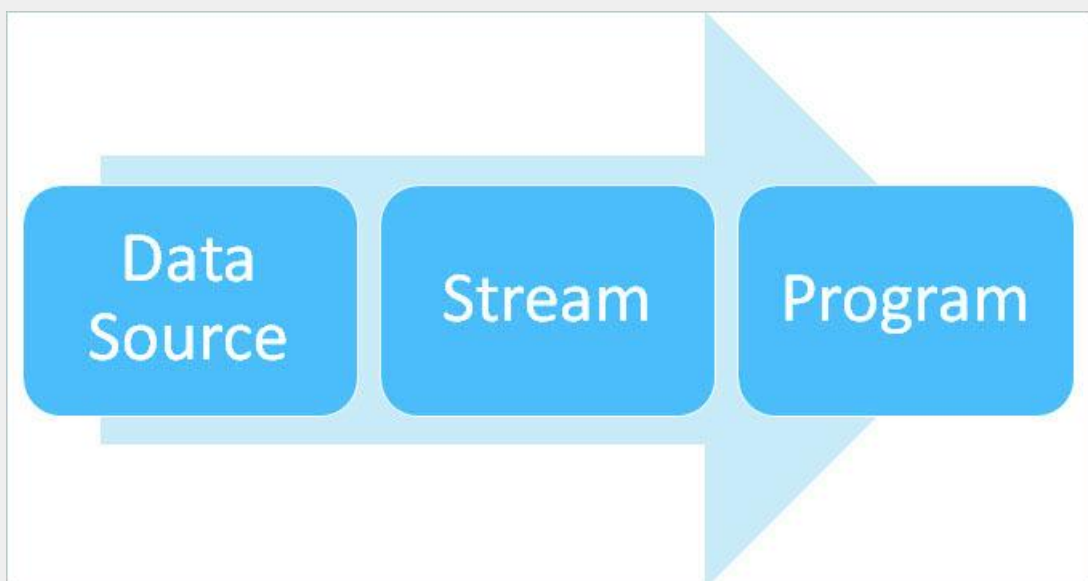


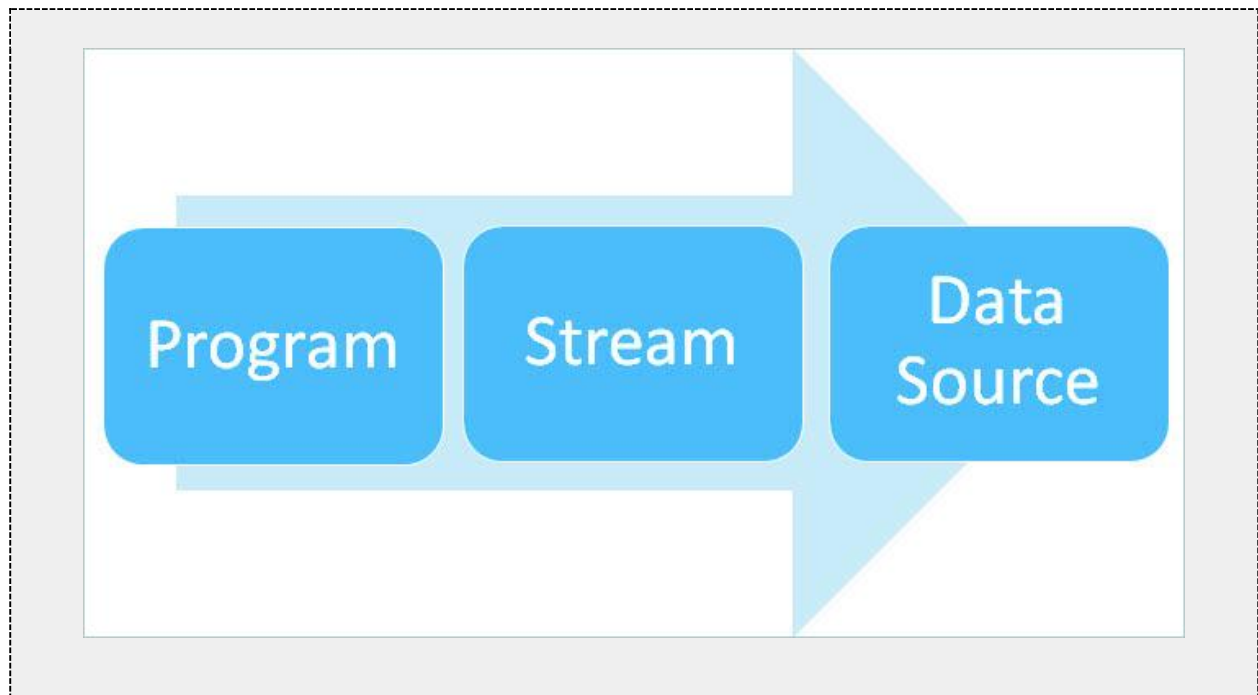
Session 5+6**File Handling in Java****&****New Features in File Handling****Phần I - Thực hiện trong 90 phút****1. Mục tiêu**

- Hiểu biết và sử dụng đối tượng File để khởi tạo thư mục, khởi tạo file.
- Biết sử dụng các đối tượng để thao tác đọc ghi file dạng text, stream.
- Có thể đọc, ghi file nội dung text, đọc ghi theo từng dòng.
- Có thể đọc, ghi file theo object, ghi dữ liệu dạng tập hợp (Collections).

Chú giải: Việc đọc, ghi dữ liệu là quá trình ngược nhau, xem hình vẽ bên dưới mô tả quá trình đọc dữ liệu từ nguồn ngoài vào chương trình:



... và quá trình ghi dữ liệu từ chương trình và nguồn ngoài:



2. Thực hiện

Bài thực hành 1: Viết chương trình tạo thư mục tên là **bkap** và file tên là **sinhvien.db**.

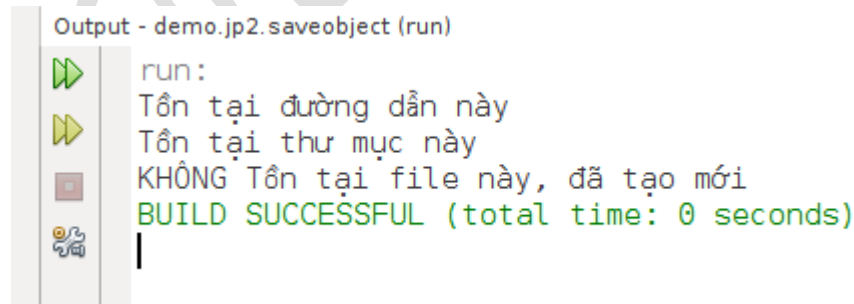
Bước 1: viết hàm createFolder()

```
void createFolder(String path) {  
    File folder = new File(path);  
    if (folder.exists()) {  
        System.out.println("Tồn tại đường dẫn này");  
        if (folder.isDirectory()) {  
            System.out.println("Tồn tại thư mục này");  
        }  
    } else {  
        // Không tồn tại thì tạo mới  
        folder.mkdirs();  
        System.out.println("KHÔNG Tồn tại đường dẫn này, đã tạo mới");  
    }  
}
```

// Bước 2: viết hàm createFile()

```
void createFile(String path) {  
    File file = new File(path);  
    if (file.exists()) {  
        System.out.println("Tồn tại file này");  
    } else {  
        try {  
            // Không tồn tại thì tạo mới  
            file.createNewFile();  
        } catch (IOException ex) {  
            System.out.println("Lỗi tạo file: " + ex.toString());  
        }  
        System.out.println("KHÔNG Tồn tại file này, đã tạo mới");  
    }  
}  
  
// Bước 3: tạo file MainClass có hàm main và viết mã gọi 2 hàm ở trên và truyền  
// tham số  
public static void main(String[] args) {  
    Sem2Jp2DemoFile demo = new Sem2Jp2DemoFile();  
    demo.createFolder("bkap/slide");  
    demo.createFile("bkap/slide/sinhvien.db");  
}
```

Bước 4: build và chạy



```
Output - demo.jp2.saveobject (run)  
run:  
Tồn tại đường dẫn này  
Tồn tại thư mục này  
KHÔNG Tồn tại file này, đã tạo mới  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bài thực hành 2: Viết chương trình ghi file dữ liệu danh sách sinh viên dạng text, sau đó đọc dữ liệu lại. Dữ liệu phải ghi thành từng dòng cho từng sinh viên.

Bước 1: tạo class NhanVien

```
public class NhanVien {  
  
    private String name;  
    private int age;  
    private String address;  
  
    public NhanVien(String name, int age, String address) {  
        this.name = name;  
        this.age = age;  
        this.address = address;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
  
    public String getAddress() {  
        return address;  
    }  
}
```

```
public void setAddress(String address) {  
    this.address = address;  
}  
  
@Override  
public String toString() {  
    return name + "(" + age + ")" + address;  
}  
  
}
```

Bước 2: Tạo class DemoJP2SaveText

```
package demo.jp2.saveobject;  
  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.OutputStreamWriter;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
/**  
 *  
 * @author minhvt  
 */  
public class DemoJP2SaveText {  
  
    public DemoJP2SaveText() {  
  
    }  
  
    //Bước 3: viết hàm ghi dữ liệu ra file text  
    public void writeText2File(List<NhanVien> lstData) throws IOException {
```

```
String filePath = "writeText2File.txt";
boolean isVietTiep = false;
File myFile = new File(filePath);
FileOutputStream fos = new FileOutputStream(myFile, isVietTiep);
OutputStreamWriter osw = new OutputStreamWriter(fos);
// Ghi ra file
for (NhanVien nhanVien : lstData) {
    osw.write(nhanVien.toString());
}
osw.flush(); // Tổng dữ liệu từ bộ nhớ tạm xuống file
osw.close(); // Đóng kết nối tới file
}

//Bước 4: viết hàm ghi dữ liệu ra file text theo từng dòng
public void writeTextByLine(List<NhanVien> lstData) throws
FileNotFoundException, IOException {
    String filePath = "writeTextByLine.txt";
    boolean isVietTiep = true;
    File myFile = new File(filePath);
    FileOutputStream fos = new FileOutputStream(myFile, isVietTiep);
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));
    // Ghi ra file
    for (NhanVien nhanVien : lstData) {
        bw.write(nhanVien.toString());
        bw.newLine(); // thêm dòng mới gọi
    }
    bw.flush(); // Tổng dữ liệu từ bộ nhớ tạm xuống file
    bw.close(); // Đóng kết nối tới file
}

//Bước 5: viết hàm đọc dữ liệu trong file theo từng dòng
public void readTextByLine() throws FileNotFoundException, IOException {
    String filePath = "writeTextByLine.txt";
    File myFile = new File(filePath);
    FileInputStream fos = new FileInputStream(myFile);
    BufferedReader br = new BufferedReader(new InputStreamReader(fos));
    String temp;
    int cnt = 1;
    while ((temp = br.readLine()) != null) {
        System.out.println(cnt + ". \t" + temp);
        cnt++;
    }
}

public static void main(String[] args) {
    //Bước 6: thêm dữ liệu ảo, gọi tới hàm, build và chạy
```

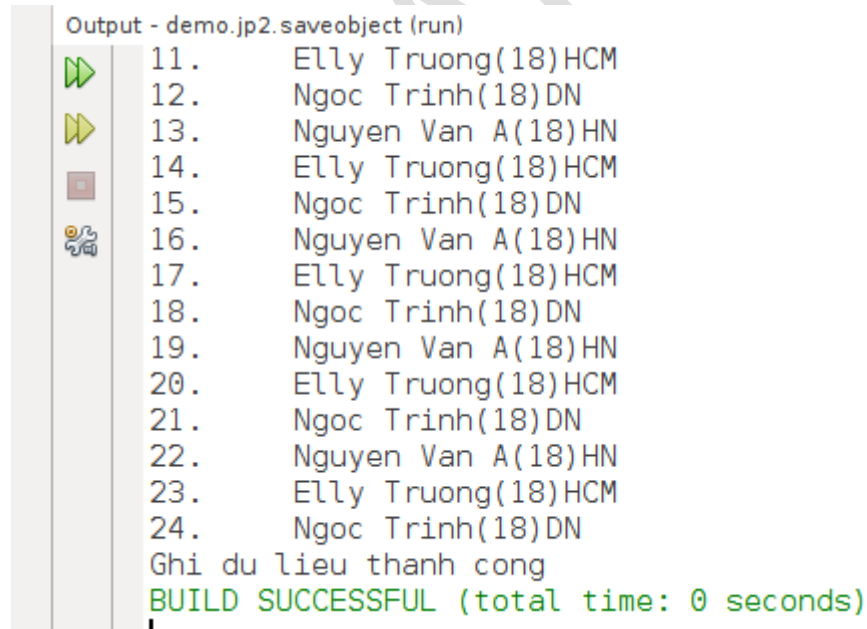
```
try {
    DemoJP2SaveText demoJP2SaveText = new DemoJP2SaveText();

    // Du lieu ao
    List<NhanVien> lstNhanVien = new ArrayList<>();
    NhanVien nv1 = new NhanVien("Nguyen Van A", 18, "HN");
    NhanVien nv2 = new NhanVien("Elly Truong", 18, "HCM");
    NhanVien nv3 = new NhanVien("Ngoc Trinh", 18, "DN");
    lstNhanVien.add(nv1);
    lstNhanVien.add(nv2);
    lstNhanVien.add(nv3);

    demoJP2SaveText.writeTextByLine(lstNhanVien);
    demoJP2SaveText.readTextByLine();
    demoJP2SaveText.writeText2File(lstNhanVien);

    System.out.println("Ghi du lieu thanh cong");
} catch (IOException ex) {
    System.out.println("Lỗi: " + ex.toString());
    Logger.getLogger(DemoJP2SaveText.class.getName()).log(Level.SEVERE,
null, ex);
}
}
```

Bước 6: Chạy và xem kết quả



```
Output - demo.jp2.saveobject (run)
11.    Elly Truong(18)HCM
12.    Ngoc Trinh(18)DN
13.    Nguyen Van A(18)HN
14.    Elly Truong(18)HCM
15.    Ngoc Trinh(18)DN
16.    Nguyen Van A(18)HN
17.    Elly Truong(18)HCM
18.    Ngoc Trinh(18)DN
19.    Nguyen Van A(18)HN
20.    Elly Truong(18)HCM
21.    Ngoc Trinh(18)DN
22.    Nguyen Van A(18)HN
23.    Elly Truong(18)HCM
24.    Ngoc Trinh(18)DN
Ghi du lieu thanh cong
BUILD SUCCESSFUL (total time: 0 seconds)
```

Chú giải:

- Stream là khái niệm chỉ luồng dữ liệu đầu vào, đầu ra, dữ liệu này có thể ở dạng nhị phân byte, ký tự hay đối tượng.
- Khi ghi file xuất qua stream, có thể cài đặt chế độ ghi đè hoặc ghi tiếp tục.

Bài thực hành 3: Viết chương trình ghi và đọc dữ liệu danh sách sinh viên dưới dạng đối tượng.

Bước 1: Tạo class NhanVien thực thi Serializable

```
public class NhanVien implements Serializable {  
    String name;  
    int age;  
    String address;
```

Bước 2: tạo file **DemoJp2Saveobject**

```
package demo.jp2.saveobject;  
  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
/**  
 *  
 * @author minhvt  
 */  
public class DemoJp2Saveobject {  
  
    //Bước 3: Viết hàm lưu mảng NhanVien  
    public void saveNhanVien() {  
        List<NhanVien> lstNhanVien = new ArrayList<>();  
  
        NhanVien nv1 = new NhanVien("Nguyen Van A", 18, "HN");  
        NhanVien nv2 = new NhanVien("Elly Truong", 18, "HCM");  
        NhanVien nv3 = new NhanVien("Ngoc Trinh", 18, "DN");
```



```
IstNhanVien.add(nv1);
IstNhanVien.add(nv2);
IstNhanVien.add(nv3);

try {
    // Ghi xuống file
    FileOutputStream file = new FileOutputStream("data.db");
    ObjectOutputStream objOut = new ObjectOutputStream(file);

    objOut.writeObject(IstNhanVien);
    objOut.flush();
    objOut.close();

} catch (Exception ex) {
    Logger.getLogger(DemoJp2Saveobject.class.getName()).log(Level.SEVERE,
null, ex);
}

}

//Bước 4: viết hàm đọc dữ liệu NhanVien
public void readNhanVien() {
    List<NhanVien> IstNhanVien = new ArrayList<>();

    try {
        FileInputStream file = new FileInputStream("data.db");
        ObjectInputStream objIn = new ObjectInputStream(file);

        IstNhanVien = (List<NhanVien>) objIn.readObject();
        objIn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }

    for (int i = 0; i < IstNhanVien.size(); i++) {
        System.out.println(IstNhanVien.get(i).toString());
    }
}

// Bước 5: gọi tới hàm đã viết ở trên
public static void main(String[] args) {
    DemoJp2Saveobject demo = new DemoJp2Saveobject();

    demo.saveNhanVien();
    demo.readNhanVien();
}
```

```
}
```

Bước 6: build và chạy chương trình

```
Output - demo.jp2.saveobject (run)
run:
Nguyen Van A(18)HN
Elly Truong(18)HCM
Ngoc Trinh(18)DN
BUILD SUCCESSFUL (total time: 0 seconds)
```

Phần II - Thực hiện trong 30 phút

1. Mục tiêu

- Tìm hiểu lớp Console.
- Biết sử dụng đối tượng hỗ trợ nén (Deflater) và giải nén (Inflater) dữ liệu
- Hiểu biết cơ bản về gói java.nio.
- Sử dụng các lớp hỗ trợ trong package java.nio để thao tác tới thư mục, file, đường dẫn, thực hiện đọc, ghi, sửa, xóa, di chuyển thư mục, file.
- Truy cập tới thuộc tính của file, thư mục, theo dõi sự thay đổi trên thư mục, file.

2. Thực hiện

Bài thực hành 1: Viết chương trình java chạy trên console yêu cầu người dùng nhập dữ liệu, password rồi hiển thị ra màn hình. Biên dịch và chạy chương trình từ Netbeans và cmd.

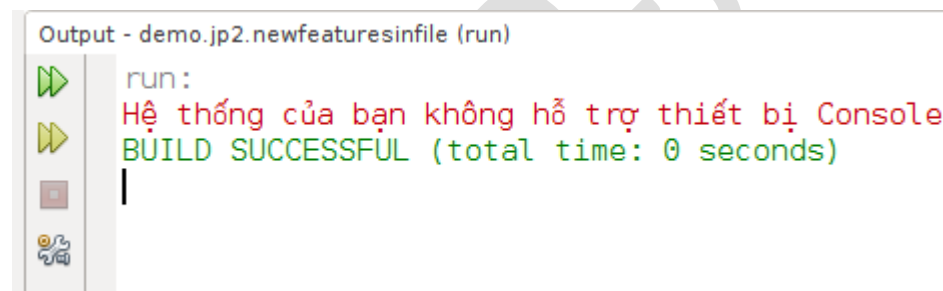
Bước 1: Viết lớp MainClass có hàm main()

```
public class MainClass {

    /**
     * @param args thecommandlinearguments
     */
    public static void main(String[] args) {
        Console console = System.console();
        if (console == null) {
            System.err.println("Hệ thống của bạn không hỗ trợ thiết bị Console");
        }
    }
}
```

```
} else {  
    try {  
        String username = console.readLine("Nhập Username: ");  
        char[] password = console.readPassword("Nhập Password: ");  
  
        System.out.println("Username" + username);  
        System.out.println("Password" + new String(password));  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}  
}
```

Bước 2: build và chạy trên Netbeans, biên dịch và chạy dùng cmd để thấy sự khác nhau.



Output - demo.jp2.newfeaturesinfile (run)

```
run:  
Hệ thống của bạn không hỗ trợ thiết bị Console  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bài thực hành 2: Viết chương trình Java nén dữ liệu là một đoạn text String, kiểm tra dung lượng dữ liệu trước và sau khi nén.

Bước 1: Viết class **CompressionUtils**.

```
package demo.jp2.newfeaturesinfile;  
  
import java.io.ByteArrayOutputStream;  
import java.io.IOException;  
import java.util.zip.DataFormatException;  
import java.util.zip.Deflater;  
import java.util.zip.Inflater;  
  
/**  
 *  
 * @author Vu Tuan Minh
```

```
*/  
public class CompressionUtils {  
  
    //Bước 2: Viết hàm nén, ghi dung lượng của dữ liệu trước và sau khi nén  
    public static byte[] compress(byte[] data) throws IOException {  
        Deflater deflater = new Deflater();  
        deflater.setInput(data);  
  
        ByteArrayOutputStream outputStream = new  
        ByteArrayOutputStream(data.length);  
  
        deflater.finish();  
        byte[] buffer = new byte[1024];  
        while (!deflater.finished()) {  
            int count = deflater.deflate(buffer); // returns the generated code... index  
            outputStream.write(buffer, 0, count);  
        }  
        outputStream.close();  
        byte[] output = outputStream.toByteArray();  
  
        deflater.end();  
  
        System.out.println("Original: " + (float) data.length / 1024f + " Kb");  
        System.out.println("Compressed: " + (float) output.length / 1024f + " Kb");  
        return output;  
    }  
  
    //Bước 3: Viết hàm giải nén, ghi dung lượng của dữ liệu trước và sau khi giải nén.  
    public static byte[] decompress(byte[] data) throws IOException,  
    DataFormatException {  
        Inflater inflater = new Inflater();  
        inflater.setInput(data);  
  
        ByteArrayOutputStream outputStream = new  
        ByteArrayOutputStream(data.length);  
        byte[] buffer = new byte[1024];  
        while (!inflater.finished()) {  
            int count = inflater.inflate(buffer);  
            outputStream.write(buffer, 0, count);  
        }  
        outputStream.close();  
        byte[] output = outputStream.toByteArray();  
  
        inflater.end();  
  
        System.out.println("Original: " + (float) data.length / 1024 + " Kb");  
    }  
}
```

```
System.out.println("Uncompressed: " + (float) output.length / 1024 + " Kb");
return output;
}
}
```

Bước 4: Viết class DeflaterClassDemo

```
import java.util.zip.Deflater;
import java.util.zip.Inflater;

/**
 * @author VuTuanMinh
 */
public class DeflaterClassDemo {

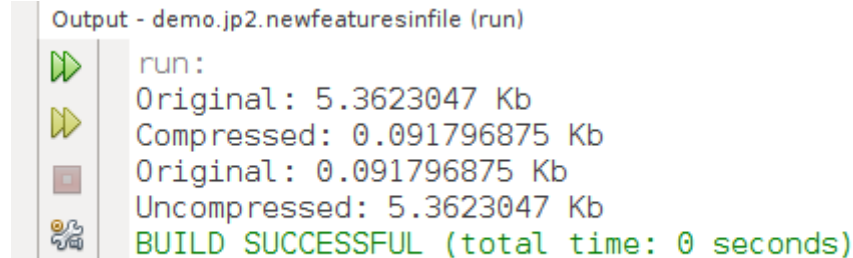
    public static void main(String[] args) throws Exception {

        DeflaterClassDemo demo = new DeflaterClassDemo();
        byte[] dataCompress = CompressionUtils.compress(demo.input.getBytes());
        CompressionUtils.decompress(dataCompress);
    }

    String input = "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam "
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
        .....
        + "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam"
```

```
+ "Cộng Hòa Xã Hội Chủ Nghĩa Việt Nam";  
}
```

Bước 5: build và chạy chương trình



```
Output - demo.jp2.newfeaturesinfile (run)  
run:  
Original: 5.3623047 Kb  
Compressed: 0.091796875 Kb  
Original: 0.091796875 Kb  
Uncompressed: 5.3623047 Kb  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Chú giải: Class Deflater và Inflater trong gói java.util.zip hỗ trợ nén và giải nén dữ liệu sử dụng thư viện ZLIP.

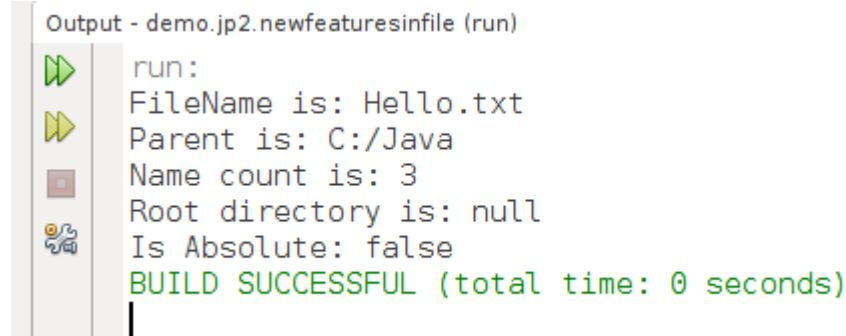
Bài thực hành 3: Viết chương trình liệt kê ra thông số cơ bản của file, đường dẫn tới file.

Bước 1: tạo file **PathApplication**.

```
import java.nio.file.Path;  
import java.nio.file.Paths;  
  
/**  
 *  
 * @author minhvufo  
 */  
public class PathApplication {  
  
    public static void main(String[] args) {  
        Path pathObj = Paths.get("C:/Java/Hello.txt");  
        System.out.printf("FileName is: %s%n", pathObj.getFileName());  
        System.out.printf("Parent is: %s%n", pathObj.getParent());  
        System.out.printf("Name count is: %d%n", pathObj.getNameCount());  
    }  
}
```

```
System.out.printf("Root directory is: %s%n", pathObj.getRoot());
System.out.printf("Is Absolute: %s%n", pathObj.isAbsolute());
}
}
```

Bước 2: build và chạy



```
Output - demo.jp2.newfeaturesinfile (run)
run:
FileName is: Hello.txt
Parent is: C:/Java
Name count is: 3
Root directory is: null
Is Absolute: false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bài thực hành 4: Viết chương trình liệt kê các file trong thư mục.

Bước 1: tạo file **ListDirApplication**.

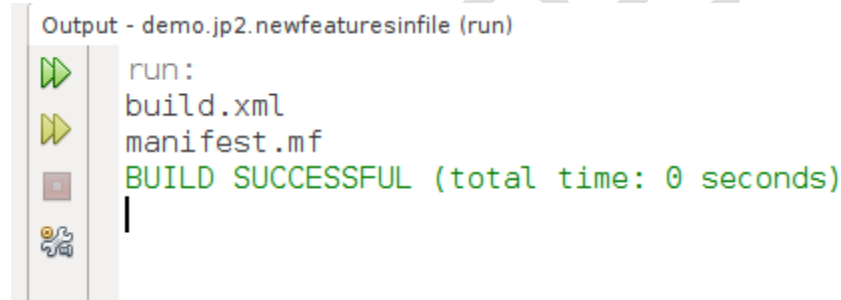
```
import java.io.IOException;
import java.nio.file.DirectoryIteratorException;
import java.nio.file.DirectoryStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Iterator;

/**
 *
 * @author minhvuvc
 */
public class ListDirApplication {

    public static void main(String[] args) {
        Path pathObj = Paths.get("test");
        // try (DirectoryStream<Path> dirStreamObj =
```

```
Files.newDirectoryStream(pathObj, "*.java")) {  
    try (DirectoryStream<Path> dirStreamObj = Files.newDirectoryStream(pathObj))  
    {  
        for (Iterator<Path> itrObj = dirStreamObj.iterator(); itrObj.hasNext();) {  
            Path fileObj = itrObj.next();  
            System.out.println(fileObj.getFileName());  
        }  
    } catch (IOException | DirectoryIteratorException ex) {  
        // IOException can never be thrown by the iteration.  
        // In this snippet, itObj can only be thrown by newDirectoryStream.  
        System.err.println(ex.getMessage());  
    }  
}
```

Bước 2: build và chạy



```
Output - demo.jp2.newfeaturesinfile (run)  
run:  
build.xml  
manifest.mf  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Bài thực hành 5: Viết chương trình theo dõi sự thay đổi của file (thêm, sửa, xóa) trong thư mục.

Bước 1: tạo file MainClass

```
import java.nio.file.*;  
import static java.nio.file.StandardWatchEventKinds.ENTRY_CREATE;  
import static java.nio.file.StandardWatchEventKinds.ENTRY_MODIFY;  
  
/**  
*
```



```
* @authorminhvt
*/
public class MainClass {

}
```

Bước 2: tạo đối tượng **MyWatchQueueReader** implements Runnable.

```
/**
 * ThisRunnableisusedtoconstantlyattempttotakefromthewatchqueue,
 * andwillreceivealleventsthatare registeredwiththefileWatcherit
 * isassociated.Inthissampleforsimplicitywejustoutputthekindof
 * eventandnameofthefileaffectedtostandardout.
 */
private static class MyWatchQueueReader implements Runnable {

    /**
     * thewatchService thatispassedinfromabove
     */
    private WatchService myWatcher;

    public MyWatchQueueReader(WatchService myWatcher) {
        this.myWatcher = myWatcher;
    }

    /**
     * Inordertoimplementafilewatcher, weloopforeverensuring
     * requestingtotakethenextitemfromthefilewatchersqueue.
     */
    @Override
    public void run() {
        try {
            // get the first event before looping

```

```
WatchKey key = myWatcher.take();
while (key != null) {
    // we have a polled event, now we traverse it and
    // receive all the states from it
    for (WatchEvent event : key.pollEvents()) {
        System.out.printf("Received %s event for file: %s\n",
            event.kind(), event.context());
    }
    key.reset();
    key = myWatcher.take();
}
} catch (InterruptedException e) {
    e.printStackTrace();
}
System.out.println("Stopping thread");
}
}
```

Bước 3: viết nội dung hàm main()

```
/**
 * changethisasaappropriateforyourfilesystemstructure.
 */
public static final String DIRECTORY_TO_WATCH = "/Home/test";

public static void main(String[] args) throws Exception {
    // get the directory we want to watch, using the Paths singleton class
    Path toWatch = null;
    try {
        toWatch = Paths.get(DIRECTORY_TO_WATCH);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
}

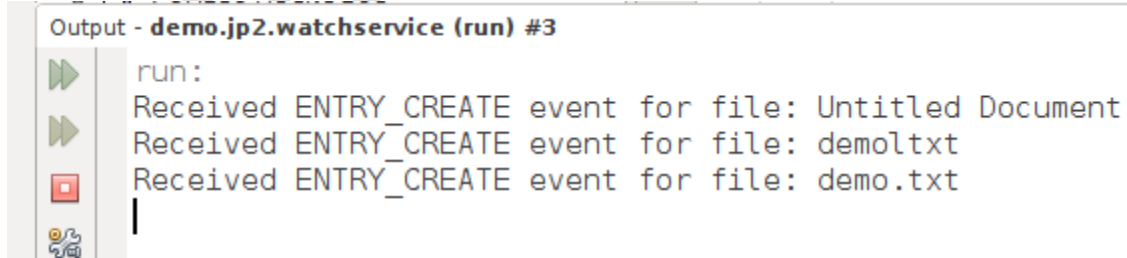
if (toWatch == null) {
    throw new UnsupportedOperationException("Không tìm thấy thư mục");
}

// make a new watch service that we can register interest in
// directories and files with.
WatchService myWatcher = toWatch.getFileSystem().newWatchService();

// start the file watcher thread below
MyWatchQueueReader fileWatcher = new MyWatchQueueReader(myWatcher);
Thread th = new Thread(fileWatcher, "FileWatcher");
th.start();

// register a file
toWatch.register(myWatcher, ENTRY_CREATE, ENTRY_MODIFY);
th.join();
}
```

Bước 4: biên dịch và chạy



```
Output - demo.jp2.watchservice (run) #3
run:
Received ENTRY_CREATE event for file: Untitled Document
Received ENTRY_CREATE event for file: demo1.txt
Received ENTRY_CREATE event for file: demo.txt
```

Phần III - Bài tập tự làm

1. Sử dụng notepad để tạo ra 1 file có tên "dulieu.txt" có nội dung như sau trong cùng thư mục của project. Hãy viết chương trình java để đọc nội dung của file này và hiển thị nội dung ra màn hình.

The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, Object, localized characters, etc.

A stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.

Java provides strong but flexible support for I/O related to Files and networks but this tutorial covers very basic functionality related to streams and I/O. We would see most commonly used example one by one:

Byte Streams

Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are , **FileInputStream** and **FileOutputStream**. Following is an example which makes use of these two classes to copy an input file into an output file:

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

public class CopyFile {

    public static void main(String args[]) throws IOException {
        FileInputStream in = null;
        FileOutputStream out = null;

        try {
            in = new FileInputStream("input.txt");
            out = new FileOutputStream("output.txt");

            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
        }
    }
}
```

```
    }  
    } finally {  
        if (in != null) {  
            in.close();  
        }  
        if (out != null) {  
            out.close();  
        }  
    }  
}
```

Now let's have a file input.txt with the following content:

This is test for copy file.

As a next step, compile above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put above code in CopyFile.java file and do the following:

```
$javac CopyFile.java
```

```
$java CopyFile
```

2. Hãy khai báo các biến số nguyên, thực double, boolean, char, String. Nhập dữ liệu vào từ bàn phím cho các biến này. Ghi dữ liệu của các biến vào file "data.dat". Viết một chương trình khác để đọc dữ liệu từ file "data.dat" và hiển thị ra màn hình.

3. Tạo một lớp Book có các thuộc tính:

```
private String isbn;  
private String bookName;  
private String author;  
private String publisher;  
private float price;
```

- Tạo đầy đủ các constructor, các phương thức get/set.
- Cài đặt hàm toString() để hiển thị thông tin các thuộc tính
- Lớp này phải thực thi giao diện Serializable để có thể ghi dữ liệu xuống file.
- Tạo lớp BookManager, trong lớp này khai báo một collection có kiểu Book.

- Nhập vào n cuốn sách sử dụng lớp `BufferedReader` (không dùng `Scanner`)
- Ghi thông tin n cuốn sách vào file `book.txt` (ghi dữ liệu dạng text)
- Tạo lớp `BookRead` để đọc thông tin các cuốn sách từ file `book.txt` ở trên.
- Hiển thị danh sách đọc được ra màn hình
- Tìm kiếm theo tên sách nhập vào

4. Tạo lớp `Product` có các thuộc tính:

```
private String proId;  
private String proName;  
private String producer;  
private int yearMaking;  
private float price;
```

- Tạo đầy đủ constructor, các phương thức `get/set`, `toString()`
- Lớp này có khả năng ghi dữ liệu xuống file
- Tạo lớp `ProductTest` để khai báo một collection kiểu `Product`
- Nhập vào n thông tin các product.
- Ghi danh sách xuống file `product.dat` (ghi dữ liệu dạng object)
- Viết chương trình để đọc danh sách từ file `product.dat`
- Hiển thị danh sách đọc được ra màn hình
- Tìm kiếm theo tên product nhập vào.