

23.03.18 11강

예외 (Exception)

프로그램을 설계할 때 실행시에 예외 상황이 발생 할 가능성이 있는 것을 예측하여, 사전에 예외 처리가 되도록 할 필요가 있음.

이럴 때 적절한 조치가 없으면 프로그램은 에러가 나며 죽어 버림.

예상 외의 상황에 적절한 조치를 취하는 것을 배운다

에러의 종류

1. 문법 에러 (syntax error)
2. 실행 시 에러 (runtime error)
3. 논리 에러 (logic error) 혹은 휴먼 에러(Human error)

	syntax error	runtime error	logic error
원인	코드의 형식적 오류	실행 중에 예상외의 사태가 발생하여 동작이 중지됨	기술한 처리 내용에 논리적인 오류가 있음
알아 채는 방법	컴파일하면 에러 남	실행하면 도중에 강제 종료 됨	실행하면 예상외의 값이 나옴
해결 방법	컴파일러의 지적을 보고 수정	에러	원인을 스스로 찾아서 해결해야 함

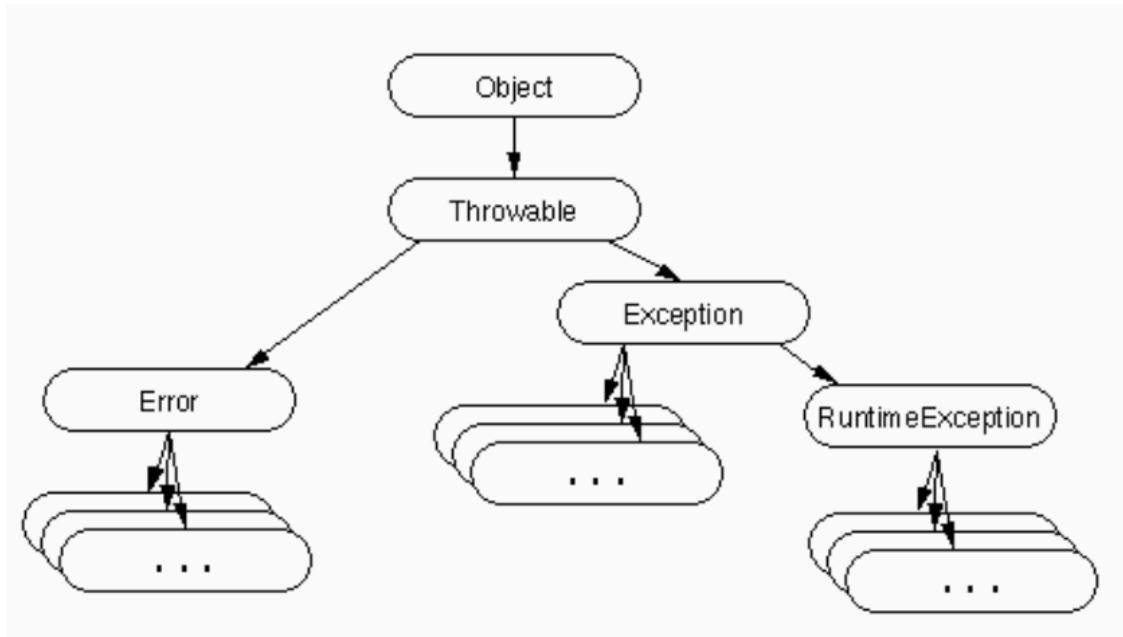
예외적 상황

- 메모리 부족
- 파일을 찾을 수 없음
- null 인 변수를 참조 했음

예외 처리의 흐름

```
try{
    //일반적인실행부분
    Thread.sleep(3000);
} catch (InterruptedException e){
    //예외가발생했을때처리를하는
    System.out.println("I에러발생!. 종료합니다");
    System.exit(1);
}
```

예외의 종류 - 클래스의 시점



대부분 Exception이다. Error는 드물다

예외 처리를 하지 않는 경우

```
import java.io.PrintWriter;

public class Main {
    public static void main(String[] args) {
        // PrintWriter는 IOException을 발생시킬 가능성이 있음
        // try-catch로 감싸지 않으면 컴파일 에러
        PrintWriter fw = new PrintWriter("data.txt");
    }
}
```

예외 발생을 준비

```
package Lesson.day11.fifteenth;

import java.io.*;

public class Main {
    public static void main(String[] args) {
        try {
            PrintWriter fw = new PrintWriter("data.txt");
        } catch (IOException e) {
            // 예외처리를 하여 프로그램이 강제 종료되지 않음
            System.out.println("에러가 발생했습니다");
        }
    }
}
```

상위 메소드에 미루는 방법

```
import java.io.*;

public class Main {
    public static void main(String[] args) throws IOException{
        // PrintWriter는 IOException을 발생시킬 가능성이 있음
        // try-catch로 감싸지 않으면 컴파일 에러
        PrintWriter fw = new PrintWriter("data.txt");
    }
}
```

main에 throws 하는 건 좋지 않다. → 예외처리를 안하겠다는 뜻

예외를 통쳐서 처리 → 비추

```
import java.io.*;

public class Main {
    public static void main(String[] args) {
        try {
            System.out.println("시작");
            something();
            System.out.println("끝");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void something() throws InterruptedException {
        Thread.sleep(3000);
    }
}
```

finally 항상 실행하는 코드

```
package Lesson.day11.fifteenth;

import java.io.*;

public class Main {
    public static void main(String[] args) {
        try {
            System.out.println("시작");
            something();
            FileWriter fw = new FileWriter("data.txt");
            fw.write("hello!!");
            System.out.println("끝");
        } catch (InterruptedException e) {
            e.printStackTrace();
        } finally {
            try {fw.close();} catch (IOException e){
                e.printStackTrace();
            }
        }
    }

    public static void something() throws InterruptedException {
        Thread.sleep(3000);
    }
}
```

```
package Lesson.day11.fifteenth;

import java.io.*;

public class Main {
    public static void main(String[] args) {
        try (FileWriter fw = new FileWriter("data.txt")) {
            System.out.println("시작");
            something();
            fw.write("hello!!");
            System.out.println("끝");
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public static void something() throws InterruptedException {
        Thread.sleep(3000);
    }
}
```

예외 인스턴스를 자신에게 던지기

예외 커스터마이징

연습문제

연습문제 15-1

다음과 같은 프로그램을 작성, 실행 후 runtime error 를 발생시키시오.

1. String 형 변수 s를 선언하고, null 을 대입한다
2. s.length() 의 내용을 표시하려고 한다

```
package Lesson.day11.fifteenth;

public class Exam15_1 {
    public static void main(String[] args) {
        String s = null;
        System.out.println(s.length());
    }
}
```

The screenshot shows the Eclipse IDE's console window. It displays a runtime exception: `Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because the variable "s" is null at Lesson.day11.fifteenth.Exam15_1.main(Exam15_1.java:6)`. The exception message is highlighted in red and blue.

연습문제 15-2

연습 15-1 에서 작성한 코드를 수정하여, try-catch() 문을 사용하여 예외처리를 하시오. 예외처리에는 다음의 처리를 수행하시오.

1. "NullPointerException 예외를 catch 하였습니다" 를 표시한다
2. "---- stack trace (여기부터) ----" 를 표시한다
3. stack trace 를 표시한다
4. "---- stack trace (여기까지) ----" 를 표시한다

```
package Lesson.day11.fifteenth;

public class Exam15_2 {
    public static void main(String[] args) {
        String s = null;
        try {
            System.out.println(s.length());
        } catch (NullPointerException e) {
            System.out.println(e + " 예외를 catch 하였습니다");
            System.out.println("---- stack trace (여기부터) ----");
            e.printStackTrace();
            System.out.println("---- stack trace (여기까지) ----");
        }
    }
}
```

```
}
}
```

```
Problems @ Javadoc Declaration Search Console ×
<terminated> Exam15_2 [Java Application] C:\Users\admin\Desktop\workspace\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\
java.lang.NullPointerException: Cannot invoke "String.length()" because "s" is null 예외를 catch 하였습니다
---- stack trace (여기부터) ----
java.lang.NullPointerException: Cannot invoke "String.length()" because "s" is null
    at Lesson.day11.fifteenth.Exam15_2.main(Exam15_2.java:6)
---- stack trace (여기까지) ----
```

연습문제 15-3

`Integer.parseInt()` 메소드를 사용하여, 문자열 "Three"의 변환결과를 `int`형 변수 `i`에 대입하는 코드를 작성하시오.

`parseInt()` 메소드가 어떤 예외를 발생시킬 가능성이 있는지 API 레퍼런스를 조사하고, 올바른 예외처리를 기술하시오.

```
package Lesson.day11.fifteenth;

public class Exam15_3 {
    public static void main(String[] args) {
        int i = Integer.parseInt("Three");
    }
}
```

```
package Lesson.day11.fifteenth;

public class Exam15_3 {
    public static void main(String[] args) {
        try {
            int i = Integer.parseInt("Three");
        } catch (NumberFormatException e) {
            e.printStackTrace();
        }
    }
}
```

```
Problems @ Javadoc Declaration Search Console ×
<terminated> Exam15_3 [Java Application] C:\Users\admin\Desktop\workspace\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\
java.lang.NumberFormatException: For input string: "Three"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
    at java.base/java.lang.Integer.parseInt(Integer.java:668)
    at java.base/java.lang.Integer.parseInt(Integer.java:786)
    at Lesson.day11.fifteenth.Exam15_3.main(Exam15_3.java:6)
```

```

/**
 * Parses the string argument as a signed decimal integer. The
 * characters in the string must all be decimal digits, except
 * that the first character may be an ASCII minus sign {@code '-'}
 * ({@code '\u005Cu002D'}) to indicate a negative value or an
 * ASCII plus sign {@code '+'} ({@code '\u005Cu002B'}) to
 * indicate a positive value. The resulting integer value is
 * returned, exactly as if the argument and the radix 10 were
 * given as arguments to the {@link #parseInt(java.lang.String,
 * int)} method.
 *
 * @param s      a {@code String} containing the {@code int}
 *                representation to be parsed
 * @return       the integer value represented by the argument in decimal
 * @throws       NumberFormatException if the string does not contain
 *                parsable integer.
 */
public static int parseInt(String s) throws NumberFormatException {
    return parseInt(s,10);
}

```

연습문제 15-4

기동 직후에 "프로그램 시작" 을 표시하고

IOException 을 임의로 발생시켜 이상 종료 되도록 프로그램을 작성하시오.

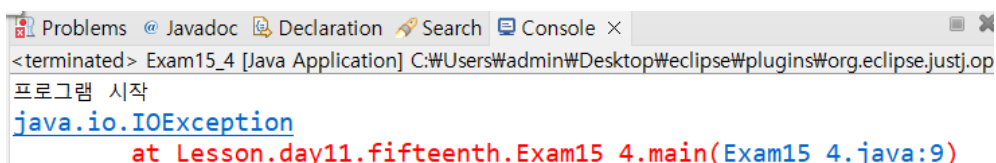
```

package Lesson.day11.fifteenth;

import java.io.IOException;

public class Exam15_4 {
    public static void main(String[] args) {
        try {
            System.out.println("프로그램 시작");
            throw new IOException();
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}

```



```

Problems @ Javadoc Declaration Search Console ×
<terminated> Exam15_4 [Java Application] C:\Users\admin\Desktop\weclipse\plugins\org.eclipse.justj.op
프로그램 시작
java.io.IOException
    at Lesson.day11.fifteenth.Exam15_4.main(Exam15_4.java:9)

```

파일 조작

```

package Lesson.day11.file;

import java.io.*;

```

```

public class Main {
    public static void main(String[] args) throws IOException {
        FileWriter fw = new FileWriter("C:\\Users\\admin\\Desktop\\하금티 광명용기원 교육\\JAVA_KOPOXHANA\\수업자료\\JAVA\\DATA.txt", true);
        fw.write("Hello world\\n");
        fw.flush();
        fw.close();
    }
}

```

flush가 매우 중요하다. ⇒ 애가 있어야 쓰기가 된다.

바이너리 파일

파일에 survivalcoding.com 2진수 01000001 을 추가로 저장하는 프로그램

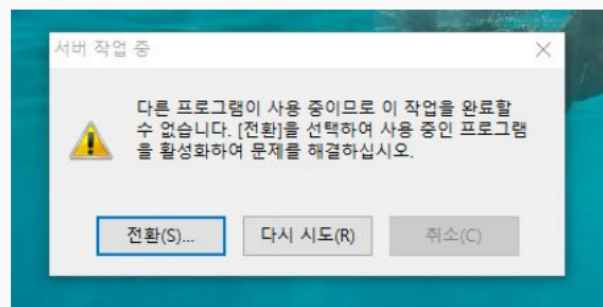
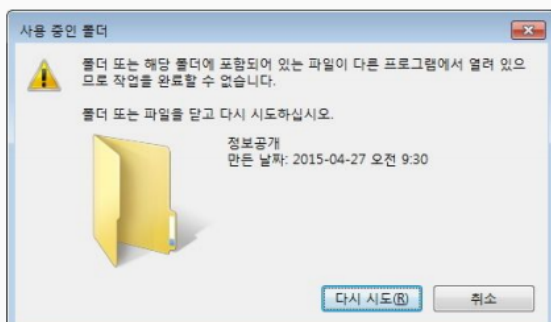
파일에는 크게 두 종류의 파일이 있다

1. 텍스트 파일
 - a. 문자로 구성
 - b. 메모장, 소스코드, HTML 등이 해당
 - c. FileReader, FileWriter 사용
2. 바이너리 파일
 - a. 문자와 데이터(바이트 배열)
 - b. Excel, Java의 class 파일, 이미지 파일, 동영상 파일, 음악 파일 등
 - c. FileInputStream, FileOutputStream 사용

close() 를 안 했을 경우

survivalc

다른 프로그램에서 해당 파일을 쓸 수 없음



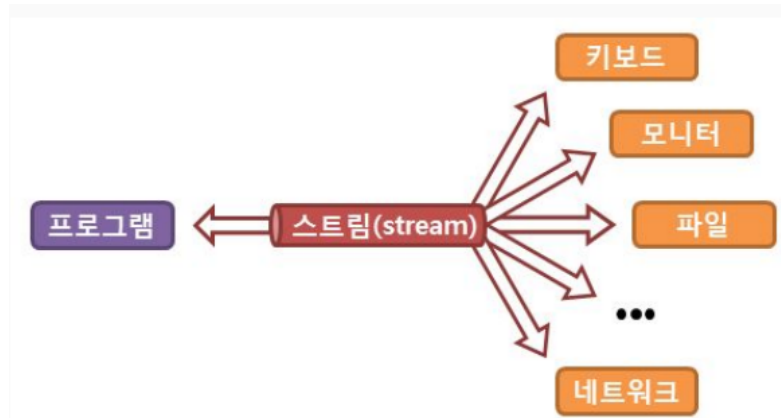
스트림

FileReader 는 데이터를 조금씩 읽거나 쓰는 API 임

한번에 10GB의 파일을 읽어버리게 되면 메모리 부족이 발생 함

스트림을 통해 데이터를 조금씩 흘려보내고 처리하면 이를 해결할 수 있음

FileReader, FileWriter, FileInputStream, FileOutputStream 는 모두 스트림계열



표준 입출력

표준 출력 : `System.out` => 모니터

표준 입력 : `System.in` => 키보드

표준 출력을 변경할 수 있음

```
> java Main > data.txt
```

출력을 `data.txt`로 변경하였고 `data.txt` 입력으로 `Main` 프로그램의 실행 결과를 받음

필터

```
// 파일 출력 스트림 생성
FileOutputStream fos = new FileOutputStream("data.txt");
// 암호화 스트림 연결
Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
CipherOutputStream cos = new CipherOutputStream(fos, c);
// 암호화하여 파일에 쓰기
cos.write(65);
```

암호화 해서 쓰기

버퍼링을 위한 필터

문자용 필터 : `BufferedReader`, `BufferedWriter`

바이트용 필터 : `BufferedInputStream`, `BufferedOutputStream`

1. 처리성능 향상

- a. 파일에 쓰거나 읽을때 마다 매번 하드디스크를 조작하는 것은 매우 느림
- b. 한 번에 모든 작업을 수행하여 쓰는 것이 효율적
- c. 버퍼링은 미리 읽어 두거나 한 번에 쓰는 작업
- d. 따라서 파일 조작은 버퍼링을 수행하는 것이 효율적

```
FileReader fr = new FileReader("save.dat");
BufferedReader br = new BufferedReader(fr);
String line = null;
while ((line = br.readLine())) != null) {
    // 한 줄씩 처리
}
```

파일 시스템의 조작

<파일 조작 메서드>

1. 삭제 : delete()
2. 이름 변경 : renameTo(File dest)
3. 존재 확인 : exists()
4. 파일인지? : isFile()
5. 디렉토리인지? : isDirectory()
6. 용량 : length()
7. 폴더
 내 파일들 : listFiles()
8. 복사기능은 제공안함

java.nio.file.Files

<제공되는 대표적인 기능>

복사, 이동, 삭제, 모든 내용 읽기, 모든 행을 읽어서 리스트로 반환 등

Path 객체를 얻는 방법

```
Path path1 = Paths.get("save.dat");
Path path2 = file.toPath();
```

연습문제

연습문제 6-1

`FileInputStream`, `FileOutputStream` 클래스를 사용하여 파일을 복사하는 프로그램을 작성하시오.

원본 파일 경로와 복사할 파일 경로는 프로그램 실행시 파라미터로 전달되는 것으로 하고, 버퍼링이나 에러 처리는 하지 않는다.

```
package Lesson.day11.file;

import java.io.*;

public class Exam_file_Main {
    public static void main(String[] args) throws Exception {
        String fileName = "Data.txt";
        String fileDir =
            "C:\\\\Users\\\\admin\\\\Desktop\\\\하금티 광명용기원 교육\\\\JAVA_KOPOXHANA\\\\수업자료\\\\JAVA\\\\";
        String fileLink = fileDir + fileName;
        String newFileLink = fileDir + "CopyOf" + fileName;
        Exam_file_1.copyFile(fileLink, newFileLink);
    }
}
```

```
package Lesson.day11.file;

import java.io.*;

public class Exam_file_1 {
    public static void copyFile(String fileLink, String copyFileLink) throws IOException {
        try (FileInputStream fis = new FileInputStream(fileLink);
            FileOutputStream fos = new FileOutputStream(copyFileLink, true);) {
            int ch = fis.read();
            while (ch != -1) {
                fos.write((char) ch);
                ch = fis.read();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

연습문제 6-2

연습문제 6-1에서 작성한 프로그램을 다음 조건을 수행하도록 수정하시오.

- `java.util.zip.GZIPOutputStream` 을 사용하여 압축한다.
- 버퍼링을 수행하시오.
- 예외처리를 하시오.

```

package Lesson.day11.file;

import java.io.*;

public class Exam_file_Main {
    public static void main(String[] args) throws Exception {
        String fileName = "Data.txt";
        String fileDir =
            "C:\\\\Users\\\\admin\\\\Desktop\\\\하금티 광명용기원 교육\\\\\\\\JAVA_KOPOXHANA\\\\\\\\수업자료\\\\\\\\JAVA\\\\\\\\";
        String fileLink = fileDir + fileName;
        String zipFileLink = fileDir + "ZipOfData.zip";
        Exam_file_2.zipFile(fileLink, zipFileLink);
        Exam_file_2.checkZipFile(zipFileLink);
    }
}

```

```

package Lesson.day11.file;

import java.io.*;
import java.util.zip.*;

public class Exam_file_2 {
    public static void zipFile(String fileLink, String zipFileLink) throws IOException {
        try {
            FileInputStream fis = new FileInputStream(fileLink);
            BufferedInputStream bis = new BufferedInputStream(fis);
            FileOutputStream fos = new FileOutputStream(zipFileLink);
            GZIPOutputStream gzipOut = new GZIPOutputStream(fos);
            BufferedOutputStream bos = new BufferedOutputStream(gzipOut) {
                byte[] buffer = new byte[1024];
                int bytesRead;
                while ((bytesRead = bis.read(buffer)) != -1) {
                    bos.write(buffer, 0, bytesRead);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public static void checkZipFile(String zipFileLink) throws IOException {
        FileInputStream fis = new FileInputStream(zipFileLink);
        BufferedInputStream bis = new BufferedInputStream(fis);
        GZIPInputStream gzipIn = new GZIPInputStream(bis);
        int size = 0;
        byte[] buffer = new byte[1024];
        while ((size = gzipIn.read(buffer)) > 0) {
            for (int i = 0; i < size; i++) {
                System.out.print((char) buffer[i]);
            }
        }
    }
}

```

여러가지 파일 형식

CSV

데이터를 콤마로 나눈 형식

StringTokenizer 클래스를 사용하면 토큰별로 데이터를 읽을 수 있어서 편리

프로퍼티 형식

Properties 클래스를 사용하여 키(key)와 값(value)의 쌍으로 읽고 쓰기가 가능

data.properties 파일의 내용

```
heroName=홍길동  
heroHp=100
```

```
Reader fr = new FileReader("data.properties");  
Properties prop = new Properties();  
prop.load(fr);    // 파일을 읽어들이  
String name = prop.getProperty("heroName");  
String hp = prop.getProperty("heroHp");  
System.out.println("용사의 이름 : " + name);  
System.out.println("용사의 HP : " + hp);  
fr.close();
```

setProperty() 메서드로 데이터를 셋팅하고 store() 메서드로 저장

```
Writer fw = new FileWriter("data.properties");  
Properties prop = new Properties();  
prop.setProperty("heroName", "한석봉");  
prop.setProperty("heroHp", "50");  
prop.setProperty("heroMp", "30");  
prop.store(fw, "용사의 정보")    // 파일 상단에 코멘트  
fw.close();
```

XML 형식

<> 태그를 활용한 기술 방식

포함관계를 기술할 수 있음

DOM Parser, SAX Parser 등을 통해 파서를 제작할 필요가 있음

자유도가 높다 요즘은 JSON에 밀려 잘 안쓰인다.

```
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>
```

JSON 형식

네트워크 통신에서 가장 많이 사용되고 있음

XML에 비해 적은 용량

[]로 리스트, {}로 객체를 표현

키(key): 값(value) 형태

JsonObject, JsonArray 클래스를 활용

Gson 등의 라이브러리 활용을 권장

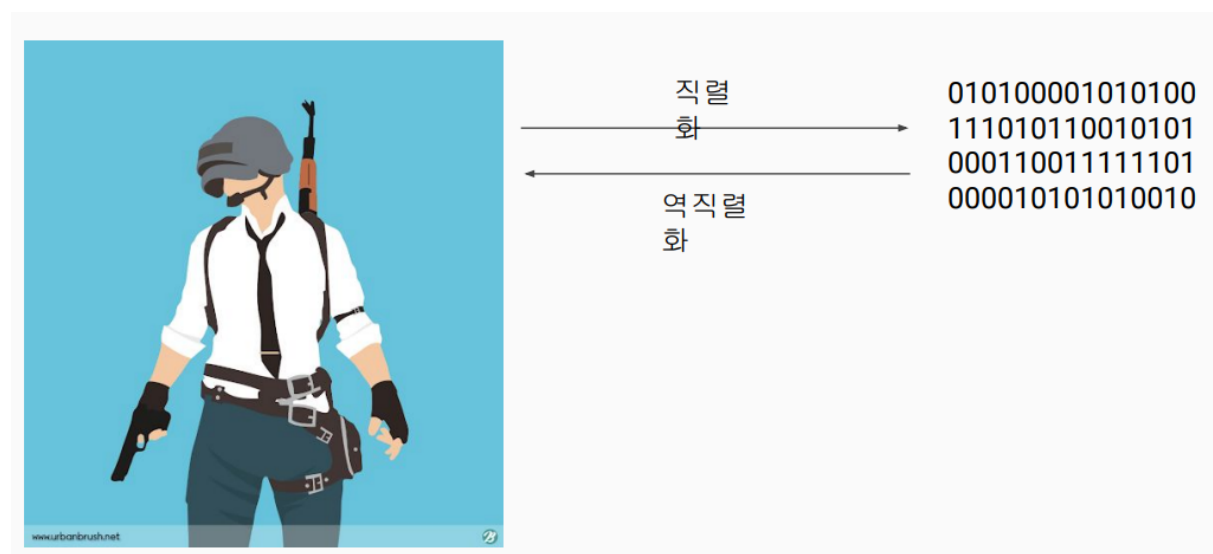
```
{
  "이름": "홍길동",
  "나이": 25,
  "성별": "여",
  "주소": "서울특별시 양천구 목동",
  "특기": ["농구", "도술"],
  "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},
  "회사": "경기 수원시 팔달구 우만동"
}
```

직렬화

게임의 세이브 파일을 저장하기 위해 적절한 파일 형식은??

정말 그 파일 형식이 편리한가??

위변조를 예방해야함



직렬화를 통해 인스턴스를 바이트 배열로 상호 변환할 수 있음

직렬화 대상에는 Serializable 인터페이스를 구현해야 함

클래스 내부의 필드도 직렬화 대상이라면 모두 직렬화 처리를 해 줘야 함

클래스 설계의 변경에 대비하려면 직렬화 버전 UID를 선언한다

```
class Name implements Serializable {}
```

내부의 클래스는 모두 직렬화 해줘야함

```
Hero hero = new Hero("홍길동", 75, 18);
```

```
// 저장
```

```
FileOutputStream fos = new FileOutputStream("save.dat");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(hero);
oos.flush();
oos.close();
```

용사를 복원

```
// 로드
```

```
FileInputStream fis = new FileInputStream("save.dat");
ObjectInputStream ois = new ObjectInputStream(fis);
Hero hero = (Hero) ois.readObject();
ois.close();
```

시리얼 버전 UID

게임이 버전업 되면서 용사 클래스의 구조가 변경되었다면 이전 버전의 세이브파일과 호환되지 않을 것이다

시리얼 버전 UID를 선언 해 두면 이런 경우 JVM에서 예외를 발생시켜 준다.

long 값으로 아무값이나 정하면 된다.

```
class Hero implements Serializable {
    private static final long serialVersionUID = 81923983183821L;
```

연습문제

연습문제 7-1

다음과 같은 내용이 담긴 파일 gradle.properties 파일이 있습니다.

```
org.gradle.jvmargs=-Xmx1536M
android.enableR8=true
android.useAndroidX=true
android.enableJetifier=true
```

이 파일을 읽어서 android.useAndroidX 값을 출력하시오.

```

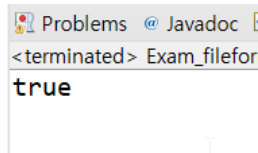
package Lesson.day11.fileformat;

import java.io.*;
import java.util.*;

public class Exam_fileformat_1 {
    public static void main(String[] args) throws IOException {
        String fileDir =
            "C:\\Users\\admin\\eclipse-workspace\\JavaExam\\src\\Lesson\\day11\\fileformat\\";
        String fileName = "gradle.properties";
        Reader fr = new FileReader(fileDir + fileName);
        Properties gradle = new Properties();
        gradle.load(fr);
        String msg = gradle.getProperty("android.useAndroidX");
        System.out.println(msg);
    }
}

```

properties의 기본 경로는 같은 폴더가 아니므로 경로를 줘야함.



연습문제 7-2

survivalcoding

다음과 같은 사원 클래스와 부서 클래스가 있습니다.

```

class Employee {
    String name;
    int age;
}

class Department {
    String name;
    Employee leader;
}

```

총무부 리더 '홍길동(41세)'의 인스턴스를 생성하고 직렬화하여 **company.dat** 파일에 쓰는 프로그램을 작성하시오.
직렬화를 위해 위의 2개 클래스를 일부 수정해도 됩니다.

```

package Lesson.day11.fileformat;

import java.io.*;
import java.util.*;

public class Exam_fileformat_2 {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        // 인스턴스 선언
        Employee hong = new Employee("홍길동", 41);
        Department chongmu = new Department("chongmubu", hong);
        // 직렬화
        FileOutputStream fos = new FileOutputStream("company.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(hong);
        oos.writeObject(chongmu);
        oos.flush();
        oos.close();
        // 확인부
        FileInputStream fis = new FileInputStream("company.dat");
        ObjectInputStream ois = new ObjectInputStream(fis);
        Employee checkEmployee = (Employee) ois.readObject();
    }
}

```

```
        Department checkDepartment = (Department) ois.readObject();  
        ois.close();  
        System.out.println(checkEmployee.toString());  
        System.out.println(checkDepartment.toString());  
    }  
}
```

Problems @ Javadoc Declaration
<terminated> Exam_fileformat_2 [Java Applet]
홍길동 : 41
chongmubu : 홍길동 : 41