DVM Network

객체지향 개발방법론

Dependable Software Laboratory

01 라이브러리 소개

02 프로젝트 적용방법

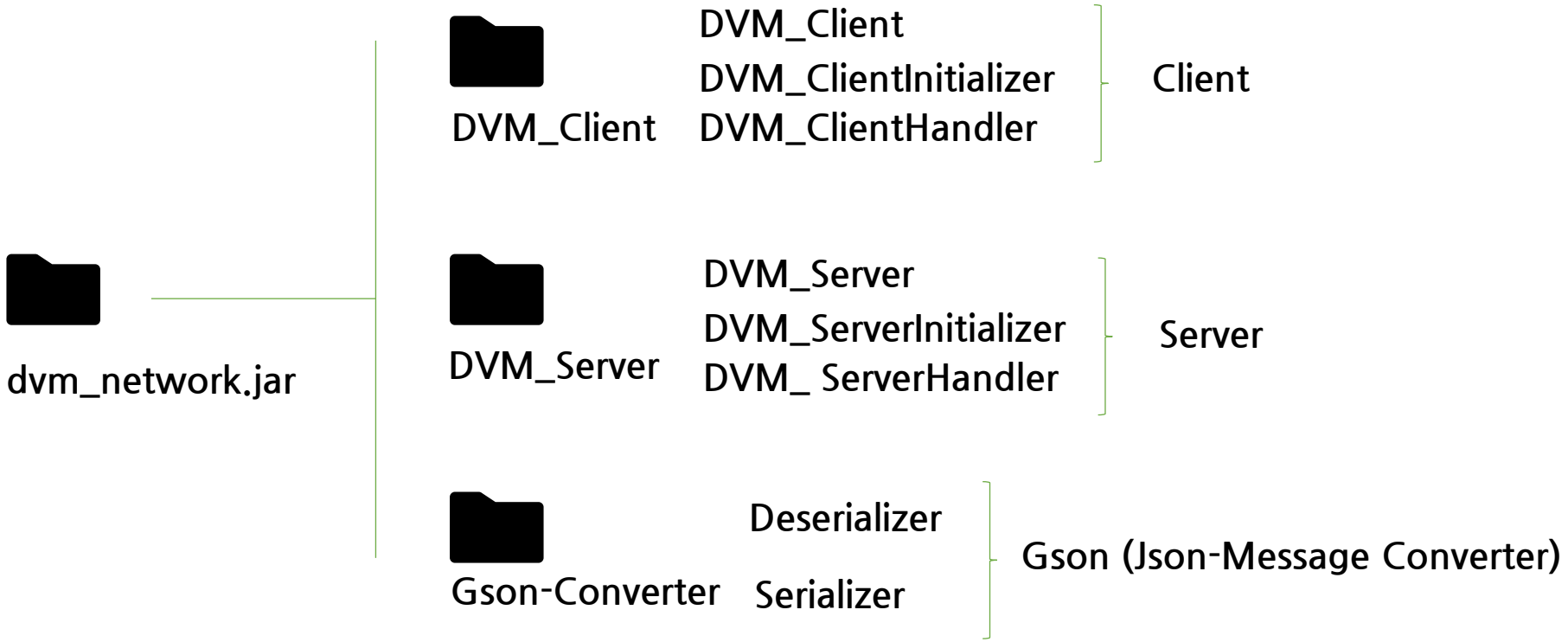
03 인터페이스 상세

테스팅 방법

04 차후 업데이트

01 라이브러리 소개

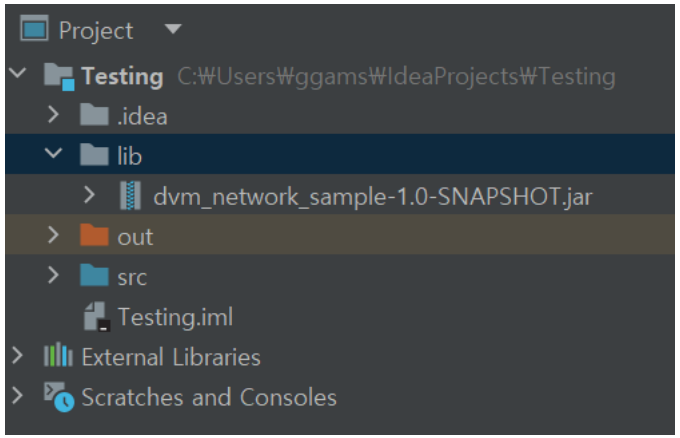
라이브러리 구조



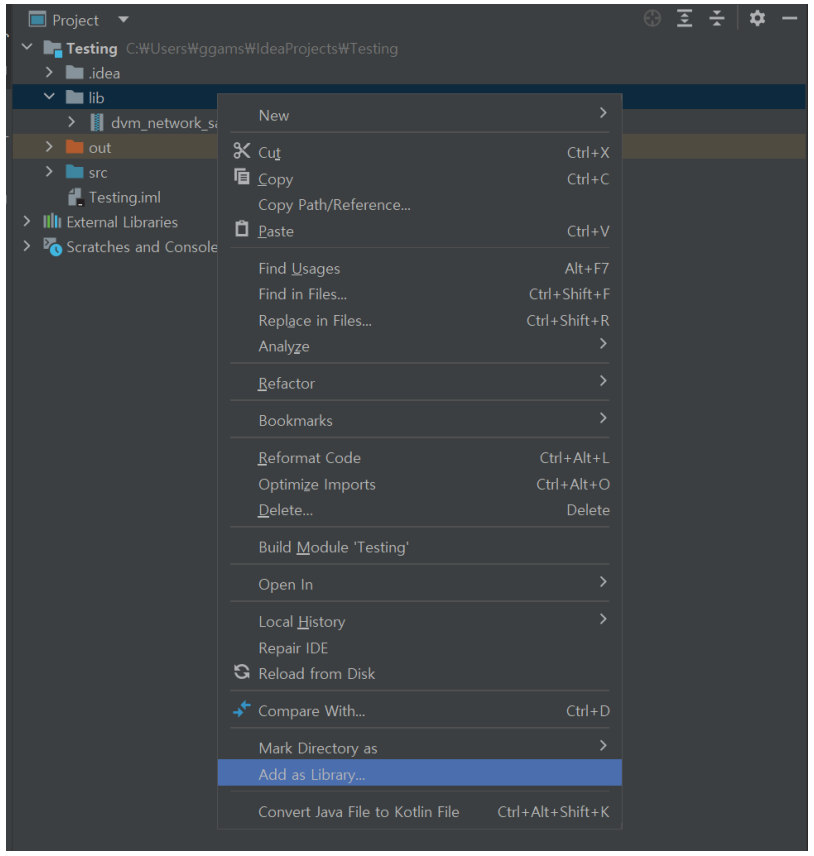
02 프로젝트 적용방법

InteliJ

1. InteliJ 프로젝트에서 lib라는 이름의 폴더 생성 후 다운로드 받은 jar 파일을 해당 폴더로 이동



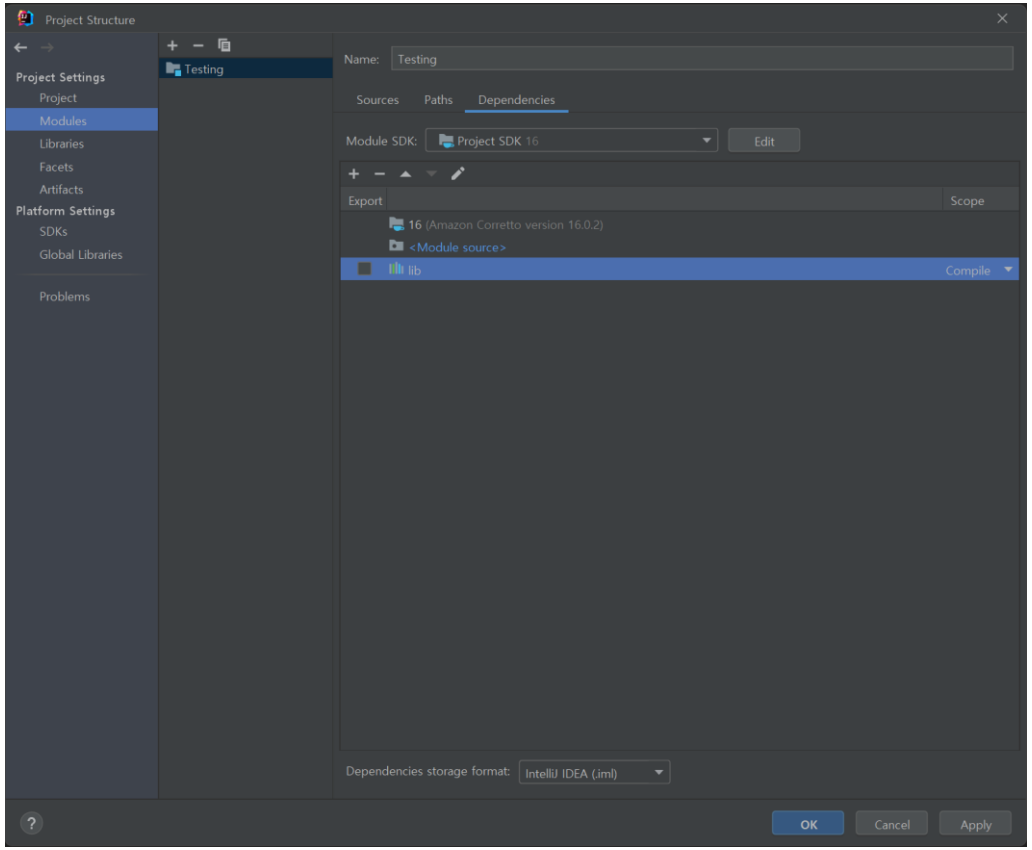
2. Lib 폴더 우클릭 -> Add as Library



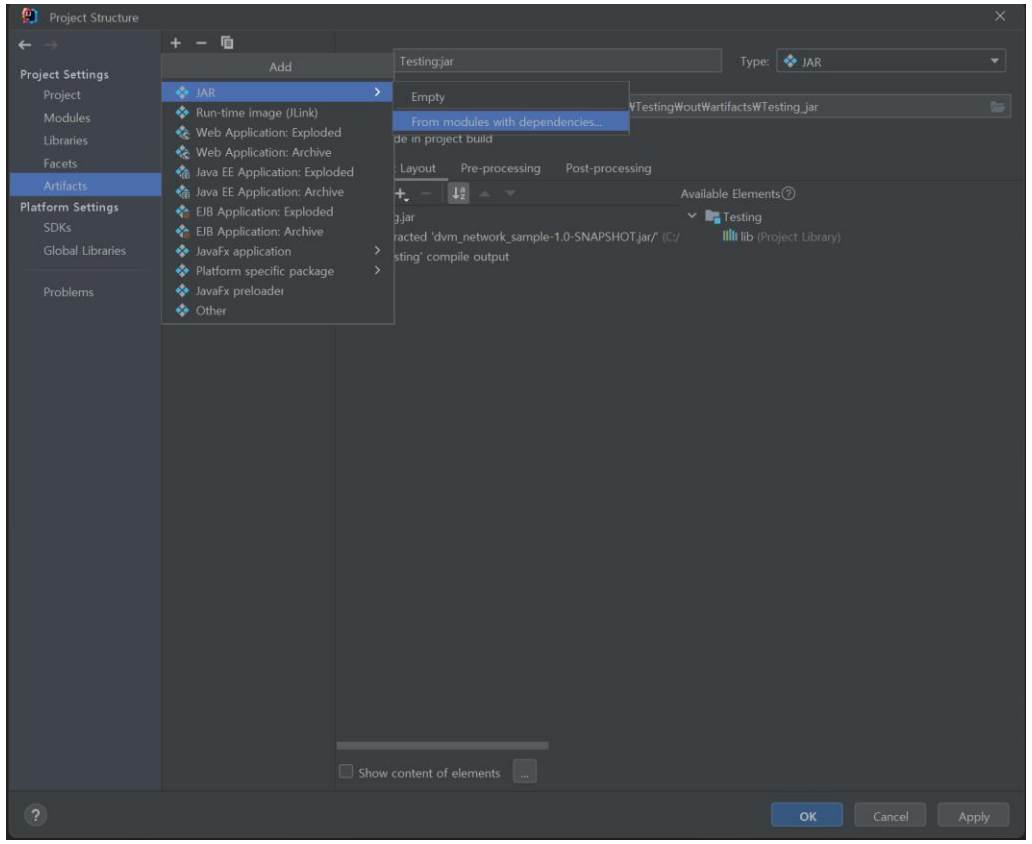
02 프로젝트 적용방법

InteliJ

3. File -> Project Structure -> Modules -> Dependencies -> lib 폴더 체크 ->Apply

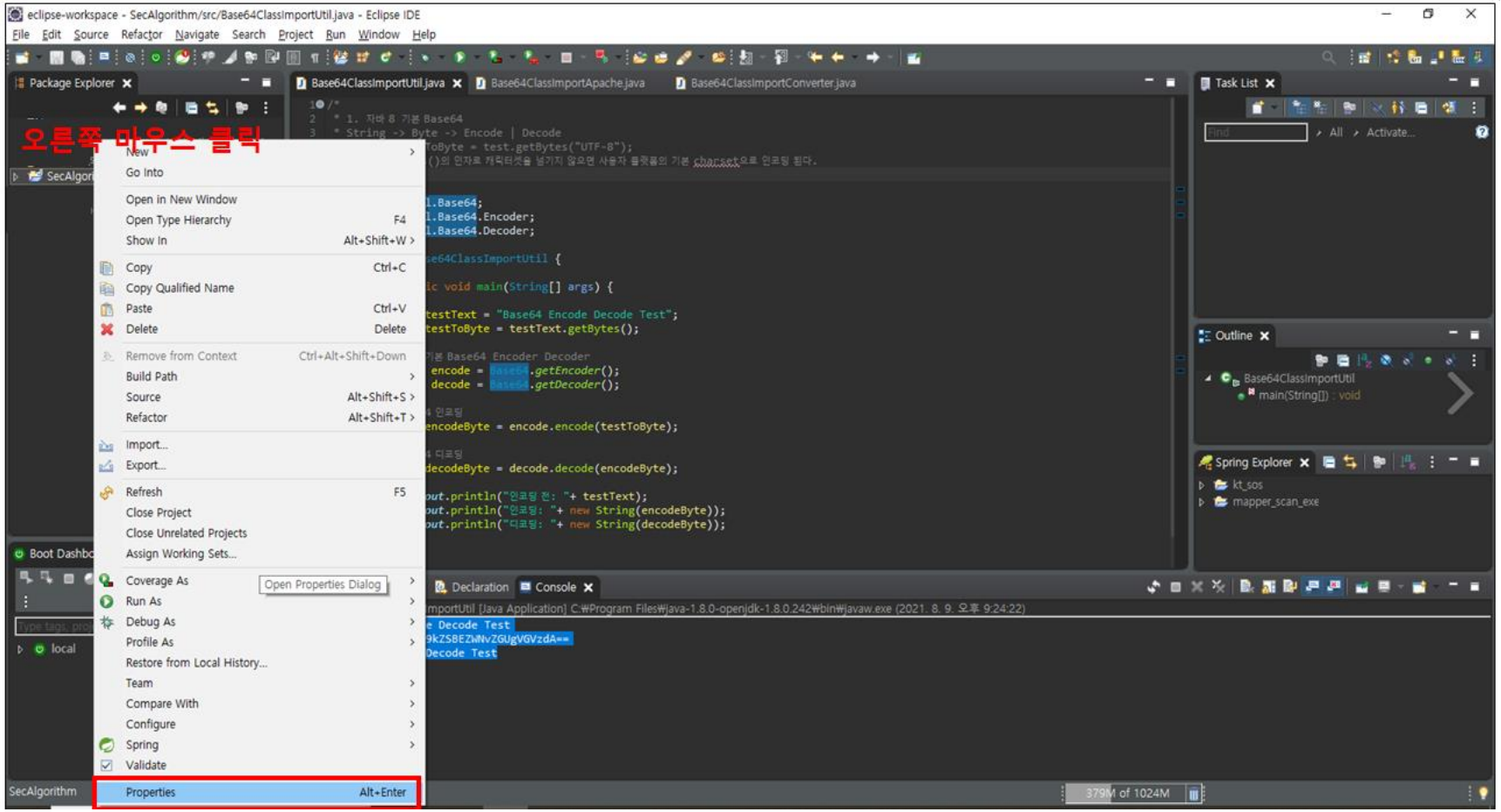


4. Artifacts -> + -> JAR -> From modules with dependencies.. -> Apply



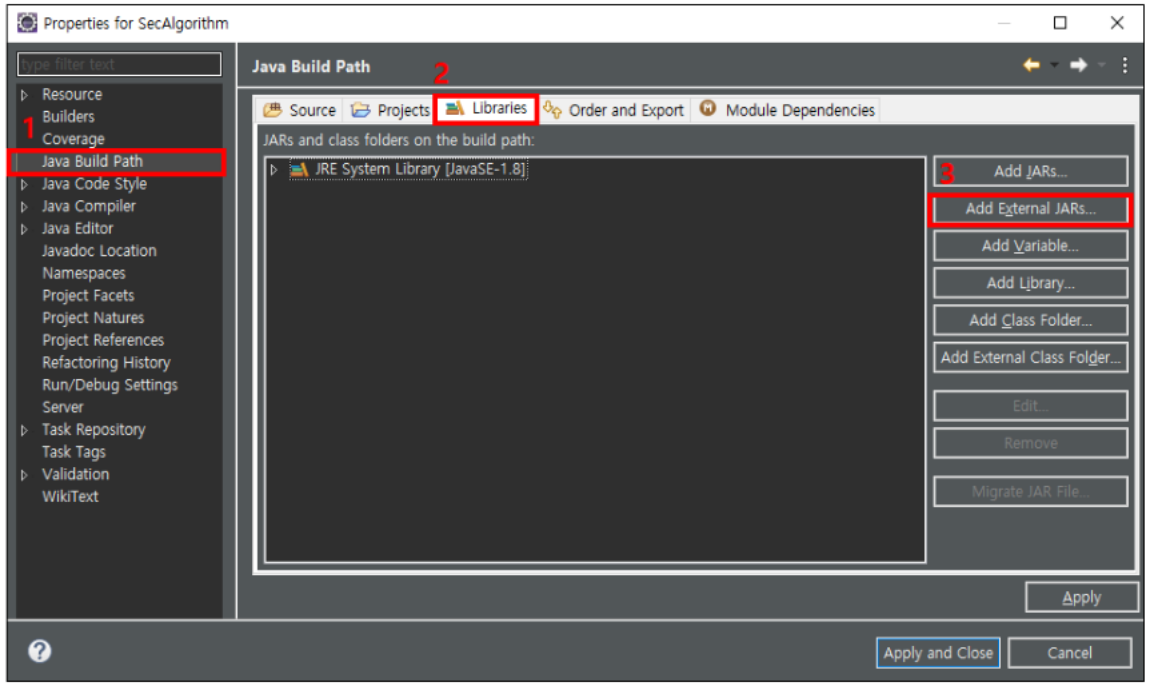
02 프로젝트 적용방법

Eclipse

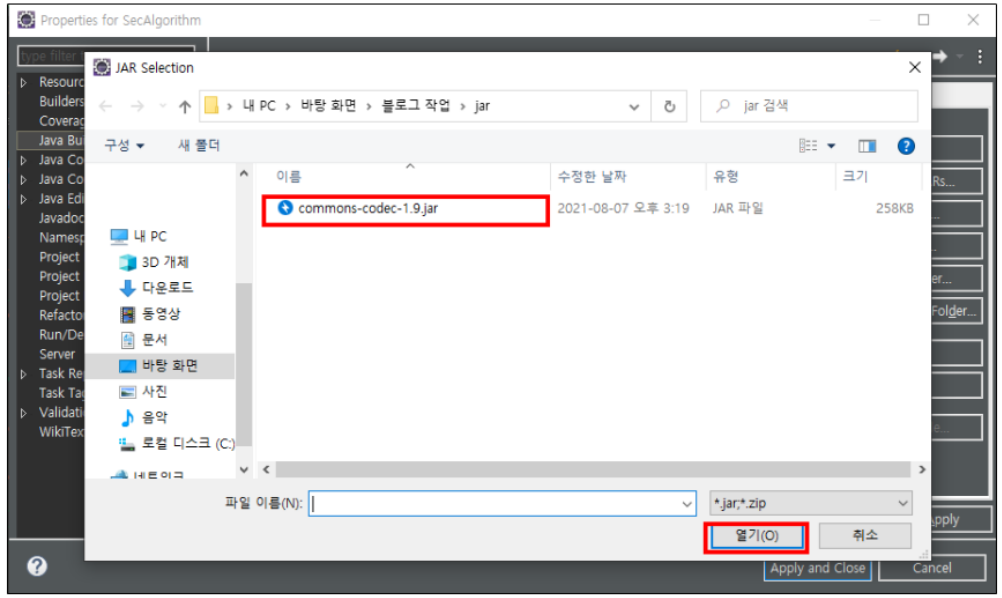


02 프로젝트 적용방법
Eclipse

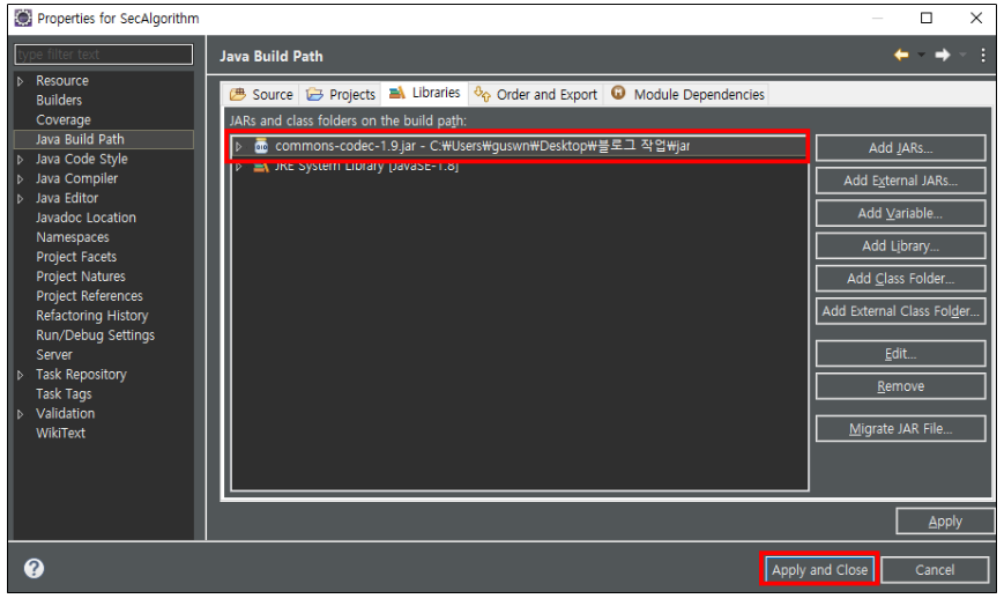
2. [Java Build Path] → [Libraries Tab] → [Add External JARs...]



3. 추가하고자 하는 jar파일을 선택 후 [열기]를 눌러 줍니다.



4. 추가가 잘 된 것을 확인 후 [Apply and Close]를 클릭해줍니다.



03 인터페이스 상세

Client

```
public DVMClient(String host, String msg);

//String host : 접속하고자 하는 서버의 ip 주소
//String msg  : 보내고자 하는 "json String"

public void run();

// 클라이언트 실행 메서드

/*-------------------------실행예제-------------------------------------*/

DVMClient client = new DVMClient("localhost", jsonMsg)
```

03 인터페이스 상세

Server

03 인터페이스 상세

Gson (Json-Message Converter)

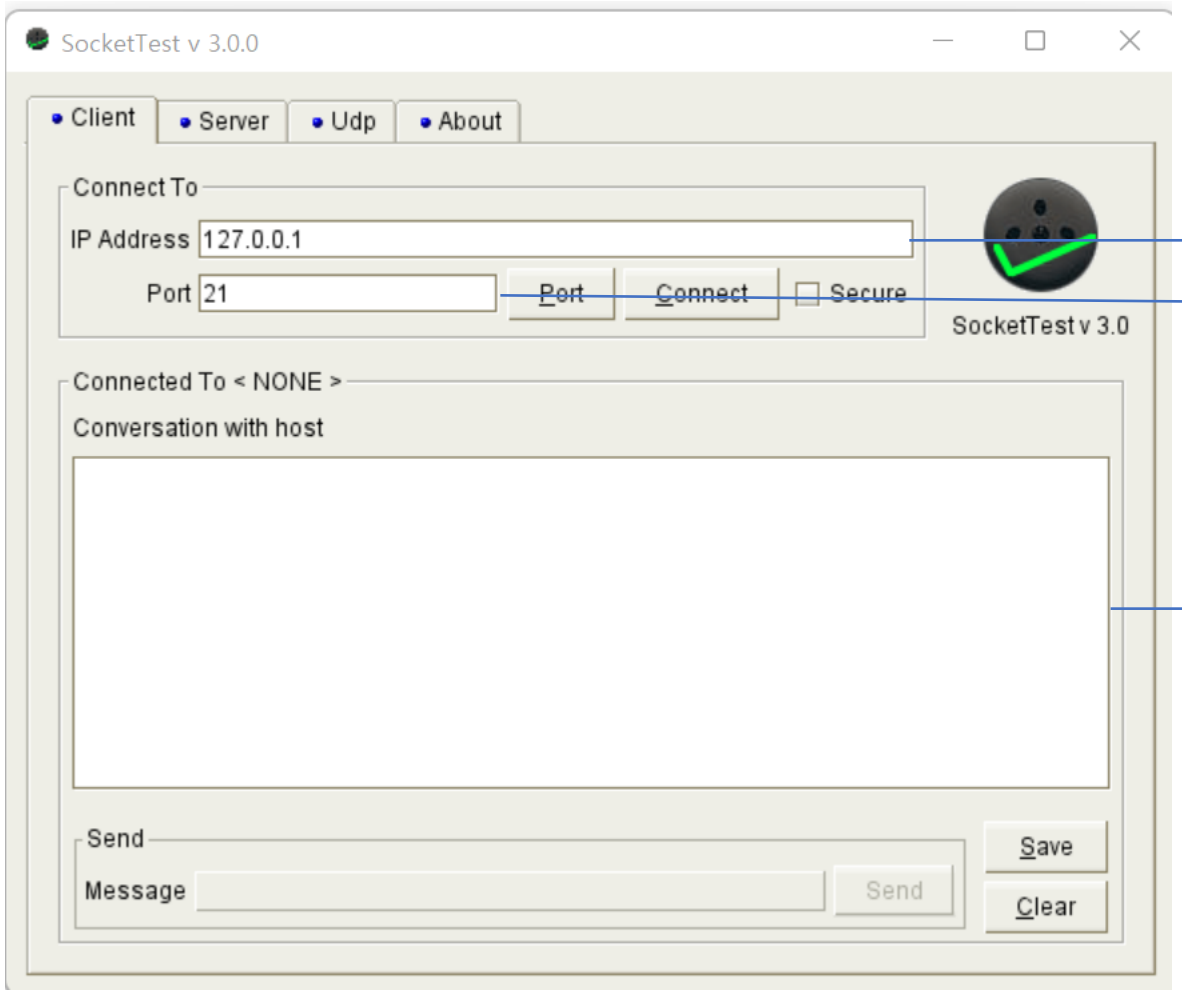
```
public class Serializer {
  // 입력받은 Message 객체를 jsonString으로 변환
  public String message2Json(Message msg);
}


public class Deserializer {
  // 입력받은 jsonString를 Message 객체로 변환
    public Message json2Message(String json)
}
```

03 인터페이스 상세

Message Object

```
//Message 객체
public Class Message{
    private int srcId;
    private int dstId;
    private String msgType;
    MsgDescription msgDescription;
    public static class MsgDescription{
        private int itemCode;
        private int itemNum;
        private int dvmXCoord, dvmYCoord;
        private int authCode;
    }
}
```

03 인터페이스 상세
SocketTest



접속하고자 하는 Server/Client 주소

접속하고자 하는 Server/Client PORT

대화내용

04 차후 업데이트

1. Server Class
2. Server Class는 금요일 오전에 업데이트 됩니다.
3. Exception Handling 추가
4. 연결이 차단되거나 실패할 시 복구방법
5. Logger

4,5번은 시간관계상 생략 할 수도 있음