hadoop 4강 하둡분산파일시스템(HDFS) 명령어와 자바 코딩 실습

3. 하둡 분산 파일 시스템(HDFS)

가. 하둡 분산 파일 시스템(Hadoop Distributed File System)

1) 대용량의 파일을 분산된 서버에 저장하고 저장된 데이터를 빠르게 처리할 수 있게 하는 파일 시스템

2) 저사양의 서버를 이용해서 구성할 수 있음

3) 블록 구조의 파일 시스템
   블록의 크기는 기존에는 64MB였으나 하둡 2.0부터 128MB로 증가됨

나. 하둡 분산 파일 시스템의 시작과 종료

1) 하둡분산파일시스템을 시작하는 명령어
   start-dfs.sh - HDFS를 구동함(NameNode, SecondaryNameNode, DataNode가 실행됨)
   start-yarn.sh - 맵리듀스를 구동함(master에서는 ResourceManager와 NodeManager가 실행되고, slave에서는 NodeManager가 실행됨)

```
start-dfs.sh
start-yarn.sh
```

2) 하둡분산파일시스템을 중지시키는 명령어

```
stop-dfs.sh
stop-yarn.sh
```

다. 하둡 분산 파일 시스템의 주요 명령어

형식) hdfs dfs -명령어 옵션

dfs : Distributed Filesystem Shell (분산 파일시스템 쉘)

1) 하둡 분산 파일시스템 도움말

```
hdfs dfs -help
```

세부적인 도움말
ls 명령어에 대한 도움말

```
hdfs dfs -help ls
```

2) 파일목록 보기

ls (LiSt)

hdfs dfs -ls [디렉토리]

경로를 지정하지 않을 경우 사용자 계정의 홈디렉토리 조회

```
hdfs dfs -ls
```

경로를 지정할 경우 해당 디렉토리의 파일 목록 조회

```
hdfs dfs -ls /input
```

-R 옵션 : 하위 디렉토리 목록까지 출력

```
hdfs dfs -ls -R /
```

대부분 하둡 분산파일 시스템 명령어는 linux명령어와 비슷하다.

```
파일(F)  편집(E)  보기(V)  검색(S)  터미널(T)  도움말(H)
[root@master ~]# hdfs dfs -ls
Found 1 items
drwxr-xr-x   - root supergroup          0 2019-01-04 11:29 c
onf
[root@master ~]# hdfs dfs -ls /conf
ls: `/conf': No such file or directory
[root@master ~]# hdfs dfs -ls conf
[root@master ~]# hdfs dfs -ls /input
Found 1 items
-rw-r--r--   2 root supergroup       1366 2019-01-04 11:30 /
input/README.txt
[root@master ~]#
```

```
[root@master ~]# hdfs dfs -ls /
Found 4 items
drwxr-xr-x   - root supergroup          0 2019-01-04 11:30 /
input
drwxr-xr-x   - root supergroup          0 2019-01-04 11:31 /
root
drwx------   - root supergroup          0 2019-01-04 11:31 /
tmp
drwxr-xr-x   - root supergroup          0 2019-01-04 11:29 /
user
[root@master ~]#
```

3) 파일용량 확인

du (Directory Usage) 디렉토리 또는 파일의 사용량을 바이트 단위로 출력

```
hdfs dfs -du /
```

du -s : 해당 디렉토리의 전체용량만 출력

```
hdfs dfs -du -s /
```

4) 파일내용 보기 : cat, text

cat (텍스트파일 전용)

text (텍스트파일과 압축파일도 읽을 수 있음)

hdfs dfs -cat 디렉토리/파일

```
hdfs dfs -cat /input/README.txt
```

hdfs dfs -text 디렉토리/파일

```
hdfs dfs -text /input/README.txt
```

텍스트 내용 출력(압축된 파일도 읽을 수 있음)

5) 디렉토리 생성

mkdir (MaKe DIRectory)

hdfs dfs -mkdir [디렉토리]

```
hdfs dfs -mkdir test

hdfs dfs -ls
```

```
Found 2 items
drwxr-xr-x   - root supergroup          0 2018-01-29 20:27 conf
drwxr-xr-x   - root supergroup          0 2018-01-29 20:49 test
```

6) 파일 복사

hdfs dfs -put 로컬시스템파일 하둡분산파일시스템파일

$HADOOP_HOME/etc/hadoop 폴더의 core-site.xml 파일을 하둡분산파일시스템의 test 폴더에 복사

```
hdfs dfs -put $HADOOP_HOME/etc/hadoop/core-site.xml test
```

복사된 파일 확인

```
hdfs dfs -ls test
```

hdfs dfs -copyFromLocal 로컬시스템파일 하둡분산파일시스템파일

로컬의 core-site.xml을 하둡분산파일시스템의 test 폴더에 a라는 파일이름으로 저장

```
hdfs dfs -copyFromLocal $HADOOP_HOME/etc/hadoop/core-site.xml test/a
```

test 디렉토리의 파일 목록 확인

hdfs dfs -put $HADOOP_HOME/etc/hadoop/core-site.xml test
란 명령어는
Local에 있는 다음과 같은 파일을
$HADOOP_HOME/etc/hadoop/core-site.xml
HDFS에
test폴더에 넣겠다는 뜻이다.

hdfs dfs -copyFromeLocal $HADOOP_HOME/etc/hadoop/core-site.xml test/a
란, 위와 같은 명령어지만, a라는 파일명으로 복사하겠다는 뜻이다.

test 디렉토리의 파일 목록 확인

```
hdfs dfs -ls test
```

hdfs dfs -get 하둡분산파일시스템파일 로컬시스템파일

하둡분산파일시스템의 파일 a를 로컬시스템의 현재폴더에 a라는 파일이름으로 복사

```
hdfs dfs -get test/a a
```

```
ls -la
```

hdfs dfs -copyToLocal 하둡분산파일시스템파일 로컬시스템파일
하둡분산파일시스템의 a 파일을 로컬의 b파일로 복사

```
hdfs dfs -copyToLocal test/a b
```

```
cat b
```

-로컬시스템으로 한개의 파일로 합쳐서 복사
hdfs dfs -getmerge 하둡분산파일시스템파일 로컬시스템파일

반대로, HDFS에 있는 파일을 local에 b라는 파일로 복사하겠다는 뜻이다.

get과 같은 명령어가 copyToLocal이다.

하둡의 test 디렉토리의 모든 파일을 로컬의 c 파일로 합쳐서 저장

```
hdfs dfs -getmerge test c
```

```
cat c
```

test폴더에 있는 파일들을 c라는 파일로 합쳐서 저장하겠다.

-파일 복사
hdfs dfs -cp 하둡분산파일시스템소스파일 하둡분산파일시스템복사파일

하둡분산파일시스템의 test 디렉토리의 core-site.xml 파일을 하둡분산파일시스템의 /user/사용자 디렉토리의 core-site-copy.xml 로 복사

```
hdfs dfs -cp test/core-site.xml core-site-copy.xml
```

```
hdfs dfs -ls

또는

hdfs dfs -ls /user/root
```

하둡에 있는 파일을 하둡에 있는 파일로 복사한다.
HDFS내에서의 복사.

경로의 default는 /user/root

7) 파일 이동

hdfs dfs -mv 이동전경로 이동후경로
하둡분산파일시스템 내에서 파일을 옮김

하둡분산파일시스템의 test 폴더 안의 a파일을 test 폴더 안의 a2 파일로 옮김

```
hdfs dfs -mv test/a test/a2
```

```
hdfs dfs -ls test
```

hdfs dfs -moveFromLocal 로컬경로 하둡경로
로컬시스템에서 하둡분산파일시스템으로 옮김(로컬시스템의 파일은 삭제됨)

```
hdfs dfs -mkdir temp
```

로컬 현재 폴더의 a 파일을 하둡분산파일시스템의 temp 폴더의 a파일로 옮김

```
hdfs dfs -moveFromLocal a temp/a
```

```
hdfs dfs -ls temp
```

8) 파일/디렉토리 삭제

hdfs dfs -rm [디렉토리]/[파일]

```
hdfs dfs -rm temp/a
```

```
hdfs dfs -ls temp
```

hdfs dfs -rm -r 디렉토리
하위 디렉토리까지 삭제

```
hdfs dfs -rm -r test
```

```
hdfs dfs -ls test
```

9) 카운트값 조회

hdfs dfs -count [디렉토리]/[파일]

```
hdfs dfs -count /
```

디렉토리갯수	파일갯수	파일사이즈	
16	17	233089	/

```
[root@master ~]# hdfs dfs -count /
          15            6             232978 /
[root@master ~]#
```

10) 파일의 마지막 내용 확인

hdfs dfs -tail [파일]

```
hdfs dfs -tail /input/README.txt
```

```
[root@master ~]# hdfs dfs -put $HADOOP_HOME/etc/hadoop/core-
site.xml   /input/test
[root@master ~]# hdfs dfs -ls /input
Found 1 items
-rw-r--r--   2 root supergroup        880 2019-01-07 09:21 /
input/test
[root@master ~]# hdfs dfs -tail /input/test
<?xml version="1.0" encoding="UTF-8"?>
```

파일이 너무 크면 뒷부분만 잘라서 보여주는 명령어 이다.

11) 권한 변경

hdfs dfs -chmod [권한옵션] [디렉토리/파일]

해당 디렉토리 또는 파일의 권한을 변경함

```
hdfs dfs -chmod 777 conf
```

conf 디렉토리의 권한이 777로 바뀐 것을 확인

```
hdfs dfs -ls
```

```
rwxrwxrwx   - root supergroup          0 2018-01-29 20:27 conf
-rw-r--r--  2 root supergroup       4684 2018-01-29 21:03 hadoop-env-copy.sh
drwxr-xr-x  - root supergroup          0 2018-01-29 21:08 temp
```

유닉스 시스템의 파일권한 - 10자리 문자

소유자권한/그룹권한/전체권한

777 (모든 권한)

555

444

ex)

```
[root@master ~]# hdfs dfs -chmod -R 777  /input
[root@master ~]# hdfs dfs -ls /input
Found 1 items
-rwxrwxrwx   2 root supergroup        880 2019-01-07 09:21 /
input/test
[root@master ~]#
```

모든 사용자가 읽기쓰기 등등 모두 가능하게 된 것을 확인할 수 있다.

12) 0바이트 파일 생성

hdfs dfs -touchz 파일이름

```
hdfs dfs -touchz test.txt
```

```
hdfs dfs -ls
```

ex)

```
[root@master ~]# hdfs dfs -touchz test.txt
[root@master ~]# hdfs dfs -ls
Found 4 items
drwxr-xr-x   - root supergroup          0 2019-01-04 11:29 c
onf
-rw-r--r--   2 root supergroup        880 2019-01-04 13:28 c
ore-site.xml
drwxr-xr-x   - root supergroup          0 2019-01-04 13:34 t
emp
-rw-r--r--   2 root supergroup          0 2019-01-07 09:32 t
est.txt
[root@master ~]#
```

라. HDFS 입출력 실습 예제

1) Java 프로젝트 생성(윈도우즈의 이클립스에서 실행)

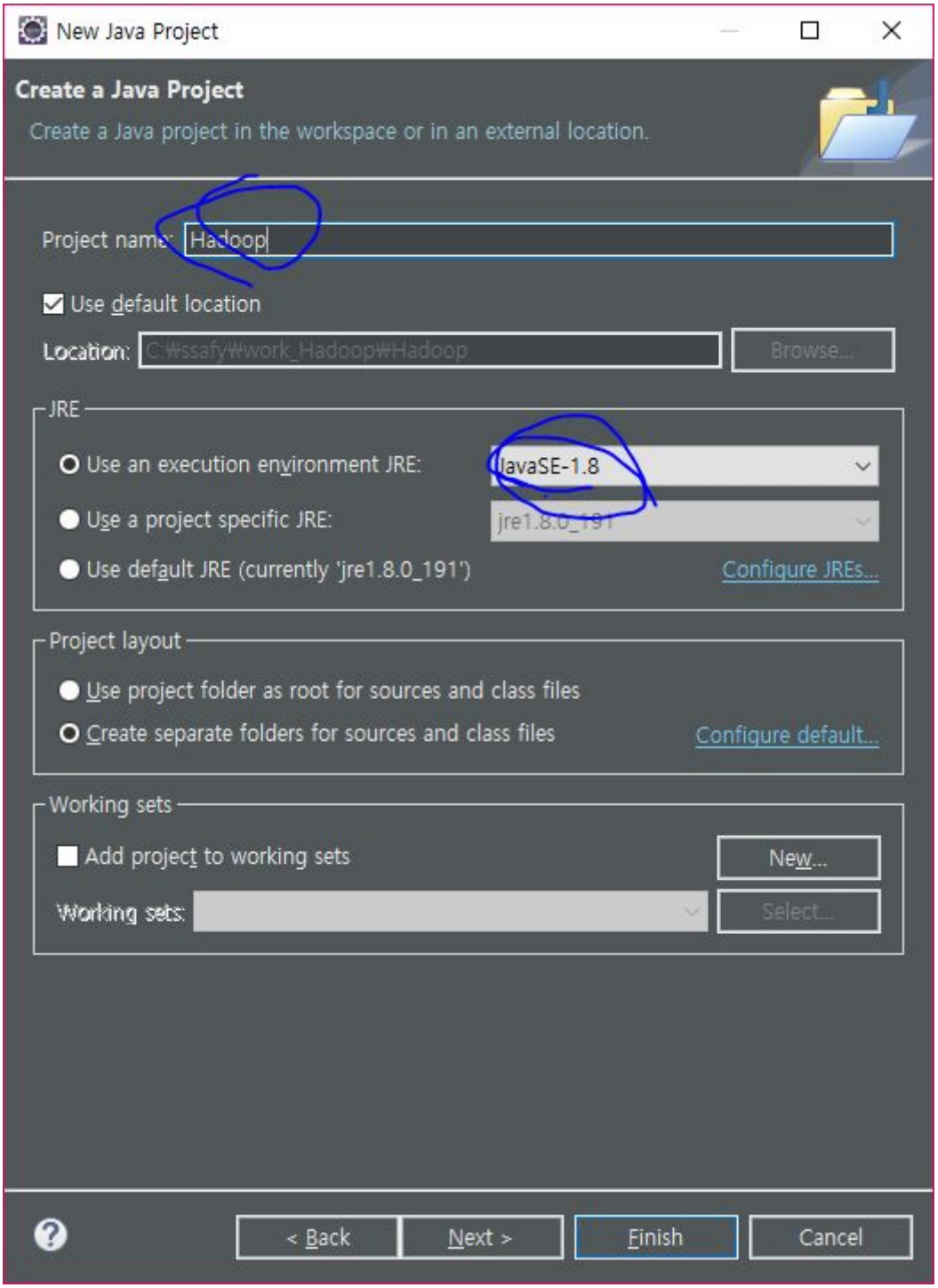
프로젝트 이름 : Hadoop
libs 디렉토리 생성
libs 디렉토리에 jar 추가
/home/centos/hadoop-2.9.2/share/hadoop/common 디렉토리의 jar 파일 3개
/home/centos/hadoop-2.9.2/share/hadoop/mapreduce 디렉토리의 jar 파일 9개

Project에서 우클릭한 후 Java Build Path의 Libraries 탭에서 Add JARs 버튼을 눌러서 9개의 jar 파일을 class path에 추가

하둡기반의 코드를 작성해서 테스트 할때는, java, 파이썬 등의 다양한 언어로 가능하다.
하지만 먼저 자바로 코딩하는 방법을 알아보도록하자.

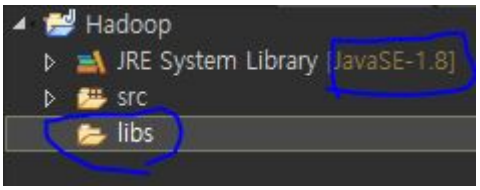
windows에서 elclipse를 실행하고, 프로젝트 이름은 중요하지 않습니다.
일단 Hadoop으로 만들고, 외부라이브러리를 저장할 libs디렉토리를 생성하고 jar파일을 추가한다.
총 12개의 파일을 추가할 것이다.

1.프로젝트생성

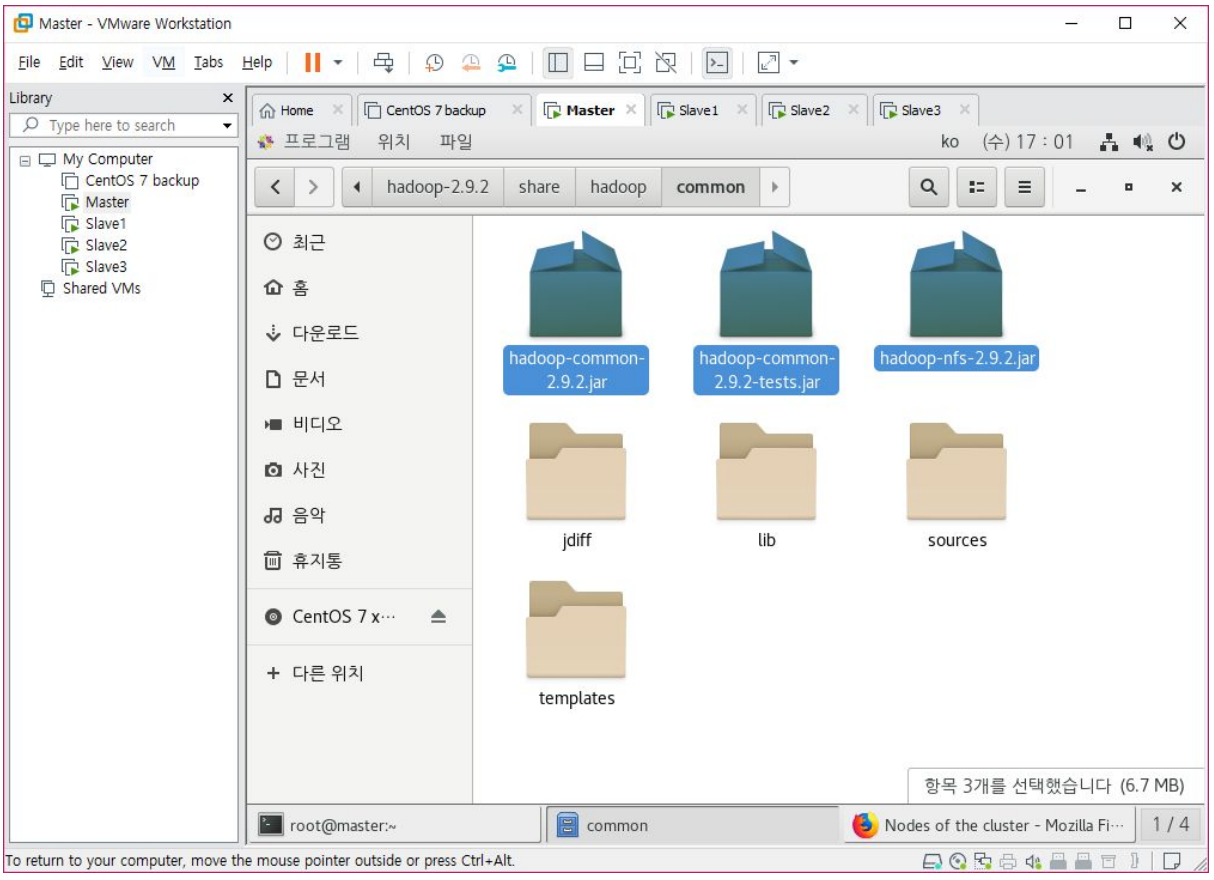


여기서 가장 중요한 부분은 Java의 버전이다. Hadoop버전에서 지원하는 java버전으로 프로젝트를 생성해야한다.

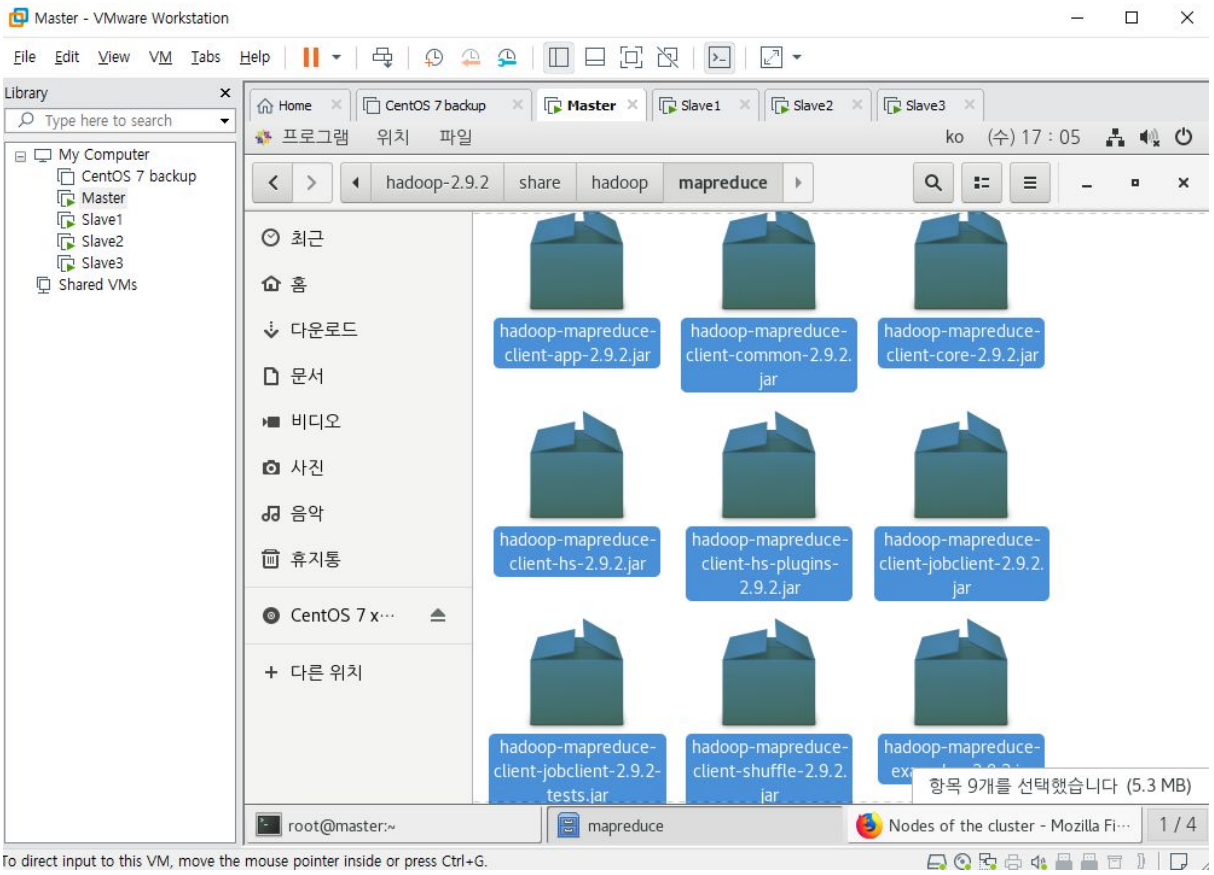
2.libs폴더를 생성한다.

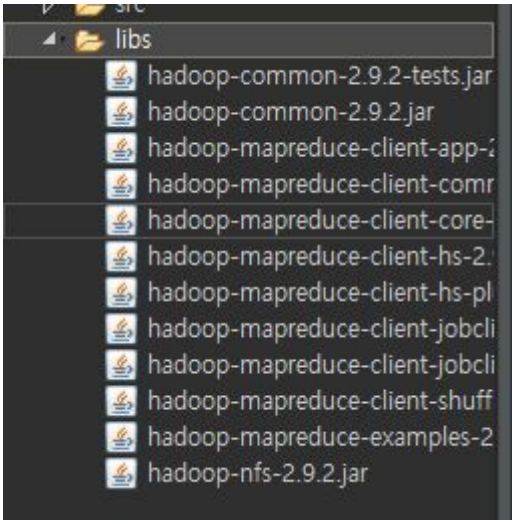


3.jar파일 복사해오기



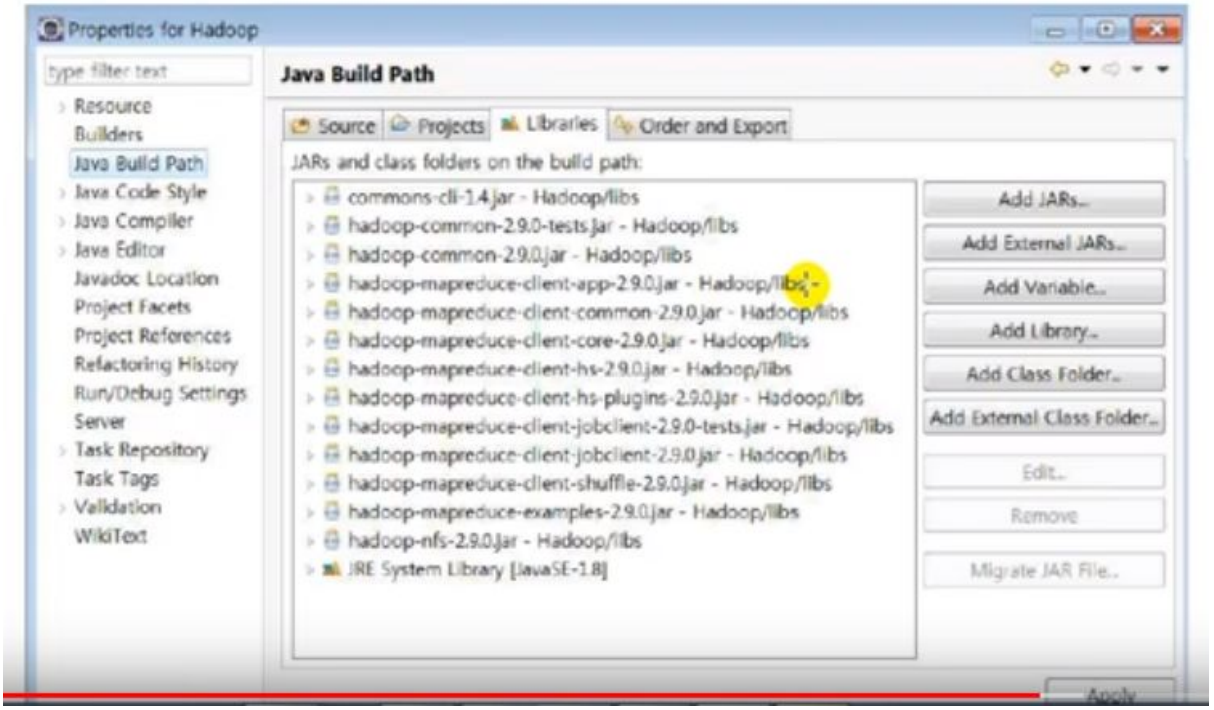
주의 사항) 파로 붙이면 해당 파일 시스템에 없다고 뜨기 때문에 바탕화면등 windows local filesystem에 복사한 후 다시 복사해서 붙인다.

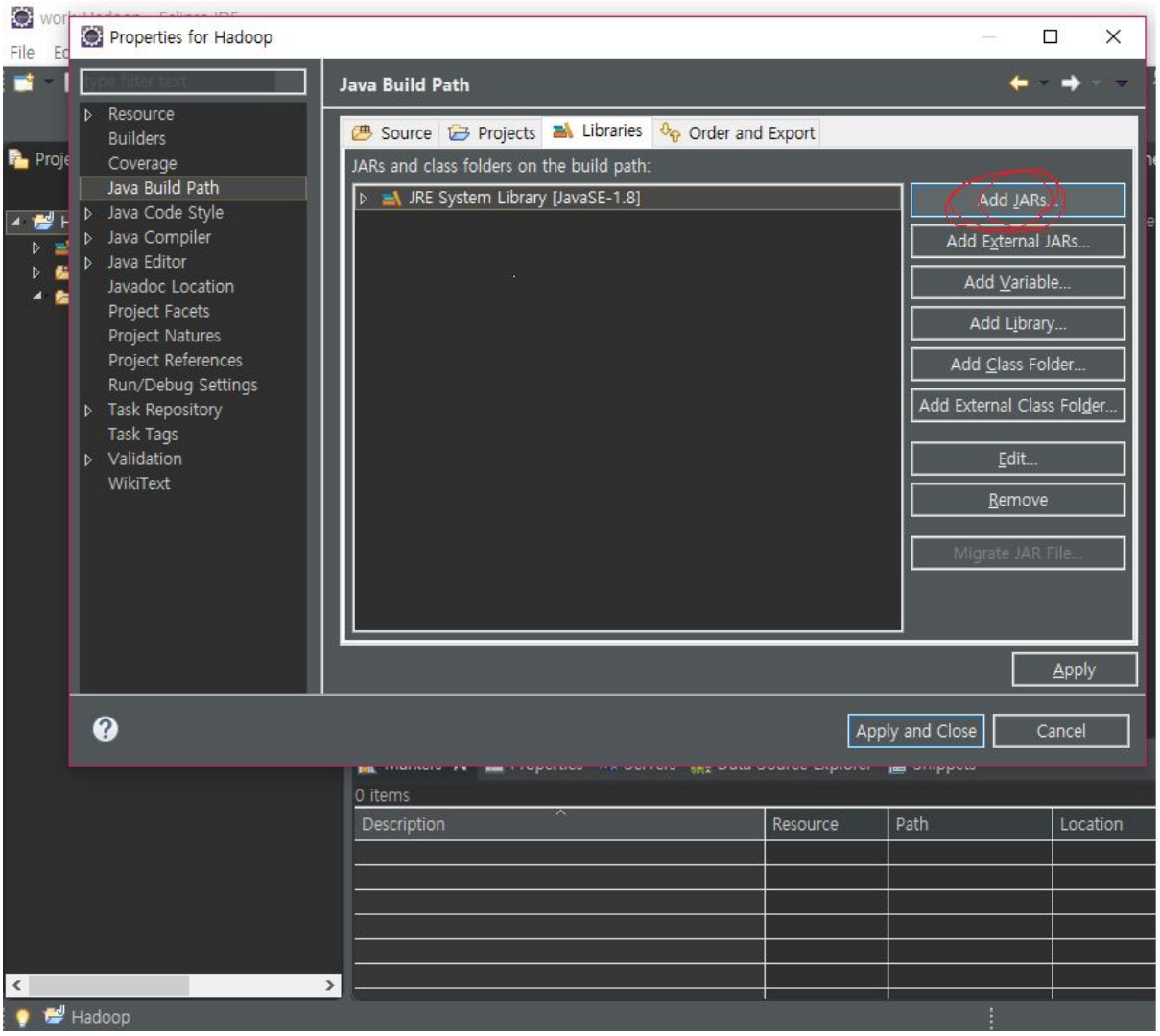


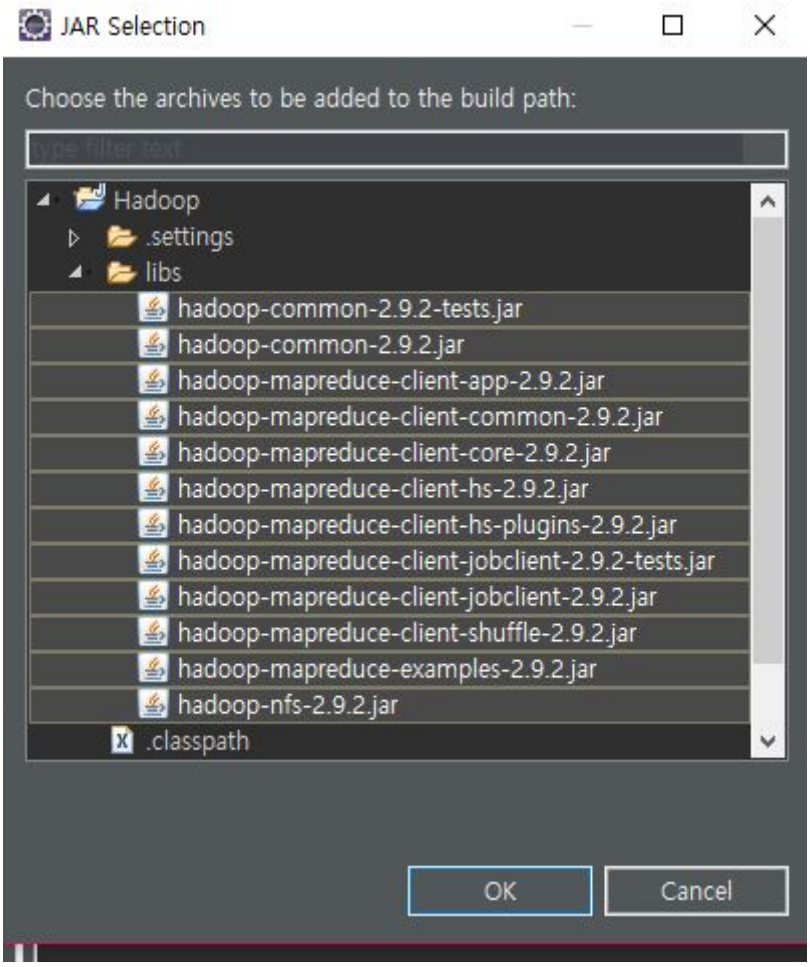


4.class path 추가

Project에서 우클릭한 후 Java Build Path의 Libraries 탭에서 Add JARs 버튼을 눌러서 9개의 jar 파일을 class path에 추가




Properties for Hadoop
Java Build Path
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project Facets
Project Natures
Project References
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
JRE System Library [JavaSE-1.8]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Apply
Apply and Close
Cancel
0 items
Description
Resource
Path
Location
Hadoop


JAR Selection
Choose the archives to be added to the build path:
type filter text
Hadoop
.settings
libs
hadoop-common-2.9.2-tests.jar
hadoop-common-2.9.2.jar
hadoop-mapreduce-client-app-2.9.2.jar
hadoop-mapreduce-client-common-2.9.2.jar
hadoop-mapreduce-client-core-2.9.2.jar
hadoop-mapreduce-client-hs-2.9.2.jar
hadoop-mapreduce-client-hs-plugins-2.9.2.jar
hadoop-mapreduce-client-jobclient-2.9.2-tests.jar
hadoop-mapreduce-client-jobclient-2.9.2.jar
hadoop-mapreduce-client-shuffle-2.9.2.jar
hadoop-mapreduce-examples-2.9.2.jar
hadoop-nfs-2.9.2.jar
.classpath
OK
Cancel

5.코드 작성(클래스)
hdfs라는 package에 HdfsFile이라는 파일을 생성한다.

2) hdfs.HdfsFile.java

```
package hdfs;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

public class HdfsFile {
    public static void main(String[] args) {
        // 입력 파라미터 확인
        if (args.length != 2) {
            System.err.println("사용 방법: HdfsFile <filename> <contents>");
            System.exit(2);
        }
```

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;


public class HdfsFile {
    //args 명령행 매개변수
    public static void main(String[] args) {
        //입력 파라미터 확인

        //2개가 아니면(입력, 출력파일)
        if(args.length !=2) {
            //에러 메시지 출력
            System.err.println("사용 방법 : HdfsFile <filename> <contents>");
            //프로그램 강제 종료
            System.exit(2);
        }

        try {
            //파일 시스템 제어 객체 생성
            Configuration conf = new Configuration();
            //하둡 분산 파일 시스템 객체
            FileSystem hdfs = FileSystem.get(conf);

            //경로체크
            Path path = new Path(args[0]);
            //경로가 존재하면
            if(hdfs.exists(path)) {
                //파일 삭제
                hdfs.delete(path, true);
            }

            //파일 저장
            FSDataOutputStream os = hdfs.create(path);
            os.writeUTF(args[1]);
            os.close();

            //파일 내용 읽기
            FSDataInputStream is = hdfs.open(path);
            String inputString = is.readUTF();
            is.close();

            //화면 출력
            System.out.println("Input Data:"+inputString);
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }
    }
}
```

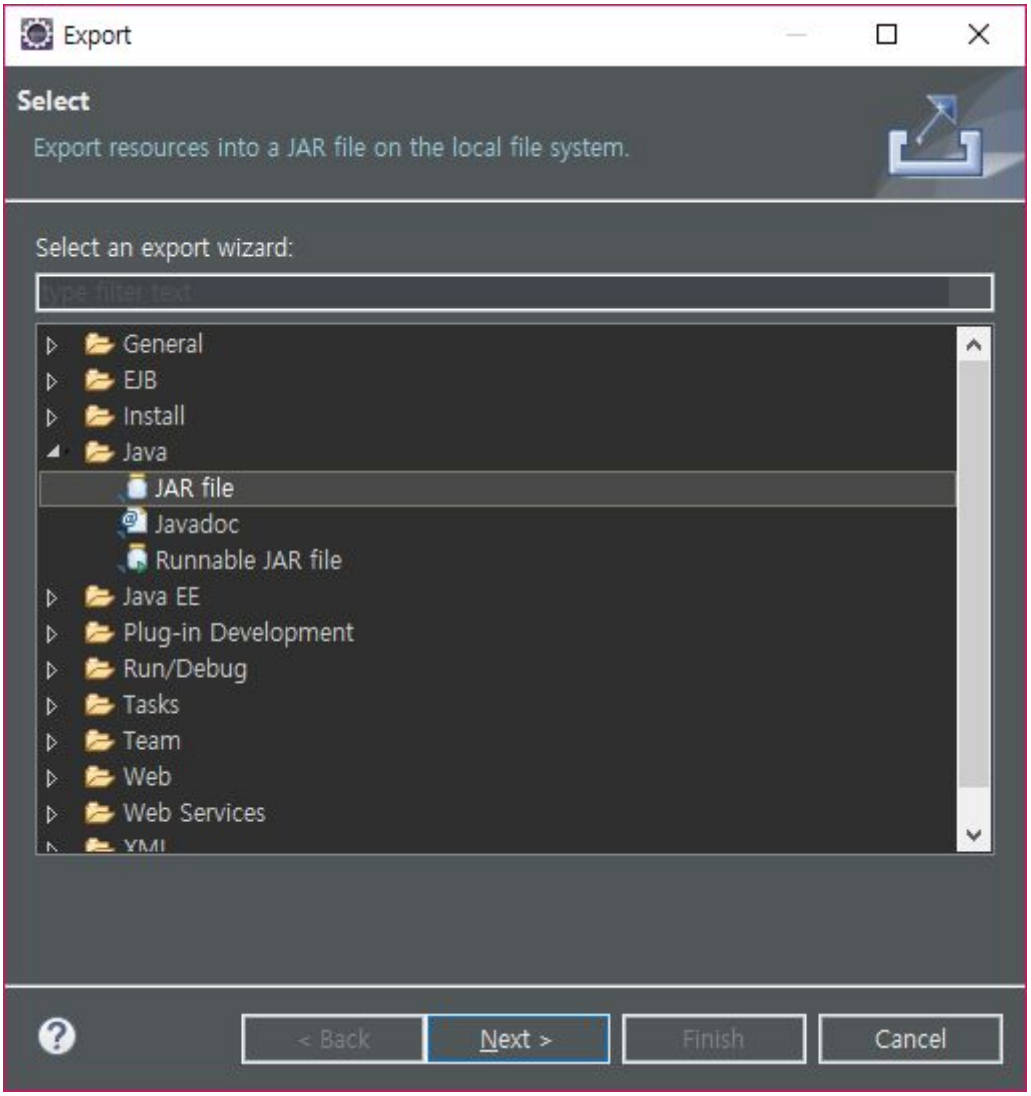
Local filesystem객체와 HDFS filesystem객체를 만든후,
입력 매개변수(1번째 매개변수)의 경로가 존재하면, 지우고 새로 만들어라.

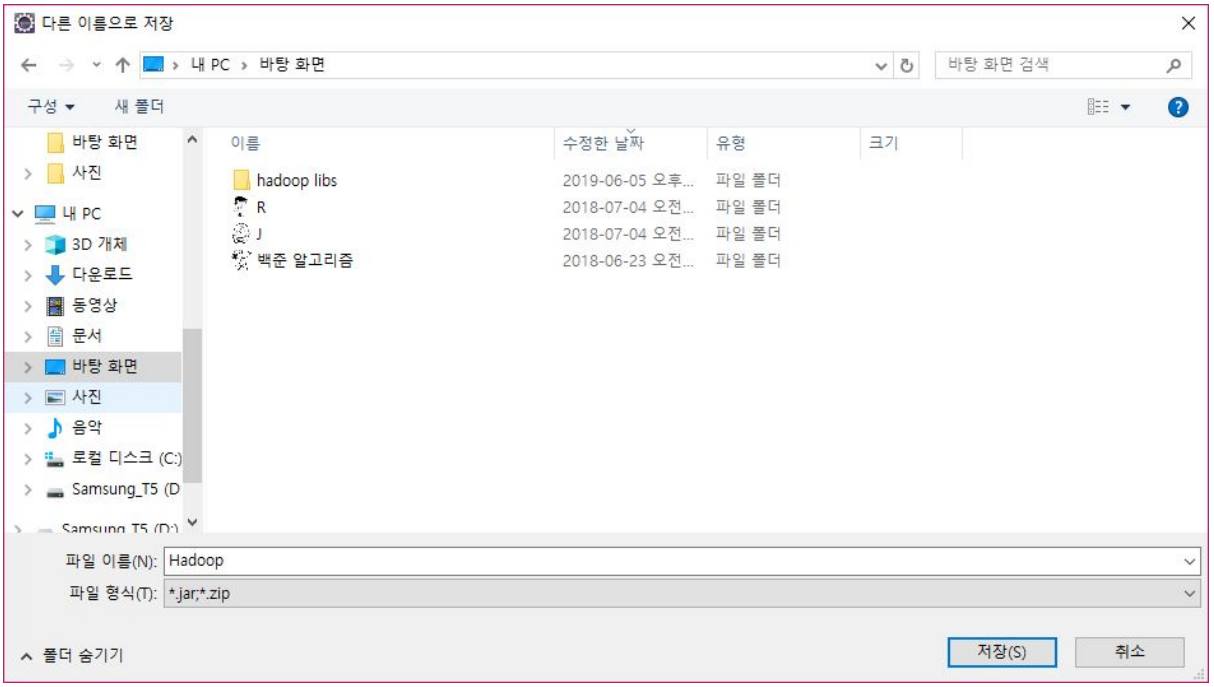
출력 매개변수로 파일을 저장한다.

저장된 파일 내용을 읽어서 화면에 출력한다.

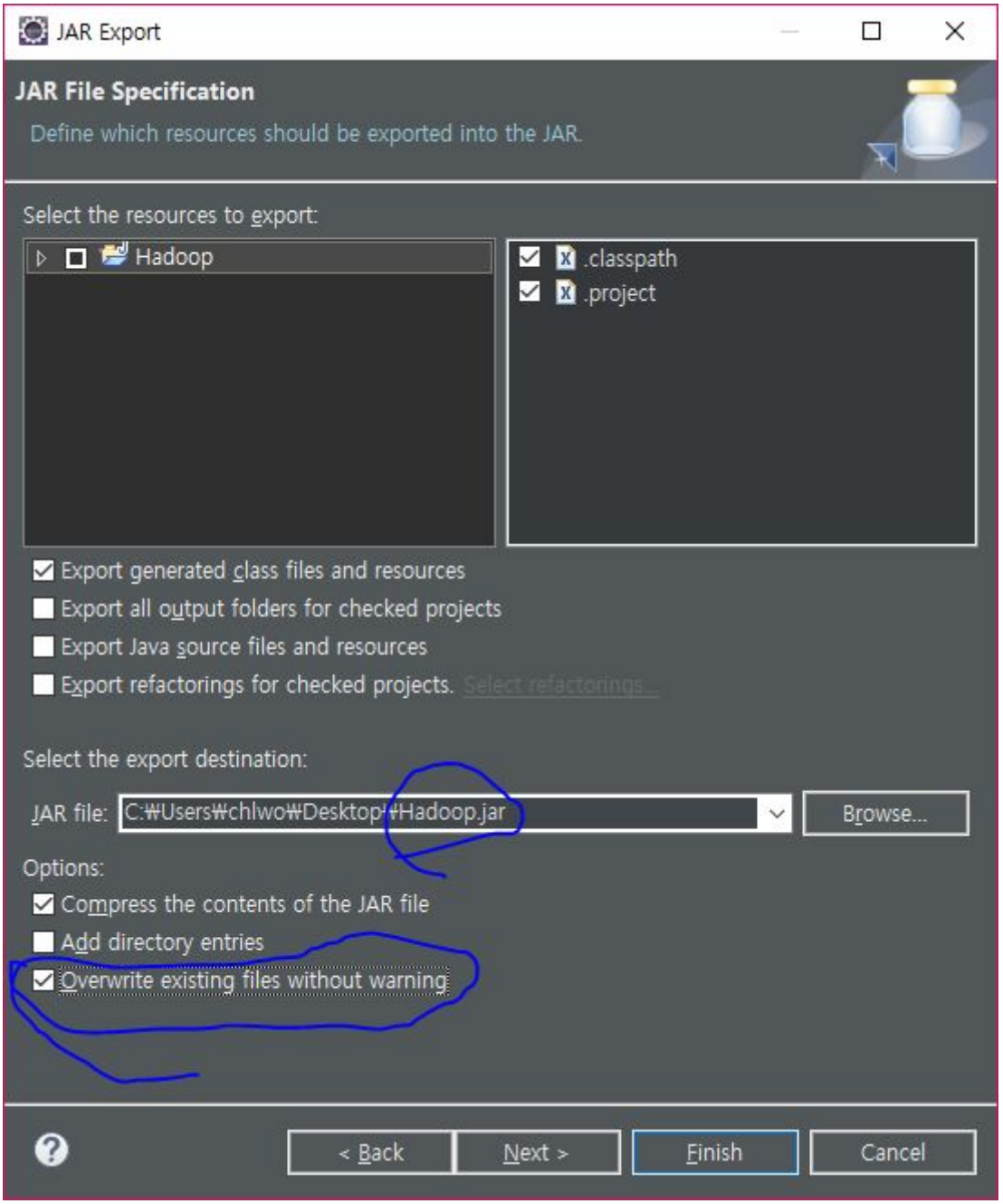
즉, 하둡 분산파일시스템에 이 java코드를 실행시키게되면, 자바에서 HDFS에 파일을 기록하고 읽어들이는 작업을 하는 것이다.

6.프로젝트를 jar파일로 export한다





파일이름을 Hadoop으로 하자.
파일이름을 나중에 코딩해야되기 때문에 꼭 기억해두어야한다.



계속 jar파일을 프로젝트에 코드를 추가하고 export할 것이기 때문에, 경고없이 jar파일을 덮어 쓰겠다는 것을 허용해둔다.

3) Hadoop.jar 파일로 export

리눅스 master node의 /home/centos/source 디렉토리에 복사

4) 실행

실행

```
hadoop jar /home/centos/source/Hadoop.jar  hdfs.HdfsFile input.txt "Hello, Hadoop"
```

파일 목록 확인

```
hdfs dfs -ls input.txt
```

실행 결과 확인

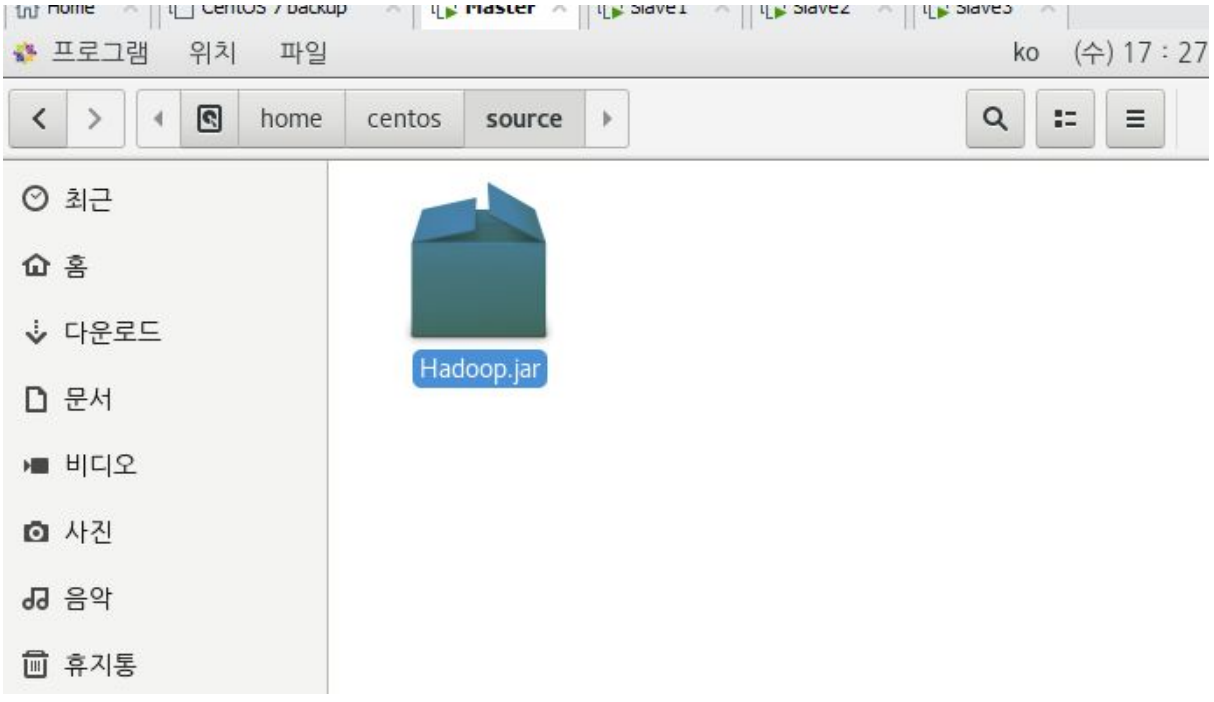
```
hdfs dfs -cat input.txt
```

커맨드 라인에서 입력한 문자열이 하둡분산파일시스템의 input.txt 파일에 저장되어 있음을 확인

디렉토리의 위치는 아무대나 해도되지만 일단 위와같이 /home/centos/source로 통일하도록 한다.

```
[root@master ~]# mkdir /home/centos/source
[root@master ~]#
```

디렉토리를 만들고
거기에 복사 붙여넣기를 한다.



```
[root@master ~]# ls -la /home/centos/source
합계 10696
drwxr-xr-x. 2 root   root         24  6월  5 17:27 .
drwx------. 5 centos centos      112  6월  5 17:26 ..
-rwxrw-rw-. 1 root   root    10952159  6월  5 17:27 Hadoop.jar
[root@master ~]#
```

4) 실행

실행

```
hadoop jar /home/centos/source/Hadoop.jar hdfs.HdfsFile input.txt "Hello, Hadoop"
```

파일 목록 확인

```
hdfs dfs -ls input.txt
```

실행 결과 확인

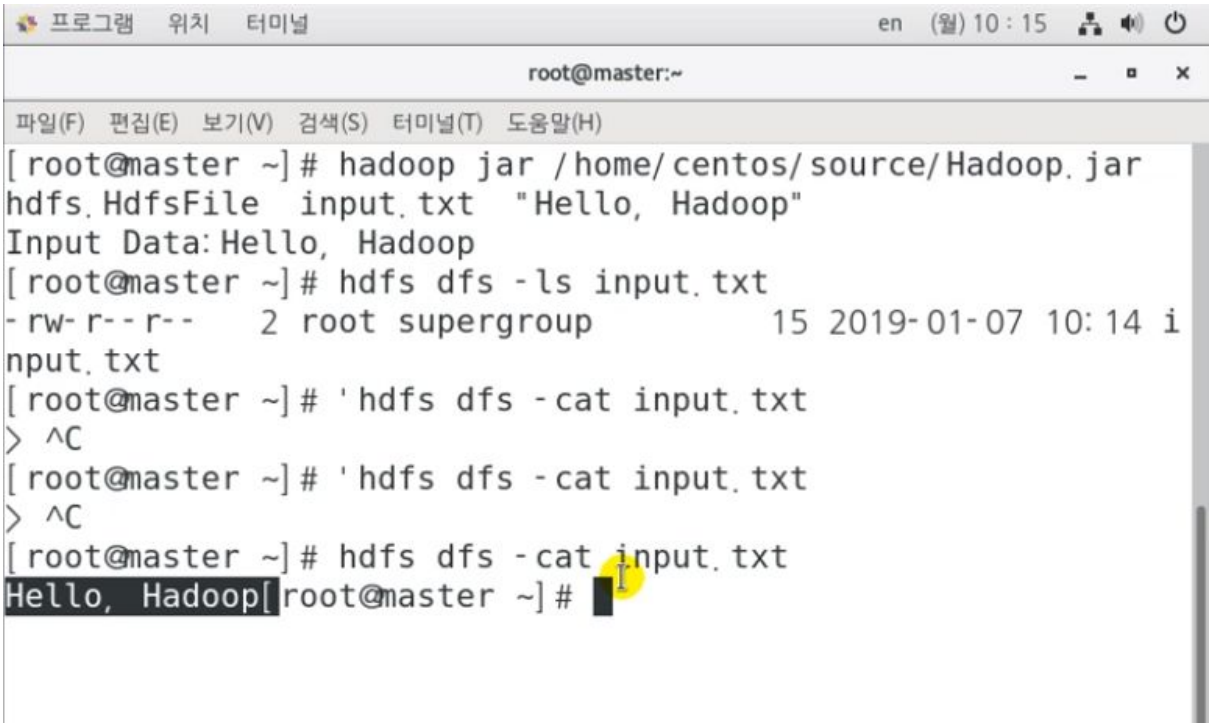
```
hdfs dfs -cat input.txt
```

커맨드 라인에서 입력한 문자열이 하둡분산파일시스템의 input.txt 파일에 저장되어 있음을 확인

hadoop jar /home/centos/source/Hadoop.jar hdfs.HdfsFile input.txt "Hello, Hadoop" 을 해석해보면
hadoop명령어를 칠껀데
jar파일을 통해 실행할 꺼야.
/home/centos/source/Hadoop.jar를 실행하면 되고
classpath는 패키지 경로를 포함해서, hdfs.HdfsFile이다.

input.txt라는 파일에 “Hello, Hadoop”을 입력해줘라.

```
[root@master ~]# hdfs dfs -ls input.txt
-rw-r--r--   2 root supergroup         15 2019-06-05 17:34 input.txt
[root@master ~]# hdfs dfs -cat input.txt
Hello, Hadoop[root@master ~]#
```



다음과 같이 결과가 나온 것을 확인 할 수 있다.