Lập trình JAVA cơ bản

Tuần 6

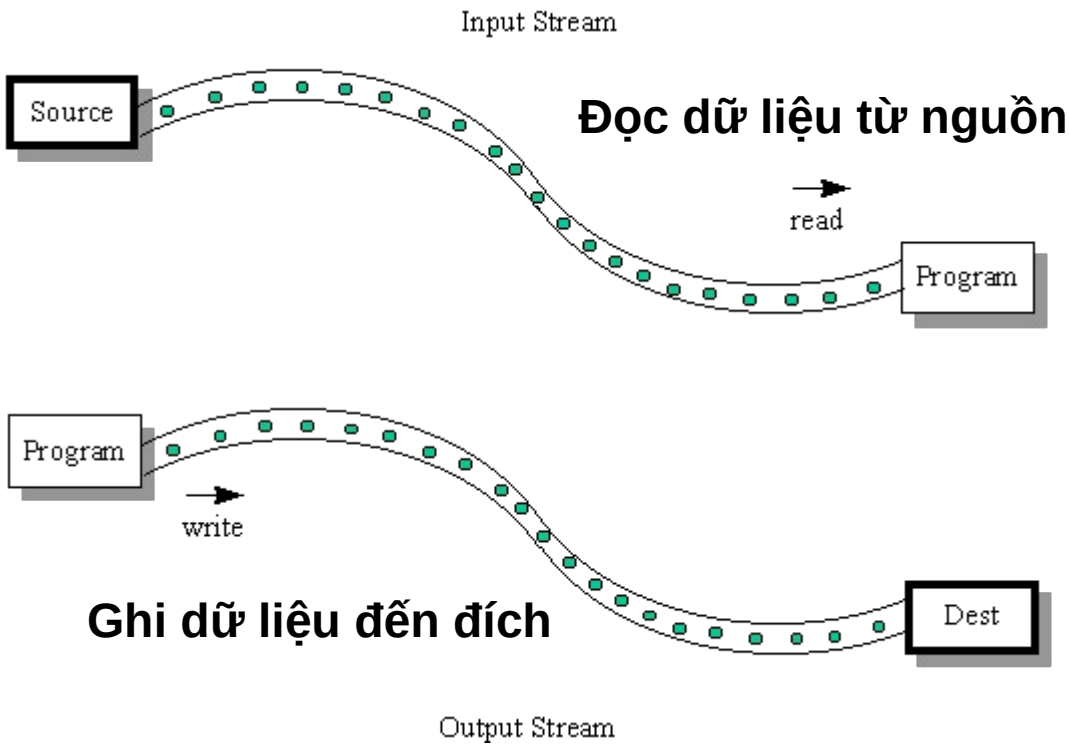
Giảng viên: Trần Đức Minh

Nội dung bài giảng

- Luồng trong Java
- Làm việc với File
  - Nhập/Xuất mảng bytes
  - Nhập/Xuất File dưới dạng byte.
- Luồng lọc
- Nhập/xuất dữ liệu sơ cấp
- System.in, System.out
- Luồng nhập/xuất đối tượng
- Luồng ký tự
- Lớp File

Luồng trong Java

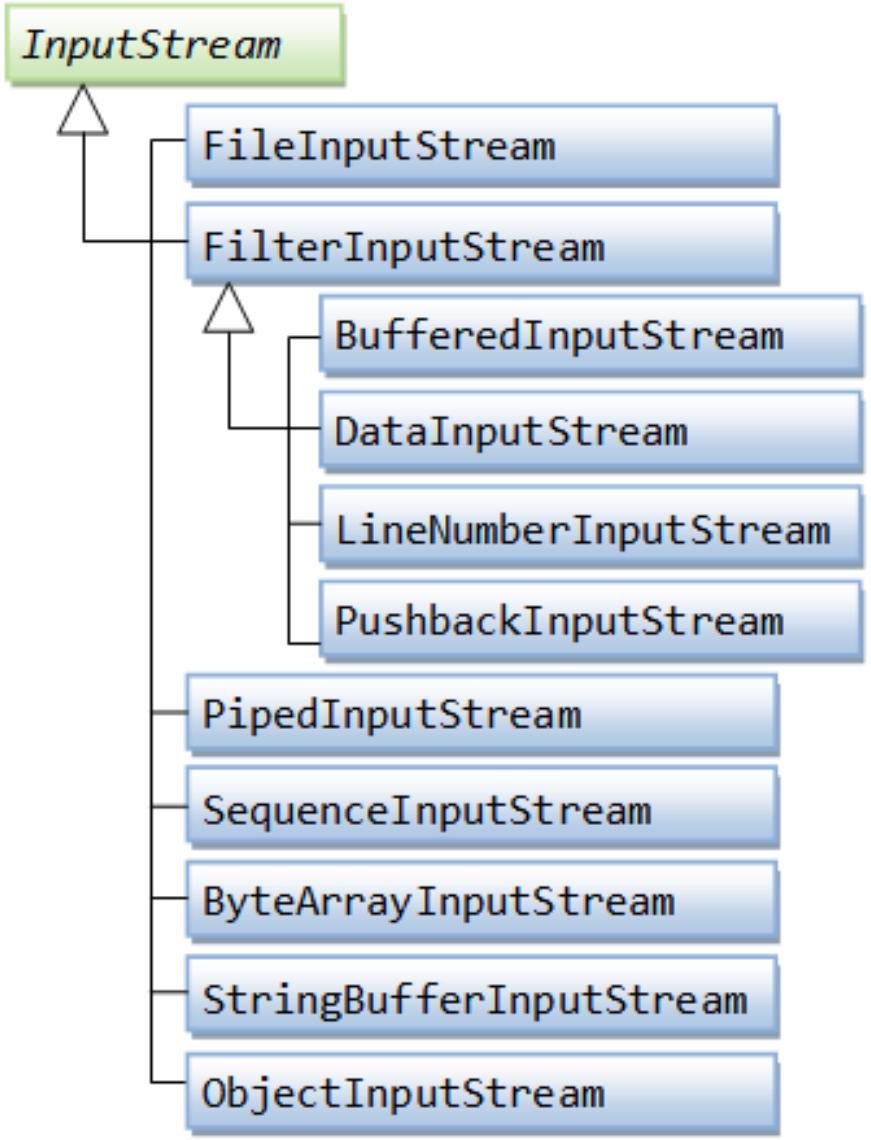
- **Luồng** (stream) trong Java là một dãy dữ liệu
- **Đọc** và **ghi** luồng trong Java

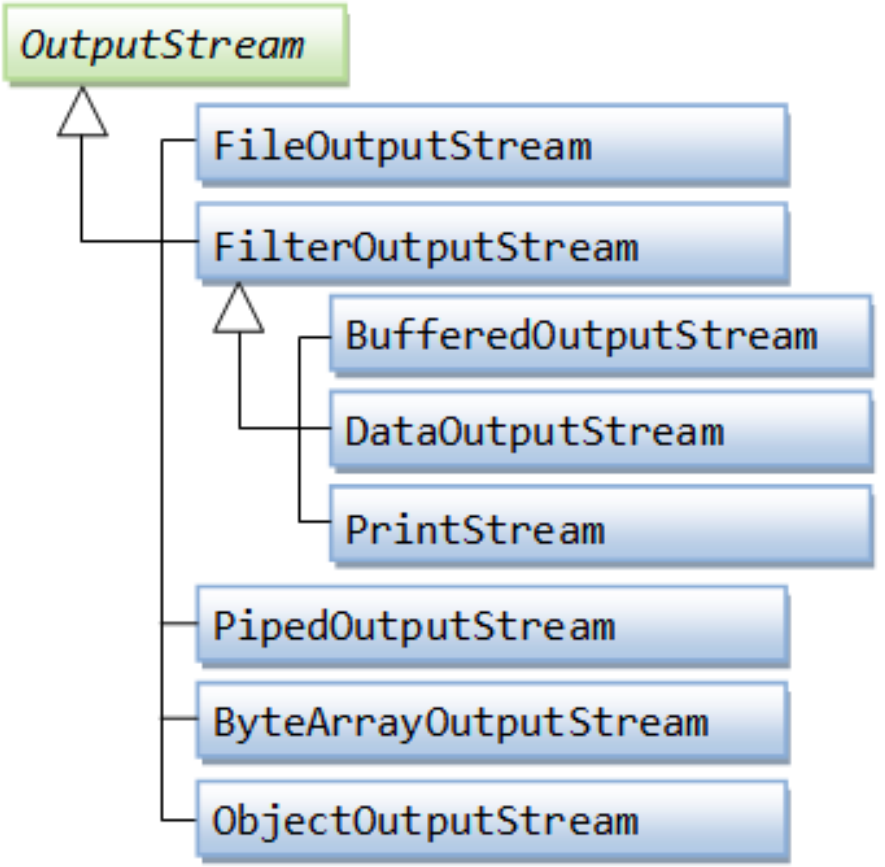


File trong Java

- Có 2 kiểu luồng làm việc với File trong Java
  - Luồng byte
    - Thực hiện nhập xuất dữ liệu dưới dạng byte.
    - Thường dùng khi làm việc với dữ liệu nhị phân.
    - Sử dụng các lớp kế thừa từ **2 lớp trừu tượng**: **InputStream** và **OutputStream**.
      - Đây là hai lớp cung cấp các phương thức để đọc/ghi với dữ liệu phi cấu trúc ở mức byte.
  - Luồng ký tự
    - Thực hiện nhập xuất dữ liệu kiểu ký tự dạng Unicode.
    - Sử dụng các lớp kế thừa từ **2 lớp trừu tượng**: **Reader** và **Writer**.
- Các lớp làm việc với luồng nằm trong gói **java.io**

Mô hình các lớp luồng nhập/xuất dữ liệu





Lớp InputStream

- Một số phương thức của lớp trừu tượng InputStream
  - abstract int read() **throws IOException**
    - Đọc một byte từ luồng.
    - Nếu cuối luồng sẽ trả về -1
  - int read(byte[] b) throws IOException
    - Đọc một dãy byte từ luồng
  - void close() throws IOException
    - Đóng luồng nhập
  - int available() throws IOException
    - Trả về số byte có thể đọc tiếp
  - long skip(long n) throws IOException
    - Bỏ qua n byte

Lớp OutputStream

- Một số phương thức của lớp OutputStream
  - abstract void write(int b) throws IOException
    - Ghi một byte ra luồng
  - void write(byte[] b) throws IOException
    - Ghi một dãy byte ra luồng
  - void close() throws IOException
    - Đóng luồng

Nhập/xuất mảng bytes

- Lớp **ByteArrayInputStream**:
  - Tạo luồng đầu vào tới bộ nhớ
  - Là một mảng các byte.
  - Không hỗ trợ các phương thức mới mà **chỉ nạp chồng lên các phương thức của lớp InputStream**.
- Lớp **ByteArrayOutputStream**:
  - Tạo ra luồng kết suất trên một mảng các byte.
  - Cung cấp các phương thức ‘toByteArrray()’ và ‘toString()’ được dùng để chuyển đổi luồng thành một mảng byte hay đối tượng chuỗi.

Ví dụ nhập/xuất mảng bytes

```
import jạva.io.*;
public class MainClass {
    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 = 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));
    }
}
```

Nhập/xuất với luồng file

- Luồng nhập từ file: **FileInputStream** được sử dụng để đọc dữ liệu dưới dạng byte.
  - FileInputStream(String name)
  - FileInputStream(File f)
- Luồng xuất ra file: **FileOutputStream** được sử dụng để ghi dữ liệu dưới dạng byte.
  - FileOutputStream(String name)
  - FileOutputStream(File f)
  - FileOutputStream(String name, boolean append)
- Phương thức nhập/xuất của các luồng file giống như của các luồng nhập xuất cơ bản

Lớp FileOutputStream

- **FileOutputStream** được sử dụng để ghi dữ liệu dưới dạng byte.

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

public class MainClass {
    public static void main(String args[]) throws IOException {
        FileOutputStream fout = null;
        try {
            fout = new FileOutputStream( name: "test_output.txt");

            String strHello = "Hello Java Class";
            int value = 65;

            fout.write(value);
            fout.write(strHello.getBytes());

            fout.close();

            System.out.println("Finish !!!");
        } catch (Exception e) {
            System.out.println(e);
        } finally {
            fout.close();
        }
    }
}
```

Lớp FileInputStream

- **FileInputStream** được sử dụng để đọc dữ liệu dưới dạng byte.

```java
import java.io.FileInputStream;
import java.io.IOException;

public class MainClass {
    public static void main(String args[]) throws IOException {
        FileInputStream fin = null;
        try {
            fin = new FileInputStream( name: "test_output.txt");
            int i = 0;
            while ((i = fin.read()) != -1) {
                System.out.print((char) i);
            }
            fin.close();
        } catch (Exception e) {
            System.out.println(e);
        } finally {
            fin.close();
        }
    }
}
```

Lớp BufferedOutputStream

- **BufferedOutputStream** được sử dụng để tạo bộ đệm (buffer) cho một luồng xuất trong bộ nhớ. Điều này giúp cho việc truy xuất dữ liệu được nhanh chóng.

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

public class MainClass {
    public static void main(String args[]) throws IOException {
        FileOutputStream fout = null;
        BufferedOutputStream bout = null;
        try {
            fout = new FileOutputStream( name: "test_output.txt");
            bout = new BufferedOutputStream(fout);

            String strHello = "Hello Java Class";
            int value = 97;

            bout.write(value);
            bout.write(strHello.getBytes());
            bout.flush();

            System.out.println("Finish !!!");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            bout.close();
            fout.close();
        }
    }
}
```

Lớp BufferedInputStream

- **BufferedInputStream** được sử dụng để đọc thông tin từ bộ đệm (buffer) từ một luồng nhập. Điều này giúp cho việc đọc dữ liệu được nhanh hơn.

```java
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.IOException;

public class MainClass {
    public static void main(String args[]) throws IOException {
        FileInputStream fin = null;
        BufferedInputStream bin = null;
        try {
            fin = new FileInputStream( name: "test_output.txt");
            bin = new BufferedInputStream(fin);
            int i;
            while ((i = bin.read()) != -1) {
                System.out.print((char) i);
            }
        } catch (Exception e) {
            System.out.println(e);
        } finally {
            bin.close();
            fin.close();
        }
    }
}
```

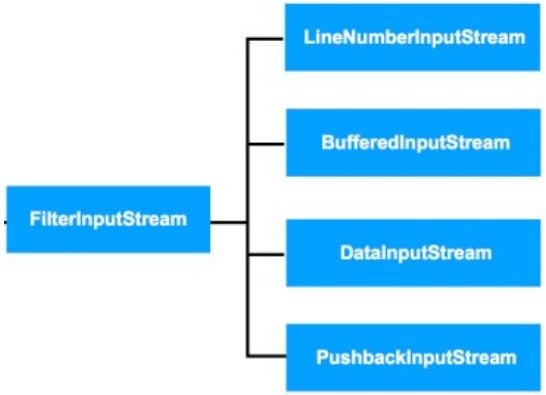
Lớp SequenceInputStream

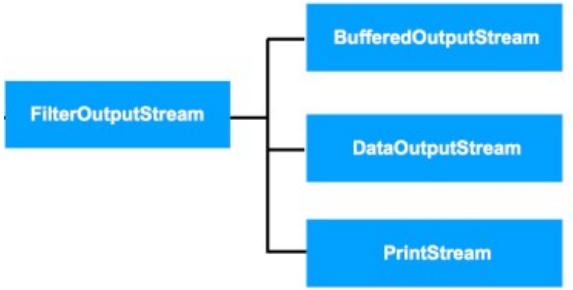
- **SequenceInputStream** được sử dụng để đọc dữ liệu từ nhiều nguồn stream.

```java
import java.io.FileInputStream;
import java.io.IOException;
import java.io.SequenceInputStream;

public class MainClass {
    public static void main(String[] args) {
        FileInputStream input1 = null;
        FileInputStream input2 = null;
        SequenceInputStream inst = null;
        try {
            input1 = new FileInputStream( name: "test_input1.txt");
            input2 = new FileInputStream( name: "test_input2.txt");
            inst = new SequenceInputStream(input1, input2);
            int j;
            while ((j = inst.read()) != -1) {
                System.out.print((char) j);
            }
        } catch (IOException ex) {
            System.out.println(ex);
        } finally {
            inst.close();
            input1.close();
            input2.close();
        }
    }
}
```

Luồng lọc (Filter Stream)

- Luồng lọc thường được sử dụng **kết nối với các luồng khác** để xử lý dữ liệu theo một hướng riêng nào đó.
- Hai lớp liên quan đến luồng lọc là **FilterInputStream** và **FilterOutputStream** được kế thừa tương ứng từ các lớp **InputStream** và **OutputStream**

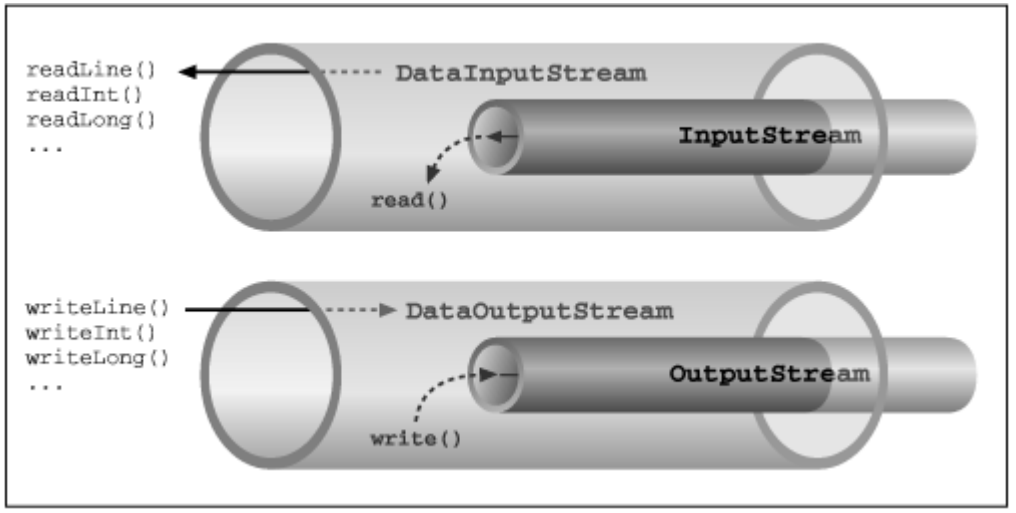




  - Các lớp kế thừa từ hai lớp **FilterInputStream** và **FilterOutputStream** sẽ được sử dụng để xử lý dữ liệu theo nhiều hướng riêng biệt.

Luồng nhập/xuất dữ liệu sơ cấp

- **DataInputStream** và **DataOutputStream** là hai lớp kế thừa từ hai lớp lọc **FilterInputStream** và **FilterOutputStream**
  - Được sử dụng để lọc (nhập/xuất) dữ liệu dưới dạng **các kiểu dữ liệu sơ cấp**.



Luồng nhập/xuất dữ liệu sơ cấp

- Một số phương thức của **DataInputStream**
  - **float readFloat()** throws IOException
  - **int readInt()** throws IOException
  - **long readLong()** throws IOException
  - **String readUTF()** throws IOException
- Một số phương thức của **DataOutputStream**
  - **void writeFloat(float v)** throws IOException
  - **void writeInt(int b)** throws IOException
  - **void writeLong(long v)** throws IOException
  - **void writeUTF(String s)** throws IOException

Luồng nhập/xuất dữ liệu sơ cấp

- Ví dụ 1: Ghi ra file **randnum.dat** 20 số nguyên ngẫu nhiên nằm trong khoảng từ 0 cho đến 1000.

```
try {
    FileOutputStream f = new FileOutputStream("randnum.dat");
    DataOutputStream outFile = new DataOutputStream(f);

    for(int i = 0; i < 20; i++)
        outFile.writeInt( (int) (Math.random()*1000) );

    outFile.close();
    f.close();
} catch (IOException ex) {
    System.out.println("Loi: " + ex);
}
```

Luồng nhập/xuất dữ liệu sơ cấp

- Ví dụ 2: Đọc tất cả dữ liệu của file **randnum.dat** được tạo ở **Ví dụ 1**, sau đó in các giá trị đọc được lên màn hình.

```
try {
    FileInputStream f = new FileInputStream("randnum.dat");
    DataInputStream inFile = new DataInputStream(f);
    int num;
    while (true)   {
        num = inFile.readInt();
        System.out.println("num = " + num);
    }
} catch (EOFException ex) {
    System.out.println("End of file");
} catch (FileNotFoundException ex) {
    ex.printStackTrace();
} catch (IOException ex) {
    ex.printStackTrace();
}
```

Ghép nối nhiều luồng

- Ta có thể dùng **kết hợp luồng lọc với luồng tạo bộ nhớ đệm** (buffer) để tăng tốc độ xử lý cho chương trình
- Ví dụ:

```
FileInputStream fStream = new FileInputStream("data.dat");
BufferedInputStream bStream = new BufferedInputStream(fStream);
DataInputStream dStream = new DataInputStream(bStream);
...
dStream.close();
```

System.in

- **System.in** là một luồng vào có kiểu **InputStream** gắn với bàn phím, được dùng để nhận thông tin từ bàn phím.
  - Ví dụ:

```java
import java.util.Scanner;

public class HelloWorld {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Nhập họ và tên: ");
        String strHoVaTen = scanner.nextLine();

        System.out.println("Nhập năm sinh: ");
        int iNamSinh = scanner.nextInt();

        System.out.println("Nhập điểm: ");
        double dDiem = scanner.nextDouble();

        scanner.close();
    }
}
```

System.out

- **System.out** là luồng ra có kiểu **PrintStream** gắn với màn hình, được dùng để kết xuất thông tin ra màn hình.
  - Lớp **PrintStream** kế thừa từ lớp **FilterOutputStream**, cho phép hiển thị biểu diễn thông tin của dữ liệu ở đầu ra.
  - Ví dụ:
    - System.out.println(“Gia tri cua x la: “ + x);

Lớp PrintStream

- Ví dụ 3: Đọc tất cả dữ liệu của file **randnum.dat** được tạo ở **Ví dụ 1**, sau đó ghi dữ liệu đó xuống file **randnum.txt** dưới dạng văn bản.

```
public class MainClass {
    public static void main(String[] args) throws IOException {
        DataInputStream inStream = null;
        PrintStream pStream = null;
        try {
            FileInputStream f = new FileInputStream( name: "randnum.dat");
            inStream = new DataInputStream(f);

            FileOutputStream ff = new FileOutputStream( name: "randnum.txt");
            pStream = new PrintStream(ff);
            int count = 0;

            while (true) {
                int num = inStream.readInt();
                count++;
                pStream.println("Integer " + count + " = " + num);
            }
        } catch (EOFException ex) {
            System.out.println("End of file");
        } finally {
            inStream.close();
            pStream.close();
        }
    }
}
```

Luồng nhập/xuất đối tượng

- Để lưu lại một đối tượng, ta có thể lưu lần lượt từng thuộc tính của nó. Khi đọc lại đối tượng ta phải tạo đối tượng mới từ các thuộc tính đã ghi.
  - => Dài dòng, kém linh hoạt.
- Java hỗ trợ đọc/ghi các đối tượng một cách đơn giản thông qua các lớp **ObjectInputStream** và **ObjectOutputStream**.
- Một đối tượng muốn có thể được đọc/ghi phải cài đặt giao tiếp **java.io.Serializable**

Lớp do người dùng tự định nghĩa cho phép đọc/ghi

```
public class Student implements Serializable {
    private String name;
    private int age;

    Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String toString() {
        String ret = "My name is " + name +  "\nI am " + age + " years old";
        return ret;
    }
}
```

Ghi đối tượng

```
try {
    FileOutputStream f = new FileOutputStream("student.dat");
    ObjectOutputStream oStream = new ObjectOutputStream(f);

    Student x = new Student("Bill Gates", 18);
    oStream.writeObject(x);
    oStream.close();
} catch (IOException e) {
    System.out.println(“Error IO file”);
}
```

Đọc đối tượng

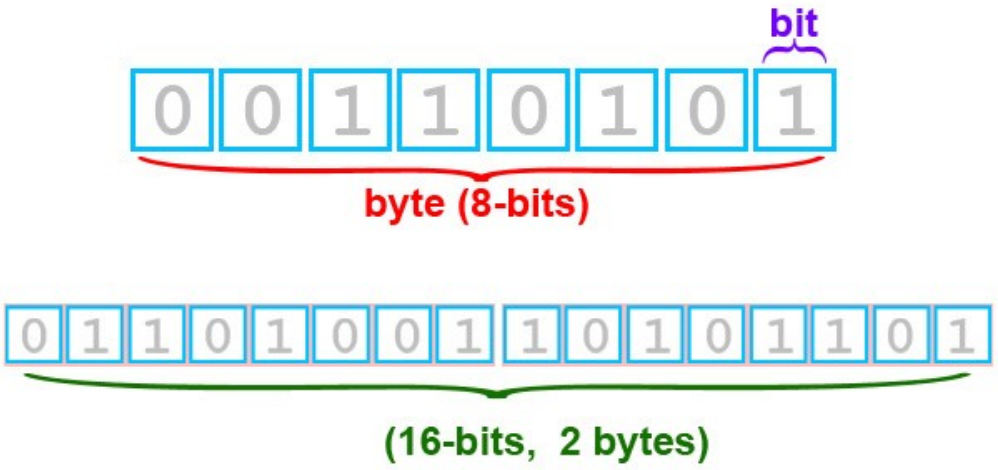
```
try {
    FileInputStream g = new FileInputStream("student.dat");
    ObjectInputStream inStream = new ObjectInputStream(g);

    Student y = (Student) inStream.readObject();
    System.out.println(y.toString());
    inStream.close();
} catch (ClassNotFoundException e) {
    System.out.println(“Class not found”);
} catch (IOException e) {
    System.out.println(“Error IO file”);
}
```

Đọc/Ghi đối tượng

- Chú ý khi đọc/ghi đối tượng
  - Đối tượng có thể cài đặt 2 phương thức sau để thực hiện đọc/ghi theo cách riêng của mình.
    - private void writeObject(ObjectOutputStream out) throws IOException
    - private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException

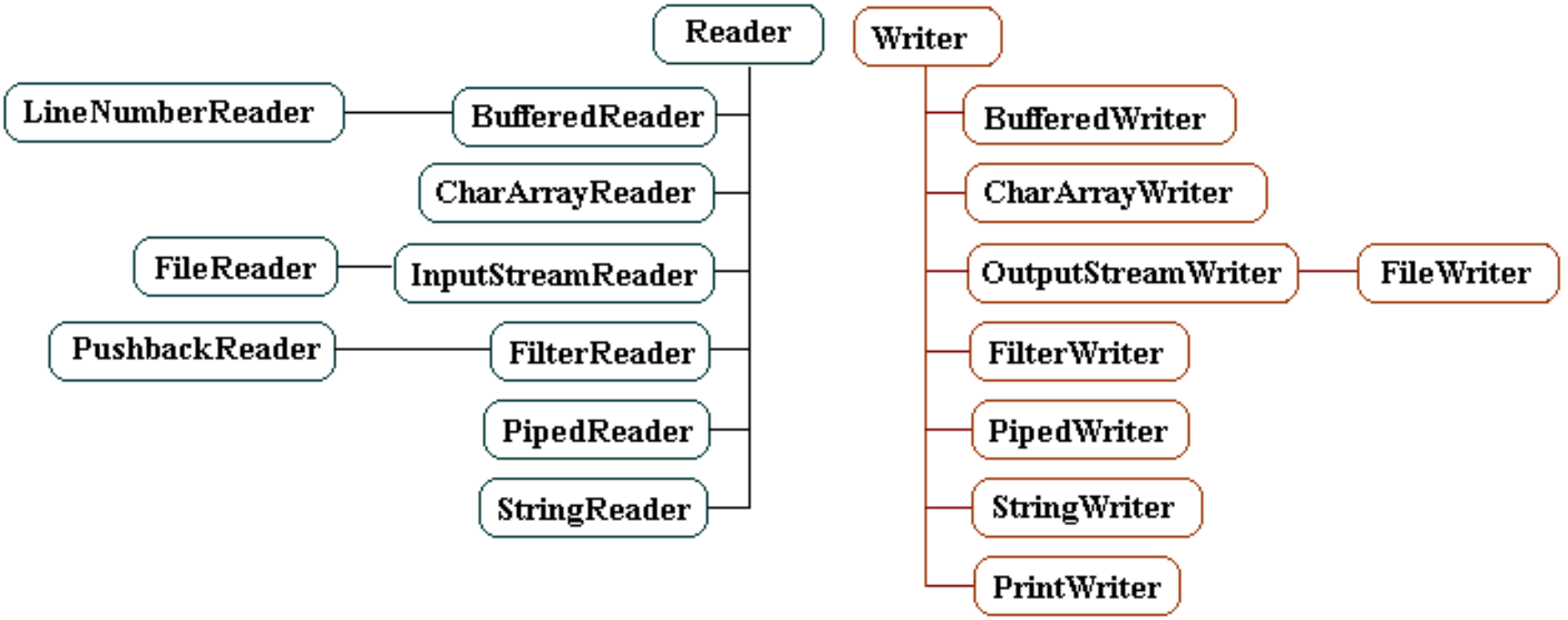
Luồng ký tự

- Luồng byte mỗi một lần đọc/ghi một byte (8 bit)
- Luồng ký tự (character stream) mỗi lần đọc/ghi một ký tự dạng Unicode, tùy thuộc vào kiểu mã hóa (UTF-8, UTF-16) mà ký tự đó tương đương với 1, 2 hay 3 bytes.



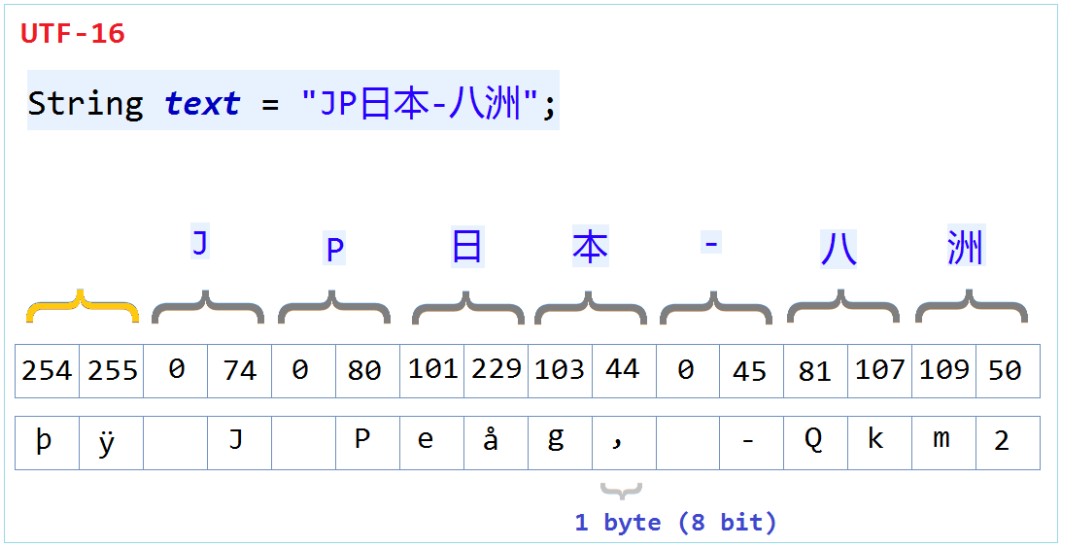
Luồng ký tự

- Sử dụng các lớp kế thừa từ 2 lớp trừu tượng: **Reader** và **Writer**.



Luồng ký tự và kiểu mã hóa

- Kiểu mã hóa **UTF-16**
  - Khi đọc từ file lên theo kiểu mã hóa UTF-16, nó sẽ **bỏ 2 bytes đầu tiên** và tiếp theo đọc liên tục cho đến hết file **mỗi lần 2 bytes** để ghép lại thành một ký tự.



Luồng ký tự và kiểu mã hóa

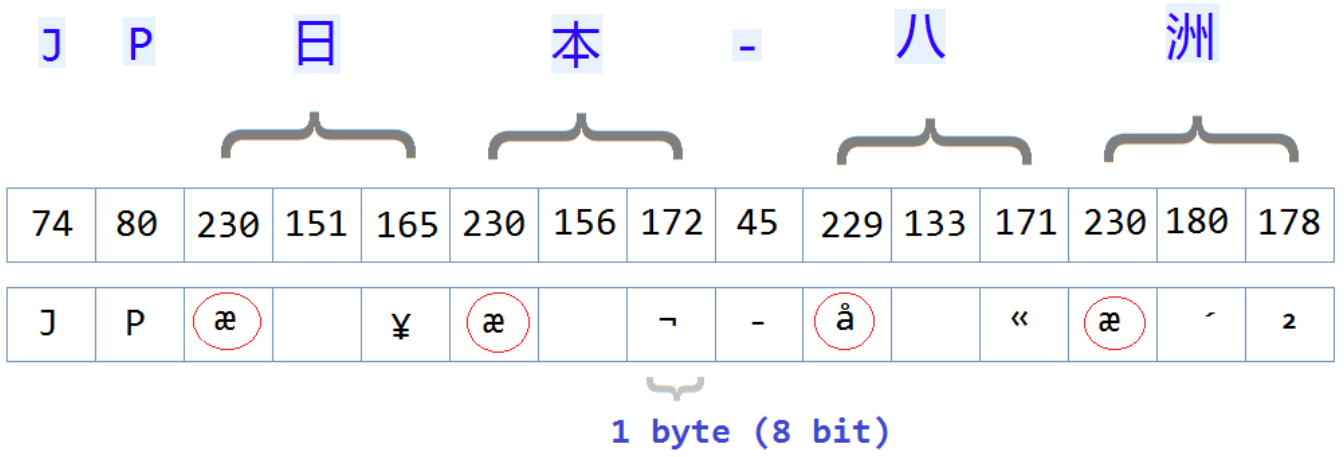
- Kiểu mã hóa **UTF-8**: Quy tắc để đọc ký tự, dựa vào một bảng mã gọi là **UTF-8 Table**.
  - Đọc byte đầu tiên, nếu <= 127, thì đó là 1 ký tự ASCII.
  - Ngược lại nếu > 127, thì nó cần đọc tiếp byte thứ 2, và kiểm tra xem 2 byte đó có ghép được thành 1 ký tự dựa vào bảng mã UTF-8 hay không.
  - Nếu 2 byte đầu tiên không tương ứng với một ký tự, nó đọc tiếp byte thứ 3 và ghép thành 1 ký tự.
  - UTF-8 dùng tối đa 3 byte để lưu trữ một ký tự.

Luồng ký tự và kiểu mã hóa

- Kiểu mã hóa UTF-8:

UTF-8

```
String text = "JP日本-八洲";
```



Kết nối luồng byte và luồng kí tự

- Có thể chuyển từ luồng byte sang luồng ký tự nhờ các lớp
  - InputStreamReader
  - OutputStreamReader
- Ví dụ:

```
BufferedReader buff = new BufferedReader(new InputStreamReader(System.in));
String s = buff.readLine();
...
```

Đọc/Ghi file Unicode

- Kết nối luồng
  - FileInputStream vào luồng InputStreamReader
  - FileOutputStream vào luồng OutputStreamWriter
  - Sau đó có thể kết nối tiếp vào luồng BufferedReader/BufferedWriter để tăng tốc độ xử lý.
- Chỉ rõ kiểu mã hóa (UTF-8 hay UTF-16)
  - Ghi xuống kiểu mã hóa nào thì khi đọc lên phải đúng kiểu mã hóa đó.

Ghi file Unicode

```
try {
    OutputStreamWriter buff = new OutputStreamWriter(new
    FileOutputStream("unicode.txt"), "UTF-16");

    buff.write('ề');
    buff.write('à');

    String s = "\r\nCộng hòa xã hội chủ nghĩa Việt Nam";
    buff.write(s, 0, s.length());
    buff.close();
} catch (IOException e) {
    System.out.println("Error IO file");
}
```

Đọc file Unicode

```
try {
    InputStreamReader buff = new InputStreamReader(new
    FileInputStream("unicode.txt"), "UTF-16");

    int ch;
    while ( (ch = buff.read()) != -1) {
        System.out.print((char)ch);
    }
    buff.close();
} catch (IOException e) {
    System.out.println(“Error IO file”);
}
```

Chú ý trong khi soạn thảo lập trình

- Muốn đưa trực tiếp tiếng Việt Unicode vào cùng các đoạn mã Java ta có thể phải sử dụng Notepad hoặc các phần mềm hỗ trợ soạn thảo tiếng Việt.
- Các bước cần thực hiện
  - Lưu file source code dưới dạng Unicode
  - Gõ lệnh biên dịch

    javac –encoding unicode filename.java
  - Lệnh thông dịch

    java filename (như bình thường)

File truy cập ngẫu nhiên

- Hai hạn chế của việc xử lý file thông qua luồng
  - Không thể đọc và ghi file cùng một lúc
  - Truy nhập file mang tính tuần tự
- Java hỗ trợ việc truy nhập và xử lý file một cách tự do thông qua lớp **RandomAccessFile**.

File truy cập ngẫu nhiên

- Các phương thức cơ bản
  - RandomAccessFile(String name, String mode)
    // cấu tử, trong đó mode có thể là “r”, “w”, “rw”
  - int readInt(); // đọc số nguyên
  - void writeInt(int v); // ghi số nguyên
  - long readLong(); // đọc số long
  - void writeLong(long v); // ghi số long
  - void seek(long pos); // di chuyển vị trí con trỏ file
  - long getFilePointer(); // lấy vị trí của con trỏ file
  - long length(); // lấy kích cỡ của file
  - void close(); // đóng file
    ...

File truy cập ngẫu nhiên

```
try {
    RandomAccessFile f = new RandomAccessFile("randfile.dat","rw");
    f.writeBoolean(true);
    f.writeInt(123456);
    f.writeChar('j');
    f.writeDouble(1234.56);
    f.seek(1);
    System.out.println(f.readInt());
    System.out.println(f.readChar());
    System.out.println(f.readDouble());
    f.seek(0);
    System.out.println(f.readBoolean());
    f.close();
} catch (IOException e) {
    System.out.println(“Error IO file”);
}
```

Chú ý khi xử lý đóng file

- Nếu để lệnh **f.close()** trong khối try thì có thể lệnh này sẽ không được thực hiện khi có lỗi ở các lệnh phía trên.
- Có thể viết lại như sau:

Chú ý khi xử lý đóng file

```
FileInputStream f = null;
try {
    f = new FileInputStream("somefile.txt");
    // đọc file
} catch (IOException e) {
    // hiển thị lỗi
} finally {
    if (f != null) {
        try {
            f.close(); // đóng file
        } catch (Exception e) {
            // thông báo lỗi khi đóng file
        }
    }
}
```

Chú ý khi xử lý đóng file

```
FileInputStream f = null;
try {
    f = new FileInputStream("somefile.txt");
    // đọc file
} catch (IOException e) {
    // hiển thị lỗi
} finally {
    if (f != null) {
        try {
            f.close(); // đóng file
        } catch (Exception e) {
            // thông báo lỗi khi đóng file
        }
    }
}
```

Lớp File

- Lớp **File** cho phép lấy thông tin về file và thư mục.
- Lớp File nằm trong gói **java.io**
- Một số phương thức của File
  - boolean exists(); // kiểm tra sự tồn tại của file
  - boolean isDirectory(); // kiểm tra xem file có phải là thư mục
  - String getParent(); // lấy thư mục cha
  - long length(); // lấy cỡ file (byte)
  - long lastModified(); // lấy ngày sửa file gần nhất
  - String[] list(); // lấy nội dung của thư mục

Ví dụ hiển thị thông tin của file

```
public static void main(String[] args) {
    File file = new File("randfile.dat");
    if ( file.exists() ) {
        System.out.println("Path is: " + file.getAbsolutePath());
        System.out.println("It's size is: " + file.length());
        Date dateModified = new Date(file.lastModified());
        System.out.println("Last update is: " + dateModified);
    } else {
        System.out.println("The file does not exist");
    }
}
```

Ví dụ hiện nội dung thư mục

```
public static void main(String[] args) {
    String strPath = ".";
    File dir = new File(strPath);
    if ( dir.isDirectory() ) {
        String[] subFiles = dir.list();

        for(int i=0; i < subFiles.length; i++)
            if (new File(strPath + subFiles[i]).isDirectory())
                System.out.println(subFiles[i] + " <DIR>");
            else
                System.out.println(subFiles[i]);
    } else {
        System.out.println("The file is not a directory");
    }
}
```

Hết Tuần 6

Cảm ơn các bạn đã chú ý lắng nghe !!!