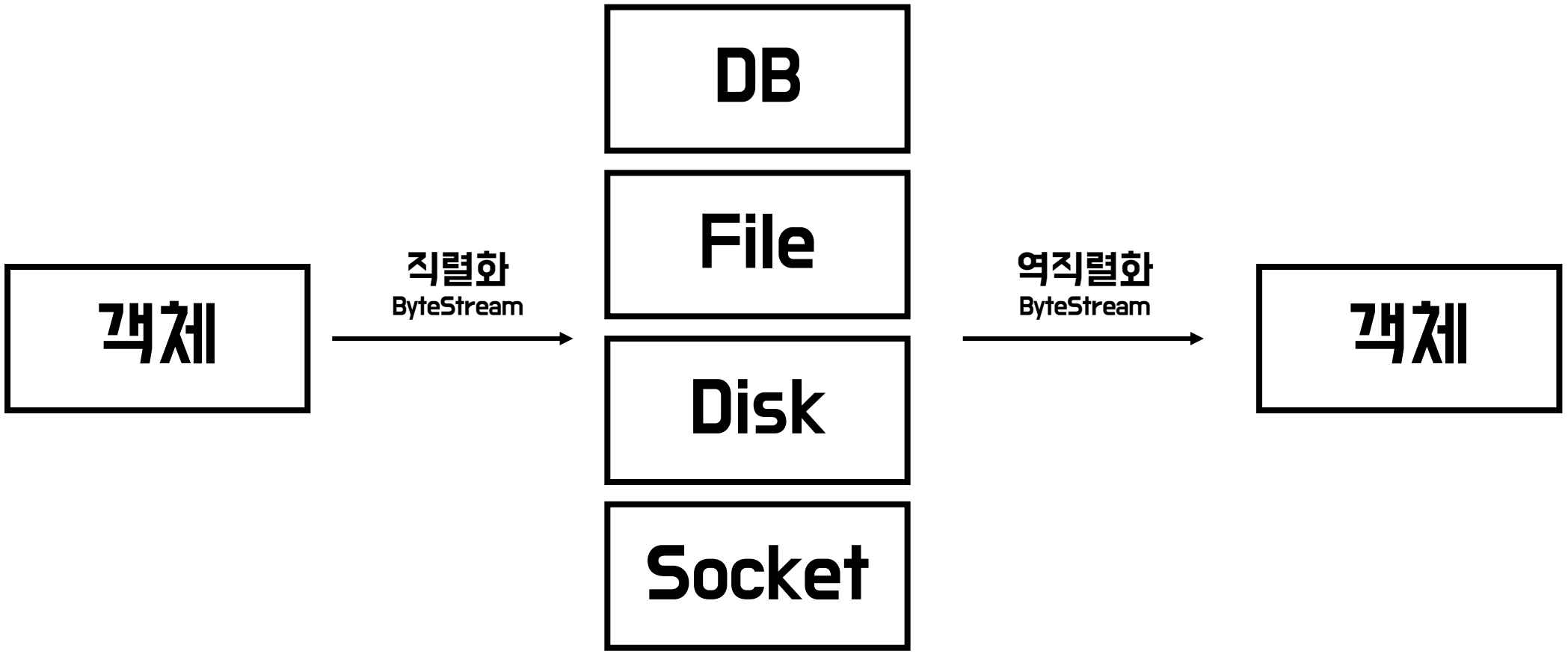
아이템 85.
자바 직렬화의 대안을 찾으라

직렬화?



객체를 바이트 스트림으로 변경하여 다른 환경에서 사용 가능!

직렬화?

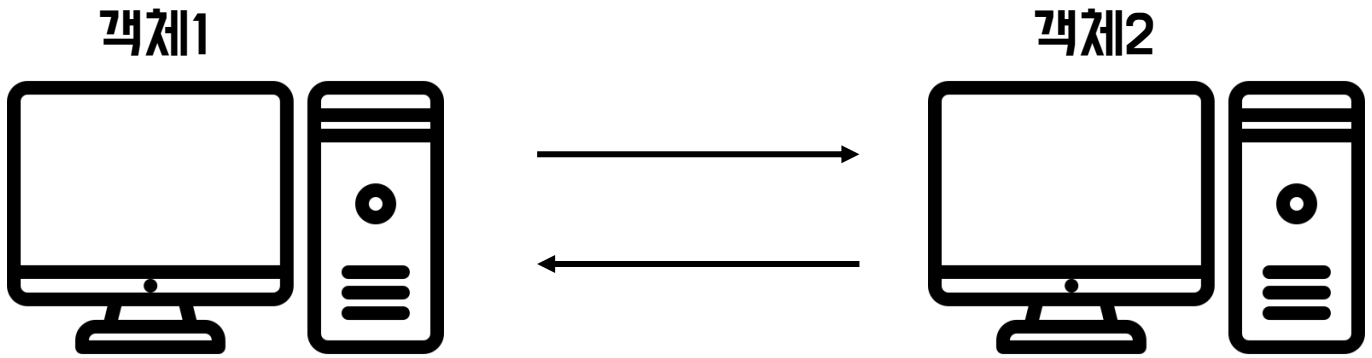
```java
public class HuniObject implements Serializable {

    private final int age;

    public HuniObject(int age) {
        this.age = age;
    }
}
```

Serializable 인터페이스를 상속받기만 하면
바로 직렬화 가능 ^^

* Serializable은 마커 인터페이스이기 때문에 구현할 것은 따로 없음

직렬화의 대안

때는 바야흐로 1997... 자바에 최초로 **직렬화**가 도입됐다.



분산 객체를 쉽게 만들 수 있다는 개념은 매력적이었으나..

직렬화의 대안

1. 보이지 않는 생성자
2. API와 구현 사이의 모호한 경계
3. 보안 문제
4. 유지 보수성

등등 여러가지 위험성이 많다..

직렬화의 대안

```
public final Object readObject()
    throws IOException, ClassNotFoundException {
    return readObject(Object.class);
}
```

ObjectOutputStream의 readObject()는 보이지 않는 생성자다!
그래서 문제가 심각하다..

직렬화의 대안

```java
public class HuniObject implements Serializable {

    private final int age;

    public HuniObject(final int age) {
        this.age = age;
        if (this.age < 26) {
            throw new IllegalArgumentException("양심을 지켜라");
        }
    }

    public int getAge() {
        return age;
    }
}
```

```java
@Test
void serializableTest() throws IOException {
    byte[] serializedHuniObject = getSerializableHuniObject( age: 26);

    try (ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(serializedHuniObject)) {
        ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
        HuniObject huniObject = (HuniObject) objectInputStream.readObject();

        assertThat(huniObject.getAge()).isEqualTo(26);
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

```java
private byte[] getSerializableHuniObject(int age) throws IOException {
    byte[] bytes;
    try (ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream()){
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);
        HuniObject huniObject = new HuniObject(age);
        objectOutputStream.writeObject(huniObject);
        bytes = byteArrayOutputStream.toByteArray();
    }
    return bytes;
}
```

직렬화 가능한 HuniObject를 직렬화하고 테스트에서 역직렬화해도 올바른 결과가 나온다!

직렬화의 대안

```java
@Test
void serializableAttackTest() throws IOException {
    byte[] serializedHuniObject = getSerializableHuniObject( age: 26);
    serializedHuniObject[62] = 25;

    try (ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(serializedHuniObject)) {
        ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
        HuniObject huniObject = (HuniObject) objectInputStream.readObject();

        assertThat(huniObject.getAge()).isEqualTo(26);
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

```
org.opentest4j.AssertionFailedError:
expected: 26
 but was: 25
Expected :26
Actual   :25
```

분명 26살 아래는 양심이 없는 거라 했는데...
공격의 대상이 되어 버린것!

직렬화의 대안

Gadget

역직렬화 과정에서 호출되어 잠재적으로 위험한 동작을 수행하는 메서드

공격자는 계속해서 gadget을 호출하여 체인을 만들어 네이티브 코드를 실행한다.

직렬화의 대안

역직렬화 폭탄

```java
static byte[] bomb()  throws IOException{
    Set<Object> root = new HashSet<>();
    Set<Object> s1 = root;
    Set<Object> s2 = new HashSet<>();

    for (int i = 0; i < 100; i++) {
        Set<Object> t1 = new HashSet<>();
        Set<Object> t2 = new HashSet<>();
        t1.add("foo");
        s1.add(t1);
        s2.add(t2);
        s2.add(t1);
        s1.add(t2);
        s1 = t1;
        s2 = t2;
    }
    byte[] bytes;
    try (ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream()){
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);
        objectOutputStream.writeObject(root);
        bytes = byteArrayOutputStream.toByteArray();
    }
    return bytes;
}
```

엄청나게 많은 역직렬화 과정이 생긴다.

역직렬화는 각 원소의 **해시코드**를 계산한다.

root의 원소는 또 다른 HashSet을 갖고
그 원소조차 또 HashSet을 가진다. 게다가 반복문의 깊이는 100

덕분에 **2의 100제곱** 이상 해시코드 메서드를 호출하게 된다.

직렬화의 대안

결론

아무것도 직렬화 하지 마라. 새로운 시스템에서 자바 직렬화 쓸 일 절대 없다.

이미 레거시에 존재한다면 역직렬화를 하지마라. 진짜 믿을만한 데이터만 역직렬화 하도록

역직렬화 필터링 등 방어 방법은 있지만 완벽하게 안정적이진 않다.

JSON이나 프로토콜 버퍼 써라.