Fpt University

FPT POLYTECHNIC

LẬP TRÌNH JAVA

BÀI 3
INPUT và OUTPUT trong Java

www.poly.edu.vn

Nhắc lại bài trước

- Ngoại lệ là các lỗi chỉ xảy ra khi chạy chương trình
- Khi gặp ngoại lệ thì chương trình lập tức dừng lại
- Dùng try… catch để xử lý ngoại lệ theo ý đồ của người lập trình.
- Dùng try có nhiều catch
- Dùng try lồng nhau
- Sử dụng try-catch-finally
- Sử dụng từ khóa throws
- Sử dụng từ khóa throw

Nội dung bài học

- Các loại luồng dữ liệu
- Xử lý nhập xuất bằng luồng byte
- Truy cập file ngẫu nhiên
- Xử lý nhập xuất bằng luồng character
- Sử dụng try... catch trong nhập/xuất
- Chuyển đổi dữ liệu kiểu số

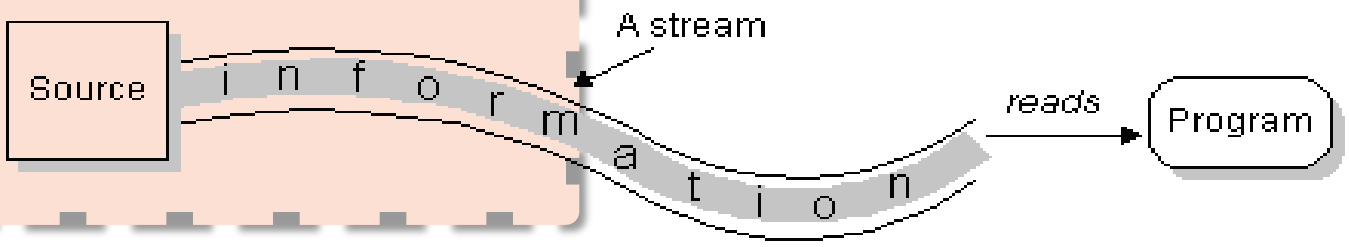
Các loại luồng dữ liệu

Các hoạt động **nhập/xuất** dữ liệu (nhập dữ liệu từ bàn phím, đọc dữ liệu từ file, ghi dữ liệu màn hình, ghi ra file, ghi ra đĩa, ghi ra máy in…) đều được gọi là **luồng** (stream).

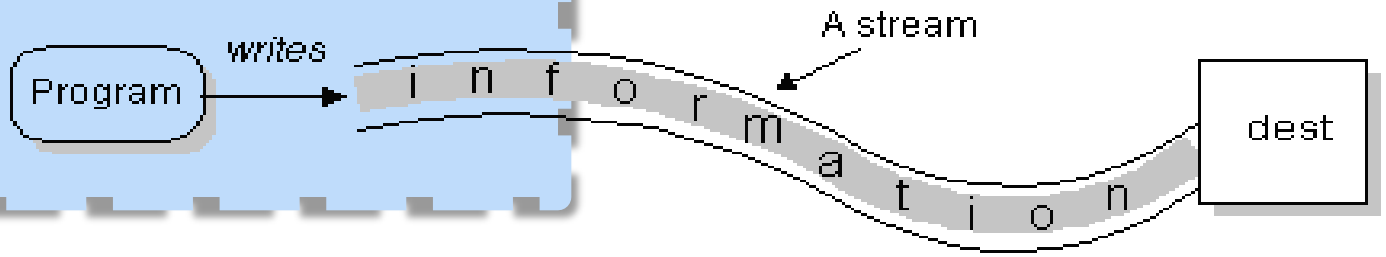
Tất cả các luồng đều có **chung một nguyên tắc** hoạt động ngay cả khi chúng được gắn kết với các thiết bị vật lý khác nhau.

Các loại luồng dữ liệu

Input Streams – lấy dữ liệu từ các nguồn: Files, Buffers và Sockets



Output Streams – ghi dữ liệu vào Files, Buffers in Memory, and Sockets



Các loại luồng dữ liệu

Luồng byte

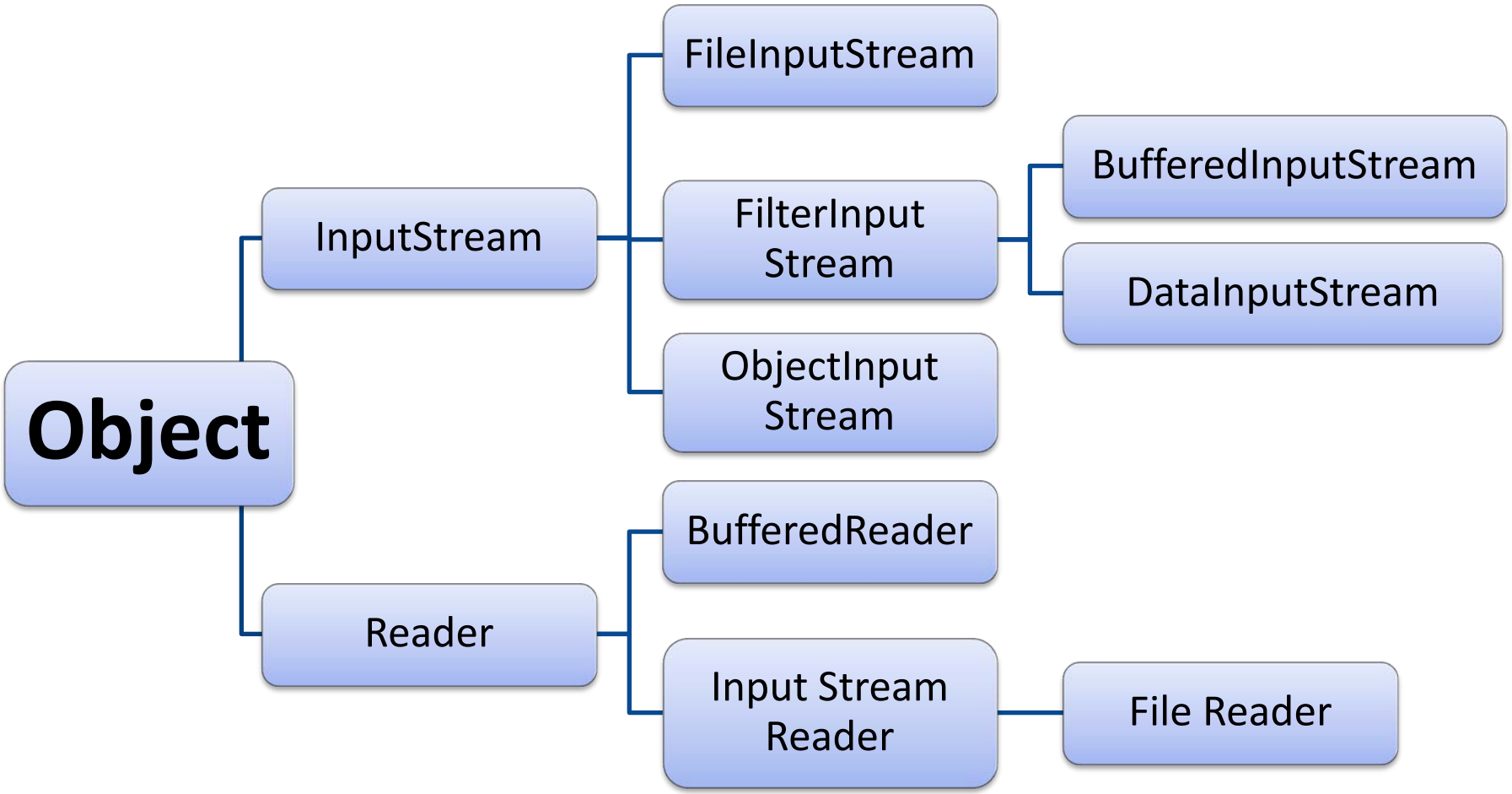
- Hỗ trợ việc xuất nhập dữ liệu trên byte,
- Thường được dùng khi đọc ghi dữ liệu nhị phân.

Luồng character

- Luồng character (ký tự) được thiết kế hỗ trợ việc xuất nhập dữ liệu kiểu ký tự (Unicode)

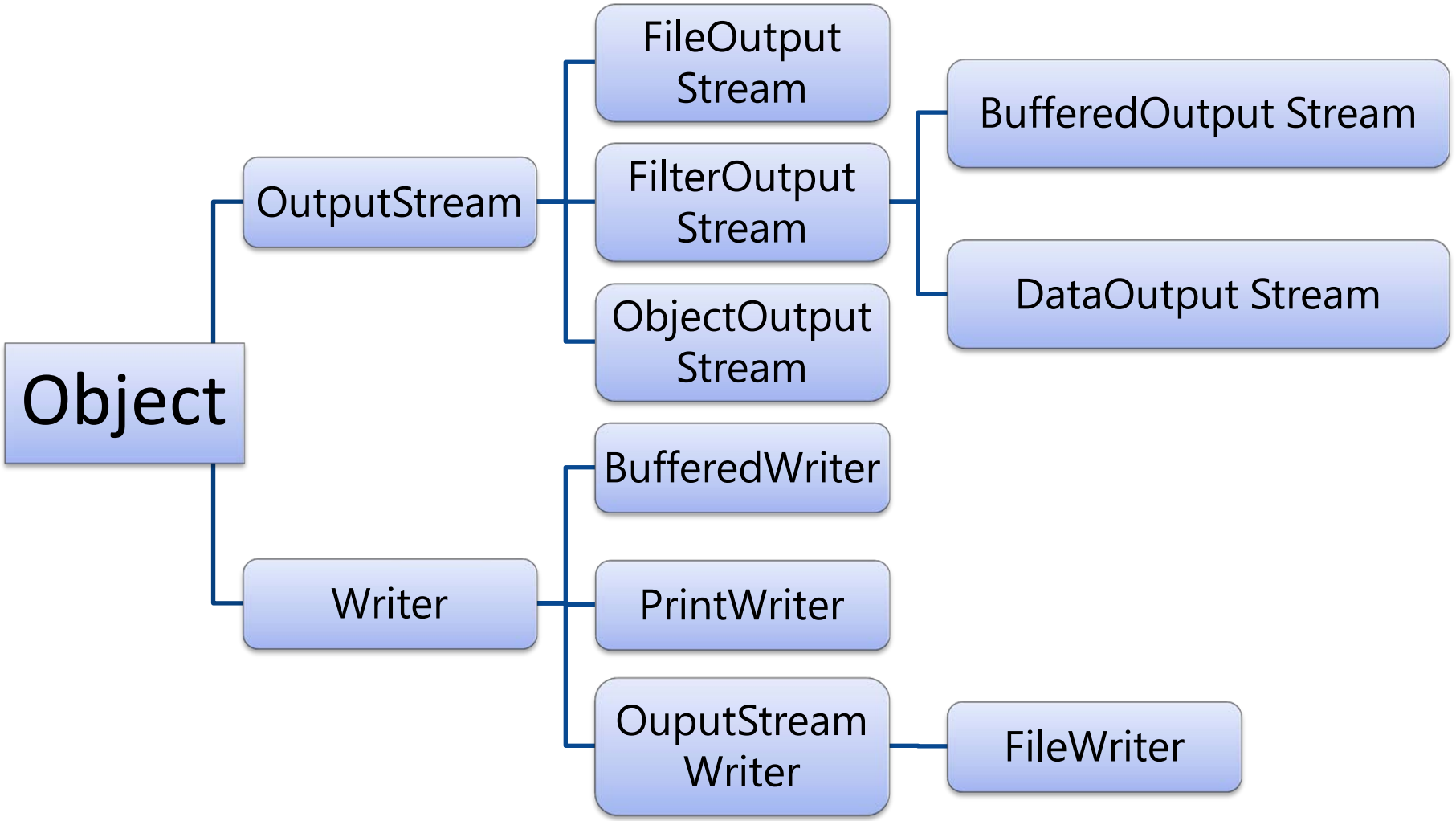
Các loại luồng dữ liệu

Kiến trúc Input Stream
(Luồng nhập dữ liệu)



Các loại luồng dữ liệu

Kiến trúc Output Stream
(Luồng xuất dữ liệu)



Các loại luồng dữ liệu

Luồng byte

- Dữ liệu dạng nhị phân
- 2 class abstract:
  - InputStream
  - OutputStream

Luồng character

- Dữ liệu dạng ký tự Unicode
- 2 class abstract:
  - Reader
  - Writer

Các loại luồng dữ liệu

Các thao tác xử lý dữ liệu:

- import java.io.*
- Tạo đối tượng luồng và liên kết với nguồn dữ liệu
- Thao tác dữ liệu (đọc hoặc ghi hoặc cả đọc và ghi)
- Đóng luồng.

Xử lý nhập xuất dữ liệu bằng luồng byte

Sử dụng luồng mỗi byte để nhập xuất dữ liệu

Tất cả các luồng byte được kế thừa từ 2 class:

- Input Stream
- Output Stream

Có nhiều class luồng byte

- File Input Stream
- File Output Stream

Chúng khác nhau về cách thức khởi tạo nhưng cách thức hoạt động là giống nhau.

Xử lý nhập xuất dữ liệu bằng luồng byte

Sử dụng luồng byte trong các trường hợp:

Nhập xuất kiểu dữ liệu nguyên thủy:

Kiểu int, float, double, String, boolean…

Nhập xuất kiểu dữ liệu khác:

Kiểu object

Xử lý nhập xuất dữ liệu bằng luồng byte

Các class Byte Stream

Byte Stream Class	Meaning
BufferedInputStream	Buffered input stream
BufferedOutputStream	Buffered output stream
ByteArrayInputStream	Input stream that reads from a byte array
ByteArrayOutputStream	Output stream that writes to a byte array
DataInputStream	An input stream that contains methods for reading the Java standard data types
DataOutputStream	An output stream that contains methods for writing the Java standard data types
FileInputStream	Input stream that reads from a file
FileOutputStream	Output stream that writes to a file
FilterInputStream	Implements **InputStream**
FilterOutputStream	Implements **OutputStream**
InputStream	Abstract class that describes stream input
ObjectInputStream	Input stream for objects
ObjectOutputStream	Output stream for objects
OutputStream	Abstract class that describes stream output
PipedInputStream	Input pipe
PipedOutputStream	Output pipe
PrintStream	Output stream that contains **print( )** and **println( )**
PushbackInputStream	Input stream that allows bytes to be returned to the stream
SequenceInputStream	Input stream that is a combination of two or more input streams that will be read sequentially, one after the other

Xử lý nhập xuất dữ liệu bằng luồng byte

Ví dụ 1: Tạo file 'file1.dat' và ghi dữ liệu

```
import java.io.FileOutputStream;
import java.io.IOException;

public class Example1 {

    public static void main(String[] args) throws IOException {
        FileOutputStream fos = new FileOutputStream("file1.dat");
        String text = "The quick brown fox jumped over the lazy dog";
        byte[] textAsBytes = text.getBytes();
        fos.write(textAsBytes);
    }
}
```

Xử lý nhập xuất dữ liệu bằng luồng byte

Ví dụ 2: Đọc thông tin từ file 'file1.dat' và in ra màn hình

```
public class Example2 {

    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("file1.dat");
        int c;
        while ((c = fis.read()) != -1) {
            System.out.print((char) c);
        }
        fis.close();
    }
}
```

Xử lý nhập xuất dữ liệu bằng luồng byte

Đọc, ghi dữ liệu nhị phân (binary data)

Khi muốn tạo file chứa các kiểu dữ liệu như short, int, long, float, double, String, boolean… thì sử dụng 2 class:

Class **DataInputStream** xử lý việc nhập dữ liệu

Class **DataOutputStream** xử lý việc xuất dữ liệu

Xử lý nhập xuất dữ liệu bằng luồng byte

Một số phương thức xử lý dữ liệu nhị phân của class **DataOutputStream**

Output Method	Purpose
void writeBoolean(boolean *val*)	Writes the **boolean** specified by *val*.
void writeByte(int *val*)	Writes the low-order byte specified by *val*.
void writeChar(int *val*)	Writes the value specified by *val* as a **char**.
void writeDouble(double *val*)	Writes the **double** specified by *val*.
void writeFloat(float *val*)	Writes the **float** specified by *val*.
void writeInt(int *val*)	Writes the **int** specified by *val*.
void writeLong(long *val*)	Writes the **long** specified by *val*.
void writeShort(int *val*)	Writes the value specified by *val* as a **short**.

Xử lý nhập xuất dữ liệu bằng luồng byte

Một số phương thức xử lý dữ liệu nhị phân của class **DataInputStream:**

Input Method	Purpose
boolean readBoolean( )	Reads a **boolean**.
byte readByte( )	Reads a **byte**.
char readChar( )	Reads a **char**.
double readDouble( )	Reads a **double**.
float readFloat( )	Reads a **float**.
int readInt( )	Reads an **int**.
long readLong( )	Reads a **long**.
short readShort( )	Reads a **short**.

Xử lý nhập xuất dữ liệu bằng luồng byte

Ví dụ 1: Ghi dữ liệu

```
public class DataStreamOutput {
    public static void main(String[] args) throws IOException {
        FileOutputStream fos = new FileOutputStream("filedata.dat");
        DataOutputStream dos = new DataOutputStream(fos);
        final int NUMBER = 5;
        dos.writeInt(NUMBER);
        for (int i = 0; i <= NUMBER; i++) {
            dos.writeInt(i);
        }
        dos.writeUTF("Hello !");
        dos.writeDouble(100.75);
        dos.flush();
        dos.close();
    }
}
```

Xử lý nhập xuất dữ liệu bằng luồng byte

Ví dụ 2: Đọc dữ liệu

```
public class DataStreamInput {

    public static void main(String[] args) throws IOException {
        FileInputStream fis = new FileInputStream("filedata.dat");
        DataInputStream dis = new DataInputStream(fis);
        int items = dis.readInt();
        for (int i = 0; i <= items; i++) {
            int n = dis.readInt();
            System.out.print(n + " ");
        }
        System.out.println(dis.readUTF());
        System.out.println(dis.readDouble());
        dis.close();
    }
}
```

Xử lý nhập xuất dữ liệu bằng luồng byte

Ví dụ 3: Đọc, ghi dữ liệu kiểu object

```
public class Stock implements Serializable {
    private int id;
    private String desc;
    private double price;
    private int quantity;
    public Stock(int id, String desc, double price, int quantity){
        this.id = id;
        this.desc = desc;
        this.price = price;
        this.quantity = quantity;
    }
    public String toString() {
        return (id+" "+desc + " " + price + " " + quantity);
    }
}
```

Xử lý nhập xuất dữ liệu bằng luồng byte

Ví dụ 3: Đọc, ghi dữ liệu kiểu object

```
public class ObjectExampleWrite {
    public static void main(String[] args) throws
            IOException, ClassNotFoundException {
        FileOutputStream fos = new FileOutputStream("fileobject.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        Stock[] stocks = {new Stock(1001, "CD ROM", 100.00, 20),
            new Stock(1002, "DRAM", 75.00, 30),
            new Stock(1003, "P4 Processor", 300.00, 100),
            new Stock(1004, "Canon Jet", 80.00, 10),
            new Stock(1005, "HP Scanner", 75.00, 90)};
        //Ghi mang doi tuong vao file 'fileobject.dat'
        oos.writeObject(stocks);
        oos.close();
    }
}
```

Xử lý nhập xuất dữ liệu bằng luồng byte

Ví dụ 2: Đọc, ghi dữ liệu kiểu object

```
public static void main(String[] args) {
    FileInputStream fis = null;
    ObjectInputStream ois = null;
    try {
        fis = new FileInputStream("fileobject.dat");
        ois = new ObjectInputStream(fis);
        Stock[] stocks1 = (Stock[]) ois.readObject();
        System.out.println("Doc tu file: ");
        for (Stock s : stocks1) {
            System.out.println(s);
        }
        ois.close();fis.close();

    } catch (Exception e) {
        System.out.println("Co loi: " + e);
    }
}
```

Truy cập file ngẫu nhiên

- Sử dụng object RandomAccessFile để truy cập ngẫu nhiên nội dung một file.
- RandomAccessFile là class thực thi 2 interface là DataInput và DataOutput trong đó có định nghĩa các phương thức input/output.

Dùng phương thức :

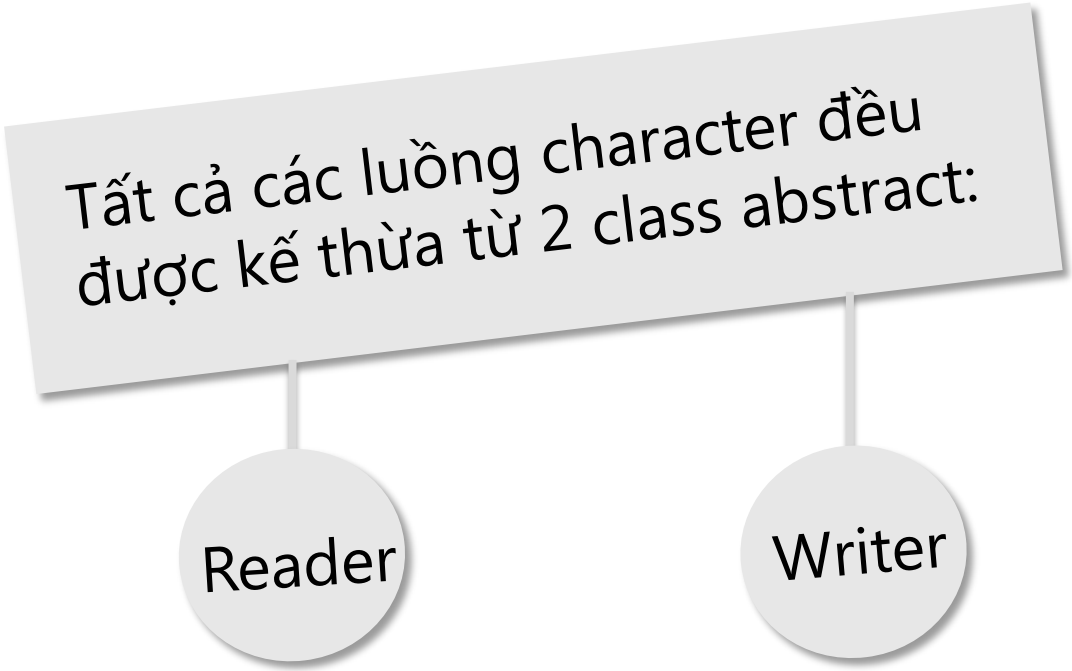
- **seek(vị_trí)**: để di chuyển con trỏ file tới vị_trí mới.
- **seek(0)**: Di chuyển con trỏ tới đầu file
- **seek(file.length())**: Di chuyển con trỏ tới cuối file

Truy cập file ngẫu nhiên

```
try {
    RandomAccessFile file = new RandomAccessFile("rand.txt", "rw");
    file.writeChar('a');file.writeInt(100);file.writeDouble(8.75);
    file.seek(0);  //Dich chuyen con tro ve dau file va doc du lieu
    System.out.println(file.readChar());
    System.out.println(file.readInt());
    System.out.println(file.readDouble());
    file.seek(2);  //Dich chuyen con tro file vao vi tri thu 2
    System.out.println("Vi tri thu 2: " + file.readInt());
    file.seek(file.length()); //Dich chuyen con tro toi cuoi file
    file.writeBoolean(true);
    file.seek(4);          //Dich chuyen con tro file vao vi tri thu 4
    System.out.println("Vi tri thu 4: " + file.readBoolean());
    file.close();
} catch (Exception e) {
    System.out.println("Co loi: " + e);
}
```

Xử lý nhập xuất dữ liệu bằng luồng character

Luồng byte rất mạnh mẽ và linh hoạt. Tuy nhiên nếu bạn muốn **lưu trữ file** chứa **văn bản Unicode** thì luồng **character** là lựa chọn **tốt nhất** vì ưu điểm của luồng character là nó thao tác trực tiếp trên ký tự Unicode.



Xử lý nhập xuất dữ liệu bằng luồng character

Các class:

Character Stream Class	Meaning
BufferedReader	Buffered input character stream
BufferedWriter	Buffered output character stream
CharArrayReader	Input stream that reads from a character array
CharArrayWriter	Output stream that writes to a character array
FileReader	Input stream that reads from a file
FileWriter	Output stream that writes to a file
FilterReader	Filtered reader
FilterWriter	Filtered writer
InputStreamReader	Input stream that translates bytes to characters
LineNumberReader	Input stream that counts lines
OutputStreamWriter	Output stream that translates characters to bytes
PipedReader	Input pipe
PipedWriter	Output pipe
PrintWriter	Output stream that contains **print( )** and **println( )**
PushbackReader	Input stream that allows characters to be returned to the input stream
Reader	Abstract class that describes character stream input
StringReader	Input stream that reads from a string
StringWriter	Output stream that writes to a string
Writer	Abstract class that describes character stream output

Xử lý nhập xuất dữ liệu bằng luồng character

Ví dụ 1

```
File filename = new File("first.txt");
try {
    FileWriter out = new FileWriter(filename);
    out.write("Doc ghi du lieu trong Java!");
    out.write("\n");                    //GHI VAO FILE
    out.write("Su dung Stream Character");
    out.close();
    //DOC TU FILE TEXT
    FileReader input = new FileReader(filename);
    System.out.println("Doc tu file first.txt:");
    int ch = input.read();
    while (ch != -1) {
        System.out.print((char) ch);
        ch = input.read();          //DOC TU FILE
    }
} catch (Exception e) {
```

Xử lý nhập xuất dữ liệu bằng luồng character

Ví dụ 2: Ghi vào file mảng String Student:

```
public void writeToFileText(String FileName) throws IOException {
      FileWriter fw = new FileWriter(FileName);
      BufferedWriter bw = new BufferedWriter(fw);
      for (int i = 0; i < this.count; i++) {
            String temp = Students[i].toString();
            bw.write(temp); //GHI VAO FILE
      }
      bw.flush();
      bw.close();
 }
```

Xử lý nhập xuất dữ liệu bằng luồng character

Ví dụ 3: Đọc dữ liệu từ file và hiển thị ra màn hình:

```
public void readFromFileText(String FileName) throws IOException,
        ClassNotFoundException {
    FileReader frr = new FileReader(FileName);
    BufferedReader br = new BufferedReader(frr);
    String text;
    while ((text = br.readLine()) != null) {
        System.out.println(text);
    }
    br.close();
}
```

Sử dụng try… catch trong nhập xuất

Khi **input/output** dữ liệu, có những **ngoại lệ 'checked'** nên bắt buộc phải **catch** khi viết **code**, thông thường các ngoại lệ đó là:

IOException

FileNotFoundException

EOFException

NotSerializableException

…

Sử dụng try... catch trong nhập xuất

```
FileOutputStream fos = null;
try {
    fos = new FileOutputStream("file1.dat");
    String text = "The quick brown fox jumped over the lazy dog";
    byte[] textAsBytes = text.getBytes();
    fos.write(textAsBytes);
} catch (FileNotFoundException ex) {
    System.out.println("Khong tim thay file: " + ex);
} catch (IOException ex) {
    System.out.println("Co loi : " + ex);
} finally {
    try {
        fos.close();
    } catch (IOException ex) {
        System.out.println("Co loi: "+ex);
    }
}
```

Chuyển đổi dữ liệu kiểu số

- Mỗi một kiểu dữ liệu nguyên thủy trong Java đều có một class dành riêng cho nó. Các class đó được gọi là lớp ‘bao bọc’, bởi vì nó “bọc" kiểu dữ liệu nguyên thủy vào một đối tượng của chính lớp đó.
- Vì vậy, có một lớp Integer chứa một biến int, có một lớp Double chứa một biến double…
- Các lớp bao bọc là một phần của gói java.lang, được import mặc định vào tất cả các chương trình Java.

Chuyển đổi dữ liệu kiểu số

Mỗi một kiểu dữ liệu nguyên thủy trong Java đều có một class dành riêng cho nó.

Primitive data type	Wrapper class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Chuyển đổi dữ liệu kiểu số

Kiến trúc của class wrapper

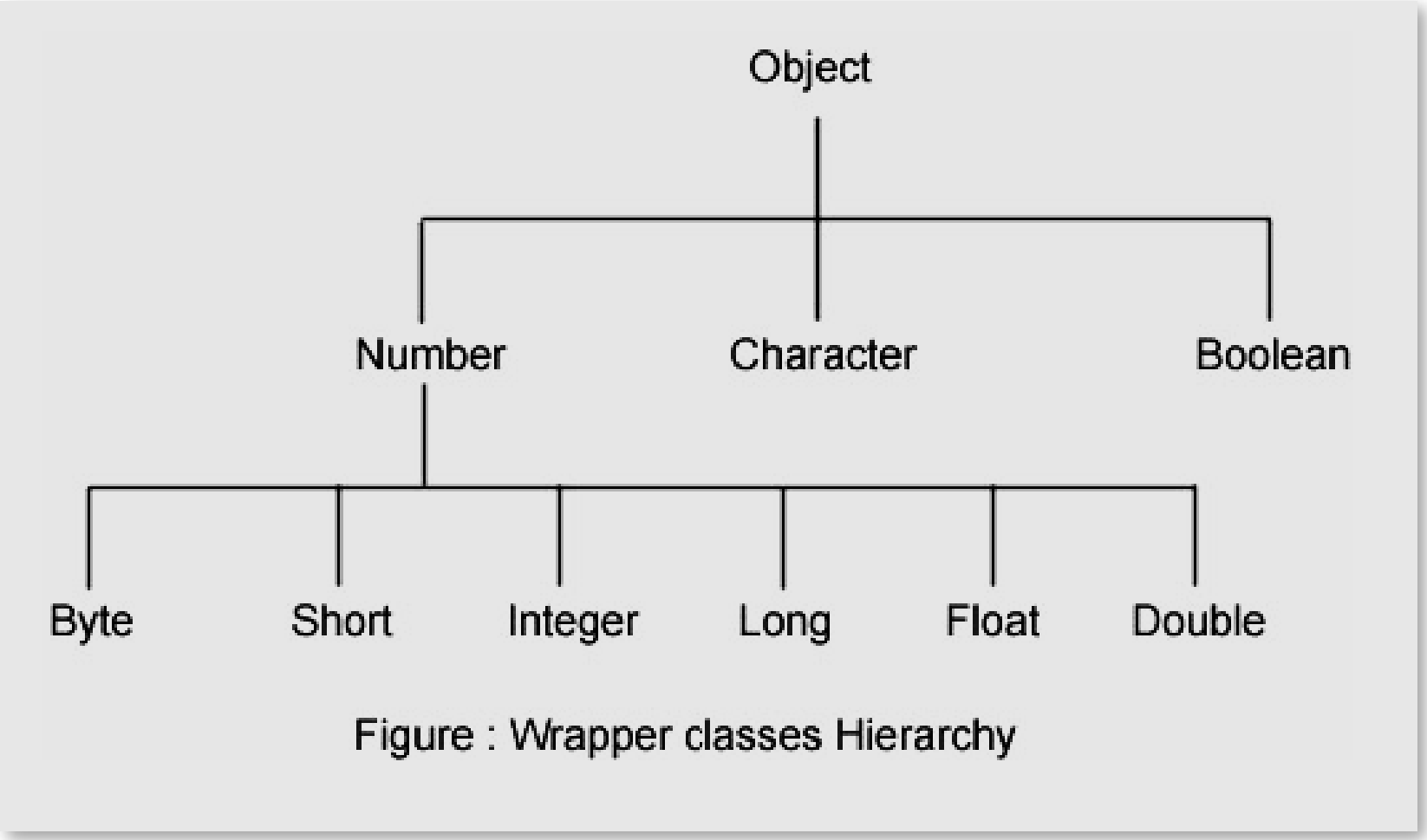


Figure : Wrapper classes Hierarchy

Chuyển đổi dữ liệu kiểu số

Ví dụ về phạm vi của các kiểu dữ liệu:

```
 1  public class FindRanges
 2  {
 3     public static void main(String[] args)
 4     {
 5        System.out.println("Integer range:");
 6        System.out.println("   min: " + Integer.MIN_VALUE);
 7        System.out.println("   max: " + Integer.MAX_VALUE);
 8        System.out.println("Double range:");
 9        System.out.println("   min: " + Double.MIN_VALUE);
10        System.out.println("   max: " + Double.MAX_VALUE);
11        System.out.println("Long range:");
12        System.out.println("   min: " + Long.MIN_VALUE);
13        System.out.println("   max: " + Long.MAX_VALUE);
14        System.out.println("Short range:");
15        System.out.println("   min: " + Short.MIN_VALUE);
16        System.out.println("   max: " + Short.MAX_VALUE);
17        System.out.println("Byte range:");
18        System.out.println("   min: " + Byte.MIN_VALUE);
19        System.out.println("   max: " + Byte.MAX_VALUE);
20        System.out.println("Float range:");
21        System.out.println("   min: " + Float.MIN_VALUE);
22        System.out.println("   max: " + Float.MAX_VALUE);
23     }
24  }
```

Chuyển đổi dữ liệu kiểu số

Có 2 ưu điểm chính của class wrapper:

- Biến đổi các kiểu dữ liệu nguyên thủy thành dữ liệu kiểu đối tượng.
- Convert kiểu String thành các dạng kiểu dữ liệu khác, là các phương thức có dạng parseXXX().

Chuyển đổi dữ liệu kiểu số

Wrapper	Conversion Method
Double	static double parseDouble(String *str*) throws NumberFormatException
Float	static float parseFloat(String *str*) throws NumberFormatException
Long	static long parseLong(String *str*) throws NumberFormatException
Integer	static int parseInt(String *str*) throws NumberFormatException
Short	static short parseShort(String *str*) throws NumberFormatException
Byte	static byte parseByte(String *str*) throws NumberFormatException

Chuyển đổi dữ liệu kiểu số

Ví dụ:

```
float a = Float.parseFloat(str1);
float b = Float.parseFloat(str2);
System.out.println("a + b = " + (a + b));
System.out.println("a - b = " + (a - b));
System.out.println("a * b = " + (a * b));
System.out.println("a / b = " + (a / b));
System.out.println("a % b = " + (a % b));
```

Tổng kết bài học

- Các loại luồng dữ liệu
- Xử lý nhập xuất bằng luồng byte
- Truy cập file ngẫu nhiên
- Xử lý nhập xuất bằng luồng character
- Sử dụng try… catch trong nhập/xuất
- Chuyển đổi dữ liệu kiểu số