AJOU UNIVERSITY 1973

AJOU-MDC 국방디지털융합학과

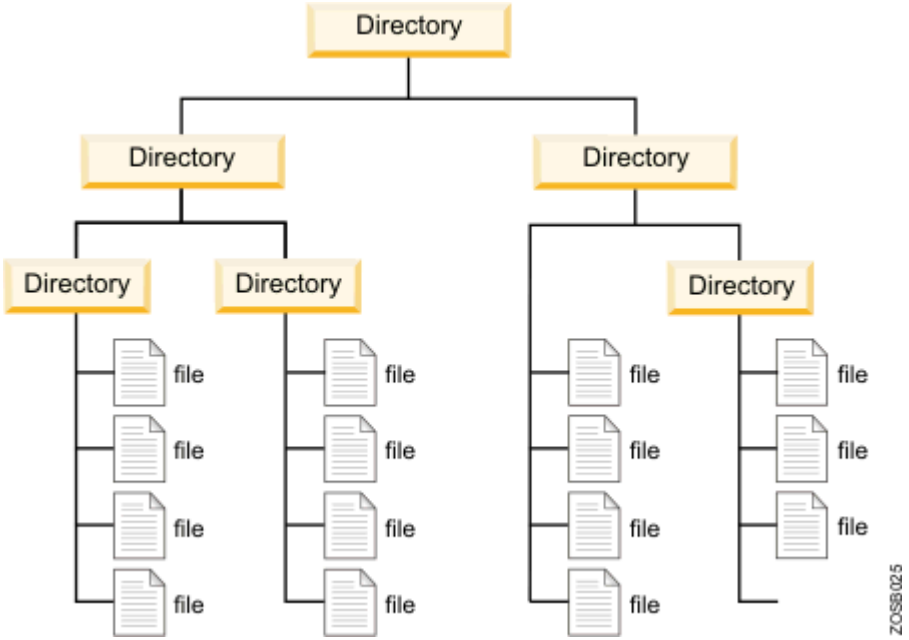
객체지향 프로그래밍 및 실습

11주차. Files, I/O Streams, NIO and XML Serialization

1. File system

- What is File system?
  - File은 데이터를 보관하는 방식
  - File System은 File 데이터를 관리 및 접근 할 수 있도록 보관 또는 조직하는 체제

  - UNIX 시스템의 파일 시스템 - 트리 구조



1. File system

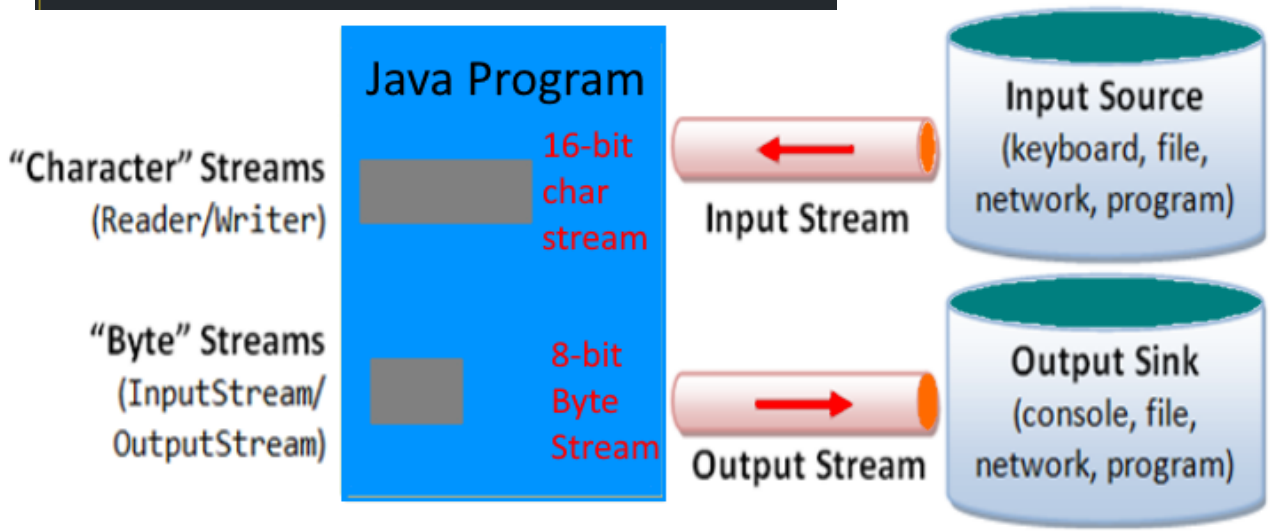
- 파일의 접근 방법
  - 파일 경로
    - /dev/file.c
    - C:₩User₩Documents
    - ./../file.c
  - 절대 경로
    - 대한민국 서울특별시 송파구 …
    - C:₩ProgramFiles/Java11/java.exe
    - /var/opt/www/file - 최상위 디렉토리 부터 나아가는 방식
  - 상대 경로
    - 내 오른쪽 집의 윗집의 …
    - ./var/file.c
    - ../bin/Main.class - '나(프로세스) ' 라는 대상을 기준으로 나아가는 방식
    - .. -> 상위 디렉토리
    - . -> 현재 디렉토리

2. Files and I/O Streams

- Stream
  - 데이터를 연속적으로 시퀀셜하게 버퍼를 사용한 입출력
  - System.out.println -> PrintStream 클래스

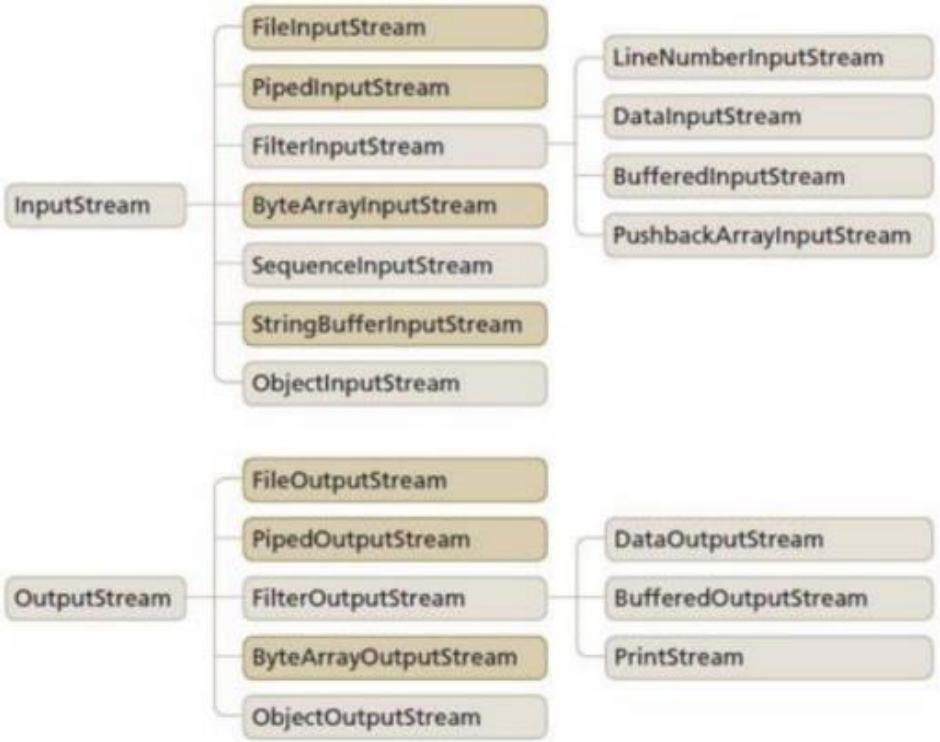
```
public class PrintStream extends FilterOutputStream
    implements Appendable, Closeable
{

    private final boolean autoFlush;
    private boolean trouble = false;
    private Formatter formatter;
```



2. Files and I/O Streams

- 바이트 단위로 입출력하는 클래스
- 추상 클래스인 InputStream과 OutputStream을 상속



2. Files and I/O Streams

- 문자 단위로 입출력하는 클래스
- 추상 클래스인 Reader와 Writer를 상속

2. Files and I/O Streams

- 다양한 방식으로의 사용
  - Stream은 다양한 객체와 사용될 수 있다.
  - 상황에 따라 적절히 사용할 객체를 선정하는 것이 중요

```
1 BufferedWriter bw1 = new BufferedWriter(new FileWriter("PATH"));
2 BufferedWriter bw2 = new BufferedWriter(new OutputStreamWriter(System.out));
3 FileWriter fw = new FileWriter("PATH");
4
5 BufferedReader br1 = new BufferedReader(new FileReader("PATH"));
6 BufferedReader br2 = new BufferedReader(new InputStreamReader(System.in));
7 FileReader fr = new FileReader("PATH");
8
9 Scanner scanFile = new Scanner(new File("PATH"));
```

2. Files and I/O Streams

- FileReader 사용해보기 – 가장 기본적인 방법

```java
public class FileReaderTest {

    public static void main(String[] args) throws InterruptedException {
        String userDir = System.getProperty("user.dir");

        try (FileReader reader = new FileReader(new File(userDir, "test.txt"))) {
            int ch;
            while ((ch = reader.read()) != -1) {
                System.out.printf("%c", (char) ch);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2. Files and I/O Streams

- FileReader 사용해보기 – BufferedReader와 결합

```java
public class BufferedTest {
    public static void main(String[] args) {
        String userDir = System.getProperty("user.dir");

        try (BufferedReader reader = new BufferedReader(new FileReader(Path.of(userDir, "test.txt").toFile()))) {
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2. Files and I/O Streams

- **실습 문제 1**
  - User directory에 위치한 problem1.txt를 읽어서,
  - 각 라인 마다의 수를 더하고
  - 각 라인의 합을 모두 곱하여 출력해보자.
  - 예시)
  - 3 7 5 4 2 (더하면 21)
  - 3 8 7 4 6 (28)
  - 9 7 8 4 1 (29)
  - 출력 => 21*28*29 = 17052

  - problem1.txt 내용

    1 3 4 7 0 3 5 7 6 5 2 3 5
    9 8 5 4 6 0 1 3 5 4 6 8 7
    0 3 5 7 4 1 8 9 5 3 6 4 2

2. Files and I/O Streams

- FileWriter 사용해보기 – BufferedWriter와 결합

```java
public class BufferedWriterTest {
    public static void main(String[] args) {
        String userDir = System.getProperty("user.dir");

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(new File(userDir, "write test.txt")))) {
            writer.write("Hello World!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2. Files and I/O Streams

- **실습 문제 2**
  - 실습 문제 1을 약간 개조해보자.

  - 실습 문제 1에서 입력된 문자열을 그대로 파일에 출력하고
  - 아래에 답을 작성한다.
  - 출력 파일명: Problem2.txt

  - 예시)

[입력]
1 3 4 7 0 3 5 7 6 5 2 3 5
9 8 5 4 6 0 1 3 5 4 6 8 7
0 3 5 7 4 1 8 9 5 3 6 4 2
[답]
250

3. NIO

- New-IO
  - java.io 클래스들의 입출력 성능 문제로 개발된 패키지

NIO에서 제공하는 패키지

NIO 패키지	포함되어 있는 내용
java.nio	다양한 버퍼 클래스
java.nio.channels	파일 채널, TCP 채널, UDP 채널 등의 클래스
java.nio.channels.spi	java.nio.channels 패키지를 위한 서비스 제공자 클래스
java.nio.charset	문자셋, 인코더, 디코더 API
java.nio.charset.spi	java.nio.charset 패키지를 위한 서비스 제공자 클래스
java.nio.file	파일 및 파일 시스템에 접근하기 위한 클래스
java.nio.file.attribute	파일 및 파일 시스템의 속성에 접근하기 위한 클래스
java.nio.file.spi	java.nio 패키지를 위한 서비스 제공자 클래스

IO 와 NIO의 차이점

구분	IO	NIO
입출력 방식	스트림 방식	채널 방식
버퍼 방식	넌버퍼	버퍼
비동기 방식	지원 안 함	지원
블로킹 / 넌블로킹 방식	블로킹 방식만 지원	블로킹 / 넌블로킹 방식 모두 지원

3. NIO

- File 클래스
  - 파일 시스템과 관련한 기능을 지원한다.

```java
import java.io.File;

public class FileTest {
    public static void main(String[] args) {
        String userDir = System.getProperty("user.dir");

        File f = new File(userDir, "test.txt");

        System.out.println(f.getAbsolutePath());
        System.out.println(f.exists());
        System.out.println(f.length());

        for (String list : new File(userDir).list()) {
            System.out.println(list);
        }
    }
}
```

3. NIO

- **실습 문제 3**
  - 키보드로 입력 받아서 파일 이름을 바꿔주는 프로그램을 개발해보자.

  - test.txt hi.txt => test.txt를 hi.txt로 이름을 바꿔줘야함

4. XML Serialization

- Serialization – 직렬화
  - 특정 데이터구조나 객체가 담고 있는 정보를 다른 시스템에 전달하고 재구성 할 수 있도록 하는 것

  - List<Student> sList = new ArrayList<>();
  - sList.add(new Student("name1", "id", age));
  - sList.add(new Student("name2", "id", age));
  - sList.add(new Student("name3", "id", age));

  - sList를 다른 시스템에 전송할 수 있는 방법이 있을까?
  - 또는 저장해두었다가 나중에 꺼내 쓸 수 있을까?

  - 이를 위한 방법이 serialization
  - Serialized된 데이터를 꺼내 쓰는 것이 Deserialization-역직렬화

4. XML Serialization

- 직렬화를 위한 방법
  - JSON
    - XML을 대체하기 위해 나온 포멧

```
{
   "이름": "홍길동",
   "나이": 25,
   "성별": "여",
   "주소": "서울특별시 양천구 목동",
   "특기": ["농구", "도술"],
   "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},
   "회사": "경기 수원시 팔달구 우만동"
}
```

  - XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE recipe PUBLIC "-//Happy-Monkey//DTD RecipeBook//EN
"http://www.happy-monkey.net/recipebook/recipebook.dtd">

<recipe>

<title>Peanutbutter On A Spoon</title>

<ingredientlist>
  <ingredient>Peanutbutter</ingredient>
</ingredientlist>

<preparation>Stick a spoon in a jar of peanutbutter, scoop
and pull out a big glob of peanutbutter.</preparation>

</recipe>
```

4. XML Serialization

- 직렬화를 위한 방법
  - XML의 문법
    - 기본 요소 = Element
    - <coffee>americano</coffee>
  - 상위 및 하위 요소

```
<StudentList>
        <Student>
                <name>name1</name>
                <id>201921166</id>
        </Student>
        <Student>
                <name>name2</name>
                <id>202215681</id>
        </Student>
</StudentList>
```

4. XML Serialization

```java
public class Book {
    private Long id;
    private String name;
    private String author;
    private Date date;

    public Book(Long id, String name, String author, Date date) {
        super();
        this.id = id;
        this.name = name;
        this.author = author;
        this.date = date;
    }

    // public getters, setters
}
```

4. XML Serialization

```java
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.annotation.XmlElement;

public class BookList {
    @XmlElement(name="book")
    private List<Book> bookList = new ArrayList<>();

    public void add(Book book) {
        bookList.add(book);
    }

    public List<Book> getBookList() {
        return bookList;
    }
}
```

4. XML Serialization

```java
public class BookTest {

    public static void main(String[] args) throws JAXBException, IOException {
        BookList list = new BookList();
        list.add(new Book(1L, "Book12", "Author1", new Date()));
        list.add(new Book(2L, "Book3", "Author12", new Date()));
        list.add(new Book(3L, "Book6", "Author13", new Date()));

        JAXB.marshal(list, new File("./book.xml"));
    }
}
```

4. XML Serialization

- Marshal
  - Serialize의 다른 표현
  - Public instance variable 이나
  - Public getters, setter가 있는 필드에 대해서 직렬화 한다.

4. XML Serialization

- 다양한 어노테이션
  - XmlType propOrder -> 엘리먼트의 순서를 지정할 수 있음

```java
@XmlType(propOrder = { "id", "name", "author", "date" })
public class Book {
    private Long id;
    private String name;
    private String author;
    private Date date;

  ...
```

4. XML Serialization

- 다양한 어노테이션
  - Private 변수에 public setter가 있을 경우,
  - Setter 위에 XmlElement name을 지정할 경우 XML 상에서의 엘리먼트 이름을 변경할 수 있다.

```
1 @XmlElement(name = "title")
2 public void setName(String name) {
3     this.name = name;
4 }
```

4. XML Serialization

- 다양한 어노테이션
  - Private 변수에 public setter가 있을 경우,
  - XmlTransient가 붙을 경우, 해당 필드가 직렬화 되지 않도록 할 수 있다.

```
1 @XmlTransient
2 public void setAuthor(String author) {
3     this.author = author;
4 }
```

4. XML Serialization

- 다양한 어노테이션
  - https://docs.oracle.com/javase/tutorial/jaxb/intro/
  - 자세한 사항은 위 documentation 참조

4. XML Serialization

- XML 역직렬화

```java
public class UnMarshalTest {
    public static void main(String[] args) {
        BookList list =  JAXB.unmarshal(new File("./book.xml"), BookList.class);

        for (Book book : list.getBookList()) {
            System.out.println(book.getName());
        }
    }
}
```

4. XML Serialization

- **실습 문제 4**
  - 네트워크를 통해 데이터 역직렬화 해보기
  - 아래 코드는 인터넷에 있는 XML 파일에 대한 InputStream을 생성하는 역할

```java
import java.io.InputStream;
import java.net.HttpURLConnection;

import java.net.URL;

public class BoxOfficeTest {
    public static void main(String[] args) throws Exception {
        URL url;
        HttpURLConnection connection = null;
        url = new URL("https://raw.githubusercontent.com/lani009/Ajou-OOP-Practice_22/main/movie.xml");
        connection = (HttpURLConnection) url.openConnection();

        connection.setRequestMethod("GET");

        InputStream inputStream = connection.getInputStream(); // InputStream to JAXB

    }
}
```