

Week 7. FILE I/O & Serialization & Deserialization

– Store Data into File.

Review.. Very Important

Method of HashSet

메서드	설명
<code>add(E e)</code>	요소 추가 (true 반환: 새로 추가됨)
<code>remove(Object o)</code>	요소 제거
<code>contains(Object o)</code>	요소 존재 여부 확인
<code>isEmpty()</code>	비어 있는지 확인
<code>size()</code>	요소 개수 반환
<code>clear()</code>	전체 요소 삭제
<code>iterator()</code>	요소 순회용 반복자 반환
<code>toArray()</code>	배열로 변환
<code>stream()</code>	Stream API 사용 가능
<code>retainAll(Collection<?>)</code>	주어진 컬렉션과의 교집합만 유지

Method of HashMap

메서드	설명
<code>put(K key, V value)</code>	key에 value 저장 (기존 값 덮어쓰기)
<code>get(Object key)</code>	key에 해당하는 value 반환 (없으면 null)
<code>remove(Object key)</code>	key-value 쌍 삭제
<code>containsKey(Object key)</code>	특정 key 존재 여부
<code>containsValue(Object value)</code>	특정 value 존재 여부
<code>isEmpty()</code>	비어 있는지 확인
<code>size()</code>	전체 key 개수 반환
<code>clear()</code>	모든 entry 삭제
<code>keySet()</code>	모든 key 집합 반환 (Set<K>)
<code>values()</code>	모든 value 모음 반환 (Collection<V>)
<code>entrySet()</code>	모든 entry (key-value 쌍) 반환 (Set<Map.Entry<K, V>>)
<code>putIfAbsent(K key, V value)</code>	key가 없을 때만 추가
<code>compute(), computeIfAbsent()</code>	key 기반 연산으로 값 추가/수정

What is Hash...?

Hash란 데이터를 고정된 크기의 고유한 숫자 값(해시값) 으로 변환하는 함수적 알고리즘입니다.

EX) “apple” 을 Hash 함수에 넣으면, 19382914 같은 값이 나온다.

```
1  int hash = 0;
2  for (int i = 0; i < s.length(); i++) {
3      hash = 31 * hash + s.charAt(i);
4  }
```

위는 실제, String 데이터에 대한, hash함수를 만드는 과정입니다.

자바 String은 31을 기반으로 하는 다항 해시(polynomial hash) 알고리즘을 사용합니다.

How to use Hash...?

2. hashCode 재보정을 통한, 분산처리

기존에 어떠한 방식으로 만들어진 hashCode나, 사용자가 오버라이딩한 hashCode가 엉망일 수 있습니다.

따라서, h라는 보정된 해쉬값을 다시 만들어서, 테이블의 index 값에 집어넣습니다.

```
1  int h = key.hashCode();  
2  int hash = (h >>> 16) ^ h;  
3  int index = (table.length - 1) & hash;
```

What is Priority Queue..?

Priority Queue는 일반적인 큐(Queue)와는 다르게, 우선순위(priority)에 따라 요소를 꺼내는 자료구조입니다.

즉, 먼저 들어온 순서가 아닌, 우선순위가 높은 요소가 먼저 나가는 큐입니다.

여러분이 우선순위를 명시해 놓으면, 그 기준을 보고 자료를 정렬해주는 클래스가 내장되어 있습니다.

Table of Contents

1. FileReading and Writing
 - A. FileReader & FileWriter
 - B. BufferedReader & BufferedWrtier
 - C. 상대 경로에 대하여
 - D. 덮어쓰기가 아닌 이어쓰기, append에 대하여
2. Serialization & Deserialization
 - A. 직렬화와 역 직렬화

File I/O

Java I / O (Input / Output)는, java.io 패키지에 있다.

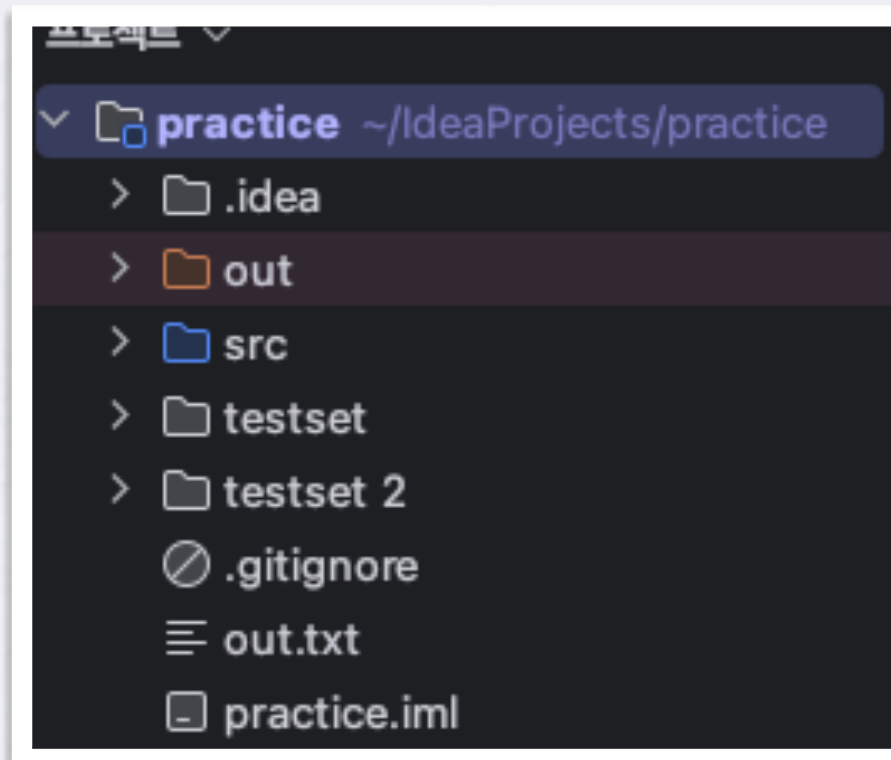
```
1 import java.io.File;  
2 import java.io.FileReader;  
3 import java.io.FileWriter;  
4 import java.io.BufferedReader;  
5 import java.io.BufferedWriter;
```

이번 시간에 우리는, Image 같은 바이트 단위로 읽는 것이 아닌 문자 단위로, Reading을 할 것이다.

```
1 BufferedImage image = ImageIO.read(new File("images/input.png"));  
2 ImageIO.write(image, "png", new File("images/output.png"));
```

File I/O

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         File file = new File("./out.txt");  
4         BufferedWriter bw = new BufferedWriter(new FileWriter(file));  
5  
6         bw.write("안녕하세요.");  
7         bw.flush(); bw.close();  
8     }  
9 }
```



프로젝트 바로 아래에,
out.txt 파일이 생겼습니다.

근데 왜 하필 프로젝트 폴더 안
에 생성될까요?

Absolute Path / Relative Path

폴더 안에는 자신을 가리키는 폴더가 존재합니다.

또한, 부모를 가리키는 폴더가 존재합니다.

```
~/s/class_programing3 ls -al
```

```
> ls -al
total 456
drwxr-xr-x  20 hyeonseok staff   640  5 22 16:18 .
drwxr-xr-x  13 hyeonseok staff   416  5  6 17:18 ..
-rw-r--r--@  1 hyeonseok staff 10244  5 16 03:46 .DS_Store
-rwxr-xr-x   1 hyeonseok staff 33432  3  7 11:17 helloworld
-rw-r--r--   1 hyeonseok staff   275  5 26 01:43 main.java
-rwxr-xr-x   1 hyeonseok staff 83560  4  2 17:07 suffix_tree
```

Absolute Path / Relative Path

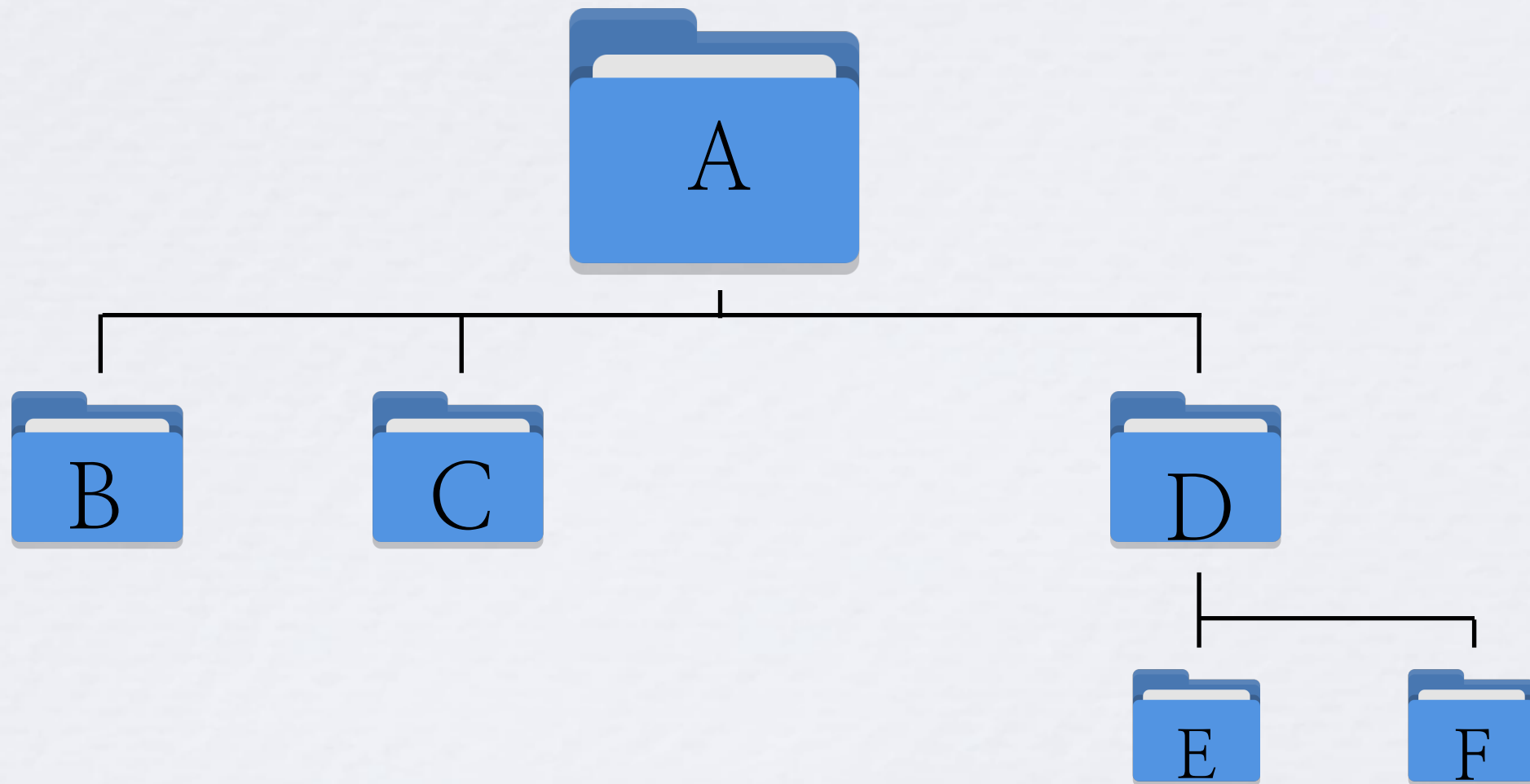
cd : 현재 작업 디렉토리를 바꾸는 명령어

cd [디렉토리 경로]

디렉토리 경로를 적을 때는, 상대경로, 절대경로를 둘다 적을 수 있다.
현재 작업 디렉토리의 주소는 cd 또는 Get-Location 명령어를 적으면 알 수 있다.

```
> pwd  
/Users/hyeonseok/IdeaProjects/practice/src
```

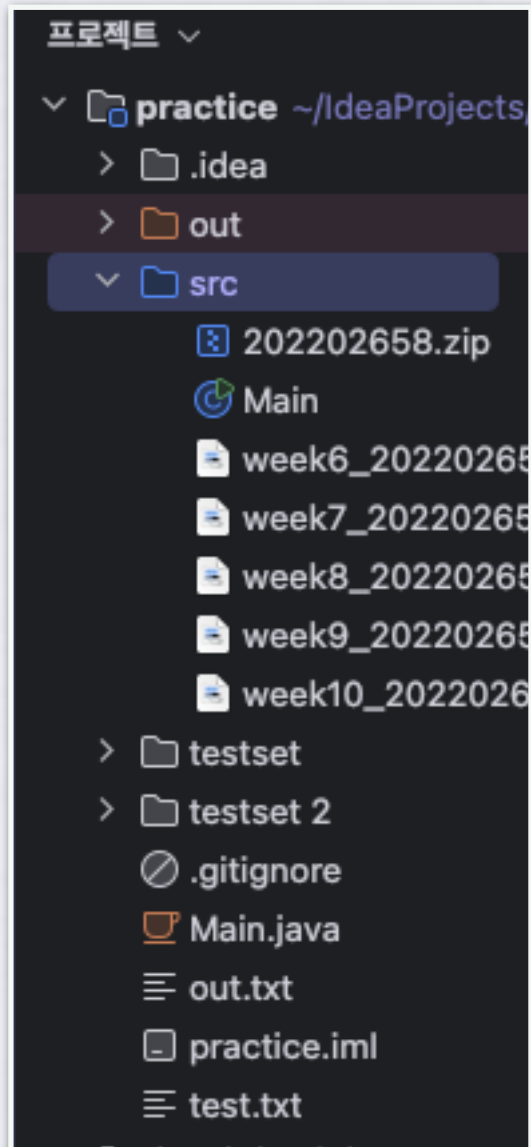
Absolute Path / Relative Path



.. 은 부모 폴더로 이동하는 것이고, .은 현재 폴더로 이동하는 것이다.

.. 을 자유롭게 쓰면, 원하는 폴더로 상대경로로 이동할 수 있다.

Absolute Path / Relative Path



```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(new File(".").getAbsolutePath());  
4     }  
5 }
```

```
/Users/hyeonseok/IdeaProjects/practice/.
```

File Class의 `getAbsolutePath()` 매서드를 활용하여, 현재 파일 객체가 무슨 파일을 가리키는지 절대경로로 출력해 볼 수 있다.

Why use relative Path..?

이식성과, 재사용성을 높이기 위함.

절대경로는 특정 사용자/컴퓨터의 디렉토리에 강하게 종속된다.

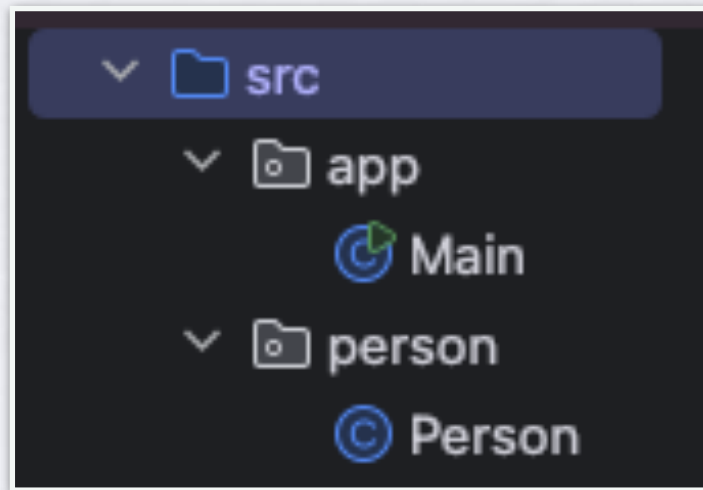
```
1 public class Main {  
2     public static void main(String[] args) {  
3         File file = new File("C:\\Users\\yourname\\Documents\\MyProject\\data\\file.txt");  
4     }  
5 }
```

여러분들의 컴퓨터 속에서, 파일을 불러오는데, 프로젝트 폴더 속, 데이터 폴더에서 가져온다고 합시다.

이를 절대경로로 가져오면, 다른 사용자의 컴퓨터에서 프로그램은 작동하지 않겠죠?

Why use relative Path..?

프로젝트 내부의 유기적인 구조 유지



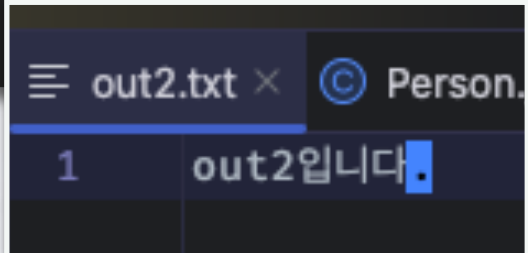
```
1 package app;
2
3 import person.Person;
4
5 public class Main {
6     public static void main(String[] args) {
7         Person person = new Person();
8     }
9 }
```

실제로 프로젝트 package import 할 때는, src가 루트 폴더로써 작동합니다.

src 기준으로 “./person/Person” 자바 파일에서, Person을 가져옵니다.

Why use Buffer???

```
1 public class Main {
2     public static void main(String[] args) throws IOException {
3         // 왜 이걸 사용하는걸까?
4         BufferedWriter bw = new BufferedWriter(new FileWriter("out.txt"));
5
6         FileWriter fw = new FileWriter("out2.txt");
7         fw.write("out2입니다.");
8         fw.flush();
9         fw.close();
10    }
11 }
```



The screenshot shows an IDE window with a project named 'practice'. The file explorer on the right lists the project structure: .idea, out, src, testset, testset 2, .gitignore, input.txt, Main.java, out.txt, and out2.txt. The 'out2.txt' file is selected and its content is displayed in the editor: 'out2입니다.'.

BufferedWriter..? 왜 구지 저런걸 사용하는건가요?
안 써도, 작업하는 데에 아무런 문제가 생기지 않는데요?

Why use Buffer???

```
1  FileWriter fw = new FileWriter("out.txt");
2  fw.write("H");
3  fw.write("e");
4  fw.write("l");
5  fw.write("l");
6  fw.write("o");
7  fw.close();
```

FileWriter는 문자 데이터를 파일로 쓰는 기능은 있지만,
문자 하나하나를 쓸 때마다 곧바로 디스크에 접근합니다.
→ 디스크 입출력(IO)은 느리고 비용이 큰 작업이기 때문에,
매번 쓰면 매우 비효율적입니다.

how to use BufferedWriter?

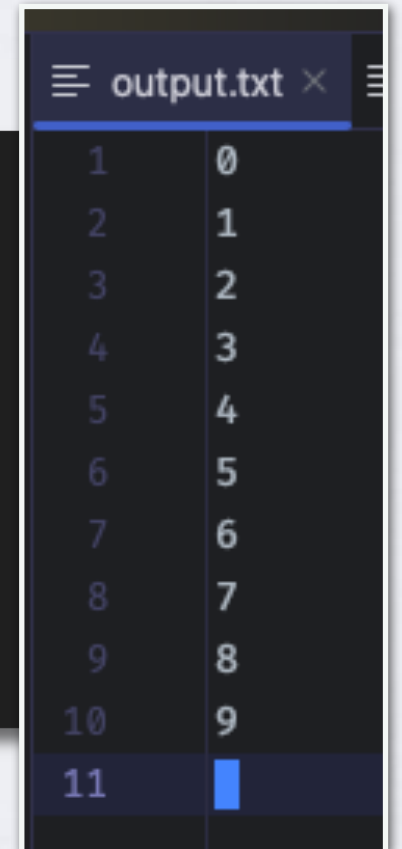
```
1  BufferedWriter bw = new BufferedWriter(new FileWriter("out.txt"));
2  bw.write("Hello"); // 메모리에 먼저 저장됨
3  bw.close();        // 한 번에 디스크에 기록됨
```

BufferedWriter는 메모리 버퍼에 데이터를 먼저 모아두었다가,
버퍼가 가득 찼거나 flush() 또는 close()가 호출될 때 실제로
한 번에 파일에 씁니다.

마지막에 무조건 flush()를 해서, 버퍼에 모든 내용을 저장하는 행동을 명시해 주어야 합니다.

how to use BufferedWriter?

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         BufferedWriter bw = new BufferedWriter(new FileWriter("output.txt"));  
4  
5         for (int i=0;i<10;i++) {  
6             bw.write(i+"\n");  
7         }  
8         bw.close();  
9     }  
10 }
```

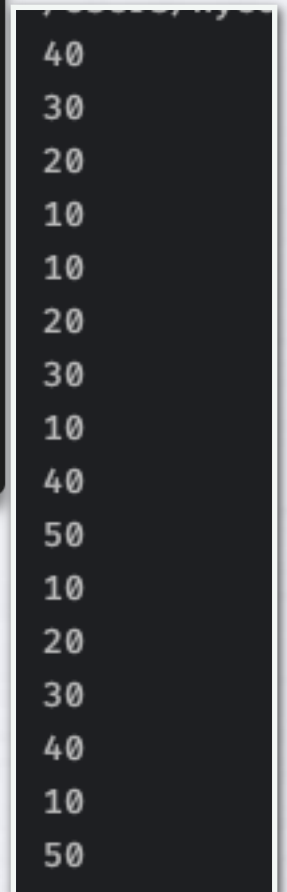
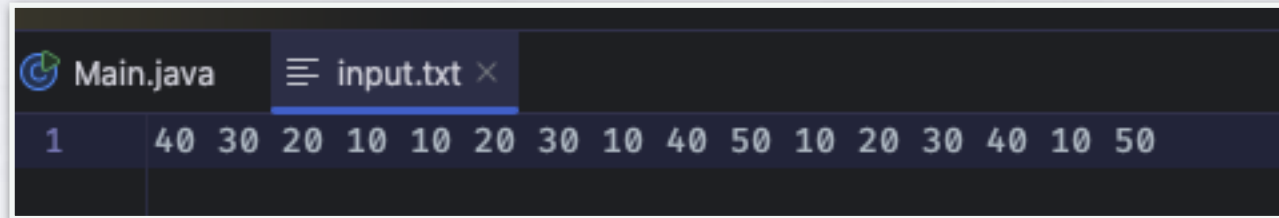
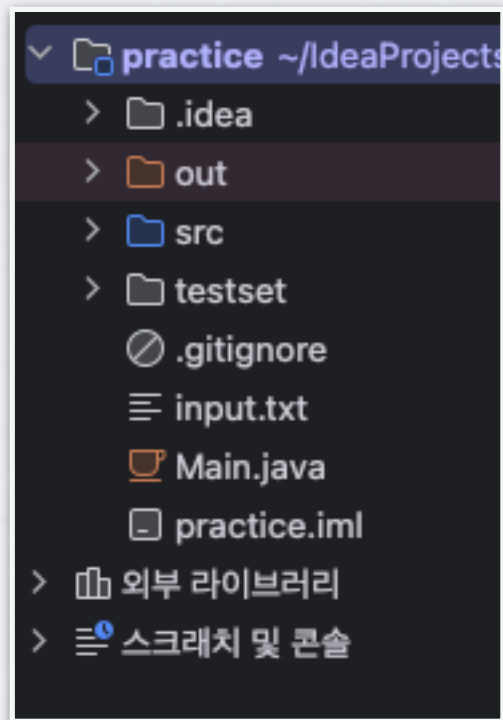


1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	

BufferedWriter는 내부적으로 파일 핸들(파일에 접근하기 위한 운영체제 자원)을 잡고 있습니다.

close()를 호출하지 않으면 이 자원이 해제되지 않아, 메모리 누수 또는 파일 잠금 현상이 발생할 수 있습니다.

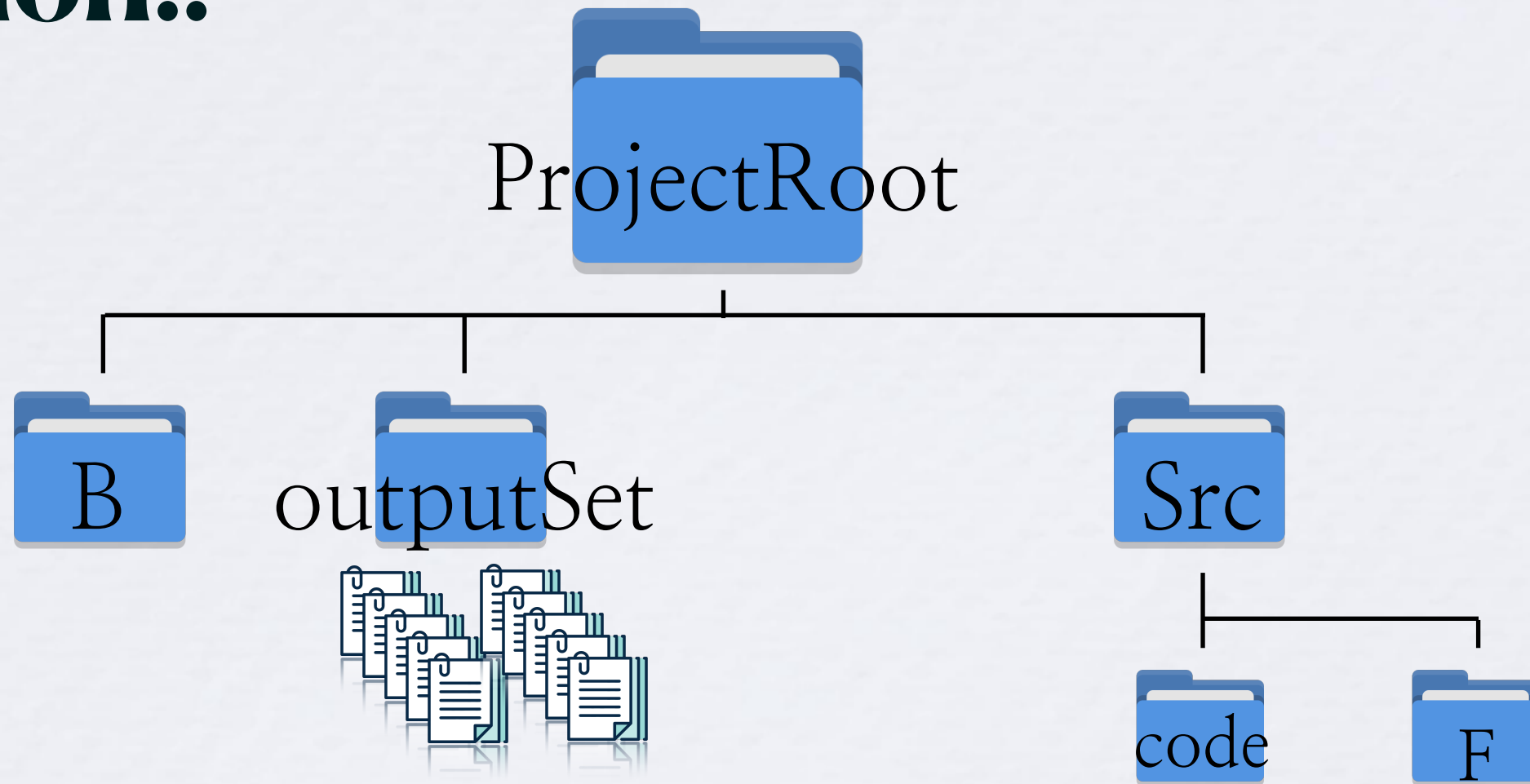
How to use BufferedReader?



기본 입출력을 할 때와 똑같이 하면 된다.

BufferedReader가 읽는 자원이, InputStreamReader()에서, FileReader()로 바뀐 것뿐이다.

Mission..



프로젝트 폴더를 루트로 하여 작업한다고 하자.

프로젝트 폴더에서 outputSet이라는 폴더를 만들어서, 1부터 10까지의 “\${number}.txt” 파일을 만들어보자.

각 텍스트 파일에는 각 텍스트 파일 숫자를 작성해 놓자.

Mission..

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         for (int i=1;i<=10;i++) {  
4             BufferedWriter bw = new BufferedWriter(new FileWriter("./outputSet/" + i + ".txt"));  
5  
6             bw.write(i);  
7             bw.close();  
8         }  
9     }  
10 }
```

루트 폴더가 기준이니까.. 그냥 바로 outputSet 안에
\${i}.txt 파일을 만들어서 i를 적고 close 하면 되겠다!!

이게 과연 잘 작동할까요???

Mission..

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         for (int i=1;i<=10;i++) {  
4             BufferedWriter bw = new BufferedWriter(new FileWriter("./outputSet/" + i + ".txt"));  
5  
6             bw.write(i);  
7             bw.close();  
            }  
        }  
    }
```

Exception in thread "main" java.io.FileNotFoundException Create breakpoint : ./outputSet/1.txt (No such file or directory)
at java.base/java.io.FileOutputStream.open0(Native Method)
at java.base/java.io.FileOutputStream.open(FileOutputStream.java:295)
at java.base/java.io.FileOutputStream.<init>(FileOutputStream.java:237)
at java.base/java.io.FileWriter.<init>(FileWriter.java:124)
at app.Main.main(Main.java:9)

FileNotFoundException..? Exception?? 파일을 못찾았다는 건가?

No such file or directory..? 디렉토리가 없다고...?

Mission..

폴더는 “구조” 이고, 파일은 “데이터” 입니다.

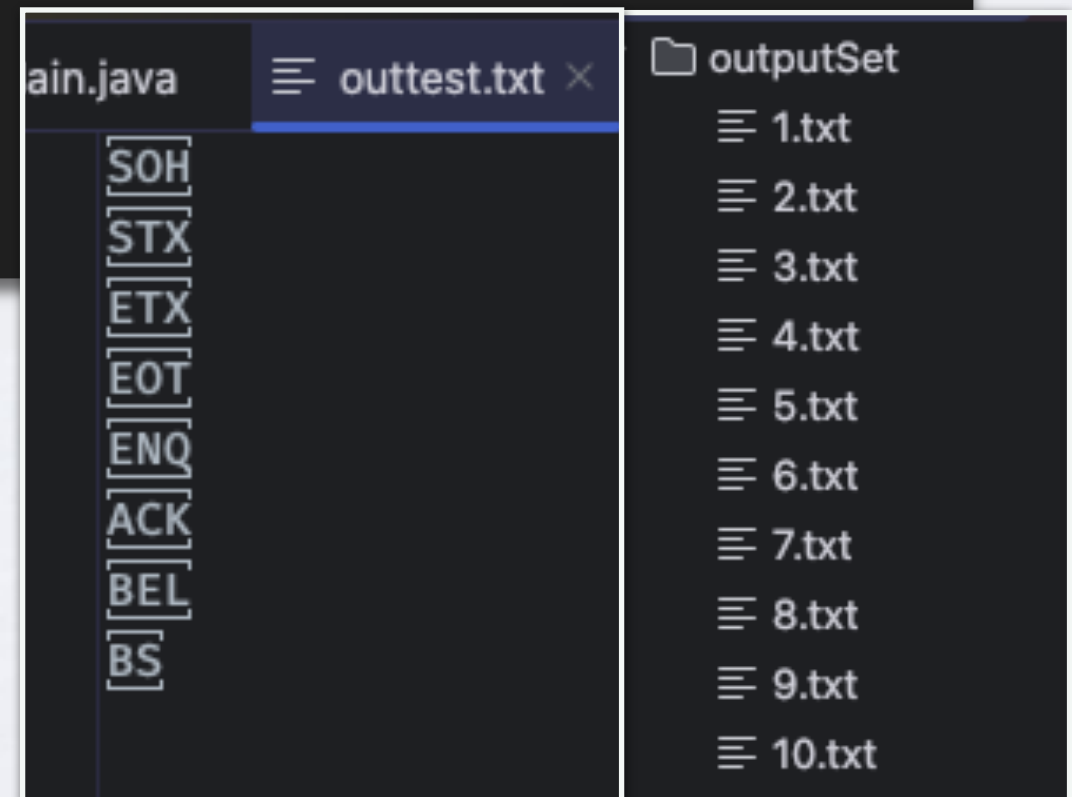
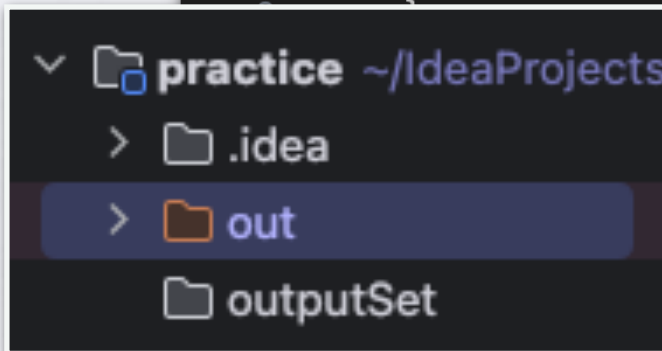
파일은 “내용을 저장하는 객체” 이고, 파일이 없으면,
그냥 만들어 버리면 됩니다.

폴더는 경로를 구성하는 상위 구조입니다. 존재하지
않은 경로에 파일을 만들려고 하면, OS가 에러를 보
냅니다.

파일 생성의 책임과 디렉터리 구조의 책임을 명확히
분리하기 위한 설계입니다.

Mission..

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         for (int i=1;i<=10;i++) {  
4             BufferedWriter bw = new BufferedWriter(new FileWriter("./outputSet/" + i + ".txt"));  
5  
6             bw.write(i);  
7             bw.close();  
8         }  
9     }  
10 }
```



bw.write()에 숫자를 적었는데 이게 뭐죠…?

Mission..

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         for (int i=1;i<=10;i++) {  
4             BufferedWriter bw = new BufferedWriter(new FileWriter("./outputSet/" + i + ".txt"));  
5  
6             bw.write(i);  
7             bw.close();  
8         }  
9     }  
10 }
```

```
© java.io.BufferedWriter  
public void write(  
    int c  
)  
throws java.io.IOException
```

BufferedWriter의 write에 int만 넣는다면, 매개변수로 int를 받는 write를 호출하게 됩니다.

이 함수는, 들어온 숫자에 해당하는 아스키코드를 write 하는 함수입니다.

Mission..

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         for (int i=1;i<=10;i++) {  
4             BufferedWriter bw = new BufferedWriter(new FileWriter("./outputSet/" + i + ".txt"));  
5  
6             bw.write(i+"");  
7             bw.close();  
8         }  
9     }  
10 }
```

write(String text), 를 호출하기 위해, i에 “” 공백 문자열을 더해, String으로 변환 합니다.

그 후, write를 호출하면 이번에는 정상적으로 적히게 됩니다.

Appending Data to a File in Java

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         BufferedWriter bw = new BufferedWriter(new FileWriter("test.txt", true));  
4     }  
5 }
```

파일이 자꾸 덮어쓰기가 되는데, 이어쓰기는 어떻게 하나요..?

FileWriter의 매개변수 뒤에, true를 하여, append 기능을 활성화할 수 있습니다.

Serialization & Deserialization

직렬화 (Serialization) : 객체를 파일 등에 저장할 수 있도록 byte 형태로 변환 하는 과정

역직렬화(Deserialization) : 저장된 byte 데이터를 다시 객체로 복원하는 과정

```
1  class Person implements Serializable{
2      public String name;
3      public int age;
4
5      public Person(String name, int age) {
6          this.name = name;
7          this.age = age;
8      }
9  }
```

Serialization & Deserialization

```
1  class Person implements Serializable{
2      public String name;
3      public int age;
4
5      public Person(String name, int age) {
6          this.name = name;
7          this.age = age;
8      }
9  }
```

Serializable : 마커 인터페이스 정말로 아무 메서드도 없는 인터페이스이다.

직렬화 메서드를 사용할 것인데, 이게 없으면 안 해준다. 직렬화 안정성을 도모하기 위해 명시적으로 개발자가 선언하게 만든 것입니다.

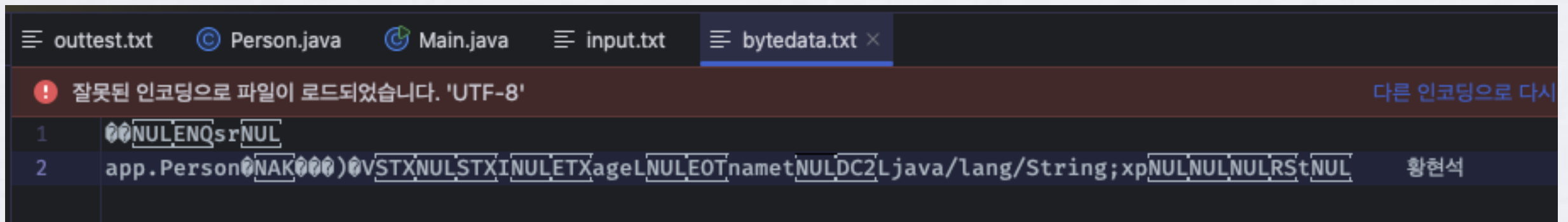
Serialization & Deserialization

```
1  public class Main {  
2      public static void main(String[] args) throws IOException {  
3          Person person = new Person("황현석", 30);  
4  
5          ObjectOutputStream oos = new ObjectOutputStream(  
6              new BufferedOutputStream(  
7                  new FileOutputStream("bytedata.txt"))));  
8  
9          oos.writeObject(person);  
10         oos.close();  
11     }  
12 }
```

byte 기반이므로, Reader가 아닌, Stream 기반 입출력 개체를 사용합니다.

똑같이 Buffer를 사용할 수 있습니다.

Serialization & Deserialization



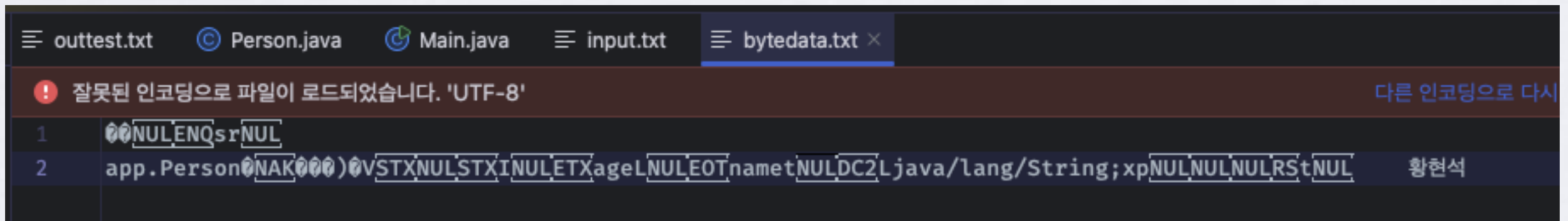
The screenshot shows an IDE window with several tabs: 'outtest.txt', 'Person.java', 'Main.java', 'input.txt', and 'bytedata.txt'. The 'bytedata.txt' tab is active and displays a warning message: '잘못된 인코딩으로 파일이 로드되었습니다. 'UTF-8'' (File loaded with incorrect encoding. 'UTF-8'). Below the warning, the file content is shown as garbled text, including 'app.Person' and 'Ljava/lang/String;'. A line number '2' is visible on the left, and the name '황현석' (Hwang Hyun-seok) is on the right.

아니 이게 뭔가요...?

ObjectOutputStream은 객체를 JVM 내부 구조에 맞는 형식으로 바이너리(binary) 데이터로 변환하여 저장합니다.

사람이 읽을 수 없는 2진수 형태의 바이너리 파일이고, 읽으려고 노력하지 않아도 됩니다..

Serialization & Deserialization



```
outtest.txt  Person.java  Main.java  input.txt  bytedata.txt ×
! 잘못된 인코딩으로 파일이 로드되었습니다. 'UTF-8'
1  app.Person
2  app.Person(NAK(000))VSTXNULSTXINULETXageLNULEEOTnametNULDC2Ljava/lang/String;xpNULNULNULRStNUL
```

포함 항목	설명
클래스 이름	직렬화된 객체의 클래스 이름
serialVersionUID	클래스의 고유 버전 식별자
필드 정보	각 필드 이름, 타입
필드 값	실제 객체 안에 들어있는 값들
중첩 객체	다른 객체를 참조하고 있을 경우 그 객체도 직렬화

Serialization & Deserialization

```
1 public class Main {  
2     public static void main(String[] args) throws IOException, ClassNotFoundException {  
3         ObjectInputStream ois = new ObjectInputStream(  
4             new BufferedInputStream(  
5                 new FileInputStream("bytedata.txt")));  
6  
7         Person person = (Person) ois.readObject();  
8         System.out.printf("%s %d\n", person.name, person.age);  
9     }  
10 }
```

황현석 30

이번에는 InputStream을 사용합니다. 구조화 되어 있어서 뭔가 비슷한 느낌을 받을 수 있습니다.

정말로 저장한 값이 꺼내와 지는 것을 볼 수 있습니다.

Serialization & Deserialization

```
1   ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("databyte.txt"));
2
3   oos.writeObject(new Person("Alice", 25));
4   oos.writeObject(new Person("Bob", 30));
5   oos.writeObject(new Person("Charlie", 28));
6
7   oos.close();
8 }
```

```
1   ObjectInputStream ois = new ObjectInputStream(new FileInputStream("databyte.txt"));
2
3   while (true) {
4       try {
5           Person p = (Person) ois.readObject();
6           System.out.println(p.name + " - " + p.age);
7       } catch (EOFException e) {
8           break; // 파일 끝에 도달하면 루프 종료
9       }
10  }
11
12  ois.close();
```


Serialization & Deserialization

```
1 List<Person> people = new ArrayList<>();
2 people.add(new Person("Alice", 25));
3 people.add(new Person("Bob", 30));
4 people.add(new Person("Charlie", 28));
5
6 ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("people.dat"));
7 oos.writeObject(people); // 리스트 전체를 직렬화
8 oos.close();
```

```
1 ObjectInputStream ois = new ObjectInputStream(new FileInputStream("people.dat"));
2 List<Person> people = (List<Person>) ois.readObject();
3 ois.close();
```

리스트 자체를 직렬화 해버려서, 한꺼번에 몽땅 저장 할 수 있습니다.

Serialization & Deserialization

```
1 HashSet<Person> people = new HashSet<>();
2 people.add(new Person("Alice", 25));
3 people.add(new Person("Bob", 30));
4
5 ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("people.dat"));
6 oos.writeObject(people);
7 oos.close();
```

```
1 try {
2     ObjectInputStream ois = new ObjectInputStream(new FileInputStream("people.dat"));
3     HashSet<Person> people = (HashSet<Person>) ois.readObject();
4
5     ois.close();
6
7     for (Person p : people) {
8         System.out.println(p.name + " - " + p.age);
9     }
10 } catch (IOException | ClassNotFoundException e) {
11     e.printStackTrace();
12 }
```

HashSet, HashMap 등도, 직렬화 가능합니다.

...?? Accident...

실제로, 2022학년도 컴퓨터 프로그래밍 2 기말 시험에 출제됨.

당시, 직렬화, 역직렬화에 대한 지식이 있는 학생들은 거의 찾을 수 없었음.

또한, 학생들은, 아무것도 모르고, 해당 함수에 대해 알지도 못했으며, 파일 읽기 쓰기도 많이 미흡한 상태 였다.

당시 시험은, 오픈북, 레퍼런스를 찾아 볼 수 있는 환경이었으나, 학생들의 수준은 그리 높지 않았다.

많은 피해자들이 발생했으며.. 응용력이 뛰어나, 레퍼런스를 빨리 뒤져가며 푼 학생들은 살아남을 수 있었다.

We've completed Week 7