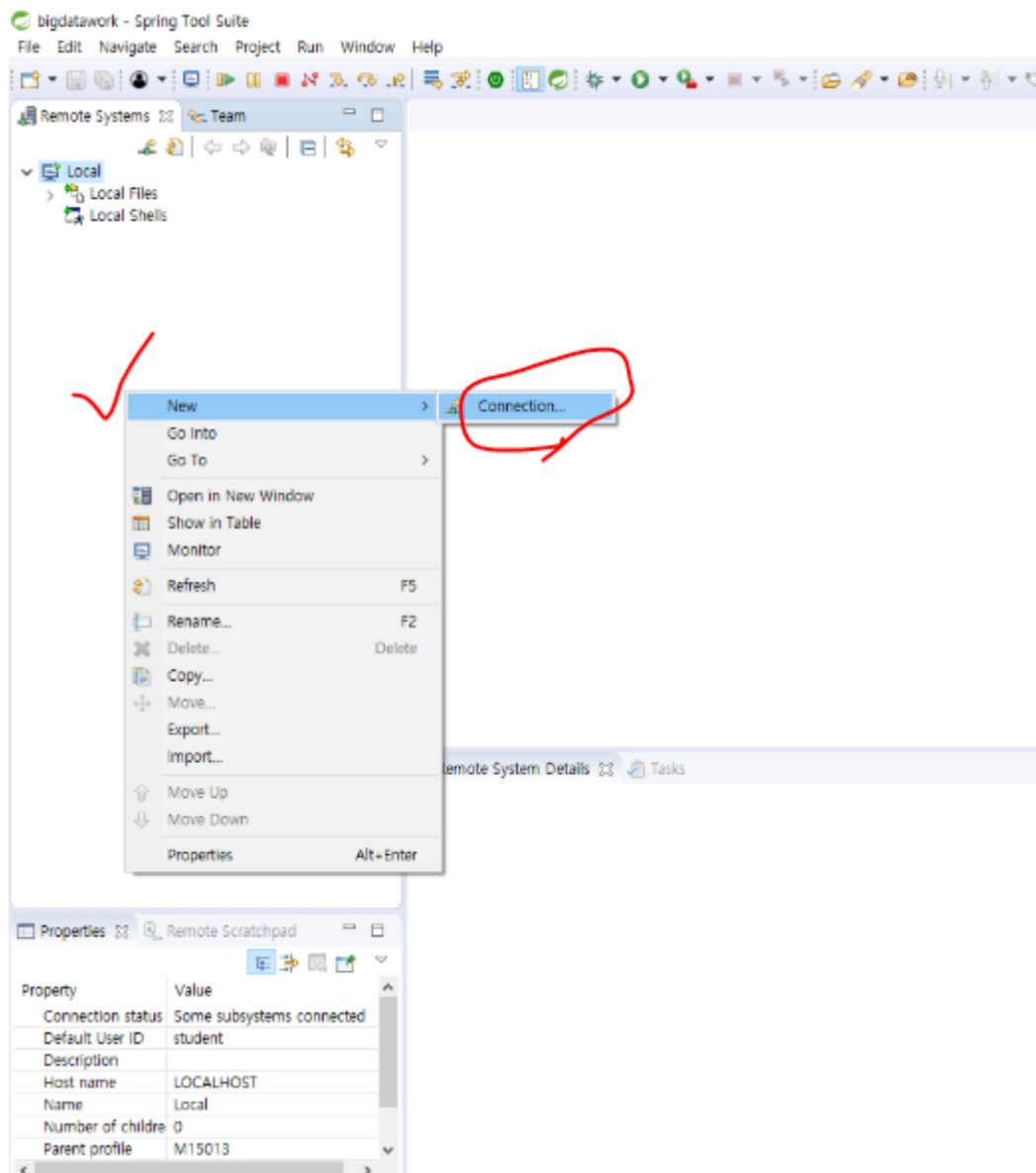
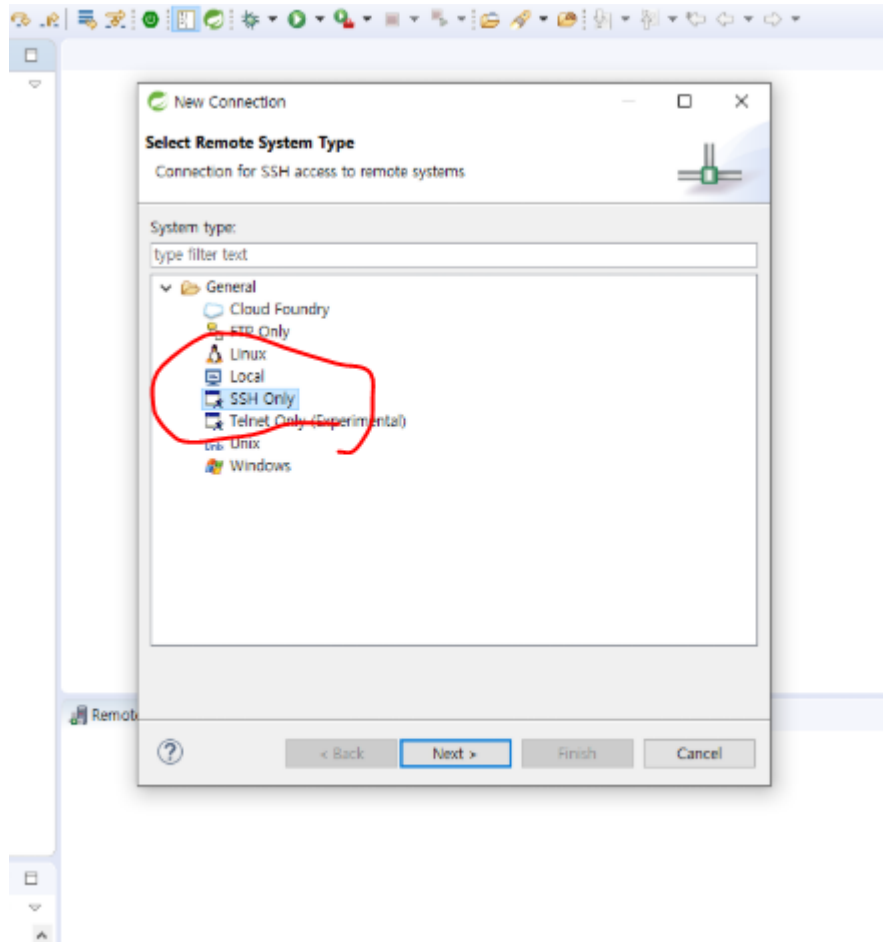
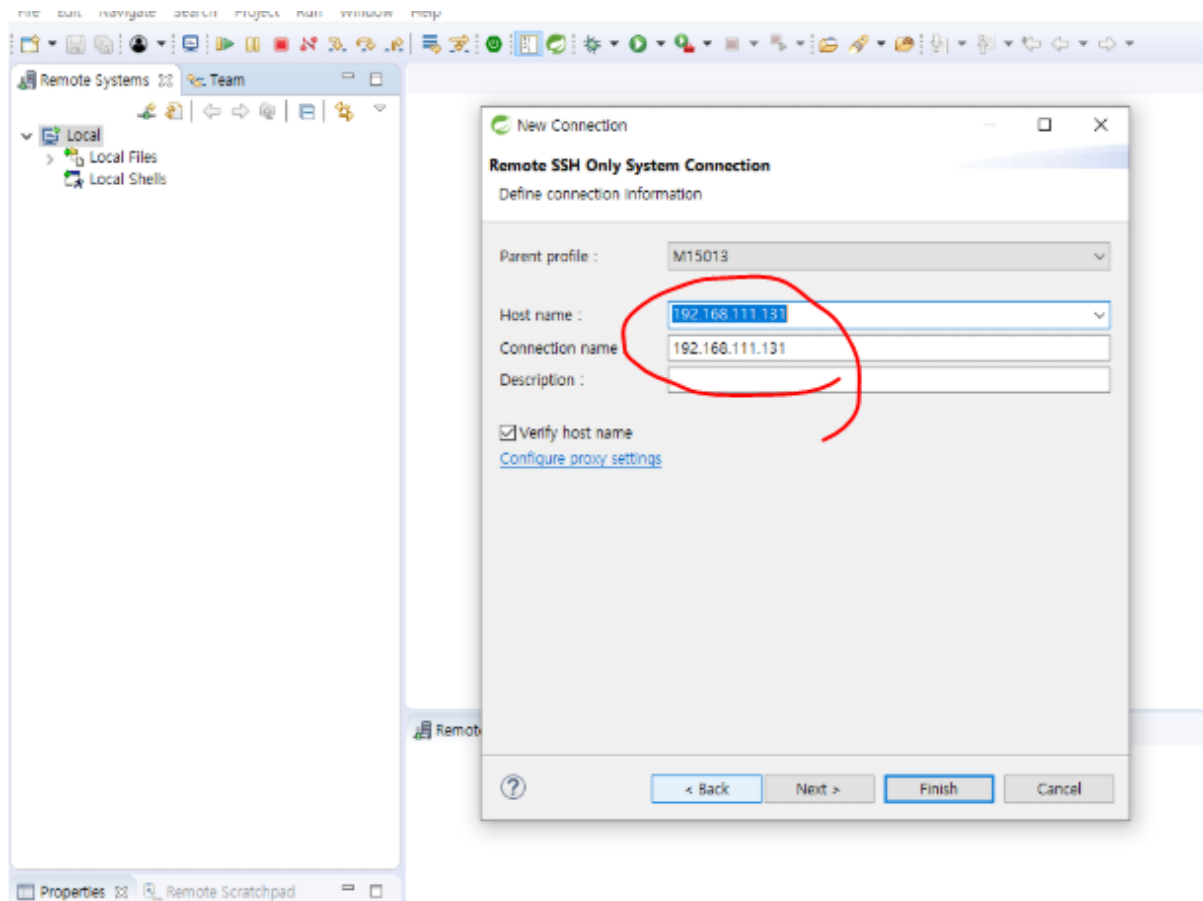


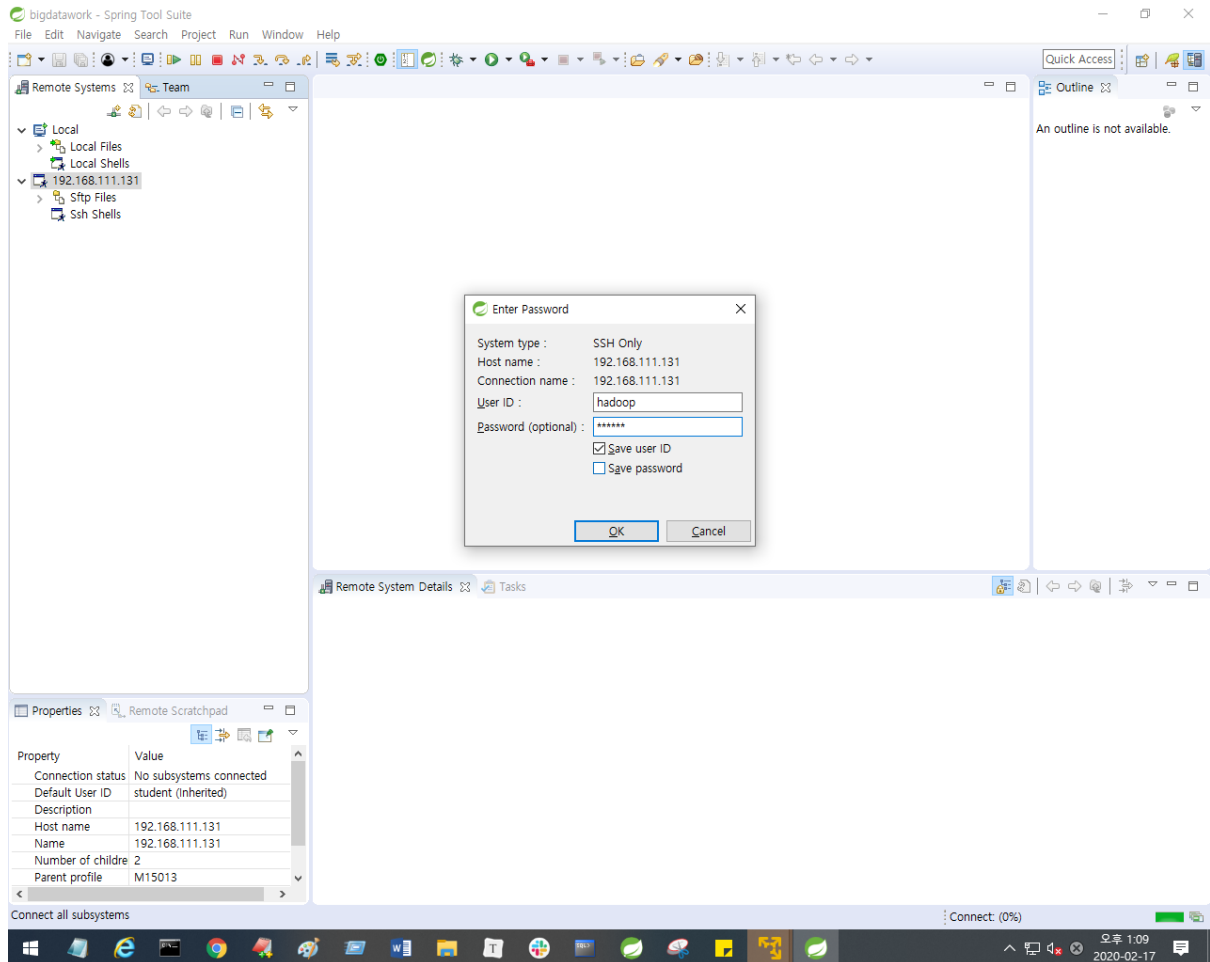
02-02-17(월) STS

STS 를 열고, Window > Perspectives > open perspective> Remote Systems Explorer 열고 Connection 해주기









```
[hadoop@hadoop01 hadoop-1.2.1]$ /home/hadoop/hadoop-1.2.1/bin/hadoop fs -copyFromLocal NOTICE.txt /input
[hadoop@hadoop01 hadoop-1.2.1]$ /bin/hadoop jar hadoop-examples-1.2.1.jar wordcount /input/NOTICE.txt /wordcount_output
20/02/17 13:54:37 INFO input.FileInputFormat: Total input paths to process : 1
20/02/17 13:54:37 INFO util.NativeCodeLoader: Loaded the native-hadoop library
20/02/17 13:54:37 WARN snappy.LoadSnappy: Snappy native library not loaded
20/02/17 13:54:37 INFO mapred.JobClient: Running job: job_202002171328_0006
20/02/17 13:54:38 INFO mapred.JobClient: map 0% reduce 0%
20/02/17 13:54:44 INFO mapred.JobClient: map 100% reduce 0%
20/02/17 13:54:51 INFO mapred.JobClient: map 100% reduce 33%
20/02/17 13:54:53 INFO mapred.JobClient: map 100% reduce 100%
20/02/17 13:54:54 INFO mapred.JobClient: Job complete: job_202002171328_0006
20/02/17 13:54:54 INFO mapred.JobClient: Counters: 29
20/02/17 13:54:54 INFO mapred.JobClient: Map-Reduce Framework
20/02/17 13:54:54 INFO mapred.JobClient: Spilled Records=22
20/02/17 13:54:54 INFO mapred.JobClient: Map output materialized bytes=173
20/02/17 13:54:54 INFO mapred.JobClient: Reduce input records=11
20/02/17 13:54:54 INFO mapred.JobClient: Virtual memory (bytes) snapshot=3888091136
20/02/17 13:54:54 INFO mapred.JobClient: Map input records=2
20/02/17 13:54:54 INFO mapred.JobClient: SPLIT_RAW_BYTES=102
20/02/17 13:54:54 INFO mapred.JobClient: Map output bytes=145
20/02/17 13:54:54 INFO mapred.JobClient: Reduce shuffle bytes=173
```

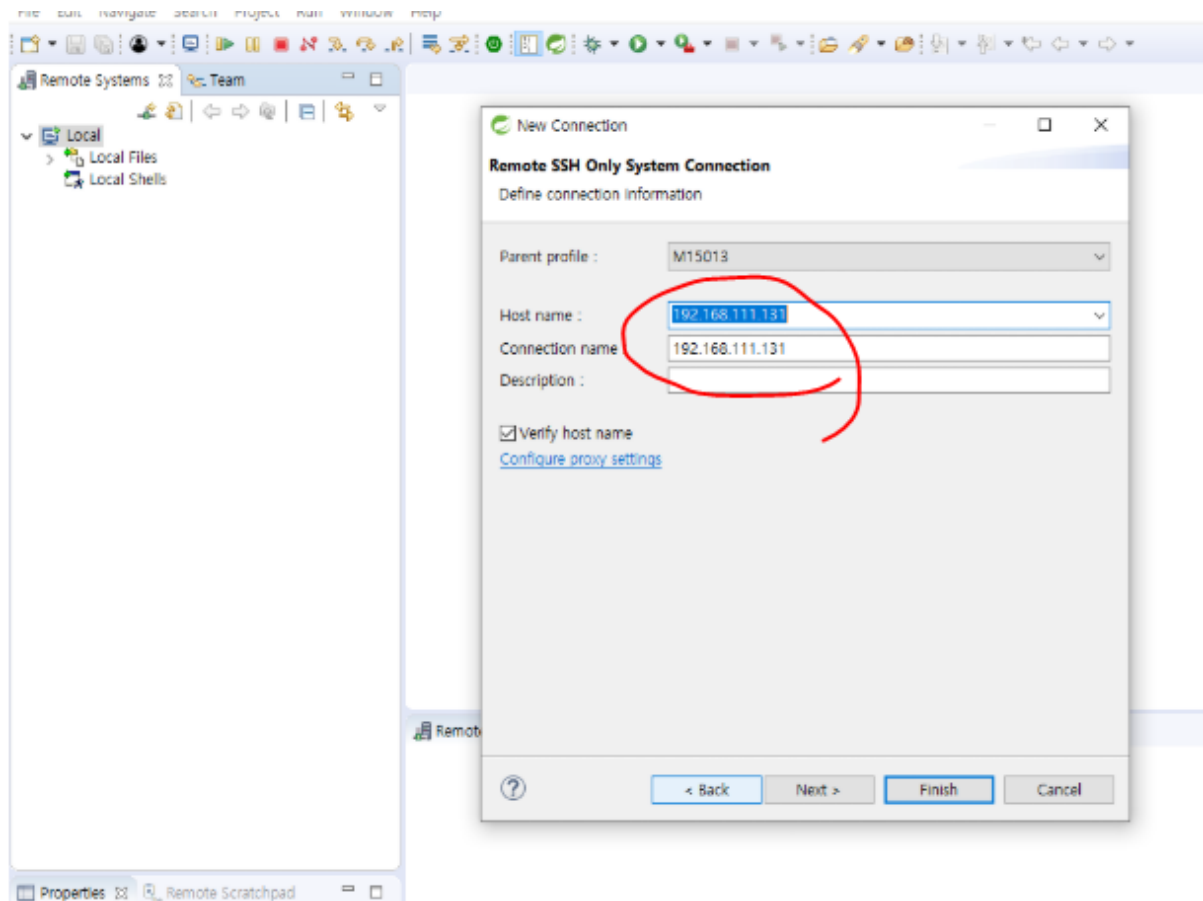
jar : 자바압축파일

(상대경로) 현재 디렉토리 밑에 있는 bin의...

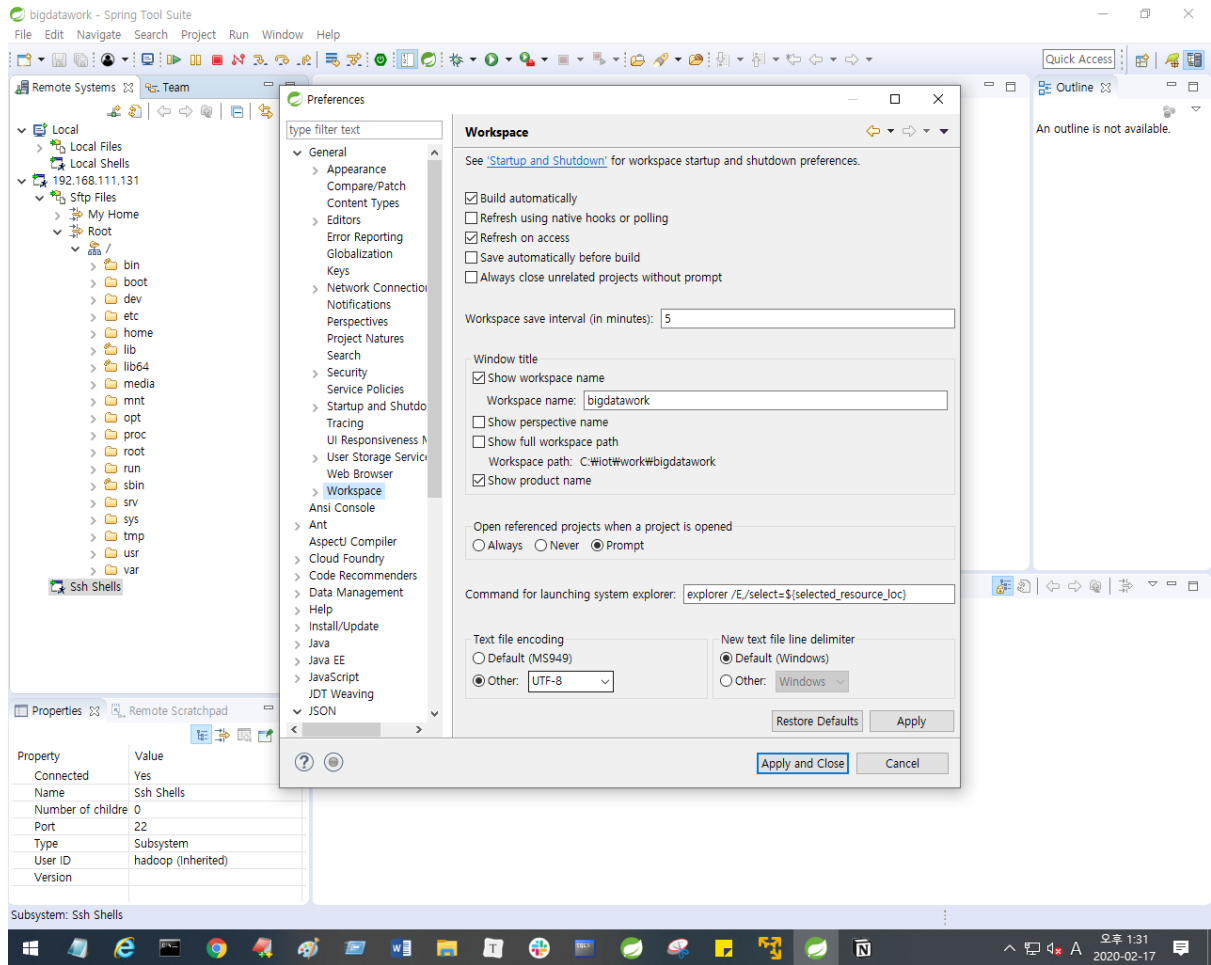
(p.62) wordcount 클래스 실행(wordcount input output 순서로 적는다)

wordcount 는 메인이 있는 클래스 그 뒤엔 input파일의 경로

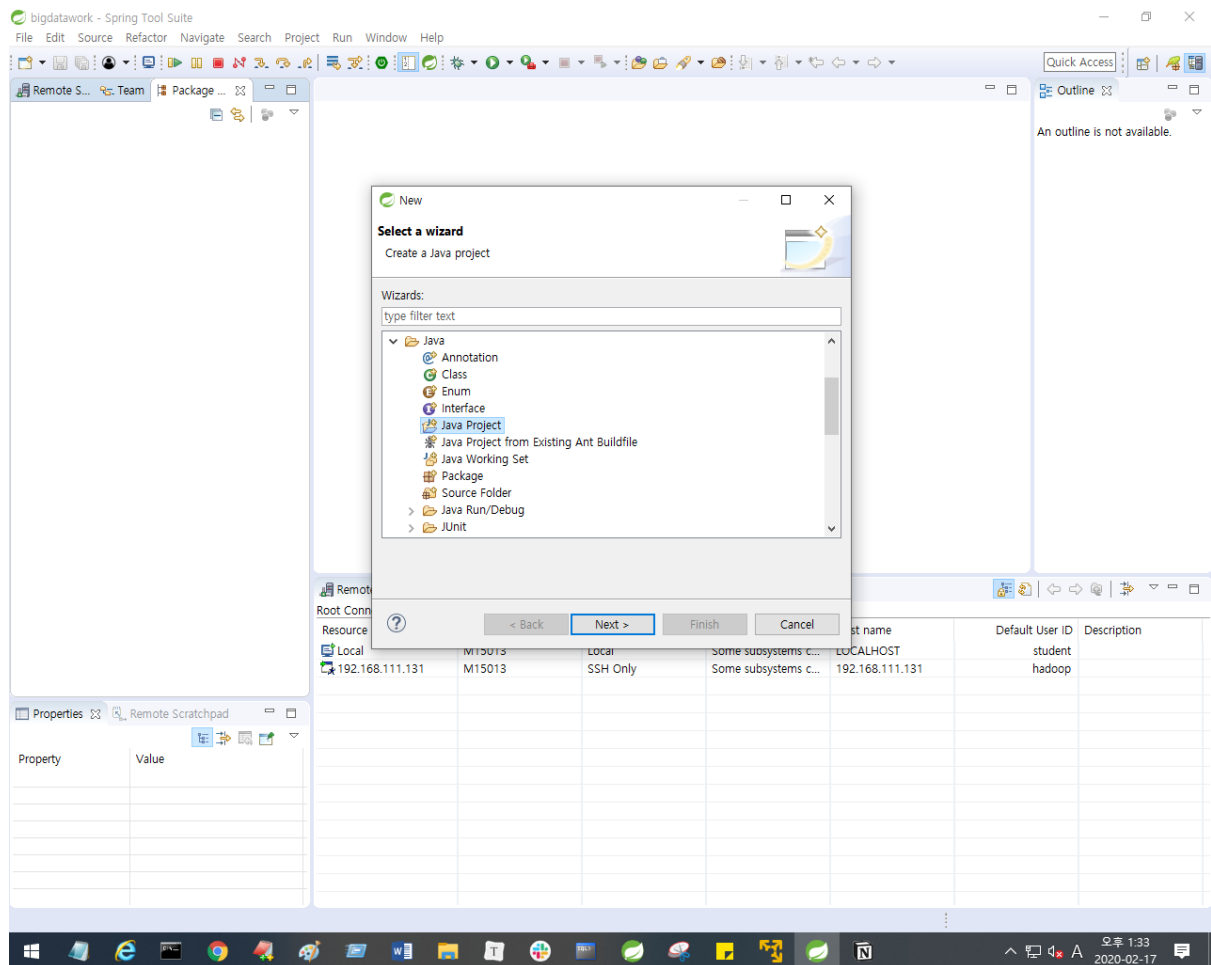
실행 후 firefox 에서 확인 시 파일 들어옴을 확인

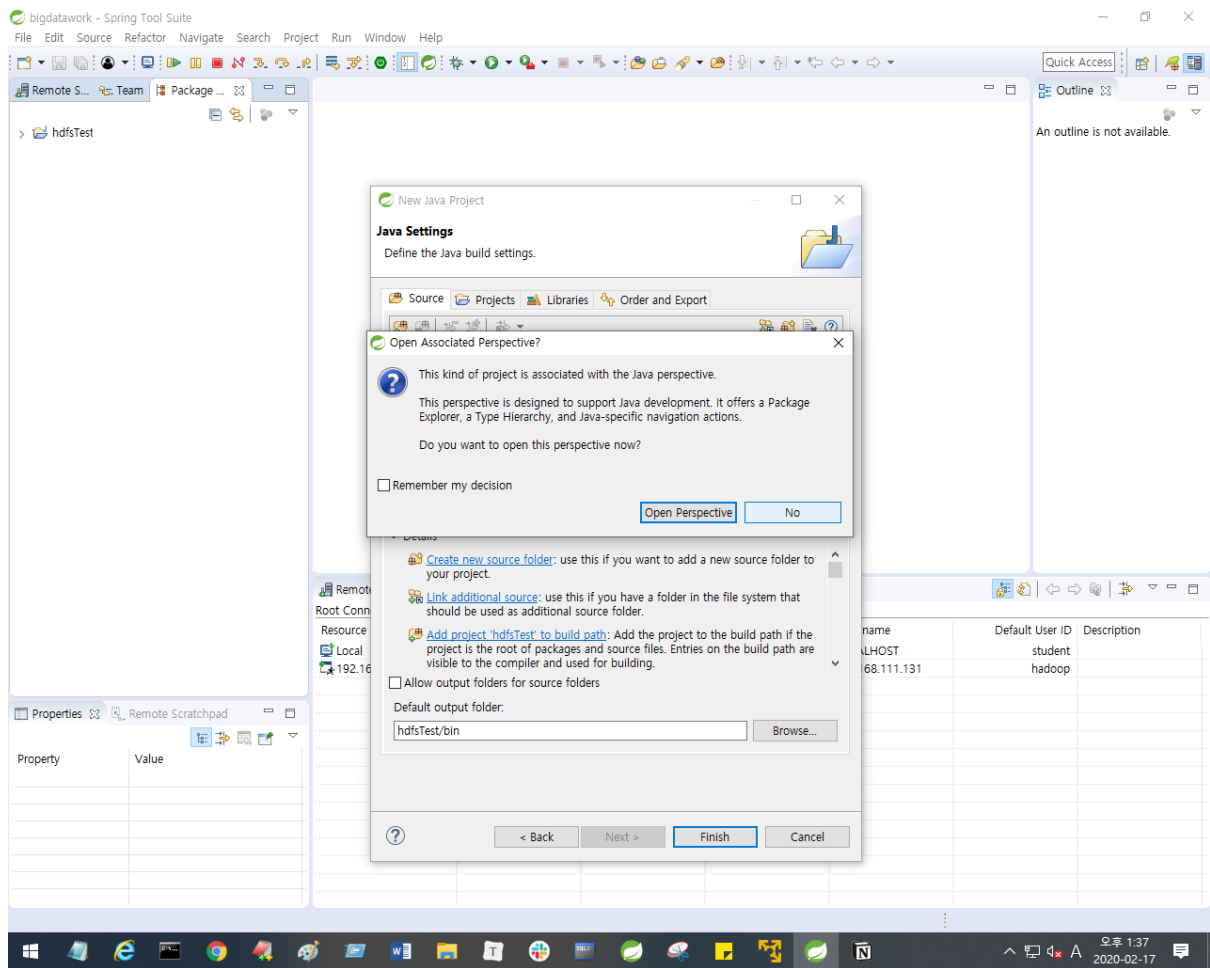


한글 설정을 해주기

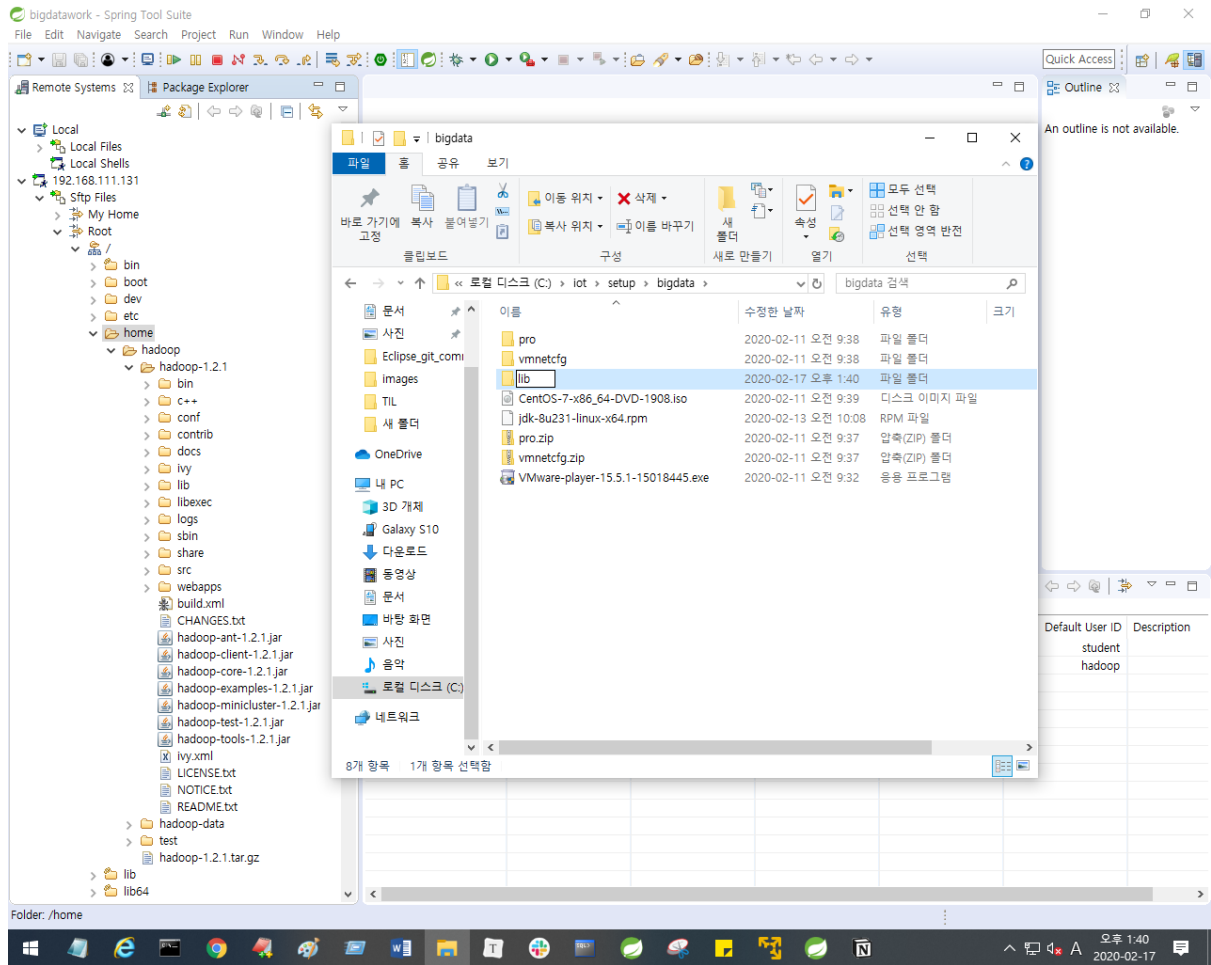


새로운 자바 프로젝트 생성



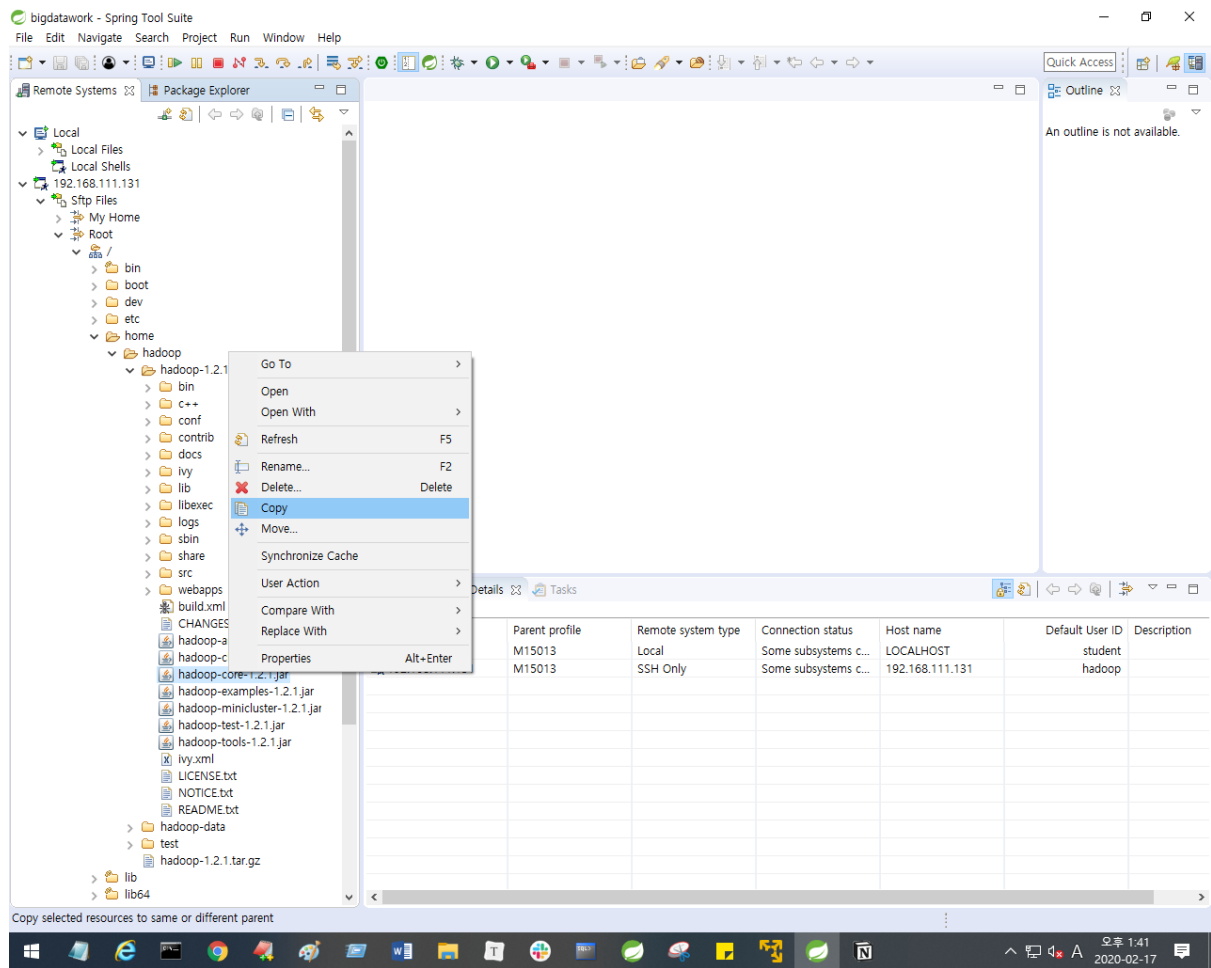


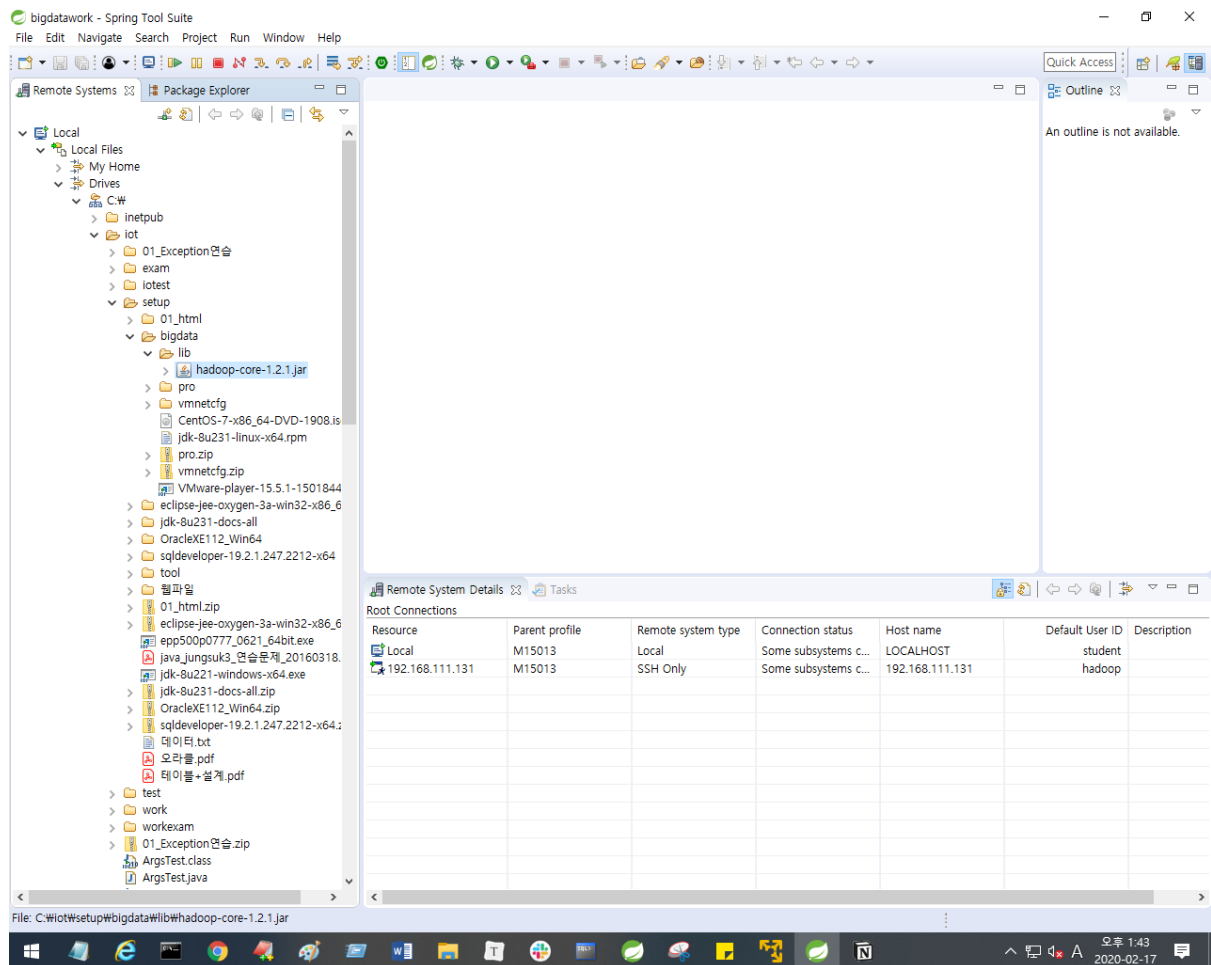
NO 선택



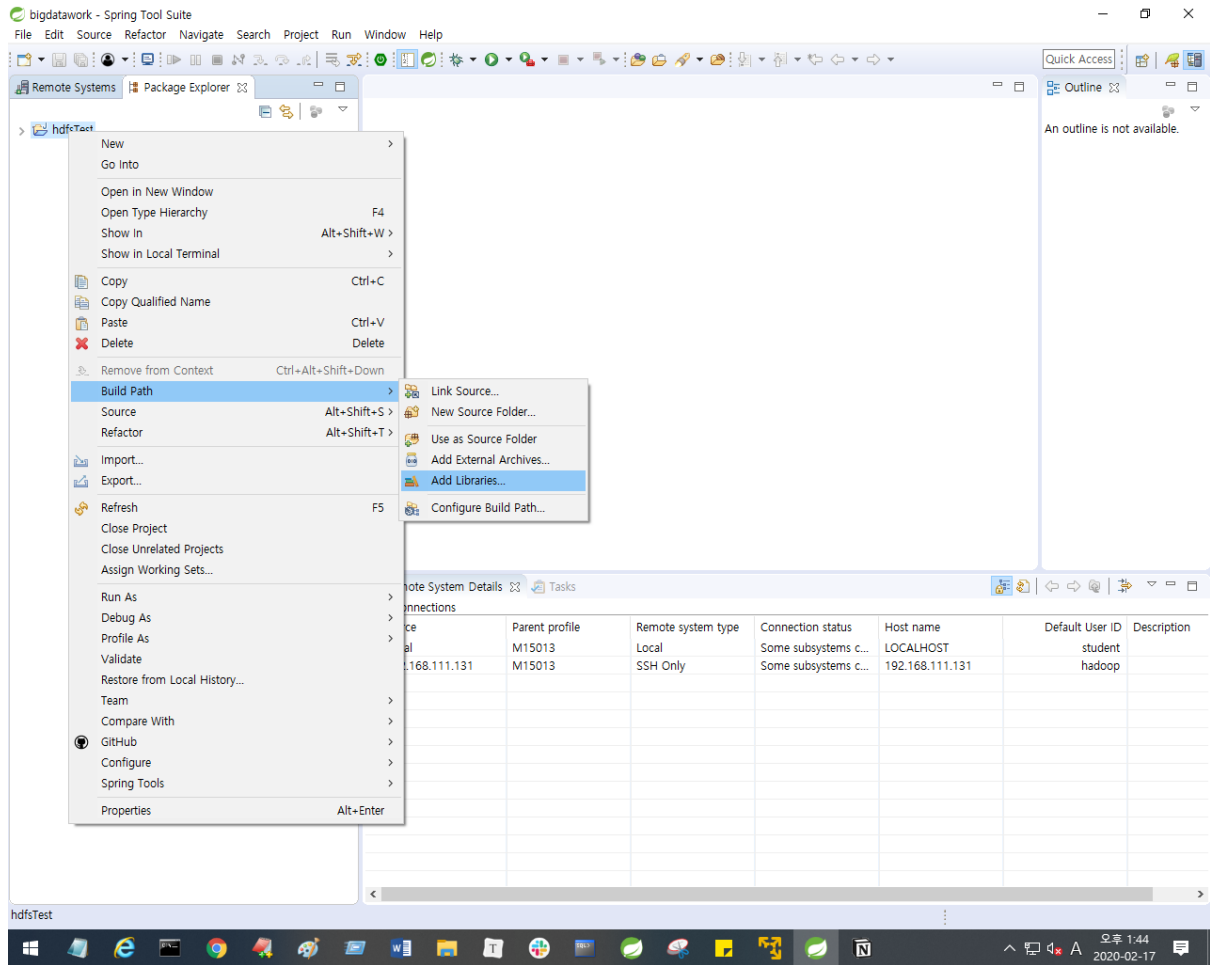
설치 폴더에 임의로 lib 폴더 생성 (여기에 hadoop-core-1.2.1.jar 파일 복사해서 넣을 것임)

hadoop-core-1.2.1.jar 파일 복사

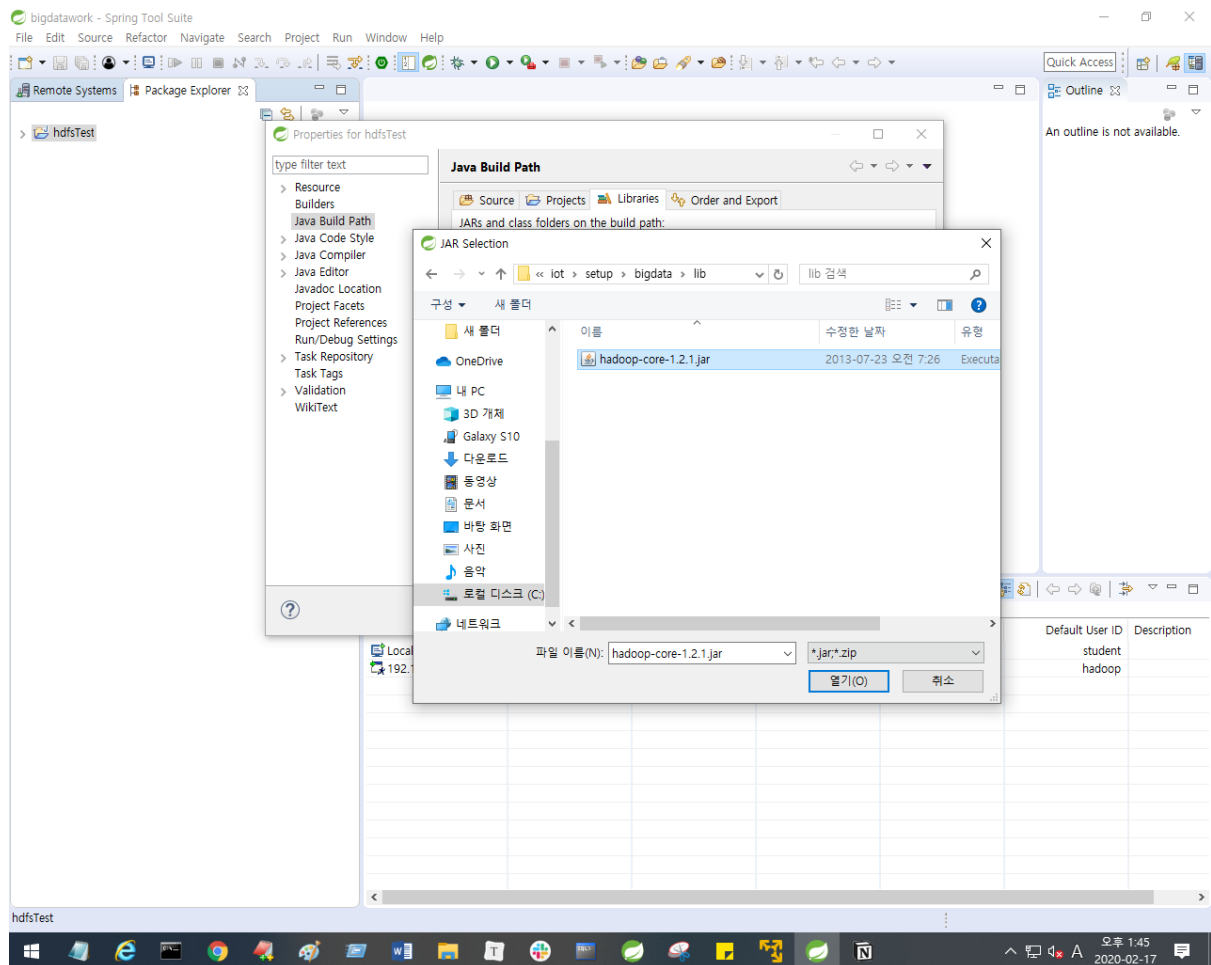


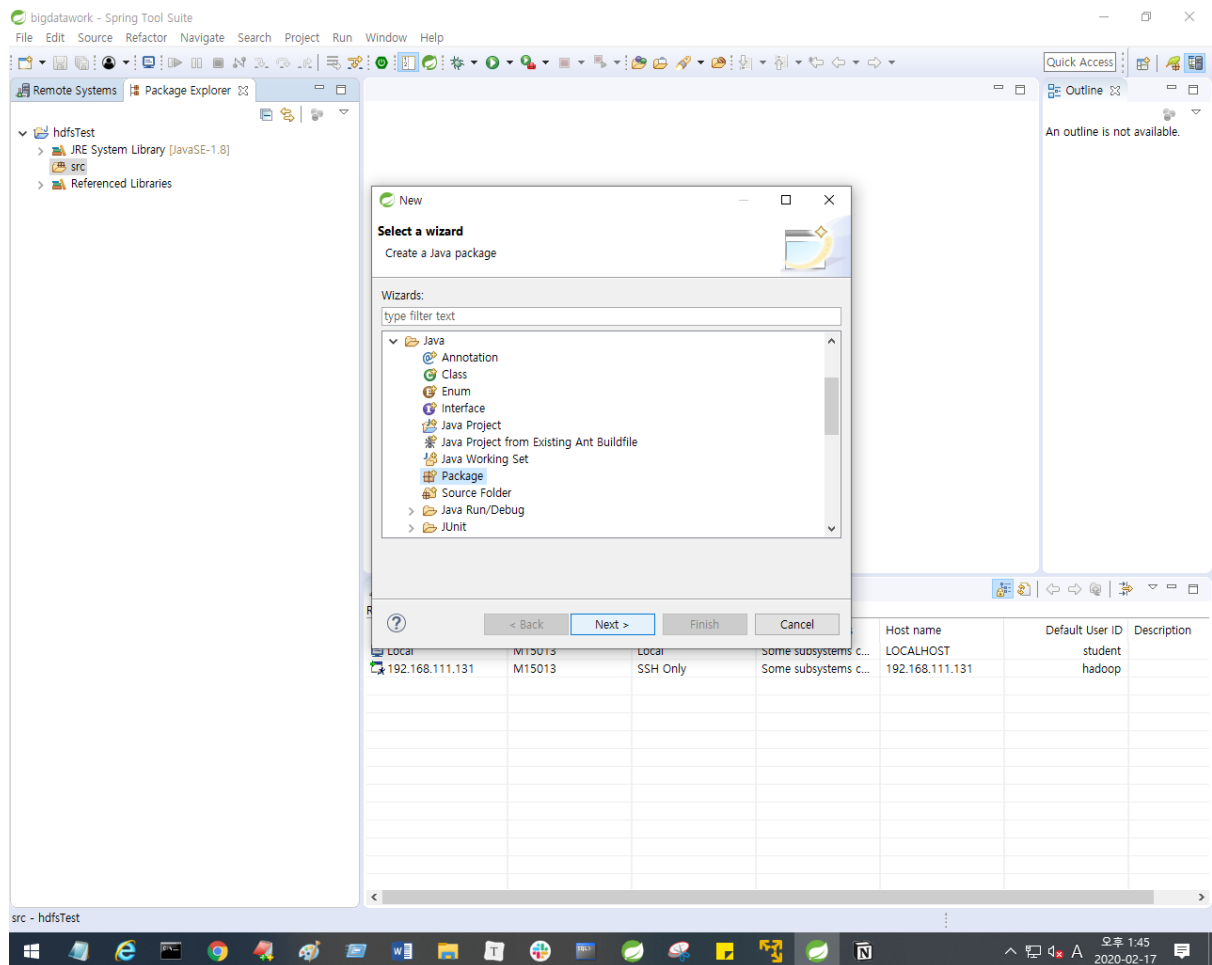


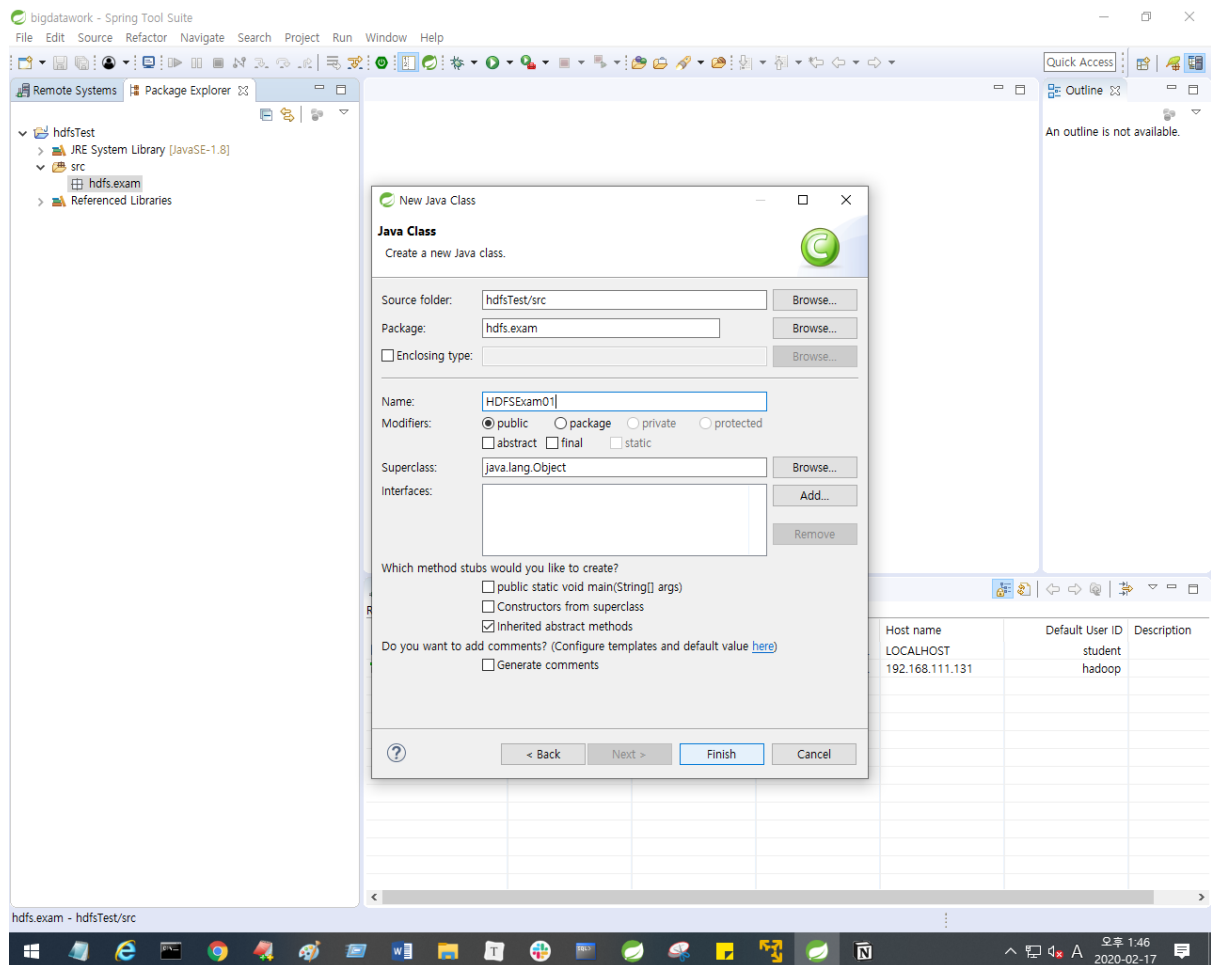
내가 만든 lib에 붙여넣기 .hadoop-core-1.2.1.jar

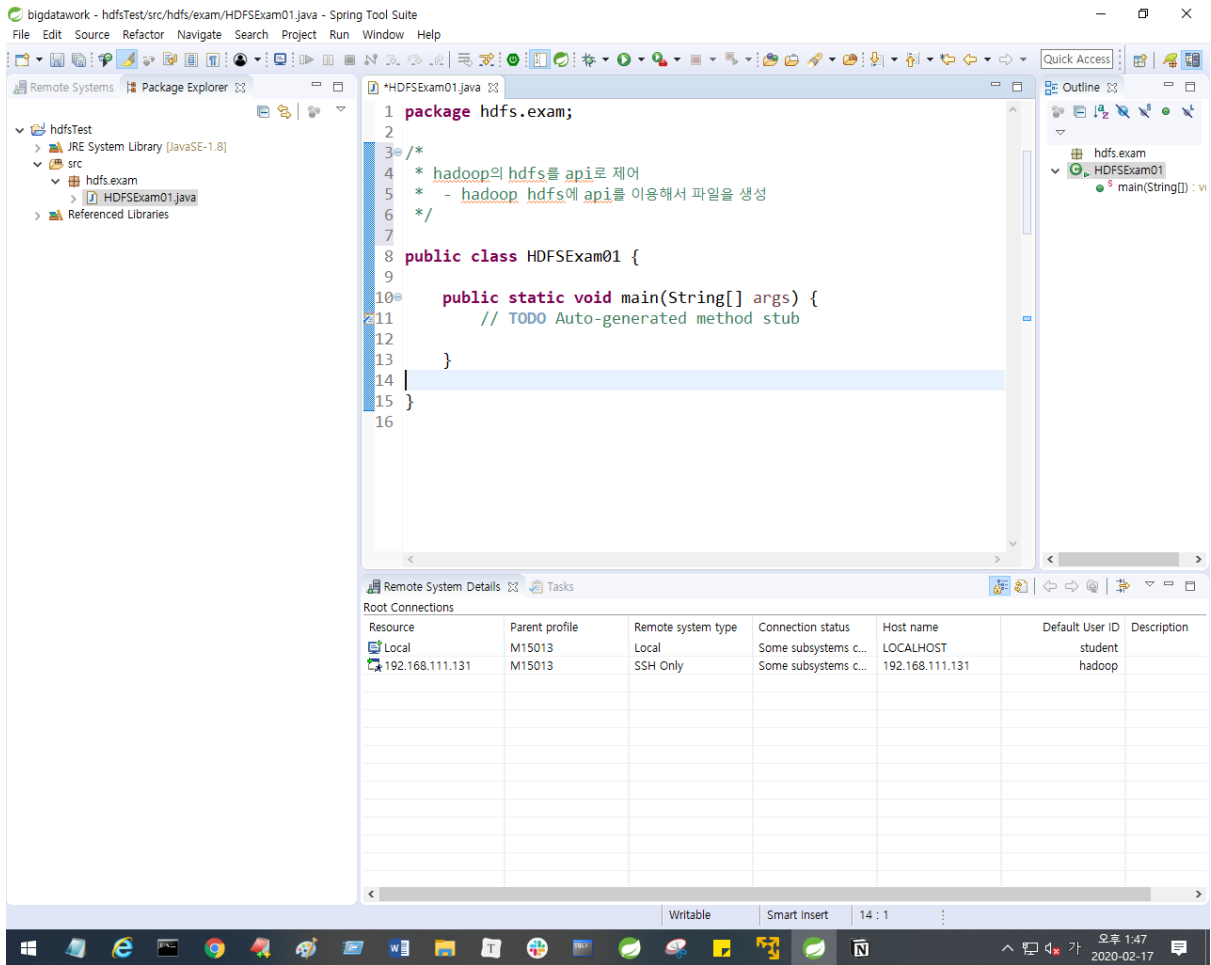


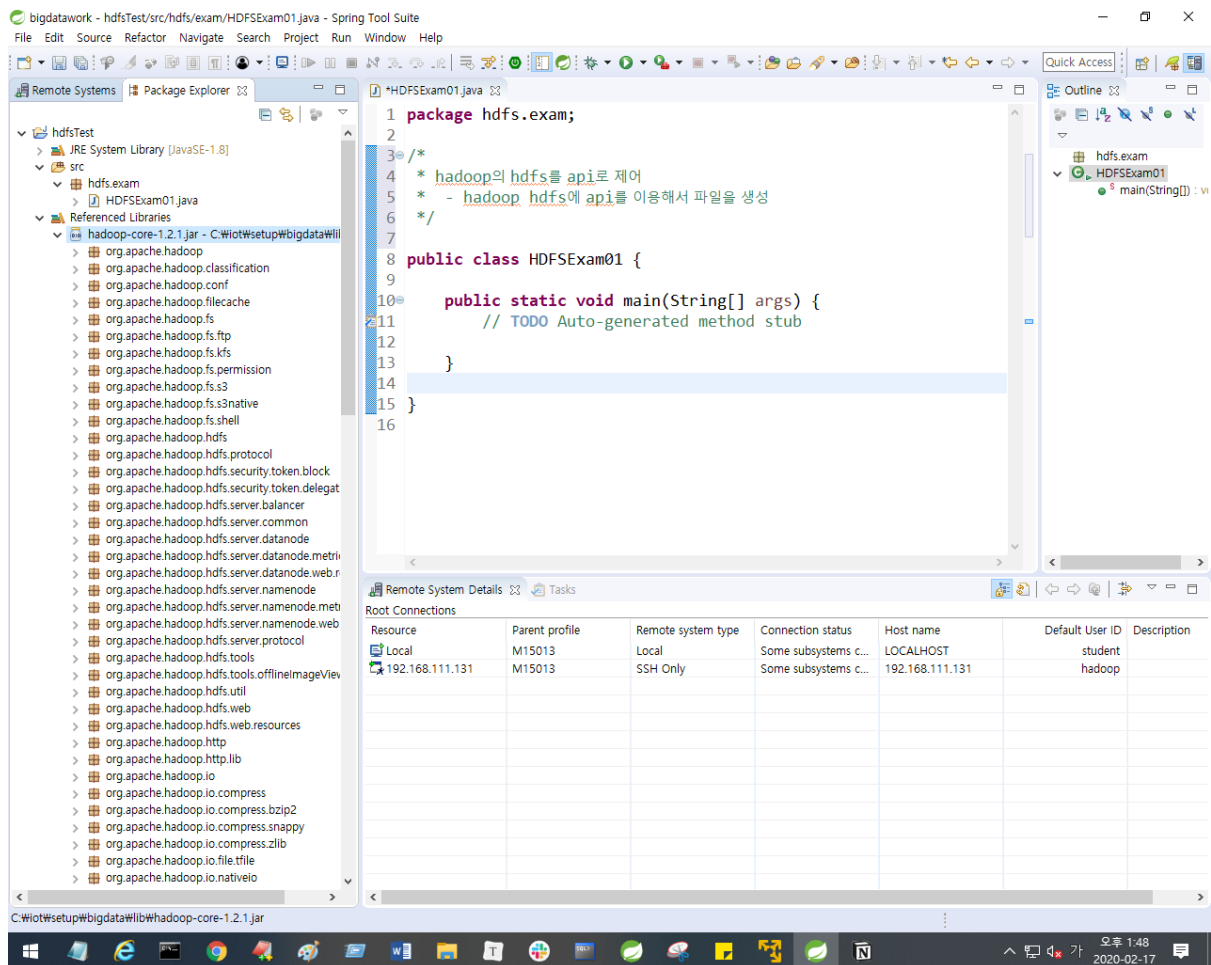
프로젝트 오른쪽버튼 눌러서 > Build Path > Configure Build Path > Add External JARS.. > 방금 붙여넣은 lib 가서 hadoop-core-1.2.1.jar 를 넣기





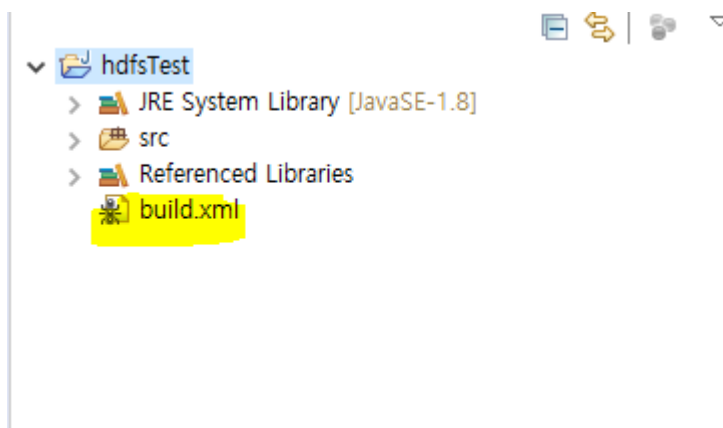






p.97~

jar를 묶는 작업



선생님 블로그에서 build.xml 복사해서 hdfsTest 프로젝트에 붙여넣기

build.xml 파일을 보자.

```
9 <property name="project-name" value="hadoop-examples" />
10 <property name="general-lib" value="${company-name}-${project-name}.jar" />
11 <property name="general-src" value="${company-name}-${project-name}-src.zip" />
12
13 <property name="build-Path" location="." />
14 <property name="src.dir.src" location="${build-Path}/src" />
15 <property name="src.dir.bin" location="${build-Path}/bin" />
16 <property name="src.dir.build" location="${build-Path}/build" />
17
18 <target name="build" depends="build-lib, build-src" />
19 <target name="clean-all" depends="clean-lib, clean-src" />
20
21 <target name="clean-lib">
22   <delete file="${src.dir.build}/${general-lib}" />
23 </target>
24
25 <target name="clean-src">
26   <delete file="${src.dir.build}/${general-src}" />
27 </target>
28
29 <target name="build-lib" depends="clean-lib">
30   <jar destfile="${src.dir.build}/${general-lib}" basedir="${src.dir.bin}">
31     <manifest>
32       <attribute name="${project-name}-Version" value="${version}" />
33     </manifest>
34   </jar>
35 </target>
36
37 <target name="build-src" depends="clean-src">
38   <zip zipfile="${src.dir.build}/${general-src}" basedir="${src.dir.src}">
39   </zip>
40 </target>
41 </project>
```

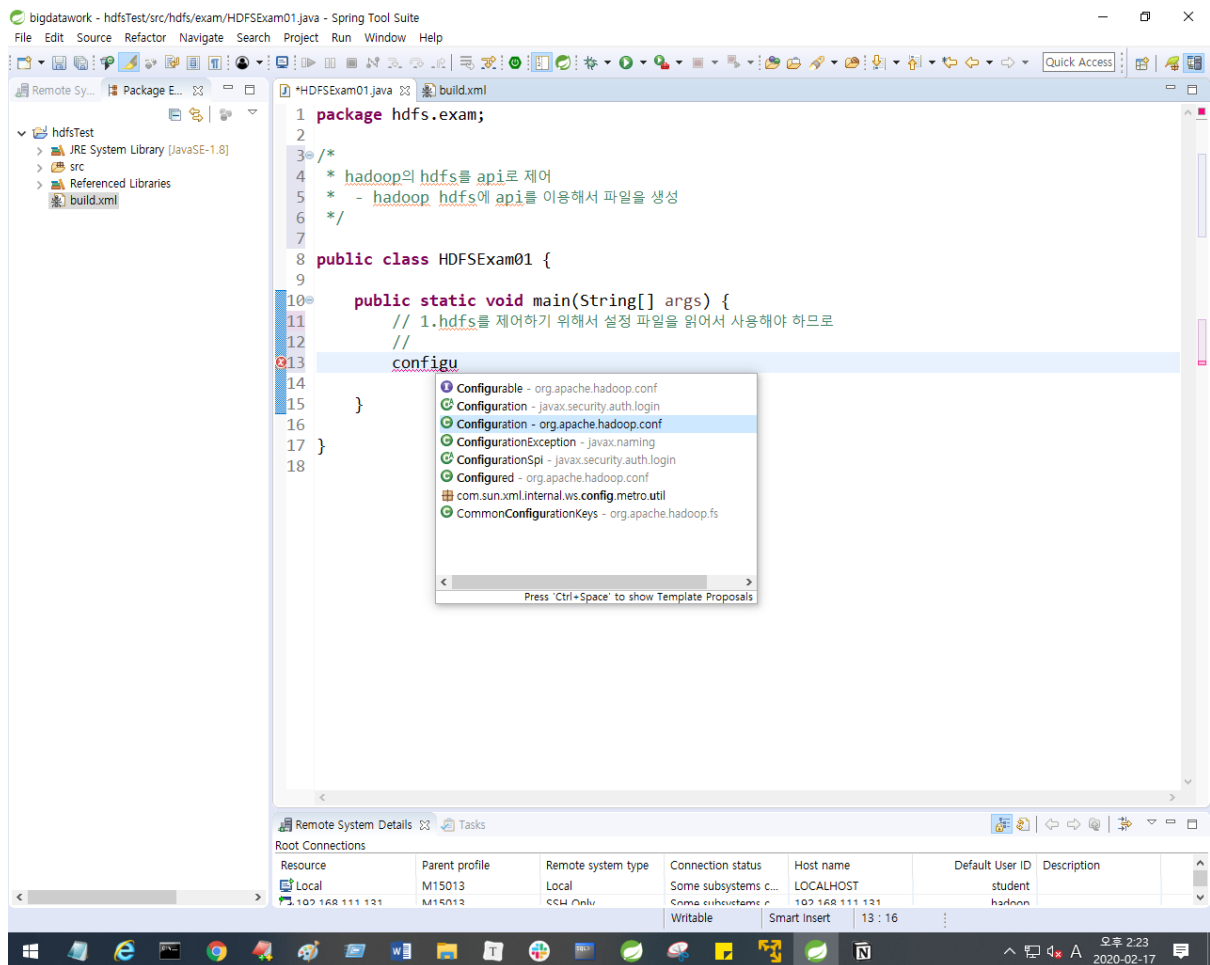
명령어가 적힌 설정파일이다.

`.` : 현재 디렉토리

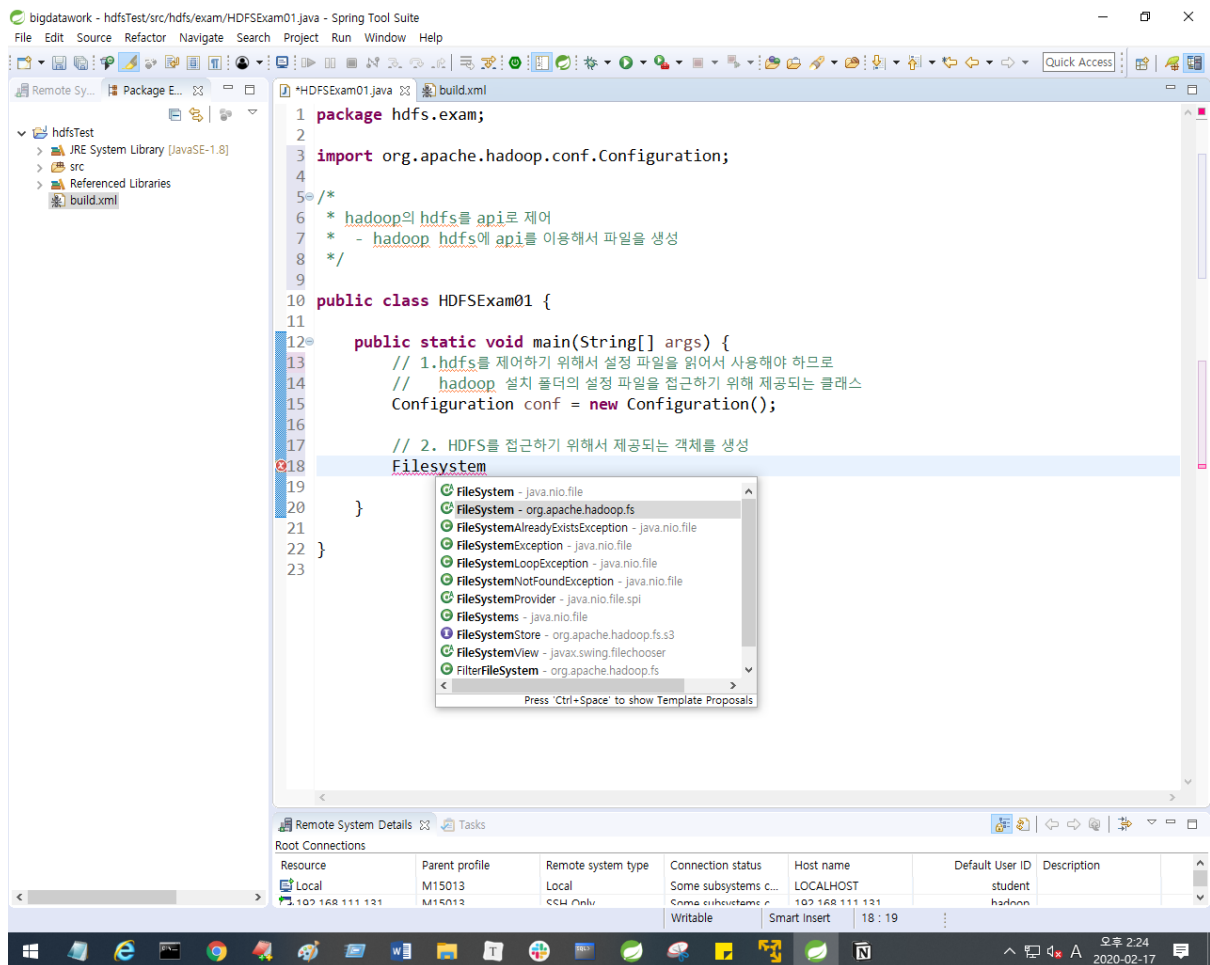
`delete` : 삭제하라는 뜻

`jar` : 가 써있으면 jar로 만들어라 라는 뜻

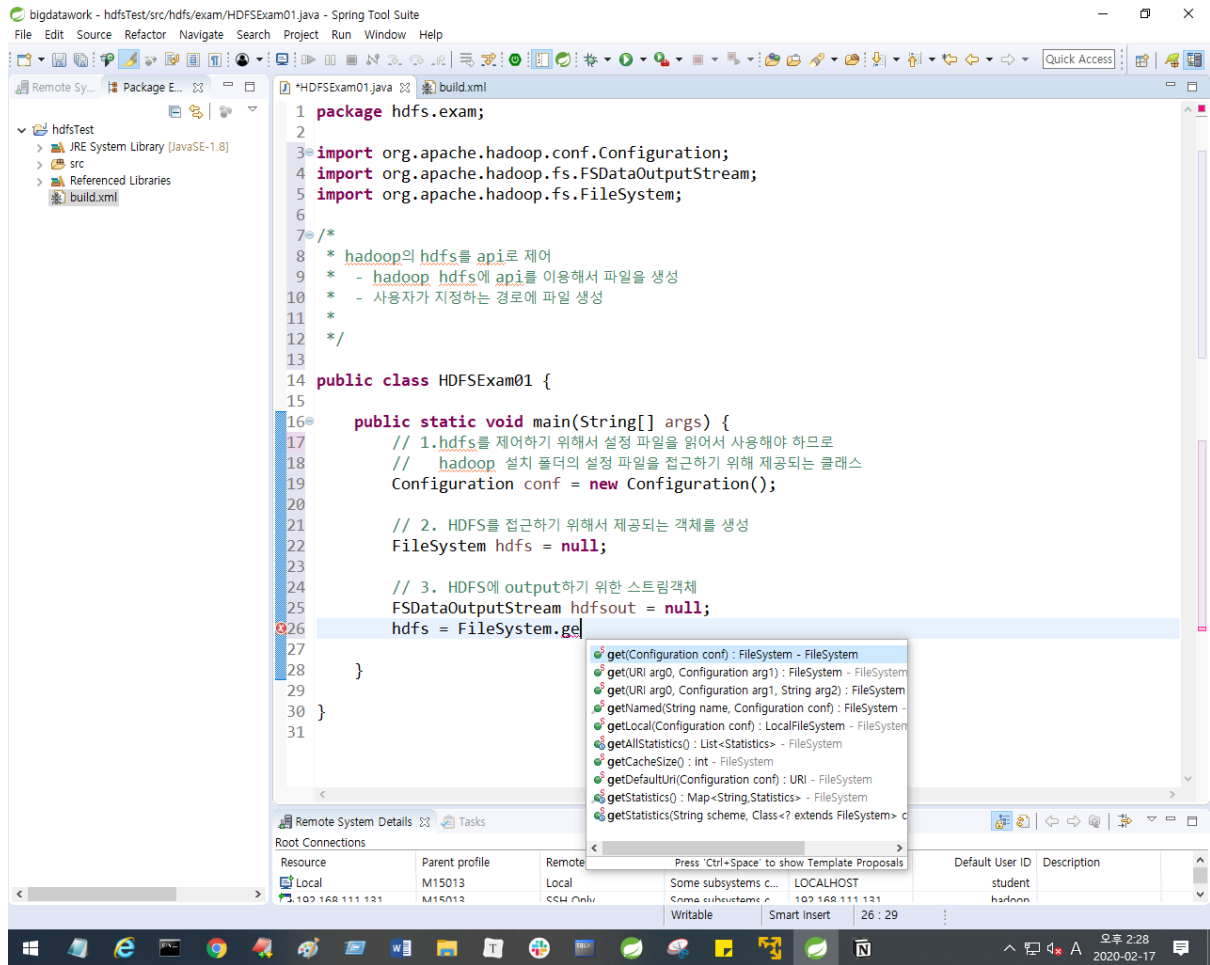
1. org.apache.hadoop 에 있는 Configuration을 사용해 설정파일을 접근한다.



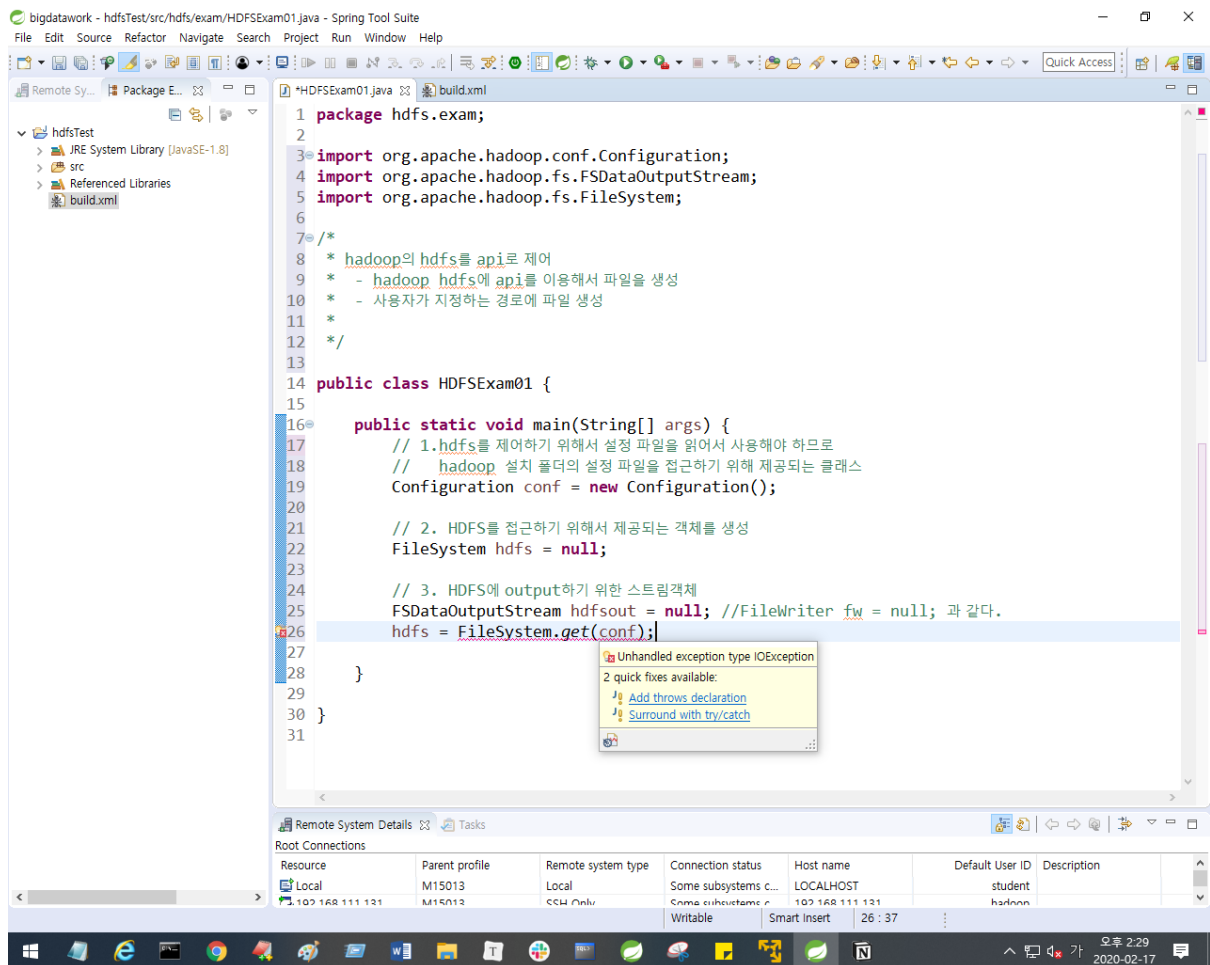
2. org.apache.hadoop 에 있는 FileSystem으로 접근



3. HDFS에 ouhtput하기 위한 스트림객체



new해서 만들지 않고, 메소드를 통해 받아오거나 그러면 이 객체는 시스템과 관련된 파일이기 때문에 싱글톤으로 운영이 된다. 따라서 변수가 1개인 것을 선택



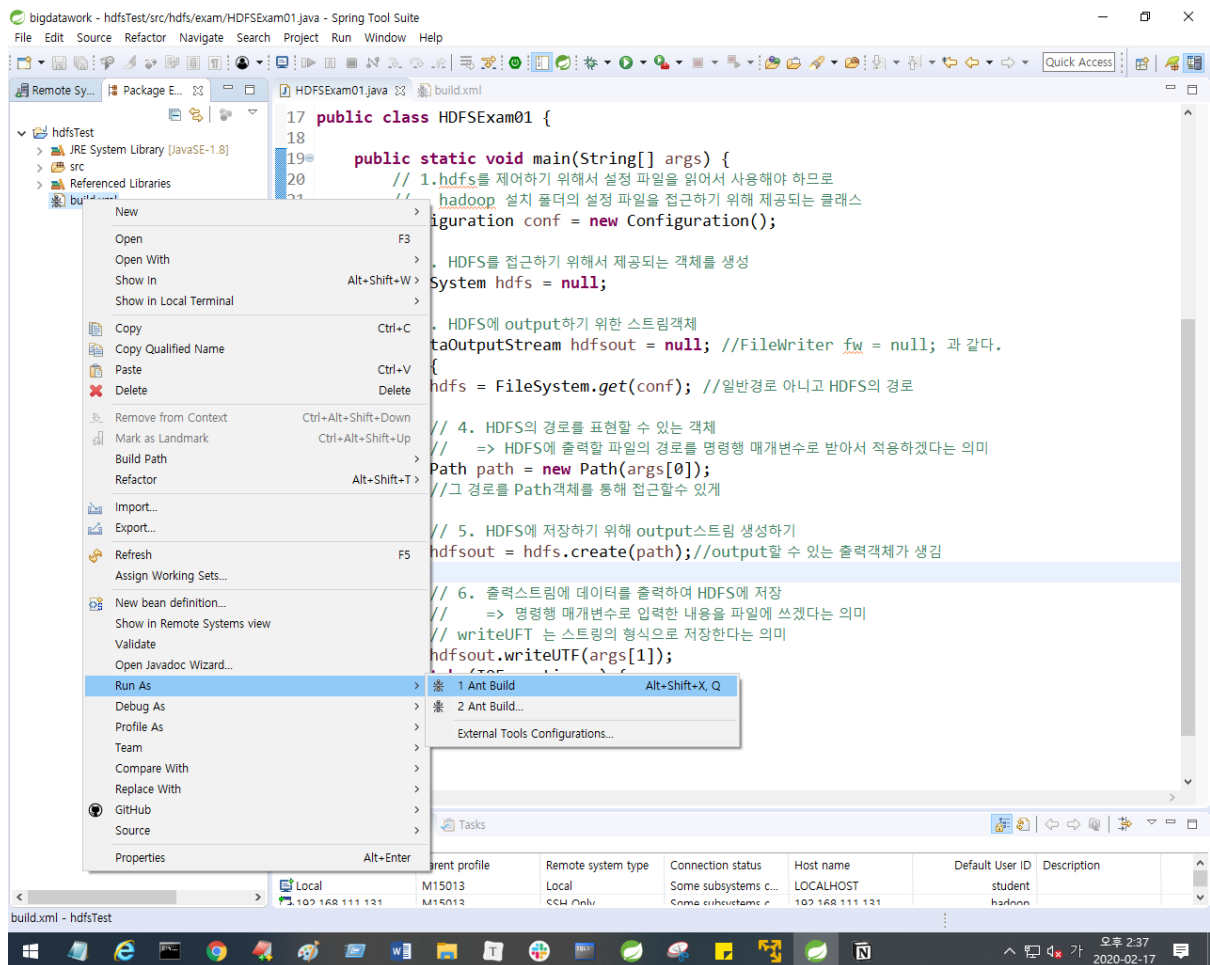
IOException에 Try , Catch 묶어준다.

create메소드 중에서 path를 전달하는 create메소드를 사용한다.

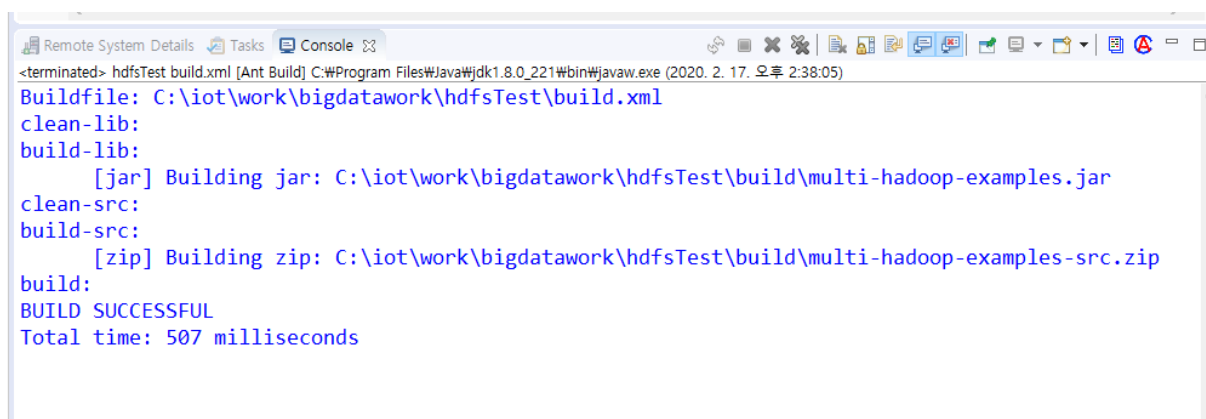
현재 파일시스템을 만들어서 (hdfs) 전달한다.

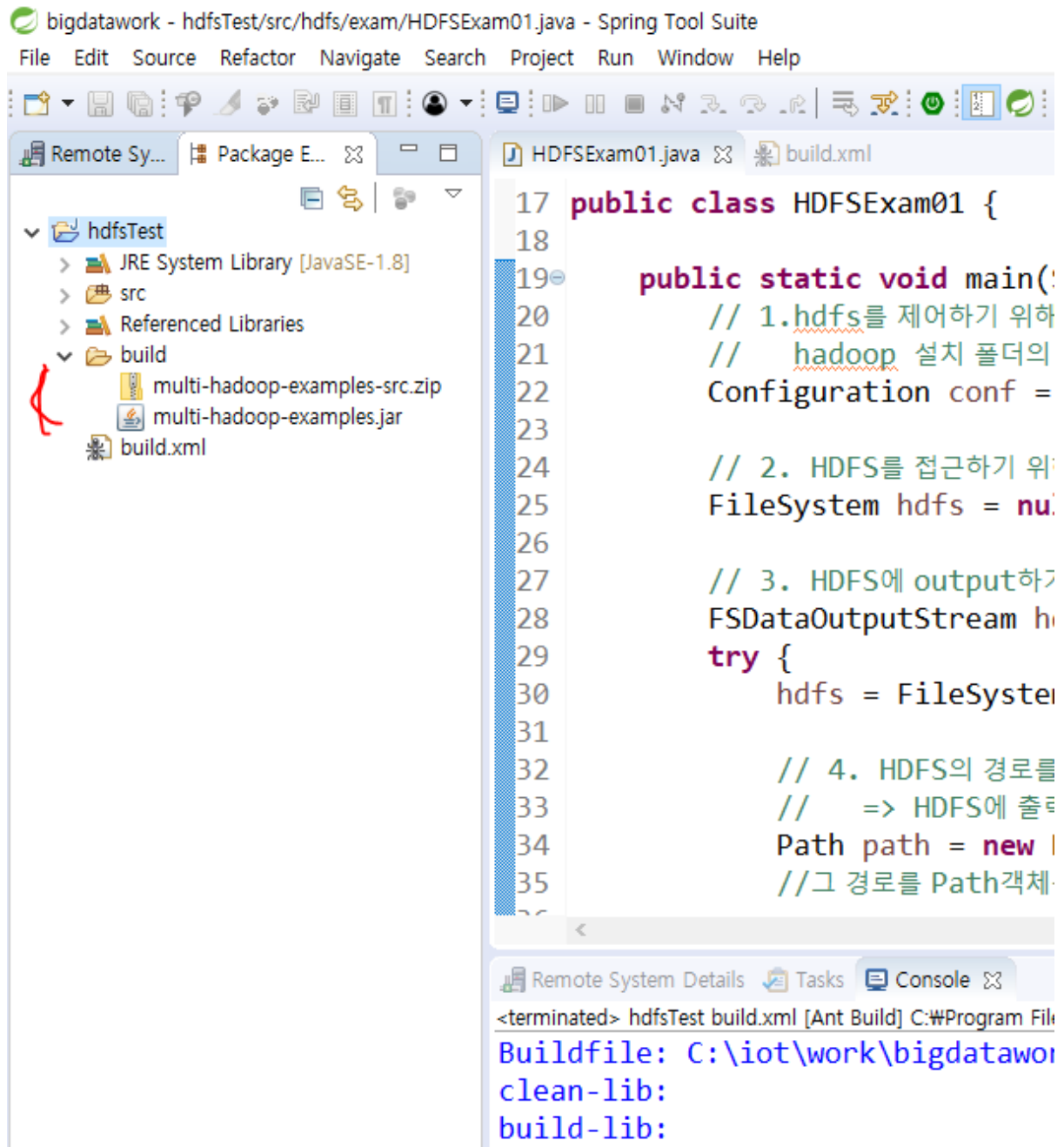
새롭게 new하지 않는다.(새로운 객체를 생성하지 않는다.)

```
HDFSExam01.java build.xml
17 public class HDFSExam01 {
18
19     public static void main(String[] args) {
20         // 1.hdfs를 제어하기 위해서 설정 파일을 읽어서 사용해야 하므로
21         //   hadoop 설치 폴더의 설정 파일을 접근하기 위해 제공되는 클래스
22         Configuration conf = new Configuration();
23
24         // 2. HDFS를 접근하기 위해서 제공되는 객체를 생성
25         FileSystem hdfs = null;
26
27         // 3. HDFS에 output하기 위한 스트림객체
28         FSDataOutputStream hdfsout = null; //FileWriter fw = null; 과 같다.
29         try {
30             hdfs = FileSystem.get(conf); //일반경로 아니고 HDFS의 경로
31
32             // 4. HDFS의 경로를 표현할 수 있는 객체
33             //   => HDFS에 출력할 파일의 경로를 명령행 매개변수로 받아서 적용하겠다는 의미
34             Path path = new Path(args[0]);
35             //그 경로를 Path객체를 통해 접근할수 있게
36
37             // 5. HDFS에 저장하기 위해 output스트림 생성하기
38             hdfsout = hdfs.create(path); //output할 수 있는 출력객체가 생김
39             |
40             // 6. 출력스트림에 데이터를 출력하여 HDFS에 저장
41             //   => 명령행 매개변수로 입력한 내용을 파일에 쓰겠다는 의미
42             // writeUTF 는 스트림의 형식으로 저장한다는 의미
43             hdfsout.writeUTF(args[1]);
44         } catch (IOException e) {
45
46             e.printStackTrace();
47         }
48     }
49 }
```

build.xml 오른쪽버튼 > Run as > Ant Build



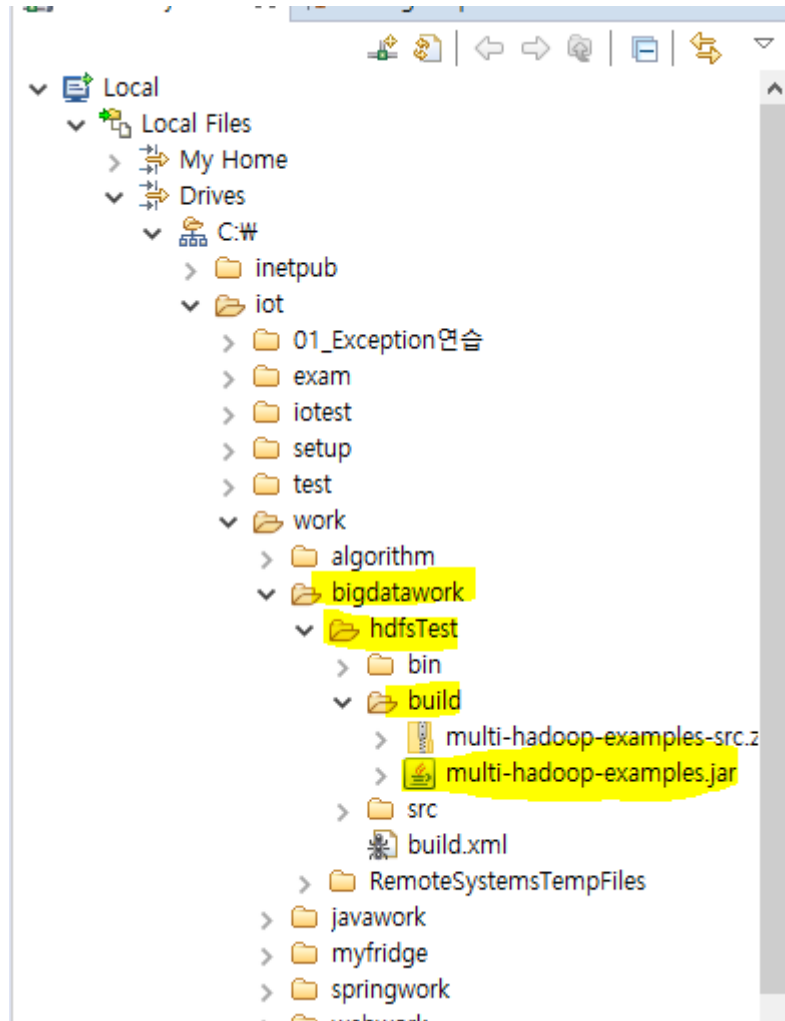


hdfsTest 프로젝트에 커서 놓고 F5누르면 build폴더가 생기고 안에 .zip파일과 .jar파일 만들어져있다.

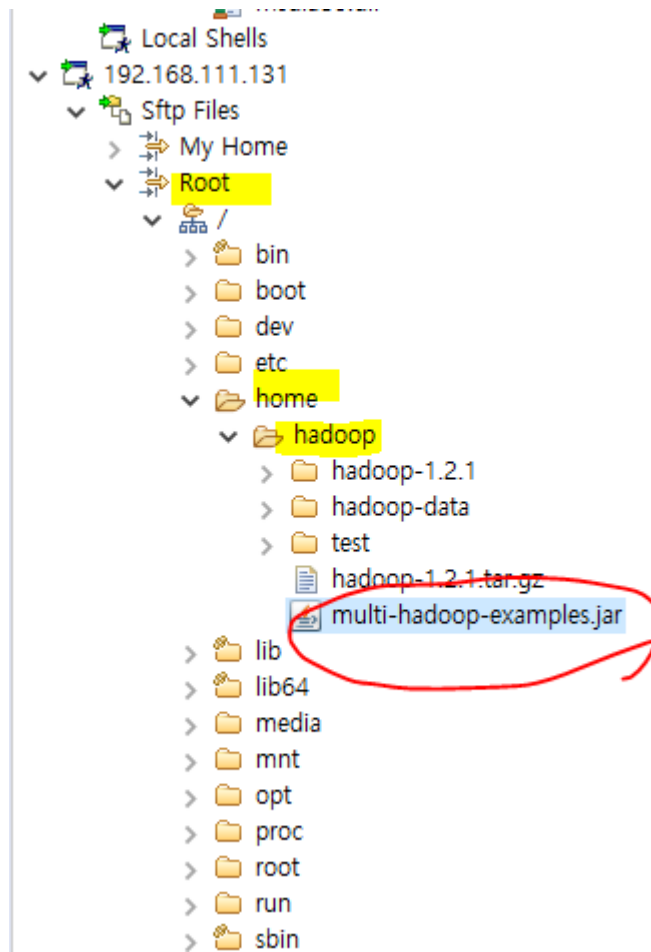
- 앞으로 무언가가 변경될 때마다, build.xml에서 ant build를 해줘야 한다.
- 그리고 만들어진 파일을 hadoop계정의 home > hadoop디렉토리에 카피해야 한다.
- sts에서 작업을 완료하고 완료된 작업물을 리눅스 머신에 복사한다. 리눅스에서 작업하지 않아도, 연동된 hadoop01 머신 Remote System을 통해서 옮기고 적용할 수

있다.

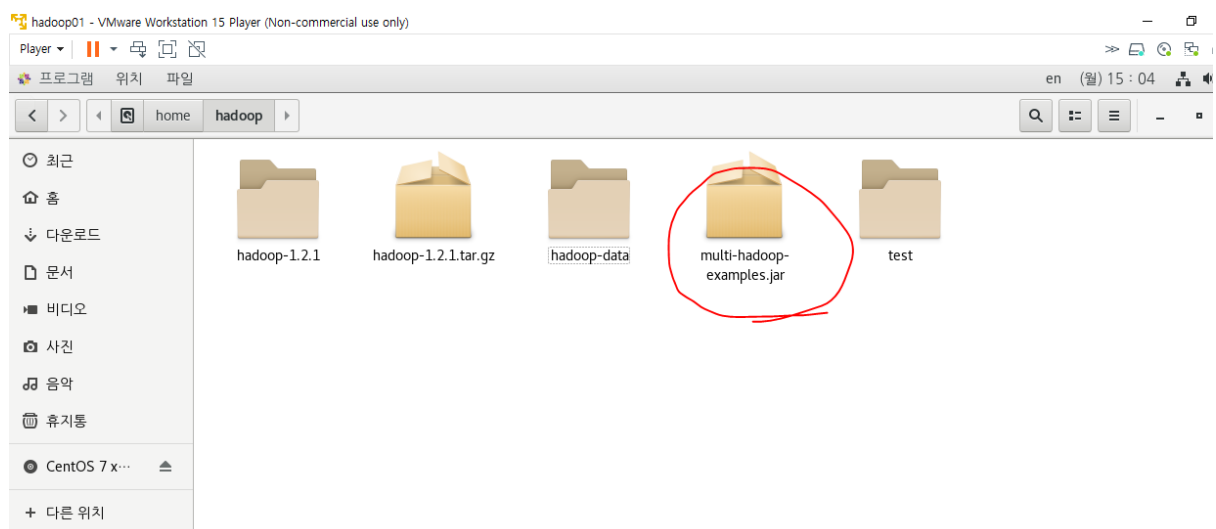
만들어진 jar 파일을 복사한다.



하둡01머신의 > home > hadoop에 만들어진 jar파일 붙여넣기



머신으로 가자



터미널창에서 실행시켜보자.

패키지명, 클래스명 카피해서 적는다.

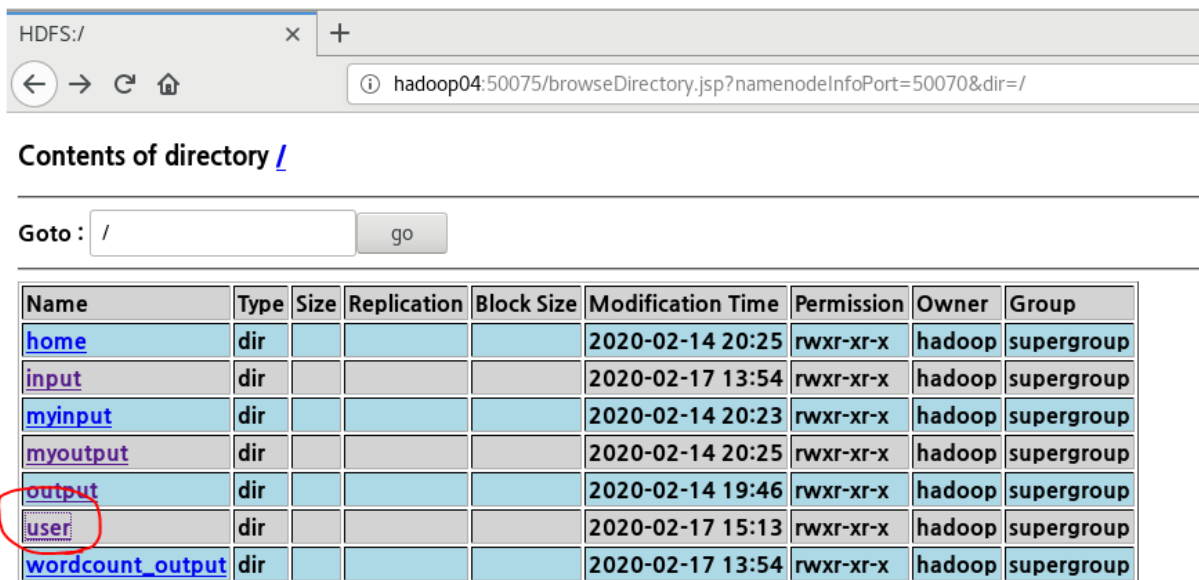
jar 파일 실행 명령어

`./hadoop-1.2.1/bin/hadoop jar multi-hadoop-examples.jar hdfs.exam HDFSEXAM0` 경로 파일에
쓸말

```
[hadoop@hadoop01 ~]$ ./hadoop-1.2.1/bin/hadoop jar multi-hadoop-examples.  
jar hdfs.exam.HDFSEXAM0 output.txt hellohadoop  
[hadoop@hadoop01 ~]$
```

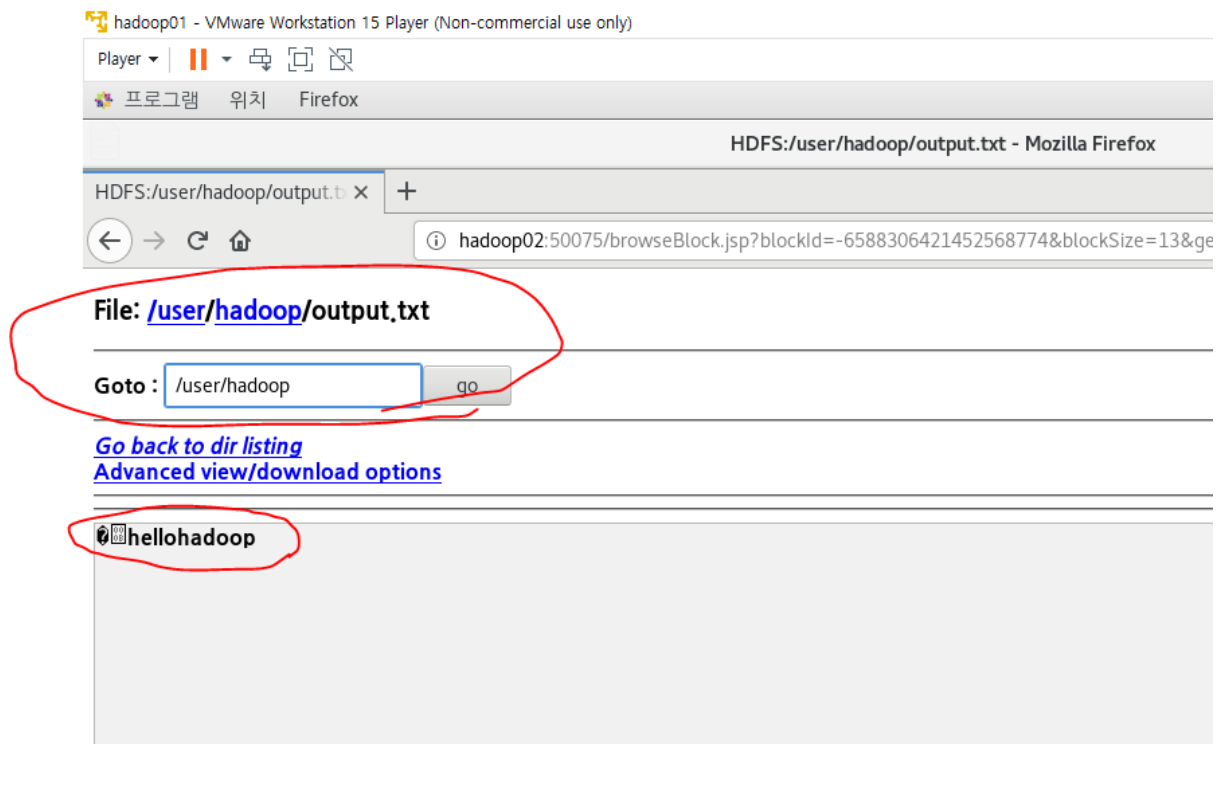
args[0] args[1]
명령어 매개변수

위 그림에서 output.txt를 보면, 경로를 정해주지 않았다. 경로를 정해주지 않았기에 새롭게 user라는 폴더가 생기고 그 안에 output.txt가 생기며, 명령행 매개변수 args[1]에 적은 hellohadoop이 적혀있음을 알 수 있다.



The screenshot shows the HDFS web interface at `hadoop04:50075/browseDirectory.jsp?namenodeInfoPort=50070&dir=/`. The 'Contents of directory /' section displays a table of files and directories. The 'user' directory is highlighted with a red circle.

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
home	dir				2020-02-14 20:25	rw-r--r--	hadoop	supergroup
input	dir				2020-02-17 13:54	rw-r--r--	hadoop	supergroup
myinput	dir				2020-02-14 20:23	rw-r--r--	hadoop	supergroup
myoutput	dir				2020-02-14 20:25	rw-r--r--	hadoop	supergroup
output	dir				2020-02-14 19:46	rw-r--r--	hadoop	supergroup
user	dir				2020-02-17 15:13	rw-r--r--	hadoop	supergroup
wordcount_output	dir				2020-02-17 13:54	rw-r--r--	hadoop	supergroup



writeUTF 말고, Read 하려면?

```

public static void main(String[] args) {
    // 1. hdfs를 제어하기 위해서 설정 파일을 읽어서 사용해야 하므로
    //   hadoop 설치 폴더의 설정 파일을 접근하기 위해 제공되는 클래스
    Configuration conf = new Configuration();

    // 2. HDFS를 접근하기 위해서 제공되는 객체를 생성
    FileSystem hdfs = null;

    // 3. HDFS에 input하기 위한 스트림객체
    FSDataInputStream hdfsout = null; //FileWriter fw = null; 과 같다.
    try {
        hdfs = FileSystem.get(conf); //일반경로 아니고 HDFS의 경로

        // 4. HDFS의 경로를 표현할 수 있는 객체
        //   => HDFS에 출력할 파일의 경로를 명령행 매개변수로 받아서 적용하겠다는 의미
        Path path = new Path(args[0]);
        //그 경로를 Path객체를 통해 접근할수 있게

        // 5. HDFS에 저장된 파일을 읽어야 하므로 input스트림 생성하기
        hdfsout = hdfs.open(path); //output할 수 있는 출력객체가 생김

        // 6. 입력스트림에 데이터를 출력하여 HDFS에 저장

        System.out.println(hdfsout.readUTF());
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

FSDataInputStream을 사용하고, open을 이용한다.

```

[hadoop@hadoop01 ~]$ ./hadoop-1.2.1/bin/hadoop jar multi-hadoop-examples.
jar hdfs.exam.HDFSExam01 output.txt
hellohadoop

```

실행 시 output.txt에 쓰여진 hellohadoop이 출력됨을 알 수 있다.

Input 과 output 동시에 해보기

```

1 package hdfs.exam;
2
3 import java.io.IOException;
4
5 public class HDFSCTest {
6
7     public static void main(String[] args) {
8         Configuration conf = new Configuration();
9
10        FileSystem hdfs = null;
11
12        FSDataOutputStream hdfsout = null;
13        FSDataInputStream hdfsin = null;
14        try {
15            hdfs = FileSystem.get(conf);
16
17            Path pathin = new Path("output.txt");
18            hdfsin = hdfs.open(pathin);
19
20            Path pathout = new Path(args[0]);
21            hdfsout = hdfs.create(pathout);
22
23            String str = hdfsin.readUTF();
24            hdfsout.writeUTF(str);
25            System.out.println(str);
26
27        } catch (IOException e) {
28            e.printStackTrace();
29        }
30    }
31 }

```