FPT Education

FPT POLYTECHNIC

Lập trình Java 2

Bài 5: Luồng dữ liệu vào/ra

Phần 1

www.poly.edu.vn

NỘI DUNG BÀI HỌC

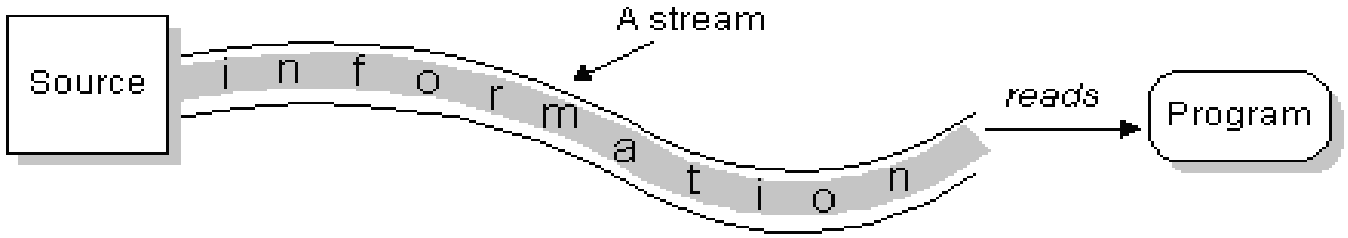
- Giải thích các loại luồng dữ liệu
- Nhập xuất các luồng byte
- Nhập xuất các luồng character
- Sử dụng try… catch trong nhập/xuất

GIỚI THIỆU LUỒNG DỮ LIỆU

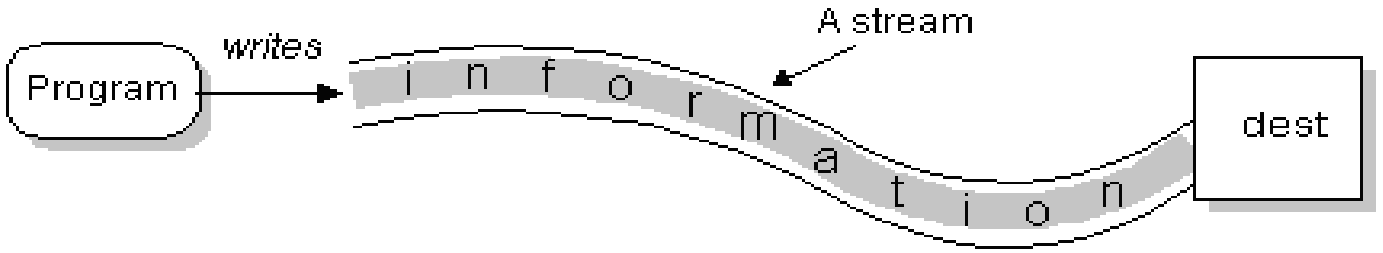
- Các hoạt động nhập/xuất dữ liệu (nhập dữ liệu từ bàn phím, đọc dữ liệu từ file, ghi dữ liệu màn hình, ghi ra file, ghi ra đĩa, ghi ra máy in…) đều được gọi là luồng (stream).
- Tất cả các luồng đều có **chung một nguyên tắc** hoạt động ngay cả khi chúng được gắn kết với các thiết bị vật lý khác nhau.

GIỚI THIỆU LUỒNG DỮ LIỆU

- Luồng vào là luồng cho phép chương trình đọc dữ liệu từ một nguồn nào đó: bán phím, file, máy scan…



- Luồng ra là luồng cho phép chương trình ghi dữ liệu lên nó để chuyển đến đích nào đó: màn hình, file, máy in…

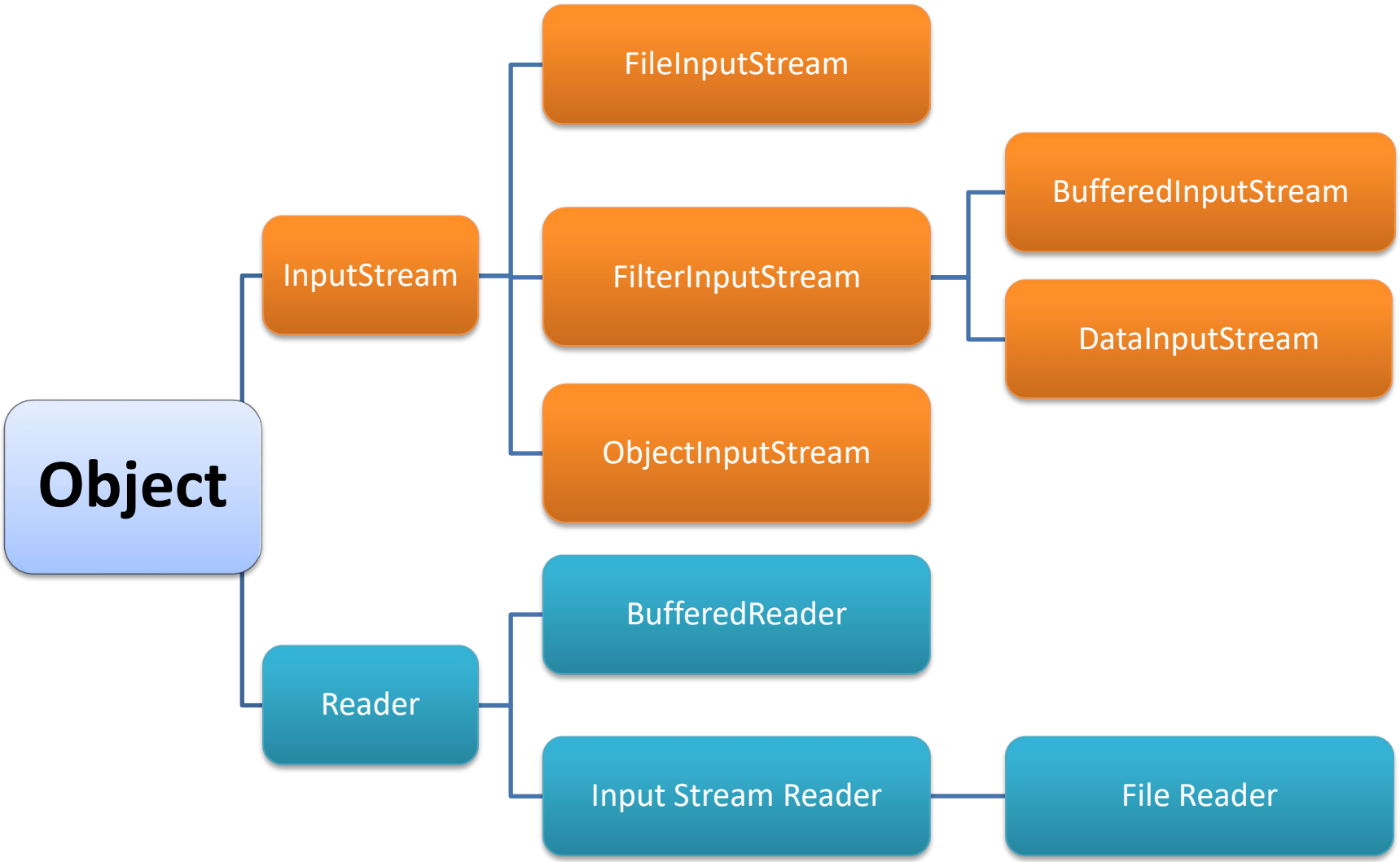


CÁC LOẠI LUỒNG DỮ LIỆU

- Có 2 kiểu luồng trong Java:
  - Luồng byte (luồng nhị phân)
  - Luồng character (luồng văn bản)
- Luồng byte
  - Hỗ trợ việc xuất nhập dữ liệu theo byte,
  - Thường được dùng khi đọc ghi dữ liệu nhị phân.
- Luồng character
  - Luồng character được thiết kế hỗ trợ việc xuất nhập dữ liệu kiểu ký tự

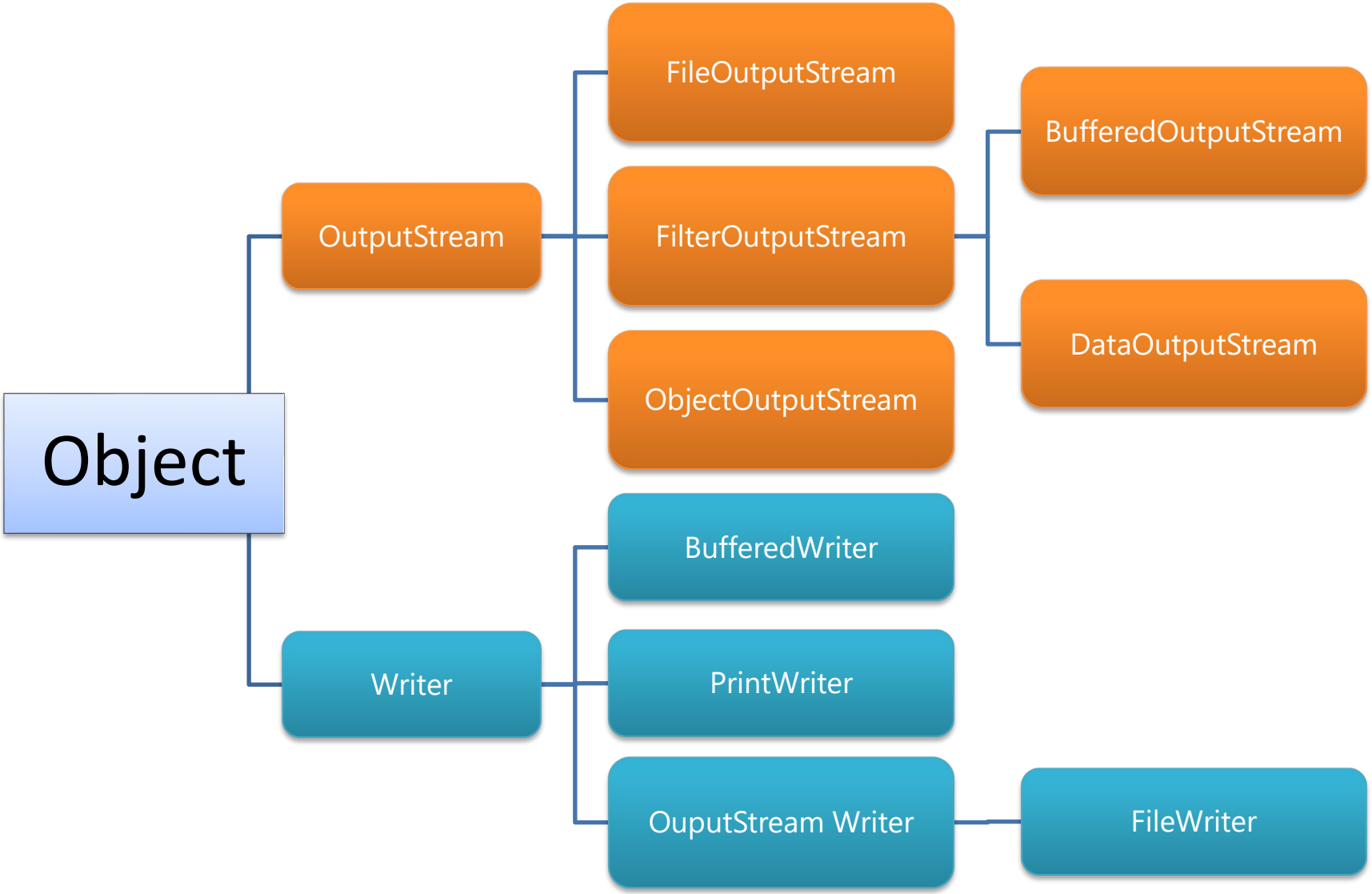
FPT POLYTECHNIC

KIẾN TRÚC PHÂN CẤP CÁC LOẠI LUỒNG VÀO



FPT POLYTECHNIC

KIẾN TRÚC PHÂN CẤP CÁC LOẠI LUỒNG RA



XỬ LÝ LUỒNG BYTE

- Sử dụng luồng mỗi byte để nhập xuất dữ liệu nhị phân
- Tất cả các luồng byte được kế thừa từ 2 class:
  - InputStream
  - OutputStream
- 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.

FileInputStream/FileOutputStream

- Cập luồng này được sử dụng để làm việc với file nhị phân
  - Sử dụng FileInputStream để đọc dữ liệu từ file nhị phân
  - Sử dụng FileOutputStream để ghi dữ liệu vào file nhị phân

FileOutputStream

- FileOutputStream là luồng ra được sử dụng để ghi dữ liệu ra file nhị phân.

```
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);
    }
}
```

- FileInputStream là luồng dữ liệu vào từ file nhị phân

```
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();
    }
}
```

DataInputStream/DataOutputStream

- 2 luồng này giúp chúng ta đọc/ghi dữ liệu nguyên thủy
- Sử dụng read**<Type>**() để đọc dữ liệu nguyên thủy từ DataInputStream
  - read**Int**()
  - read**Double**()…
- Sử dụng write**<Type>**(Type) để ghi dữ liệu nguyên thủy lên DataOutputStream
  - write**Boolen**(boolean)
  - write**Int**(int)…

DATAOUTPUTSTREAM

```
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();
    }
}
```

DataInputStream

```
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();
    }
}
```

ObjectInputStream/Object OutputStream

- Cập luồng này giúp chúng ta đọc/ghi đối tượng
- Sử dụng readObject() để đọc đối tượng từ DataInputStream
- Sử dụng writeObject(Serializable) để ghi đối tượng lên DataOutputStream
- Chú ý:
  - Chỉ các đối tượng được tạo từ các lớp có thực thi theo interface Serializable mới có thể đọc ghi được.

LỚP THỰC THI INTERFACE SERIALIZABLE

```
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);
    }
}
```

OBJECTOUTPUTSTREAM

```
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();
    }
}
```

OBJECTINPUTSTREAM

```
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);
    }
}
```

FPT Education

FPT POLYTECHNIC

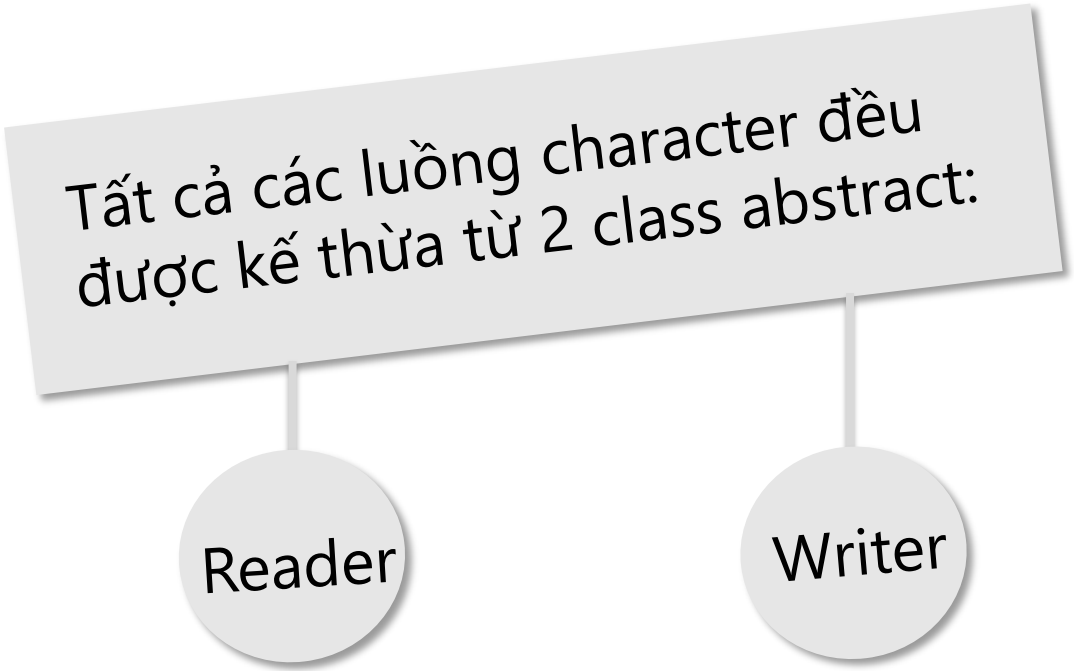
LẬP TRÌNH JAVA 2

BÀI 5: LUỒNG DỮ LIỆU VÀO/RA

PHẦN 2

www.poly.edu.vn

- 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.



FileReader/FileWriter

- Cập luồng này được sử dụng để làm việc với với luồng character
- Sử dụng FileReader để làm việc với luồng vào file văn bản
- Sử dụng FileWriter để làm việc với luồng ra file văn bản

FileReader/FileWriter

```
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) {
```

BufferedReader/BufferedWriter

- Cập luồng này được sử dụng để làm việc với với luồng đệm character
- Sử dụng BufferedReader để làm việc với luồng đệm văn bản vào
- Sử dụng BufferedWriter để làm việc với luồng đệm văn bản ra

BUFFEREDWRITER

```
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();
  }
```

BUFFEREDREADER

```
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:
  - FileNotFoundException
  - EOFException
  - NotSerializableException
  - IOException

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);
    }
}
```

TỔNG KẾT NỘI DUNG BÀI HỌC

- Giải thích các loại luồng dữ liệu
- Nhập xuất các luồng byte
- Nhập xuất các luồng character
- Sử dụng try... catch trong nhập/xuất