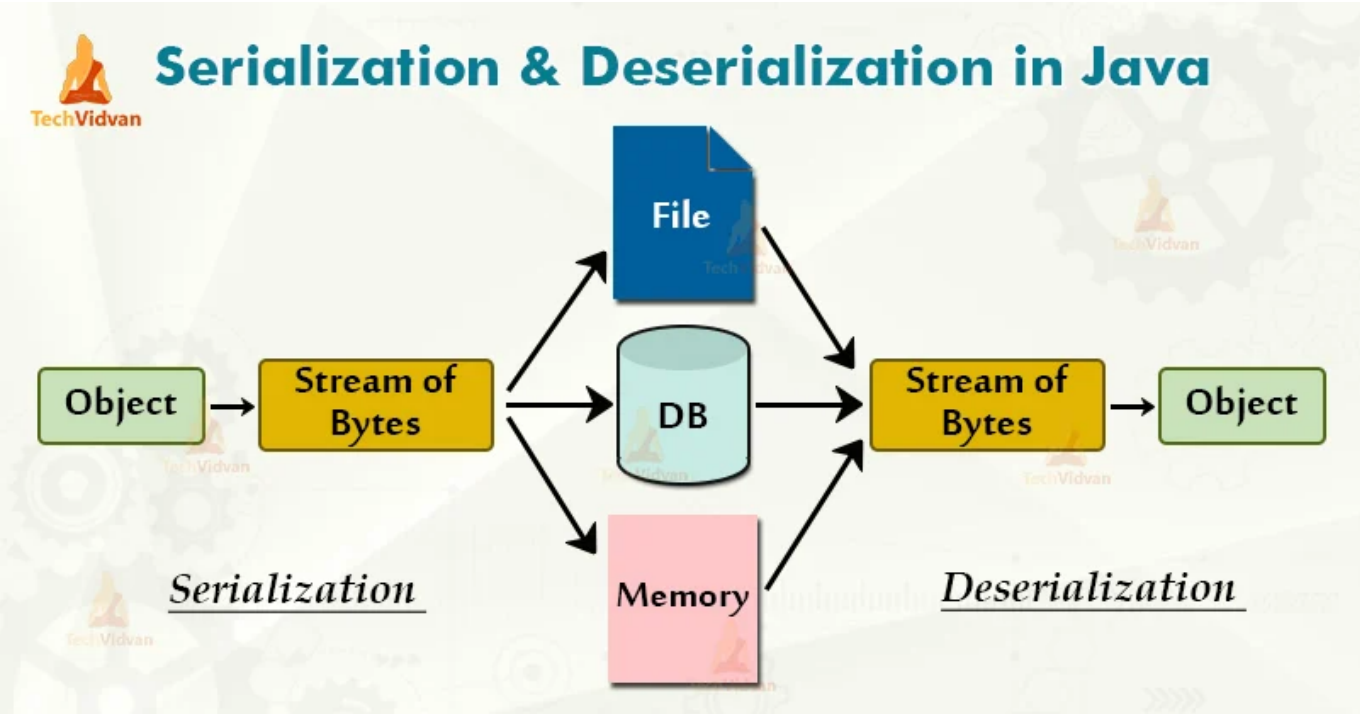
# [Java] 직렬화(Serialization)

> 자바 시스템 내부에서 사용되는 객체 또는 데이터를 외부의 자바 시스템에서도 사용할 수 있도록 바이트(byte) 형태로 데이터 변환하는 기술

각자 PC의 OS마다 서로 다른 가상 메모리 주소 공간을 갖기 때문에, Reference Type의 데이터들은 인스턴스를 전달 할 수 없다.

따라서, 이런 문제를 해결하기 위해선 주소값이 아닌 Byte 형태로 직렬화된 객체 데이터를 전달해야 한다.

직렬화된 데이터들은 모두 Primitive Type(기본형)이 되고, 이는 파일 저장이나 네트워크 전송 시 파싱이 가능한 유의미한 데이터가 된다. 따라서, 전송 및 저장이 가능한 데이터로 만들어주는 것이 바로 **'직렬화(Serialization)'**이라고 말할 수 있다.




## 직렬화 조건

자바에서는 간단히 `java.io.Serializable` 인터페이스 구현으로 직렬화/역직렬화가 가능하다.

> 역직렬화는 직렬화된 데이터를 받는쪽에서 다시 객체 데이터로 변환하기 위한 작업을 말한다.

**직렬화 대상** : 인터페이스 상속 받은 객체, Primitive 타입의 데이터

Primitive 타입이 아닌 Reference 타입처럼 주소값을 지닌 객체들은 바이트로 변환하기 위해 Serializable 인터페이스를 구현해야 한다.

## 직렬화 상황

- JVM에 상주하는 객체 데이터를 영속화할 때 사용
- Servlet Session
- Cache
- Java RMI(Remote Method Invocation)

## 직렬화 구현

```
@Entity
@AllArgsConstructor
@toString
public class Post implements Serializable {
private static final long serialVersionUID = 1L;

private String title;
private String content;
```

`serialVersionUID`를 만들어준다.

```
Post post = new Post("제목", "내용");
byte[] serializedPost;
try (ByteArrayOutputStream baos = new ByteArrayOutputStream()) {
    try (ObjectOutputStream oos = new ObjectOutputStream(baos)) {
        oos.writeObject(post);

        serializedPost = baos.toByteArray();
    }
}
```

`ObjectOutputStream`으로 직렬화를 진행한다. Byte로 변환된 값을 저장하면 된다.

## 역직렬화 예시

```
try (ByteArrayInputStream bais = new ByteArrayInputStream(serializedPost)) {
    try (ObjectInputStream ois = new ObjectInputStream(bais)) {
```

```
            Object objectPost = ois.readObject();
            Post post = (Post) objectPost;
        }
    }
```

`ObjectInputStream`로 역직렬화를 진행한다. Byte의 값을 다시 객체로 저장하는 과정이다.

### 직렬화 serialVersionUID

위의 코드에서 `serialVersionUID`를 직접 설정했었다. 사실 선언하지 않아도, 자동으로 해시값이 할당된다.

직접 설정한 이유는 기존의 클래스 멤버 변수가 변경되면 `serialVersionUID`가 달라지는데, 역직렬화 시 달라진 넘버로 Exception이 발생될 수 있다.

따라서 직접 `serialVersionUID`을 관리해야 클래스의 변수가 변경되어도 직렬화에 문제가 발생하지 않게 된다.

> `serialVersionUID`을 관리하더라도, 멤버 변수의 타입이 다르거나, 제거 혹은 변수명을 바꾸게 되면 Exception은 발생하지 않지만 데이터가 누락될 수 있다.

### 요약

- 데이터를 통신 상에서 전송 및 저장하기 위해 직렬화/역직렬화를 사용한다.
- `serialVersionUID`는 개발자가 직접 관리한다.
- 클래스 변경을 개발자가 예측할 수 없을 때는 직렬화 사용을 지양한다.
- 개발자가 직접 컨트롤 할 수 없는 클래스(라이브러리 등)는 직렬화 사용을 지양한다.
- 자주 변경되는 클래스는 직렬화 사용을 지양한다.
- 역직렬화에 실패하는 상황에 대한 예외처리는 필수로 구현한다.
- 직렬화 데이터는 타입, 클래스 메타정보를 포함하므로 사이즈가 크다. 트래픽에 따라 비용 증가 문제가 발생할 수 있기 때문에 JSON 포맷으로 변경하는 것이 좋다.

  > JSON 포맷이 직렬화 데이터 포맷보다 2~10배 더 효율적

**[참고자료]**

- 링크
- 링크
- 링크