IOException
    {
        // creating an output file abc.txt
        FileOutputStream os = new FileOutputStream("abc.txt");
        String s = "Welcome to File Input Output Stream " ;
        for(int i = 0; i< s.length( ); + +i) .
        os. write(s.charAt(i));
        os.close();
        // opening abc.txt for input
        FileInputStream is = new FileInputStream("abc.txt");
        int ibyts = is.available( );
        System.out.println("Input Stream has " + ibyts + " available bytes");
        byte ibuf[ ] = new byte[ibyts];
        int byrd = is.read(ibuf, 0, ibyts);
        System.out.println("Number of Bytes read are: " + byrd);
        System.out.println("They are: " + new String(ibuf));
        is.close();
        File fl = new File("abc.txt");
        fl.delete();
    }
}
```

Hình 9.2 hiện kết xuất của đoạn mã nguồn trên:



Hình 9.2 sử dụng `FileInputStream`, `FileOutputStream`, và các lớp `File`

9.3.5 Nhập xuất đã lọc

Một 'Filter' là một kiểu luồng sửa đổi cách điều khiển một luồng hiện tồn tại. Các lớp, các luồng nhập xuất đã lọc của java sẽ giúp ta lọc I/O theo một số cách. Về cơ bản, các bộ lọc này dùng để thích ứng các luồng theo các nhu cầu của chương trình cụ thể.

Bộ lọc nằm giữa một luồng nhập và một luồng xuất. Nó thực hiện xử lý một tiến trình đặc biệt trên các byte được truyền từ đầu vào đến kết xuất. Các bộ lọc có thể phối hợp thực hiện dãy tuần tự các tùy chọn lọc ở đó mọi bộ lọc tác động như kết xuất của một bộ lọc khác.

➤ Lớp `FilterInputStream`

Đây là lớp trừu tượng. Nó là cha của tất cả các lớp luồng nhập đã lọc. Lớp này cung cấp khả năng tạo ra một luồng từ luồng khác. Một luồng có thể được đọc và cung cấp dưới dạng kết xuất cho luồng khác. Biến 'in' được sử dụng để làm điều này. Biến này được dùng để duy trì một đối tượng tách biệt của lớp `InputStream`. Lớp `FilterInputStream` được thiết kế sao cho có thể tạo nhiều bộ lọc kết xích [chained filters]. Để thực hiện điều này chúng ta dùng vài tầng lồng ghép. đến lượt mỗi lớp sẽ truy cập kết xuất của lớp trước đó với sự trợ giúp của biến 'in'.

➤ Lớp `FilterOutputStream`

Lớp này là một dạng hỗ trợ cho lớp `FilterInputStream`. Nó là lớp cha của tất cả các lớp luồng xuất đã lọc. Lớp này tương tự như lớp `FilterInputStream` ở chỗ nó duy trì đối tượng của lớp `OutputStream` làm một biến 'out'. Dữ liệu ghi vào lớp này có thể sửa đổi theo nhu cầu để thực hiện tác vụ lọc và sau đó được chuyển gửi tới đối tượng `OutputStream`.

9.3.6 I/O có lập vùng đệm

Vùng đệm là kho lưu trữ dữ liệu. Chúng ta có thể lấy dữ liệu từ vùng đệm thay vì quay trở lại nguồn ban đầu của dữ liệu.

Java sử dụng cơ chế nhập/xuất có lập vùng đệm để tạm thời lập cache dữ liệu được đọc hoặc ghi vào/ra một luồng. Nó giúp các chương trình đọc/ghi các lượng dữ liệu nhỏ mà không tác động ngược lên khả năng thực hiện của hệ thống.

Trong khi thực hiện nhập có lập vùng đệm, số lượng byte lớn được đọc tại thời điểm này, và lưu trữ trong một vùng đệm nhập. khi chương trình đọc luồng nhập, các byte dữ liệu được đọc từ vùng đệm nhập.

Tiến trình lập vùng đệm kết xuất cũng thực hiện tương tự. khi dữ liệu được một chương trình ghi ra một luồng, dữ liệu kết xuất được lưu trữ trong một vùng đệm xuất. Dữ liệu được lưu trữ đến khi vùng đệm trở nên đầy hoặc các luồng kết xuất được xả trống. Cuối cùng kết xuất có lập vùng đệm được chuyển gửi đến đích của luồng xuất.

Các bộ lọc hoạt động trên vùng đệm. Vùng đệm được phân bố nằm giữa chương trình và đích của luồng có lập vùng đệm.

➤ Lớp `BufferedInputStream`

Lớp này tự động tạo ra và chứa đựng vùng đệm để hỗ trợ vùng đệm nhập. Nhờ đó chương trình có thể đọc dữ liệu từng luồng theo byte một mà không ảnh hưởng đến khả năng thực hiện của hệ thống. Bởi lớp '`BufferedInputStream`' là một bộ lọc, nên có thể áp dụng nó cho một số đối tượng nhất định của lớp `InputStream` và cũng có thể phối hợp với các tập tin đầu vào khác.

Lớp này sử dụng vài biến để thực hiện các cơ chế lập vùng đệm đầu vào. Các biến này được khai báo là `protected` và do đó chương trình không thể truy cập trực tiếp. Lớp này định nghĩa hai phương thức thiết lập. Một cho phép chỉ định kích cỡ của vùng đệm nhập trong khi đó phương thức thiết lập kia thì không. Nhưng cả hai phương thức thiết lập đều tiếp nhận đối tượng của lớp `InputStream` và `OutputStream` làm đối số. lớp này chồng lên các phương thức truy cập mà `InputStream` cung cấp và không làm nảy sinh bất kì phương thức mới nào.

Lớp `BufferedInputStream`. Lớp này cũng định nghĩa hai phương thức thiết lập. nó cho phép chỉ định kích cỡ của vùng đệm xuất trong một phương thức thiết lập cũng như cung cấp một kích cỡ vùng đệm ngầm định. Nó chồng lên tất cả các phương thức của `OutputStream` và không làm nảy sinh bất kì phương thức nào.

Chương trình 9.3 dưới đây mô tả cách dùng các luồng nhập/xuất có lập vùng đệm:

Chương trình 9.3

```
import javaJang.* ;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.SequenceInputStream;
import java.io.IOException;
publicI class buff exam
{
    public static void main(String args[ ]) throws IOException
    {
        // defining sequence input stream
        SequenceInputStream Seq3;
        FileInputStream Fis 1 ;
        Fis1 = new FileInputStream("byteexam.java");
        FileInputStream Fis2;
        Fis2= new FileInputStream("fileioexam.java");
        Seq3 = new SequenceInputStream(Fis1, Fis2);
```



```

// create buffered input and output streams
BufferedInputStream inst;
inst = new BufferedInputStream(Seq3);
BufferedOutputStream oust;
oust = new BufferedOutputStream(System.out);
inst.skip(1000);
boolean eof = false;
int bytcnt = 0;
while(!eof)
{
    int num = inst.read();
    if(num == -1)
    {
        eof = true;
    }
    else
    {
        oust.write((char) num);
        ++ bytcnt;
    }
}
String bytrd = String.valueOf(bytcnt);
bytrd += "bytes were read";
oust.write(bytrd.getBytes(). 0, bytrd.length());

// close all streams.
inst.close();
oust.close();
Fis1.close();
Fis2.close();
}
}

```

Hình 9.3 hiện kết xuất của chương trình trên:



Hình 9.3 Sử dụng các lớp vùng đệm luồng nhập và xuất.

9.3.7 Lớp Reader và Writer

Đây là các lớp trừu tượng. Chúng nằm tại đỉnh của hệ phân cách lớp, hỗ trợ việc đọc và ghi các luồng ký tự unicode.java 1.1 thực tế đã giới thiệu các lớp này.

➤ Lớp Reader

Lớp này hỗ trợ các phương thức:

- read()
- reset()
- skip()
- mark()
- markSupported()
- close()

Lớp này cũng hỗ trợ phương thức gọi 'ready()'. Phương thức này trả về giá trị kiểu boolean nếu rõ tác vụ đọc kế tiếp có tiếp tục mà không phong tỏa hay không.

➤ Lớp Writer

Lớp này hỗ trợ các phương thức:

- write()
- flush()
- close()

9.3.8 Nhập/ xuất chuỗi và xâu ký tự

Các lớp 'CharArrayReader' và 'CharArrayWriter' cũng tương tự như các lớp ByteArrayInputStream và ByteArrayOutputStream ở chỗ chúng hỗ trợ nhập/xuất từ các vùng đệm nhớ. Các lớp CharArrayReader và CharArrayWriter hỗ trợ nhập/ xuất ký tự 8 bit.

CharArrayReader không hỗ trợ bổ sung các phương pháp sau đây vào các phương thức của lớp Reader cung cấp. Lớp CharArrayWriter bổ sung các phương thức sau đây vào các phương thức của lớp Writer.

➤ reset()

thiết lập lại vùng đệm

➤ size()

trả về kích cỡ hiện hành của vùng đệm

➤ toCharArray()

Trả về bản sao mảng ký tự của vùng đệm xuất

➤ toString()

Chuyển đổi vùng đệm xuất thành một đối tượng String

➤ writeTo()

Ghi vùng đệm ra một luồng xuất khác.

Lớp StringReader trợ giúp luồng nhập ký tự từ một chuỗi. Nó không bổ sung phương thức nào vào lớp Reader.

Lớp StringWriter trợ giúp ghi luồng kết xuất ký tự ra một đối tượng StringBuffer. Lớp này bổ sung hai phương thức có tên là 'getBuffer()' và 'toString()' . Phương thức 'getBuffer()' trả về đối tượng StringBuffer tương ứng với vùng đệm xuất, trong khi đó phương thức toString() trả về một bảng sao chuỗi của vùng đệm xuất.

Chương trình 9.4 dưới đây thực hiện các tác vụ nhập/xuất mảng ký tự:

Chương trình 9.4

```
import java.lang.System;
import java.io.CharArrayReader;
import java.io.CharArrayWriter;
import java.io.IOException;
public class test1
{
    public static void main(String args[ ]) throws IOException
    {
        CharArrayWriter ost = new CharArrayWriter( );
        String s = "Welcome to Character Array Program";

        for(int i= 0; i<s.length( ); ++i) ;
        ost.write(s.charAt(i));

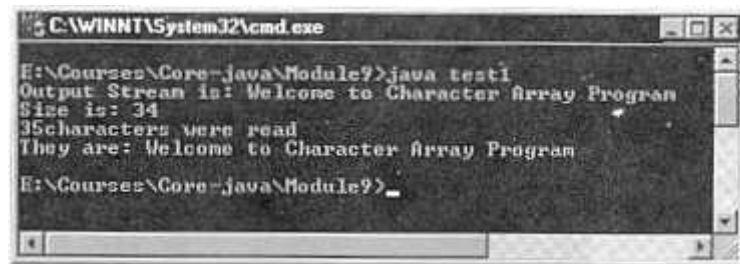
        System.out.println("Output Stream is: " + ost);
        System.out.println("Size is: " + ost.size( ));

        CharArrayReader inst;
        inst = new CharArrayReader(ost.toCharArray( ));
        int a= 0;
        String Buffer sbI = new String Buffer(" ");

        while((a = inst.read( )) != -1)
        sbI.append((char) a);

        s = sbI.toString( );
        System.out.println(s.length() + "characters were read");
        System.out.println("They are:" + s);
    }
}
```

Hình 9.4 Hiện kết xuất chương trình:



Hình 9.4 Các tác vụ nhập và xuất mảng các ký tự

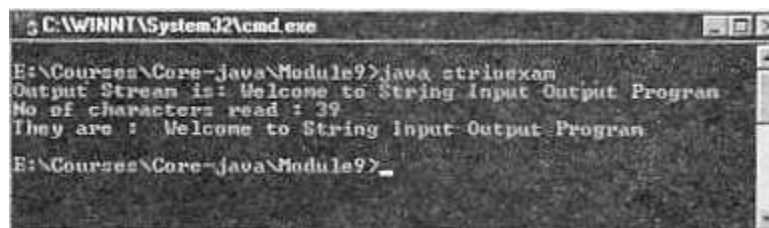
Chương trình 9.5 Mô tả tiến trình nhập/xuất chuỗi.

Chương trình 9.5

```
import java.lang.System;
import java.io.StringReader;
import java.io.StringWriter;
import java.io.IOException;
import java.io. * ;

public class strioexam
{
    public static void main(String args[ ]) throws IOException
    {
        StringWriter ost = new StringWriter( );
        String s = "Welcome to String Input Output Program";
        for(int i= 0; i <s.length( ); ++i)
            ost.write(s.charAt(i)) ;
        System.out.println("Output Stream is: " + ost);
        StringReader inst;
        inst = new StringReader(ost.toString( ));
        int a= 0;
        StringBuffer sb1 = new StringBuffer(" ");
        while((a = inst.read( )) != -1)
            sb1.append((char) a);
        s = sb1.toString( ); ,
        System.out.println("No of characters read: " +s.length( ));
        System.out.println("They are: " + s);
    }
}
```

Hình 9.5 Hiện kết xuất chương trình:



Hình 9.5 Nhập và xuất sâu chuỗi

9.3.9 Lớp PrinterWriter

Lớp 'PrintStream' thực hiện việc kết xuất dữ liệu. Lớp này có các phương thức bổ sung, trợ giúp cho việc in ấn dữ liệu cơ bản.

Lớp 'PrinterWriter' là một thay thế của lớp 'PrintStream'. Nó thực tế cải thiện lớp 'PrintStream' bằng cách dùng dấu tách dòng phụ thuộc nền tảng để in các dòng thay vì

ký tự '\n'. Lớp này cũng cấp hỗ trợ các ký tự Unicode so với PrinterStream. Phương thức 'checkError()' được sử dụng kiểm tra kết xuất được xả sạch và được kiểm tra các lỗi. Phương thức setError() được sử dụng để thiết lập lỗi điều kiện. Lớp PrintWriter cung cấp việc hỗ trợ in ấn các kiểu dữ liệu nguyên thủy, các mảng ký tự, các chuỗi và các đối tượng.

9.3.10 Giao diện DataInput

Giao diện DataInput được sử dụng để đọc các byte từ luồng nhị phân và xây dựng lại các kiểu dữ liệu dạng nguyên thủy trong Java.

DataInput cũng cho phép chúng ta chuyển đổi dữ liệu từ định dạng sửa đổi UTF-8 tới dạng chuỗi. Chuẩn UTF cho định dạng chuyển đổi Unicode. Nó là kiểu định dạng đặt biệt giải mã các giá trị Unicode 16 bit. UTF lạc quan ở mức thấp giả lập trong hầu hết các trường hợp, mức cao 8 bit Unicode sẽ là 0. Giao diện DataInput được định nghĩa là số các phương thức, các phương thức bao gồm việc đọc các kiểu dữ liệu nguyên thủy trong java.

Bảng 9.3 tóm lược vài phương thức. Tất cả các phương thức được kích hoạt IOException trong trường hợp lỗi:

Tên phương thức	Mô tả
boolean readBoolean()	Đọc một byte nhập, và trả về đúng nếu byte đó không phải là 0, và sai nếu byte đó là 0.
byte readByte()	Đọc một byte
char readChar()	Đọc và trả về một giá trị ký tự
short readShort()	Đọc 2 byte và trả về giá trị short
long readLong()	Đọc 8 byte và trả về giá trị long.
float readFloat()	Đọc 4 byte và trả về giá trị float
int readInt()	Đọc 4 byte và trả về giá trị int
double readDouble()	Đọc 8 byte và trả về giá trị double
String readUTF()	Đọc một chuỗi
String readLine()	Đọc một dòng văn bản

Bảng 9.3 Các phương thức của giao diện DataInput

9.3.11 Giao diện DataOutput

Giao diện DataOutput được sử dụng để xây dựng lại các kiểu dữ liệu nguyên thủy trong java vào trong dãy các byte. nó ghi các byte này lên trên luồng nhị phân.

Giao diện DataOutput cũng cho phép chúng ta chuyển đổi một chuỗi vào trong java được sửa đổi theo định dạng UTF-8 và ghi nó vào luồng.

Giao diện DataOutput định nghĩa số phương thức được tóm tắt trong bảng 9.4. Tất cả các phương thức sẽ kích hoạt IOException trong trường hợp lỗi.

Tên phương thức	Mô tả
void writeBoolean(boolean b)	Ghi một giá trị Boolean vào luồng

<code>void writeByte(int value)</code>	Ghi giá trị 8 bit thấp
<code>void writeChar(int value)</code>	Ghi 2 byte giá trị kiểu ký tự vào luồng
<code>void writeShort(int value)</code>	Ghi 2 byte, biểu diễn lại giá trị dạng short
<code>void writeLong(long value)</code>	Ghi 8 byte, biểu diễn lại giá trị dạng long
<code>void writeFloat(float value)</code>	Ghi 4 byte, biểu diễn lại giá trị dạng float
<code>void writeInt(int value)</code>	ghi 4 byte
<code>void writeDouble(double value)</code>	Ghi 8 byte, biểu diễn lại giá trị dạng double
<code>void writeUTF(String value)</code>	Ghi một sâu dạng UTF tới luồng.

Bảng 9.4 Các phương thức của giao diện `DataOutput`

9.3.12 Lớp `RandomAccessFile`

Lớp `RandomAccessFile` cung cấp khả năng thực hiện I/O theo một vị trí cụ thể bên trong một tập tin. Trong lớp này, dữ liệu có thể đọc hoặc ghi ở vị trí ngẫu nhiên bên trong một tập tin thay vì một kho lưu trữ thông tin liên tục. Hơn thế nữa lớp này có tên `RandomAccess`. Phương thức `'seek()'` hỗ trợ truy cập ngẫu nhiên. Kết quả là, biến trở tương ứng với tập tin hiện hành có thể ấn định theo vị trí bất kỳ trong tập tin.

Lớp `RandomAccessFile` thực hiện cả hai việc nhập và xuất. Do vậy, có thể thực hiện I/O bằng các kiểu dữ liệu nguyên thủy. Lớp này cũng hỗ trợ cho phép đọc hoặc ghi tập tin cơ bản, điều này cho phép đọc tập tin theo chế độ chỉ đọc hoặc đọc-ghi. Tham số `'r'` hoặc `'rw'` được gán cho lớp `RandomAccessFile` chỉ định truy cập `'chỉ đọc'` và `'đọc-ghi'`. Lớp này giới thiệu vài phương thức mới khác với phương pháp đã thừa kế từ các lớp `DataInput` và `DataOutput`.

Các phương thức bao gồm:

➤ **`seek()`**

Thiết lập con trỏ tập tin tới vị trí cụ thể bên trong tập tin.

➤ **`getFilePointer()`**

Trả về vị trí hiện hành của con trỏ tập tin.

➤ **`length()`**

Trả về chiều dài của tập tin tính theo byte.

Chương trình dưới đây minh họa cách dùng lớp `RandomAccessFile`. Nó ghi một giá trị boolean, một int, một char, một double tới một file có tên `'abc.txt'`. Nó sử dụng phương pháp `seek()` để tìm vị trí định vị 1 bên trong tập tin. Sau đó nó đọc giá trị số nguyên, ký tự và double từ tập tin và hiển thị chúng ra màn hình.

Chương trình 9.6

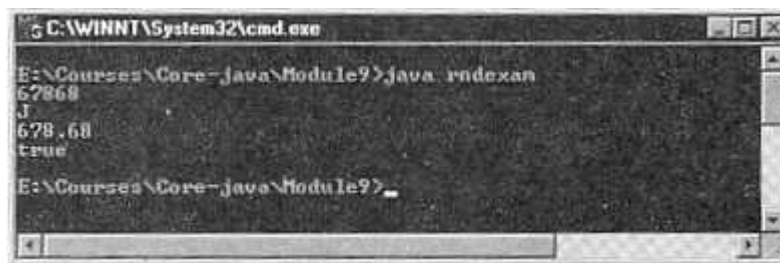
```
import java.lang.System;
import java.io.RandomAccessFile;
import java.io.IOException;
public class mdexam
{
    public static void main (String args[ ]) throws IOException
    {
```

```

RandomAccessFile rf;
rf= new RandomAccessFile("abc.txt", "rw");
rf.writeBoolean(true);
rf.writeInt( 67868) ;
rf.writeChars("J");
rf.writeDouble(678.68);
// making use of seek( ) method to move to a specific file location
rf.seek(1);
System.out.println(rf.readInt( ));
System.out.println(rf.readChar( ));
System.out.println(rf.readDouble( ));
rf.seek(0);
System.out.println(rf.readBoolean( ));
rf.close( );
}
}

```

Hình 9.5 Hiện kết xuất chương trình:



Hình 9.6: Lớp RandomAccessFile

9.4 Gói java.awt.print

Đây là gói mới mà java JDK 1.2 cung cấp. Nó thay thế khả năng in của JDK 1.1. Nó bao gồm dãy các giao diện:

- **Pageable**
- **Printable**
- **PrinterGraphics**

Giao diện ‘Pageable’ định nghĩa các phương thức được sử dụng cho đối tượng mô tả lại các trang sẽ được in. Nó cũng chỉ định số lượng trang sẽ được in cũng như sẽ được in trang hiện hành hay một miền trang.

Giao diện ‘Printable’ chỉ định phương thức print() được dùng để in một trang trên một đối tượng Graphics.

Giao diện ‘PrinterGraphics’ cung cấp khả năng truy cập đối tượng ‘PrinterJob’. Nó cung cấp các lớp sau đây:

- **Paper**
- **Book**
- **PageFormat**

➤ **Printerjob**

Lớp ‘Page’ định nghĩa các đặc tính vật lý của giấy in. Ngoài ra nó cũng cung cấp khổ giấy và vùng vẽ.

Lớp ‘Book’ là một lớp con của đối tượng duy trì một danh sách các trang in. Lớp này cũng cung cấp các phương thức để bổ sung và quản lý các trang cũng như thực thi giao diện Pageable.

Lớp ‘PageFormat’ định nghĩa lề của trang như các lề ‘Top’, ‘Bottom’, ‘Left’ và ‘Right’. Nó cũng chỉ định kích cỡ và hướng in như ‘Portrait’ (khô dọc) hoặc ‘Landscape’ (khô ngang).

Lớp ‘Printerjob’ là một lớp con của đối tượng khởi tạo, quản lý, và điều khiển yêu cầu máy in. Lớp này cũng chỉ định các tính chất in.

Dưới đây là ngoại lệ và lỗi mà gói java.awt.print kích hoạt:

➤ **PrinterException**

➤ **PrinterIOException**

➤ **PrinterAbortException**

‘PrinterException’ mở rộng lớp java.lang.Exception nhằm cung cấp một lớp cơ sở để in các ngoại lệ liên quan.

‘PrinterIOException’ mở rộng lớp ‘PrinterException’ nêu rõ một lỗi trong I/O.

‘PrinterAbortException’ là lớp con của lớp PrinterException nêu rõ khối in đã được bỏ ngang.

Tóm tắt buổi học

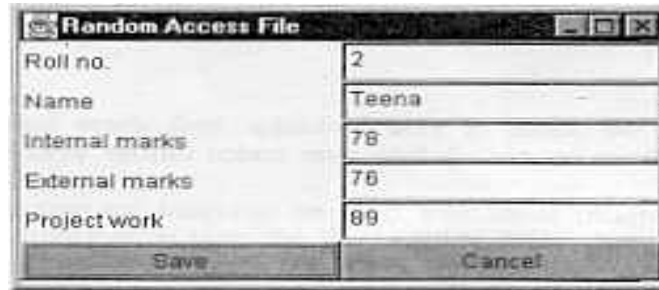
- Một luồng là một lộ trình qua đó dữ liệu di chuyển trong một chương trình java.
- Khi một luồng dữ liệu được gửi hoặc nhận. Chúng ta xem nó như đang ghi và đọc một luồng theo thứ tự nêu trên.
- Luồng nhập/xuất bao gồm các lớp sau đây:
 - Lớp System.out
 - Lớp System.in
 - Lớp System.err
- Lớp InputStream là một lớp trừu tượng định nghĩa cách nhận dữ liệu.
- Lớp OutputStream cũng là lớp trừu tượng. Nó định nghĩa ghi ra các luồng được kết xuất như thế nào.
- Lớp ByteArrayInputStream tạo ra một luồng nhập từ vùng đệm bộ nhớ trong khi ByteArrayOutputStream tạo một luồng xuất trên một mảng byte.
- Java hỗ trợ tác vụ nhập/xuất tập tin với sự trợ giúp của các File, FileDescriptor, FileInputStream và FileOutputStream.
- Các lớp Reader và Writer là lớp trừu tượng hỗ trợ đọc và ghi các luồng ký tự Unicode.
- CharArrayReader, CharArrayWriter khác với ByteArrayInputStream, ByteArrayOutputStream hỗ trợ định dạng nhập/xuất 8 bit, Trong khi ByteArrayInputStream, ByteArrayOutputStream hỗ trợ nhập/xuất 16bit.
- Lớp PrintStream thực thi một kết xuất. lớp này có phương thức bổ sung, giúp ta in các kiểu dữ liệu cơ bản.
- Lớp RandomAccessFile cung cấp khả năng thực hiện I/O tới vị trí cụ thể trong một tập tin.

Kiểm tra mức độ tiến bộ

1. ----- là các dàn ống (pipelines) để gửi và nhận thông tin trong các chương trình java.
2. ----- là luồng lỗi chuẩn.
3. Phương thức ----- đọc các byte dữ liệu từ một luồng.
4. Phương thức ----- trả về giá trị boolean, nêu rõ luồng có hỗ trợ các khả năng mark và reset hay không.
5. Phương thức ----- xả sạch luồng.
6. Nhập/xuất mảng byte sử dụng các lớp ----- và -----
7. Lớp ----- được sử dụng truy cập các đối tượng thư mục và tập tin.
8. -----là một khi chưa để lưu giữ dữ liệu.

Bài tập

1. Viết chương trình nhận một dòng văn bản từ người dùng và hiển thị đoạn văn bản đó lên màn hình.
2. Viết chương trình sao chép nội dung một tập tin tới tập tin khác.
3. Viết chương trình tạo ra một tập tin truy cập ngẫu nhiên. kết xuất hiển thị phía dưới đây.



A screenshot of a Windows-style dialog box titled "Random Access File". It contains a table with five rows of data. The first column lists fields: "Roll no:", "Name", "Internal marks", "External marks", and "Project work". The second column contains the corresponding values: "2", "Teena", "76", "76", and "89". At the bottom of the dialog, there are two buttons: "Save" on the left and "Cancel" on the right.

Roll no:	2
Name	Teena
Internal marks	76
External marks	76
Project work	89

Save Cancel

Các bản ghi nên được lưu ở dạng tập tin '.dat', vì vậy người dùng truy cập chúng nhanh hơn.