메이븐과 워드 카운트

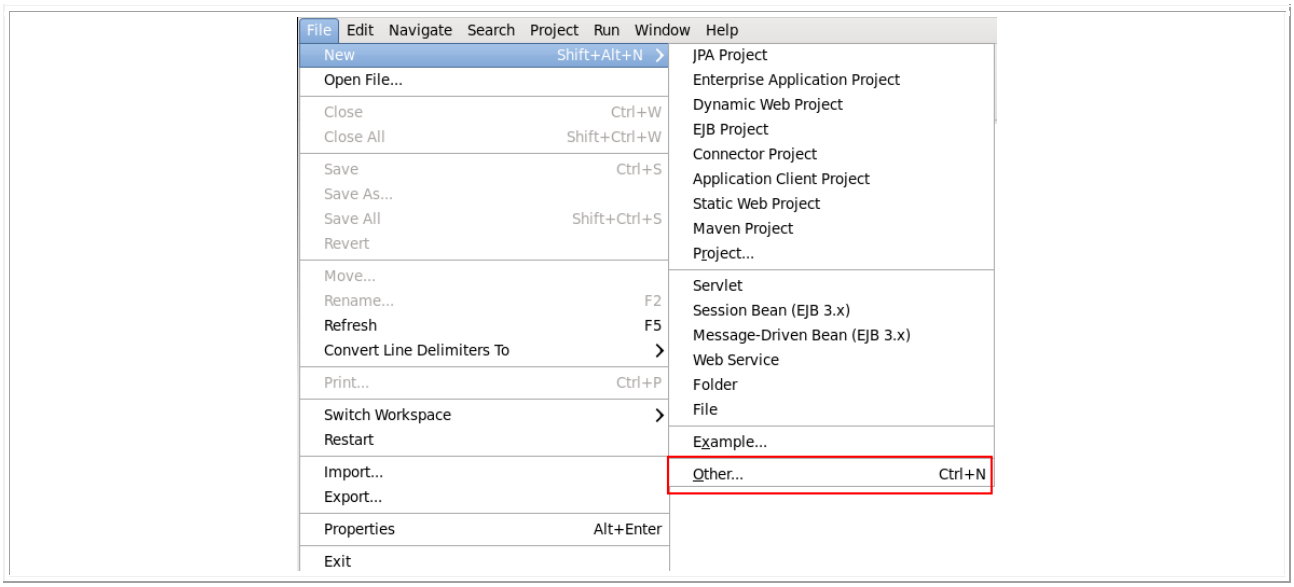
메이븐이란 관련된 라이브러리들을 서로 관리해주고, 의존성이 있는 경우 상위 라이브러리까지 알아서 사용자 홈디렉터리 아래 jar 파일을 모아 주기 위한 관리 툴이다.

참조 사이트 : http://repo1.maven.org/maven2/

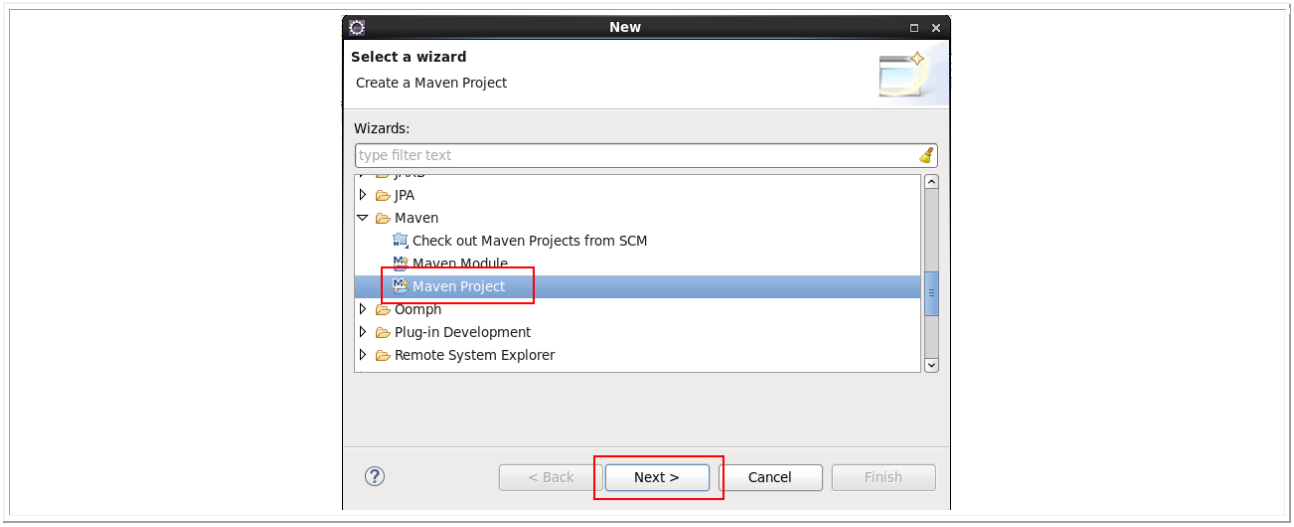
메이븐 프로젝트 생성하기

이클립스를 실행한다.
참고로 기본 워크 스페이스의 경로는 "/home/hadoop/workspace"이다.
다음 순서대로 메이븐 프로젝트를 생성한다.

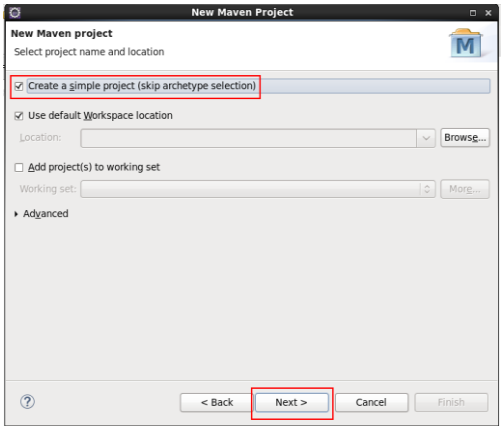
풀다운 메뉴 [File] - [Other]를 클릭하여 Maven 프로젝트를 선택하도록 한다.



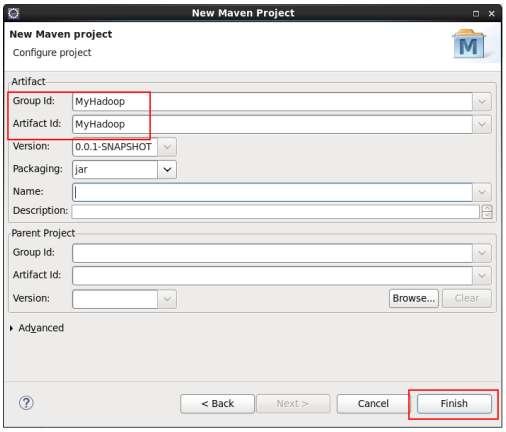
Maven 폴더 하위에 [Maven Project] 항목을 클릭한다.



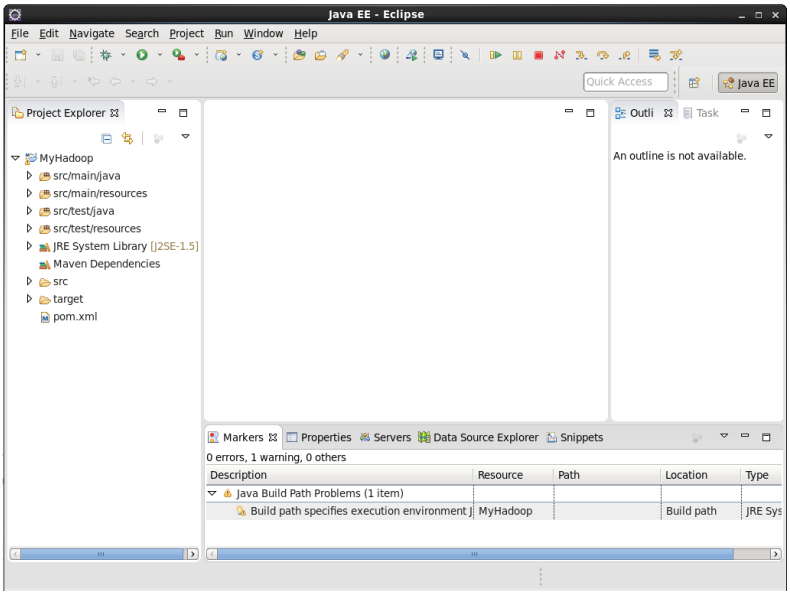
그림과 같이 첫 번째 체크 박스를 체크하고 [Next] 버튼을 클릭한다.



GroupId 와 Artifact Id 를 입력하고 [Finish] 버튼을 클릭한다.
설정하는 데 약간의 시간이 걸린다.



다음과 같이 프로젝트가 생성이 되었다.



실습 : pom.xml 파일 작성

pom.xml 파일은 관련 library 파일들을 다운 로드 받아서 로컬 디렉토리에 설치하기 위한 설정 파일이다.

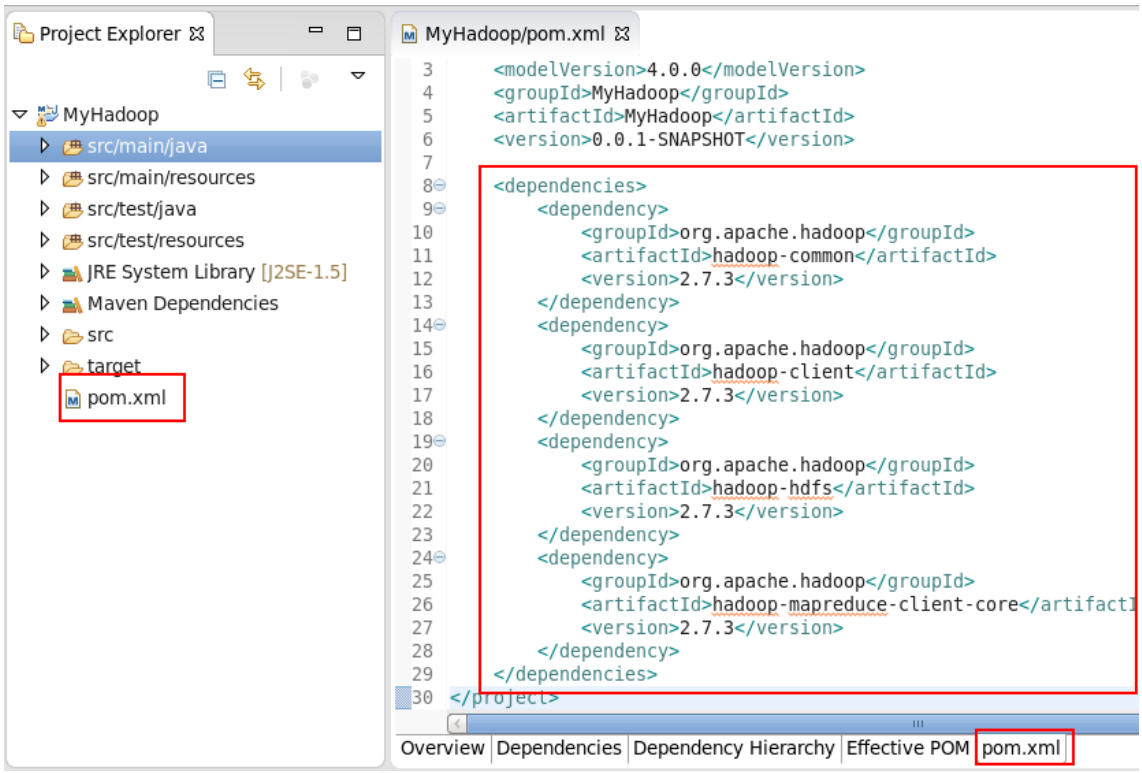
pom.xml에 다음을 입력하고 파일을 저장한다.
저장하고 나면 시간이 다소 걸린다.
이유는 관련 jar 라이브러리들을 읽어 들이기 때문이다.

코드

```
<dependencies>
	<dependency>
		<groupId>org.apache.hadoop</groupId>
		<artifactId>hadoop-common</artifactId>
		<version>2.7.3</version>
	</dependency>

	<dependency>
		<groupId>org.apache.hadoop</groupId>
		<artifactId>hadoop-client</artifactId>
		<version>2.7.3</version>
	</dependency>

	<dependency>
		<groupId>org.apache.hadoop</groupId>
		<artifactId>hadoop-hdfs</artifactId>
		<version>2.7.3</version>
	</dependency>

	<dependency>
		<groupId>org.apache.hadoop</groupId>
		<artifactId>hadoop-mapreduce-client-core</artifactId>
		<version>2.7.3</version>
	</dependency>
</dependencies>
```

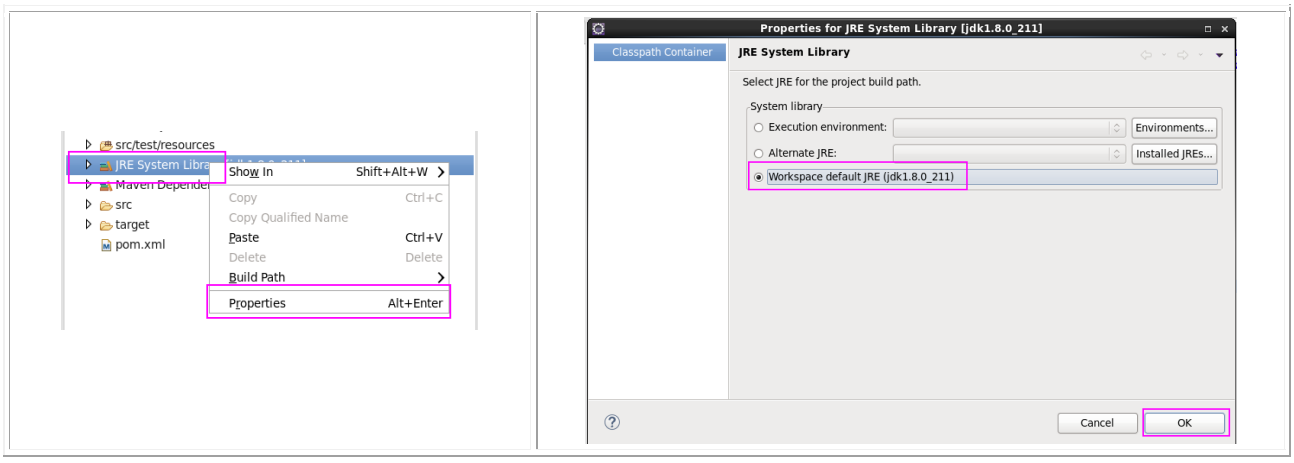
pom.xml 파일 내용 확인

다음 화면은 pom.xml 파일에 코딩이 완료된 그림이다.



실습 : jdk 버전 맞추기

메이븐을 설치하게 되면 기본 값으로 jdk 1.5버전에 맞춰진다.
이것을 시스템 버전에 맞추려면 다음과 같이 설정하면 된다.



WordCount 프로그램

WordCount는 주어진 입력 파일에 들어 있는 텍스트를 단어별로 나누어서 빈도수를 계산해주는 프로그램이다.
텍스트를 단어별로 나눌 때, 공백, tab과 newline과 같은 white-character를 기준으로 사용한다.
이클립스를 이용하여 WordCount 기능을 테스트 해보도록 한다.

myhadoop.wordcount 패키지를 설치하고 다음 파일들을 코딩한다.

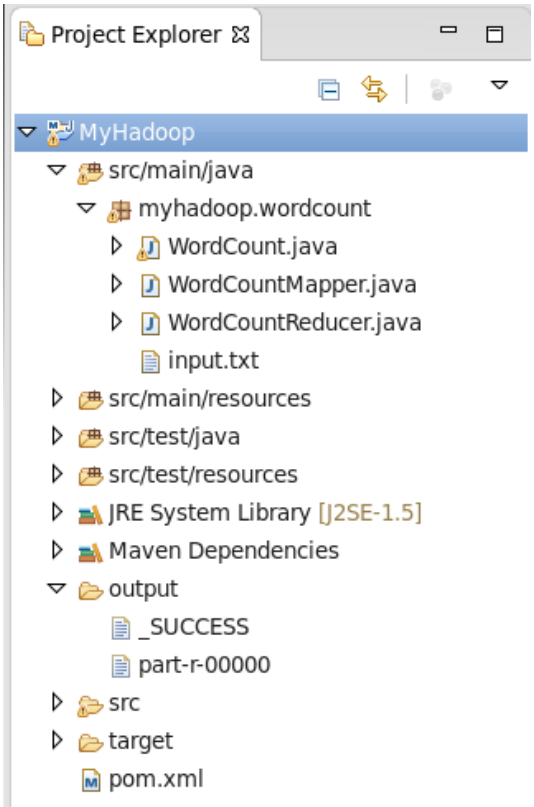
파일 현황

다음 파일들은 이번 실습에 필요한 파일들의 목록이다.

항목	설명
input.txt	워드 카운트를 테스트 하기 위한 텍스트 파일이다.
WordCount.java	워드 카운트 메인 프로그램 클래스 파일이다.
WordCountMapper.java	워크 카운트 매퍼 클래스 파일이다.
WordCountReducer.java	워크 카운트 리듀스 클래스 파일이다.
output 폴더	프로그램 실행 결과물이 저장될 폴더 이름이다.(임의의 이름 가능)

참조 그림

다음과 같이 프로젝트를 만들고 해당 클래스 파일 및 텍스트 파일들을 구현해 보도록 한다.



실습 : 테스트 용 텍스트 파일 만들기

다음과 같이 어떠한 단어들로 구성된 텍스트 파일을 작성하도록 한다.

소스 코드 : input.txt

```
brave lion
yellow lion
the lion ate the cow
now the lion is happy
```

데이터 타입 참조 사이트 :

참조 사이트

http://hadoop.apache.org/docs/r2.6.5/api/index.html

실습 : 매퍼 클래스 만들기

Mapper 클래스의 Generic 는 Mapper<LongWritable, Text, Text, IntWritable>이다.
입력되는 Key 는 해당 라인의 파일 오프셋(숫자 타입)이고, Value 는 한 줄(Text)이기 때문이다.
출력되는 Key 는 단어 1 개이고, Value 는 카운터 수(숫자 타입)이기 때문이다.

소스 코드 : WordCountMapper.java

```
package myhadoop.wordcount;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends
                Mapper<LongWritable, Text, Text, IntWritable> {

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(LongWritable key, Text value, Context context)
                        throws IOException, InterruptedException {
                StringTokenizer itr = new StringTokenizer(value.toString());
                while (itr.hasMoreTokens()) {
                        word.set(itr.nextToken());
                        context.write(word, one);
                }
        }
```

```
}
```

실습 : 리듀스 클래스 만들기

맵퍼의 출력 key와 value는 리듀서의 입력이 되어야 한다.
리듀서의 출력은 해당 단어와 합산된 단어의 빈도 수를 반환하면 된다.
Reducer<Text, IntWritable, Text, IntWritable>

소스 코드 : WordCountReducer.java

```
package myhadoop.wordcount;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends
                Reducer<Text, IntWritable, Text, IntWritable> {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
                        throws IOException, InterruptedException {
                int sum = 0;
                for (IntWritable val : values) {
                        sum += val.get();
                }
                result.set(sum);
                context.write(key, result);
        }
}
```

실습 : 워드 카운트 메인 프로그램 만들기

매퍼 클래스와 리듀스 클래스 작성 이후 다음과 같이 메인 워드 카운트 클래스를 작성하도록 한다.

소스 코드 : WordCount.java

```
package myhadoop.wordcount;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
```

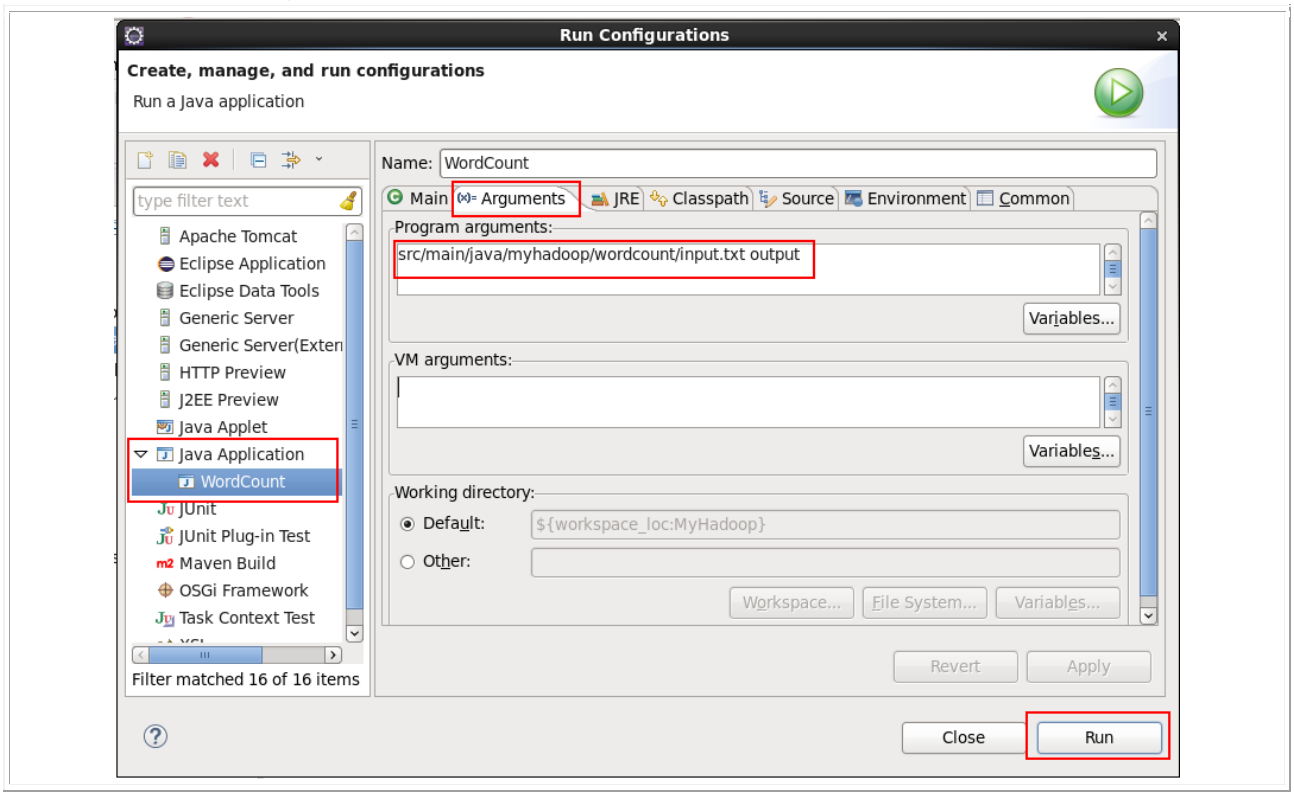
```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {
	public static void main(String[] args) throws Exception {
		Configuration conf = new Configuration();
		if (args.length != 2) {
			System.err.println("Usage: WordCount <input> <output>");
			System.exit(2);
		}
		Job job = new Job(conf, "WordCount");

		job.setJarByClass(WordCount.class);
		job.setMapperClass(WordCountMapper.class);
		job.setReducerClass(WordCountReducer.class);

		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);

		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);

		// 파일 시스템 제어 객체 생성
		FileSystem hdfs = FileSystem.get(conf);
		// 경로 체크
		Path path = new Path(args[1]);
		if (hdfs.exists(path)) {
			hdfs.delete(path, true);
		}
		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));

		job.waitForCompletion(true);
		System.out.println("finished");
	}
}
```

실습 : 명령행 매개 변수 입력하기

WordCount 프로그램은 명령행 매개 변수 2개를 요구한다.
WordCount 파일을 마우스 우측 클릭한 후, [Run As] - [Run Configurations]를 클릭한다.

Program Argument 입력란에 다음과 같은 형태로 입력한다.

전체경로/파일이름.확장자 출력될_폴더_이름
src/main/java/myhadoop/wordcount/input.txt output



확인 : WordCount 프로그램 실행 결과

실행이 정상적으로 완료 되고 나면 출력될 폴더(예시에서는 output) 아래에 part-r-00000 파일이 생성된다.
해당 파일 part-r-00000 을 열어 보면 결과는 다음과 같다.

출력 결과

```
ate     1
brave   1
cow     1
happy   1
is      1
lion    4
```

```
now     1
the     3
yellow  1
```

실습 : 이클립스의 모든 소스를 윈도우로 복사하기 :

현재 디렉토리는 MyHadoop 이라는 디렉토리라고 가정한다.
참고로 MyHadoop 디렉토리는 내가 진행중인 프로젝트 이름이다.

윈도우로 복사하기

```
[hadoop@master  MyHadoop]$  pwd
/home/hadoop/workspace/MyHadoop

labs 폴더에 source_code 라는 폴더가 있다고 가정한다.
다음과 같이 복사 명령어를 수행한다.
[hadoop@master  MyHadoop]$  cp  .  -r  /mnt/hgfs/shared/source_code/
```

TokenizerMapper.java

```
cd  /usr/local/hadoop

hadoop  fs  -mkdir  /input

hadoop  fs  -mkdir  /output

#  HDFS 위에 input이란 디렉토리를 만들고, 거기에 input.txt 파일을 복사한다.
hadoop  fs  -copyFromLocal  input.txt  /input

최종적으로 /input 디렉토리의 내용을 확인한다.
hadoop  fs  -ls  /input
```