BÀI 5.
VÀO RA DỮ LIỆU

1

Vào ra dữ liệu

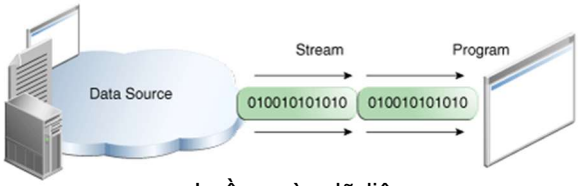
- Các luồng vào-ra dữ liệu
- Vào-ra dữ liệu trên thiết bị chuẩn
- Vào-ra dữ liệu trên file nhị phân
- Vào-ra dữ liệu trên file văn bản

2

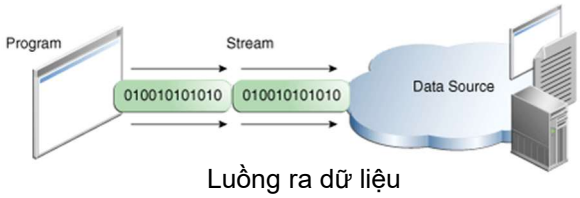
1. CÁC LUỒNG VÀO RA DỮ LIỆU

Các luồng vào ra dữ liệu

- Vào-ra dữ liệu: trao đổi dữ liệu với các thiết bị ngoại vi
  - Bàn phím, màn hình, thiết bị nhớ, cạc mạng...
- Java cung cấp cơ chế vào ra dữ liệu theo các luồng

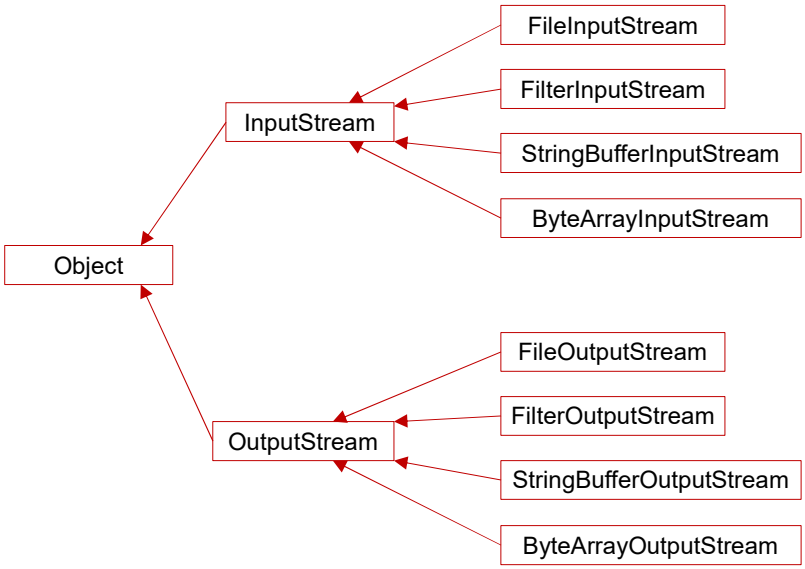


Luồng vào dữ liệu



Luồng ra dữ liệu

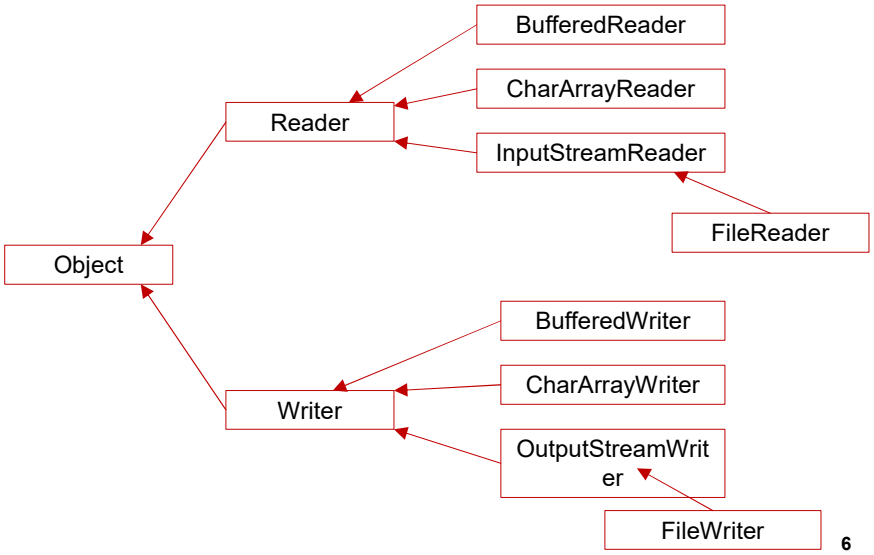
Các luồng vào ra dữ liệu theo byte



5

5

Các luồng vào ra dữ liệu theo ký tự



6

6

Vào ra dữ liệu trên thiết bị chuẩn

- Vào dữ liệu từ thiết bị chuẩn (bàn phím): `System.in`
  - Một đối tượng của lớp `InputStream` → đọc ghi theo luồng byte
  - Các phương thức rất hạn chế
  - Thường được sử dụng để khởi tạo các đối tượng luồng khác để xử lý dễ dàng hơn:

```
new BufferedReader(new
                   InputStreamReader(System.in))
new Scanner(System.in)
```

- Ra dữ liệu trên thiết bị chuẩn (màn hình): `System.out`
  - Một đối tượng của lớp `PrintStream`
  - Cung cấp các phương thức đầy đủ

7

7

2. VÀO-RA DỮ LIỆU TRÊN FILE NHỊ PHÂN

8

8

File nhị phân

- Dữ liệu được tổ chức và xử lý theo dạng bit-by-bit
- Thuận tiện cho các chương trình khi vào ra dữ liệu
- Vào-ra dữ liệu trên file nhị phân:
  - `new FileOutputStream(filePath)`: ghi dữ liệu theo luồng
    - `filePath`: đường dẫn tới file (bao gồm tên file)
    - Phương thức `write(int)`
  - `new FileInputStream(filePath)`: đọc dữ liệu theo luồng
    - Phương thức `int read()` trả về -1 nếu đọc hết file
  - `new DataOutputStream(outputStreamObject)`: ghi dữ liệu nguyên thủy
    - Phương thức `writeInt(), writeDouble(), writeChars(),...`
  - `new DataInputStream(inputStreamObject)`: đọc dữ liệu nguyên thủy
    - Phương thức `readInt(), readDouble(),...`

9

9

Vào ra trên file nhị phân – Ví dụ 1

```
/** Copy data from source file into destination file
*@param srcFilePath the path of the source file
*@param dstFilePath the path of the destiantion file
*/
private static void copy(String srcFilePath,
                         String dstFilePath) throws IOException{
        FileInputStream in = null;
        FileOutputStream out = null;
        in = new FileInputStream(srcFilePath);
        out = new FileOutputStream(dstFilePath);
        int data;
        while((data = in.read()) != -1)
                out.write(data);
        in.close();
        out.close();
}
```

Đóng luồng sau khi hoàn thành

10

10

Ví dụ 1 - Xử lý ngoại lệ

```
private static void copy(String srcFilePath,
                         String dstFilePath){
   FileInputStream in = null;
   FileOutputStream out = null;
   try {
       in = new FileInputStream(srcFilePath);
   } catch (FileNotFoundException e) {
       System.out.println("Source file not found");
   }
   try {
       out = new FileOutputStream(dstFilePath);
   } catch (FileNotFoundException e) {
       System.out.println("Destination file not found");
   }
}
```

11

11

Ví dụ 1 - Xử lý ngoại lệ (tiếp)

```
   int data = -1;
   try {
       while((data = in.read()) != -1)
          out.write(data);
   } catch (IOException e) {
       System.out.println("Cannot access file");
   }
   try {
       in.close();
       out.close();
   } catch (IOException e) {
       System.out.println("Cannot close files");
   }
}//end method
```

Chưa thể chắc chắn việc luồng đã được đóng khi ngoại lệ xảy ra

12

12

Ví dụ 1 - Xử lý ngoại lệ(tiếp)

```
private static void copy(String srcFilePath,
                    String dstFilePath){
   try(FileInputStream in = new
                        FileInputStream(srcFilePath);
       FileOutputStream out = new
                        FileOutputStream(dstFilePath)
   ){
       while((data = in.read()) != -1)
          out.write(data);
   } catch (FileNotFoundException e) {
       System.out.println(e.getMessage());
   } catch (IOException e) {
       System.out.println(e.getMessage());
   }
```

try-with-resources: đảm bảo luồng luôn được đóng

13

13

Vào ra trên file nhị phân – Ví dụ 2

```
public static void main(String args[]) {
   int seqNumber = 1;
   String fullName = "Nguyen Van An";
   double mark = 9.5;

   try(DataOutputStream out = new DataOutputStream(new
                        FileOutputStream("test.bin"));
       DataInputStream in = new DataInputStream(new
                    FileInputStream("test.bin"))
   ){
       out.writeInt(seqNumber);
       out.writeChar(':'); //write delimiter
       out.writeChars(fullName);
       out.writeChar(':'); //write delimiter
       out.writeDouble(mark);
```

14

14

Vào ra trên file nhị phân – Ví dụ 2 (tiếp)

```
        System.out.println("No:" + in.readInt());
        in.readChar(); //ignore ':'
        char chr;
        StringBuffer name = new StringBuffer(30);
        while((chr = in.readChar()) != ':')
           name.append(chr);
        System.out.println("Fullname: " + name.toString());
        System.out.println("Mark: " + in.readDouble());
    } catch (FileNotFoundException e) {
        System.out.println(e.getMessage());
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

15

15

Ghi đè và ghi nối tiếp

- Sử dụng cờ khi khởi tạo các đối tượng luồng ra:
  - `true`: ghi tiếp
  - `false`: ghi đè (mặc định)
- Ví dụ:

```
out = new FileOutputStream(dstFilePath, true);
DataOutputStream out = new DataOutputStream(new
            FileOutputStream("test.bin", true));
```

16

16

Vào-ra sử dụng bộ đệm

- Các phương thức vào ra đã đề cập đến được xử lý trực tiếp bởi HĐH → kém hiệu quả
- Ghi dữ liệu sử dụng bộ đệm: `BufferedOutputStream`
  - Khởi tạo: `BufferedOutputStream(outputStreamObject)`
  - Phương thức `flush()`: xóa bộ đệm
  - Phương thức `void write(int)`: ghi dữ liệu
- Đọc dữ liệu sử dụng bộ đệm: `BufferedInputStream`
  - Khởi tạo: `BufferedInputStream(inputStreamObject)`
  - Phương thức `available()`: trả về 0 nếu đọc hết dữ liệu
  - Phương thức `int read()`: trả về -1 nếu đọc hết dữ liệu

17

17

Ví dụ- Vào ra qua bộ đệm

```
try(BuferredInputStream in = new BuferredInputStream
                  (new FileInputStream(srcFilePath));
    BufferedOutputStream out = new BufferedOutputStream
                  (new FileOutputStream(dstFilePath))
){
    int data;
    while (in.available()>0){
       data = in.read();
       out.flush();
       out.write(data);
    }
} catch (FileNotFoundException e) {
    System.out.println(e.getMessage());
} catch (IOException e) {
    System.out.println(e.getMessage());
}
```

18

18

3. VÀO-RA DỮ LIỆU TRÊN FILE VĂN BẢN

19

FileReader và FileWriter

- Đọc và ghi dữ liệu trên file văn bản.
- `FileReader`
  - Khởi tạo: `FileReader(filePath)`
  - Phương thức `read()`: đọc theo từng ký tự, trả về int → ép kiểu thành char
    - Trả về -1 nếu hết file
- `FileWriter`
  - Khởi tạo: `FileWriter(filePath)`
  - Phương thức `write()`: ghi dữ liệu vào file

20

Ví dụ

```
public static void main(String args[]) {
   try(
       FileWriter wr = new FileWriter("test.txt");
       FileReader rd = new FileReader("test.txt")
   ){
       wr.write(String.valueOf(1));
       wr.write(":Nguyen Van An:");
       wr.write(String.valueOf(9.5));
       char ch;
       while((ch = (char) rd.read()) != -1)
              System.out.print(ch);
   } catch (FileNotFoundException e) {
       System.out.println(e.getMessage());
   } catch (IOException e) {
       System.out.println(e.getMessage());
   }
}
```

21

21

Xử lý theo từng dòng văn bản

- Ghi từng dòng văn bản: Sử dụng PrintWriter
  - Khởi tạo: `new PrintWriter(writerObject)`
  - Phương thức: `print(), printf(), println()`
- Ghi từng dòng văn bản: Sử dụng BufferedWriter
  - Khởi tạo: `new BufferedWriter(writerObject)`
  - Phương thức: `void write(int),void write(String), void writeLine()`
- Đọc từng dòng văn bản: Sử dụng BufferedReader
  - Khởi tạo: `new BufferedReader(readerObject)`
  - Phương thức: `String readLine()` trả về null nếu đọc hết file

22

22

java.util.StringTokenizer

- Phân tách xâu ký tự thành các xâu phần tử theo dấu hiệu phân cách (delimiter)
  - Delimiter: mặc định là dấu cách trắng \s
  - Có thể định nghĩa lại trong phương thức khởi tạo
- Phương thức khởi tạo
  - Mặc định: `StringTokenizer(String input)`
  - Định nghĩa lại dấu hiệu phân cách

    `StringTokenizer(String input, String delimiter)`
- `nextToken()`: trả lại xâu phần tử tiếp theo
- `hasMoreTokens()`: trả về `false` nếu không còn xâu phần tử
- `countTokens()`: trả về số xâu phần tử tách được

23

23

Vào-ra file văn bản – Ví dụ

```
public static void main(String[] args) {
   int seqArr[] = {1,2,3};
   String nameArr[] = {"Nguyen Van A", "Tran Thi B",
                                    "Le Van C"};
   double markArr[] = {7.0, 8.0, 9.5};
   try(PrintWriter writer  = new PrintWriter(new
             FileWriter("D:\\Samsung\\data.txt",true),true)
   ){
       for(int i = 0; i < 3; i++)
          writer.println(seqArr[i] + ":" + nameArr[i] + ":"
                         + markArr[i]);
   }catch(FileNotFoundException e){
       System.out.println(e.getMessage());
   }catch(IOException e){
       System.out.println(e.getMessage());
   }
```

24

24

Vào-ra file văn bản – Ví dụ(tiếp)

```
try(BufferedReader reader = new BufferedReader(
    new FileReader("D:\\Samsung\\data.txt"))
){
    String line;
    StringTokenizer readData;
    while((line = reader.readLine()) != null){
       readData = new StringTokenizer(line,":");
       while(readData.hasMoreTokens())
           System.out.printf("%s ", readData.nextToken());
           System.out.println();
    }
}catch(FileNotFoundException e){
    System.out.println(e.getMessage());
}catch(IOException e){
    System.out.println(e.getMessage());
}
}
```

25

25

4. LỚP FILE

26

26

Lớp File

- Cung cấp các phương thức thao tác với file, thư mục trên máy tính
  - Thư mục về bản chất cũng là file
- Các phương thức khởi tạo:
  - File(String `filePath`): Tạo đối tượng file với đường dẫn (và tên file)
  - File(String `path`, String `filePath`): Tạo đối tượng file nằm trong thư mục cha `path`
- Lưu ý: tạo đối tượng file trong chương trình không có nghĩa là tạo một file mới trên hệ thống

27

27

Các phương thức

- `boolean mkdir()`: tạo thư mục có tên chỉ ra khi khởi tạo đối tượng File. Trả về false nếu không thành công
- `boolean mkdirs()`: tạo thư mục có tên chỉ ra khi khởi tạo đối tượng File, bao gồm cả thư mục cha nếu cần thiết.
- `createNewFile()`: tạo file mới
- `boolean isDirectory()`: trả về true nếu là thư mục
- `boolean isFile()`: trả về true nếu là file
- `boolean canRead()`: trả về true nếu có quyền đọc
- `boolean canWrite()`: trả về true nếu có quyền ghi
- `boolean canExecute()`: trả về true nếu có quyền thực thi
- `String getName()`: trả về tên file/thư mục
- `String getParent()`: trả về thư mục cha

28

28

Các phương thức

- `String[] list()`: trả về tên các thư mục con và file
- `String[] list(FileNameFilter filter)`: trả về tên các thư mục con và file có chứa `filter`
- `File[] listFiles()`
- `File[] listFiles(FileFilter filter)`: trả về các đối tượng file thỏa mãn `filter`
- `boolean exists()`: trả về true nếu tồn tại file, thư mục
- `long length()`: trả về kích thước của file (byte)
- boolean `delete()`
- void `deleteOnExit()`: xóa khi tắt máy ảo JVM
- boolean `renameTo(File dest)`: đổi tên
- boolean `setReadOnly()`: thiết lập thuộc tính read-only

29

29

Ví dụ - Tìm file theo tên

```
package samsung.java.file.operator;
/** The SearchingByName class demonstrates searching the
file containing the keyword*/
public class SearchingByName {
    Scanner inputData = new Scanner(System.in);
    System.out.print("Search in directory: ");
    String dirPath = inputData.nextLine();
    File dirInSearch = new File(dirPath);
    if(dirInSearch.isDirectory()){
        System.out.print("Keyword: ");
        String key = inputData.nextLine();
        NameFilter filter = new NameFilter(key);
        String[] children;
        children = dirInSearch.list(filter);
        for(String child:children)
            System.out.println(child);
    }
}
```

30

30

Ví dụ (tiếp)

```
package samsung.java.file.operator;
/** The NameFilter class presents a file filter by name*/
public class NameFilter implements FileNameFilter{
   private String key;
   /** Construct a new filter with keyword
   *@param initKey the keyword
   */
   public NameFilter(String initKey){
       this.key = initKey;
   }
   @Override
   public boolean accept(File dir, String name) {
       return name.contains(key);
   }
}
```

31

31

Sử dụng luồng vào-ra trên đối tượng File

- Các luồng vào-ra đều cung cấp phương thức khởi tạo với đối tượng File
- `FileInputStream(File file)`
- `FileOutpuStream(File file)`
- `FileReader(File file)`
- `FileWriter(File file)`

32

32

4. LỚP FILES

33

Lớp Files

- Java 7 trở lên cung cấp thêm lớp `Files` với các phương thức tiện dụng và hiệu quả hơn
- Tiện dụng vì các phương thức đều là `static`
- Khi sử dụng lớp `Files`, thường phải truyền đối số là đối tượng từ lớp `Path` để định vị file trên hệ thống
  - Tạo một đối tượng Path từ đường dẫn file: Paths.get(String filePath)

34

Các phương thức

- `boolean isDirectory(Path)`: trả về true nếu là thư mục
- `boolean isRegularFile(Path)`: trả về true nếu là file
- `boolean isReadable(Path)`: trả về true nếu được phép đọc
- `boolean isWritable(Path)`
- `boolean isExecutable(Path)`
- `Path createFile(Path, FileAttribute)`: tạo file
- `Path createDirectory(Path, FileAttribute)`: tạo thư mục
- `Path createDirectories(Path, FileAttribute)`: thạo thư mục, bao gồm cả thư mục cha nếu không tồn tại

35

35

Các phương thức

- `void deleteIfExist(Path)`: xóa
- `boolean notExist(Path)`: trả về true nếu file không tồn tại
- `long size(Path)`: trả về kích thước file (byte)
- `Path copy(Path source, Path target, CopyOption options)`
- `Path move(Path source, Path target, CopyOption... options)`

36

36

Các phương thức đọc từ file

- `byte[] readAllBytes(Path)`: đọc nội dung file vào mảng byte
- `BufferedReader newBufferedReader(Path, Charset)`: mở file và trả lại đối tượng BufferedReader, hỗ trợ bảng mã khác (US-ASCII, UTF-16)mặc định(UTF-8)
- `InputStream newInputStream(Path, OpenOption)`: mở file và trả lại đối tượng InputStream

37

37

Các phương thức ghi vào file

- `Path write(Path, byte[], OpenOption)`: ghi mảng byte vào file
- `BufferedWriter newBufferedWriter(Path, OpenOption)`: mở và tạo một đối tượng BufferedWriter để ghi
- `BufferedWriter newBufferedWriter(Path, Charset, OpenOption)`
- `OutputStream newOutputStream(Path, OpenOption)`: mở và tạo một đối tượng OutputStream để ghi

38

38

Các tùy chọn

- Tùy chọn mở OpenOption:
  - APPEND: ghi tiếp
  - CREATE: tạo file mới và ghi
  - READ: mở để đọc
  - WRITE: mở để ghi
  - DELETE_ON_CLOSE: xóa khi đóng file
  - DSYNC và SYNC: yêu cầu đồng bộ hóa khi có nhiều luồng cũng truy cập vào file
- Tùy chọn CopyOption:
  - COPY_ATTRIBUTES: Sao chép cả thuộc tính
  - REPLACE_EXISTING: chép đè lên file cũ (nếu có)

39

39

Ví dụ - Xóa file, thư mục

```
Path path = Paths.get(filePath);//filePath is a String
try {
    Files.delete(path);
} catch (NoSuchFileException x) {
    System.err.format("%s: no such" + " file or
directory%n", path);
} catch (DirectoryNotEmptyException x) {
    System.err.format("%s not empty%n", path);
} catch (IOException x) {
    // File permission problems are caught here.
    System.err.println(x);
}
```

40

40

Ví dụ - Đọc-ghi qua bộ đệm

```
public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   System.out.print("Open file for reading: ");
   String srcPath = sc.nextLine();
   Path srcFile = Paths.get(srcPath);
   Charset cs = Charset.forName("US-ASCII");
   try(BufferedReader reader =
                   Files.newBufferedReader(srcFile, cs)
   ){
       String line = null;
       while ((line = reader.readLine()) != null)
          System.out.println(line);
   }catch(IOException e){
       System.err.format("IOException: %s%n", e);
   }
}
```

41

41

Ví dụ - Ghi qua bộ đệm

```
try(BufferedWriter writer = Files.newBufferedWriter(srcFile,
       cs,StandardOpenOption.APPEND, StandardOpenOption.WRITE)
){
   String line;
   System.out.println("Write to file:");
   while((line = sc.nextLine()).length() != 0){
       writer.write(line);
       writer.flush();
   }
}catch(IOException e){
   System.err.format("IOException: %s%n", e);
}
```

42

42