# Big Data Hadoop Training

Session 3 Assignment 1 Solution:

**Task 1:**

**Q) Write a Java program, to take an HDFS Path as input and display all the files and sub-directories in that HDFS path.**

A) **Java Program**: Takes HDFS Path as input and display all the files and sub-directories in that HDFS path

(**Note:** Assuming the HDFS path provided as input has Read permission for displaying the content of the directory)

import java.io.IOException;

import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileStatus;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.FileUtil;

import org.apache.hadoop.fs.Path;

public class list\_all\_files\_in\_hdfs\_recursively{

public static void main(String[] args) throws IOException, URISyntaxException

{

//Single HDFS path will be given as input – accessed using args[0] by main method

//1. Get the file path instance

Path filepath = new Path(args[0]);

//2. Get the Configuration instance

Configuration configuration = new Configuration();

//3. Get the instance of the HDFS - file system

FileSystem filesystem = FileSystem.get(filepath.toUri(), configuration);

//4. Get the metadata of the desired directory

FileStatus[] fileStatus = filesystem.listStatus(filepath);

//5. Using FileUtil, getting the Paths for all the FileStatus

Path[] paths = FileUtil.stat2Paths(fileStatus);

//6. Iterate through the directory and display the files in it

System.out.println("\*\*\*\*\* Contents of the Directory \*\*\*\*\*");

for(Path path : paths)

{

System.out.println(path);

}

}

}

**Input format: “**hdfs://<hostname>:<port\_no>/<HDFS\_File\_Path>**”**

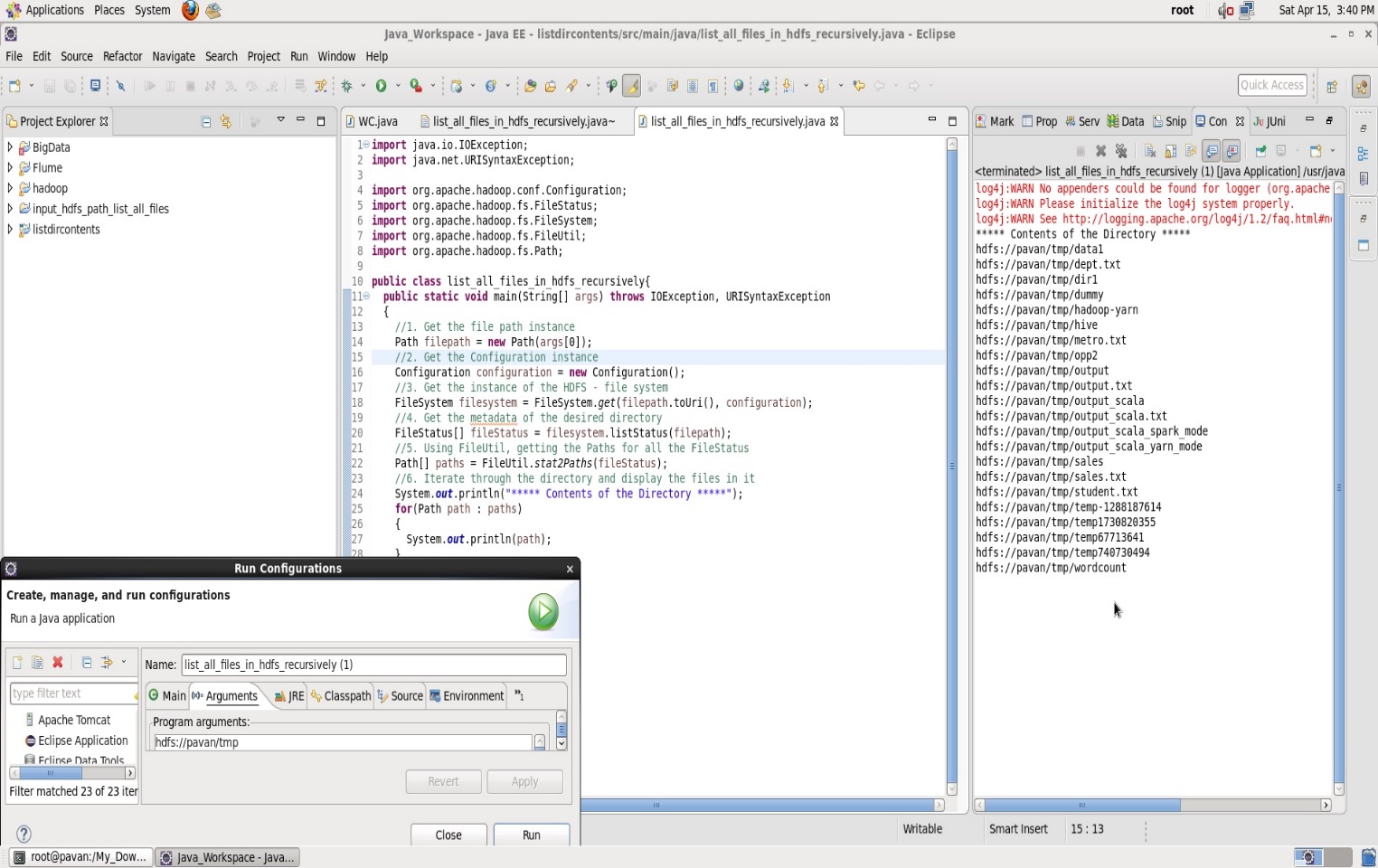
**Ex: “**hdfs://pavan/tmp**” here, hostname :** pavan , **port\_no: 8020 specified in core-site.xml HDFS\_File\_Path :** tmp

**Running in Eclipse :**

* Input given in Run Configurations Argument Tab as : “hdfs://pavan/tmp”

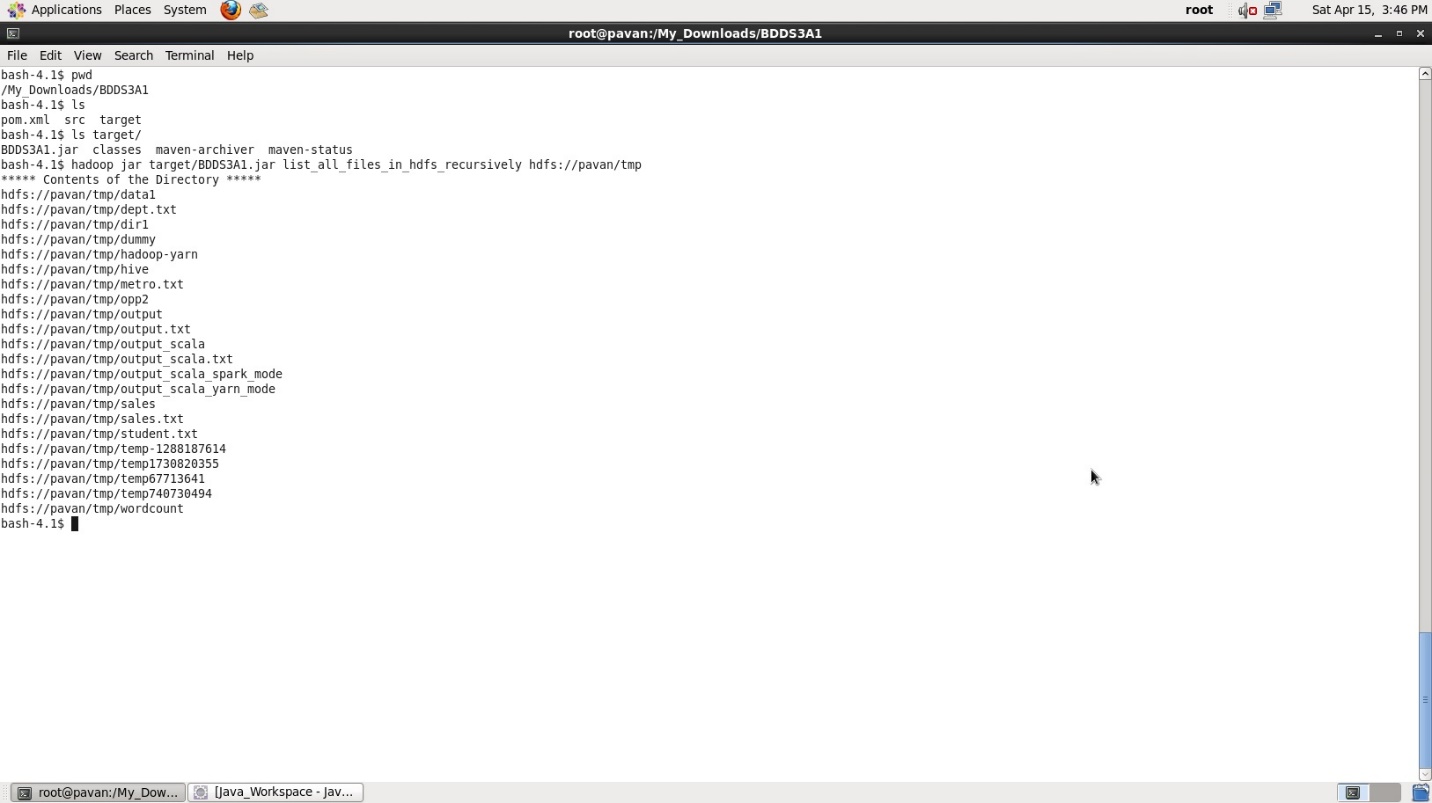
**Note: If port\_no not specified, it takes the default port\_no : 8020 specified in core-site.xml**

* Output in Console (Right side) - displays all the files and sub-directories in that HDFS path



**Running in Command Line:**

* Make the folder structure BDDS3A1/src/main/java/list\_all\_files\_in\_hdfs\_recursively.java
* Then in BDDS3A1 folder, open terminal
* Create the Jar file target/BDDS3A1.jar using Maven as **root user** # mvn clean install
* Run the Hadoop Job : “hadoop jar <jar\_file\_path> <class\_name> <input\_hdfs\_file\_path>”
* In our case : “hadoop jar target/BDDS3A1.jar list\_all\_files\_in\_hdfs\_recursively hdfs://pavan/tmp”
* **Note: If port\_no not specified, it takes the default port\_no : 8020 specified in core-site.xml**
* Output gets displayed on the screen



**Task 2:**

**Q) Modify the previous program to list all the files and sub-directories in the HDFS path recursively.**

A) **Java Program**: Takes HDFS Path and display all the files and sub-directories in that HDFS path **recursively**

(**Note:** Assuming the HDFS path along with its sub-directories have Read permission for displaying the content)

import java.io.FileNotFoundException;

import java.io.IOException;

import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileStatus;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.FileUtil;

import org.apache.hadoop.fs.Path;

public class list\_all\_files\_in\_hdfs\_recursively{

public static void main(String[] args) throws IOException, URISyntaxException

{

//1. Get the file path instance

Path filepath = new Path(args[0]);

//2. Get the Configuration instance

Configuration configuration = new Configuration();

//3. Get the instance of the HDFS - file system

FileSystem filesystem = FileSystem.get(filepath.toUri(), configuration);

//4. Get the metadata of the desired directory

FileStatus[] fileStatus = filesystem.listStatus(filepath);

//5. Call the function which displays the content of the directory

displayDirectoryContents(fileStatus);

}

public static void displayDirectoryContents(FileStatus[] status) throws FileNotFoundException, IOException {

for (int i = 0; i < status.length; i++) {

FileStatus fileStatus = status[i];

if (fileStatus.isDirectory()) {

Configuration conf = new Configuration();

FileSystem filesystem = FileSystem.get(fileStatus.getPath().toUri(), conf);

FileStatus[] subStatus = filesystem.listStatus(fileStatus.getPath());

System.out.println("directory:" + fileStatus.getPath());

displayDirectoryContents(subStatus);

} else {

System.out.println(" file:" + fileStatus.getPath());

}

}

}

}

**Input format: “**hdfs://<hostname>:<port\_no>/<HDFS\_File\_Path>**”**

**Ex: “**hdfs://pavan/tmp/dummy**” here, hostname :** pavan , **HDFS\_File\_Path :** tmp/dummy

**Note: If port\_no not specified, it takes the default port\_no : 8020 specified in core-site.xml**

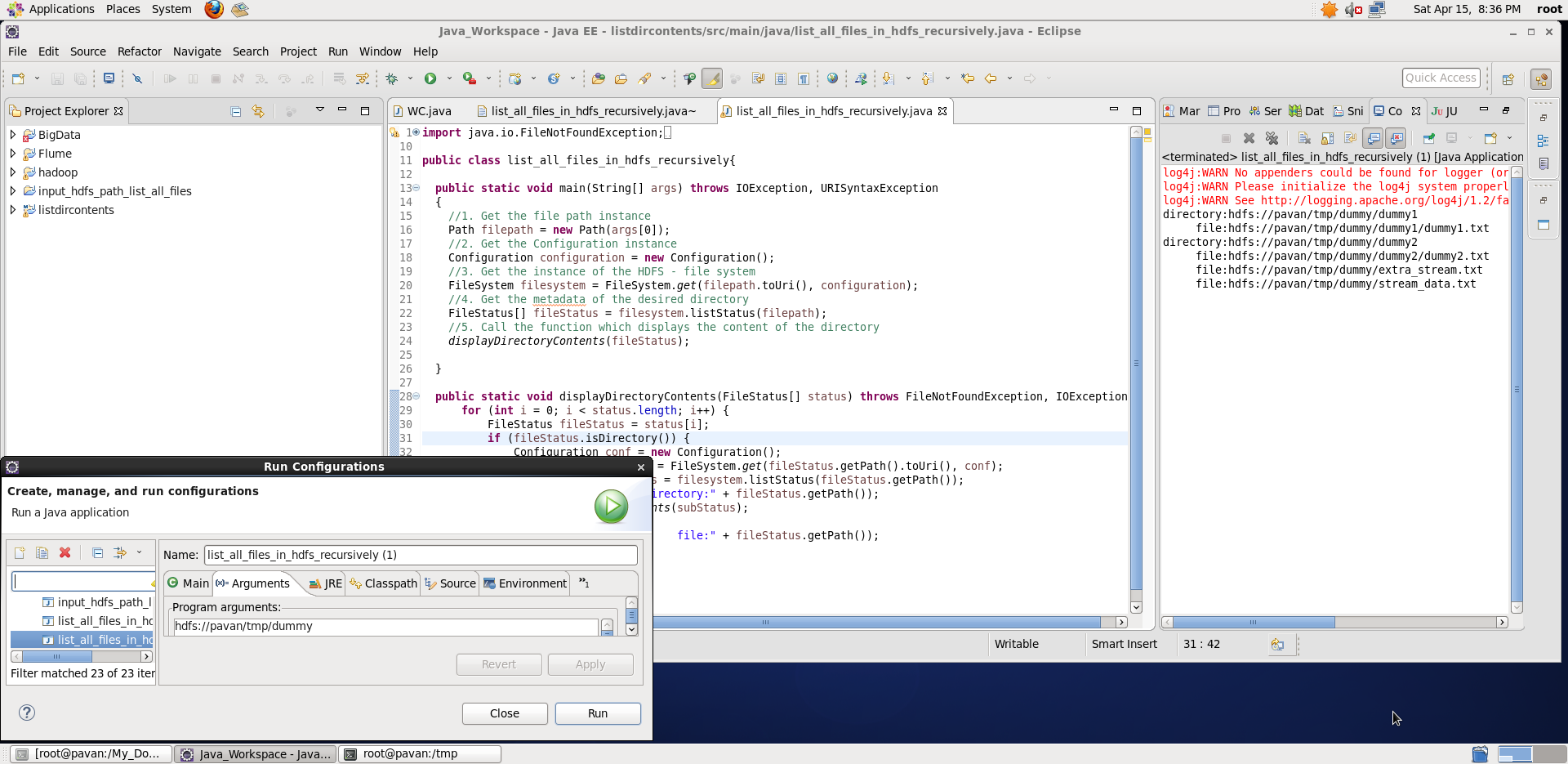
**Files Structure in HDFS is as follows:**

/tmp/dummy/ - **input directory**

* extra\_stream.txt - file
* stream\_data.txt - file
* dummy1 - directory
  + - dummy1.txt - file
* dummy2 - directory
  + - dummy2.txt - file

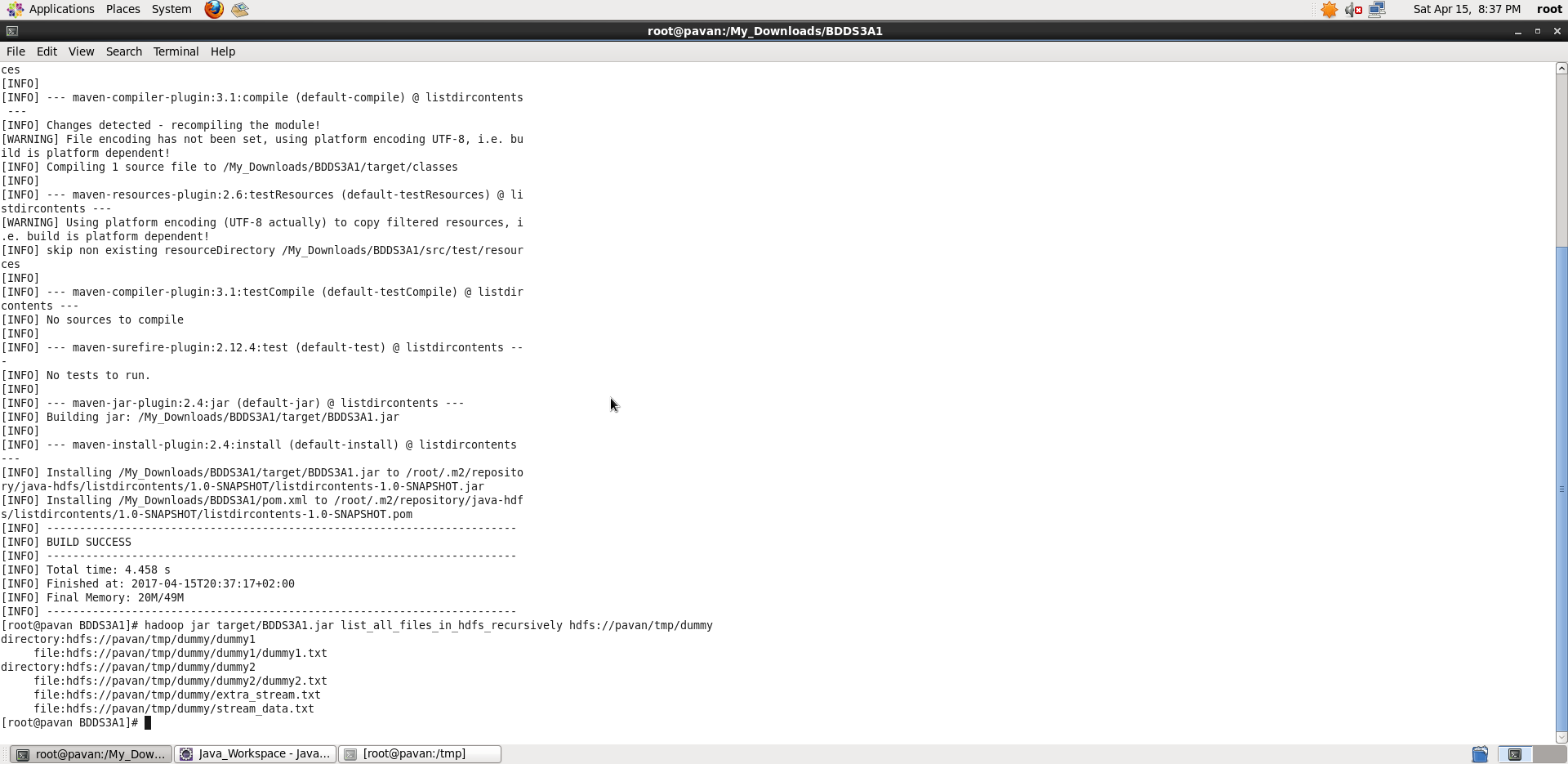
**Running in Eclipse:**

* Input given in Run Configurations Argument Tab as : “hdfs://pavan/tmp/dummy”
* Output in Console (Right side) - displays all the files and sub-directories recursively in HDFS path



**Running in Command Line:**

* Make the folder structure BDDS3A1/src/main/java/list\_all\_files\_in\_hdfs\_recursively.java
* Then in BDDS3A1 folder, open terminal
* Create the Jar file target/BDDS3A1.jar using Maven as **root user** # mvn clean install
* Run the Hadoop Job : “hadoop jar <jar\_file\_path> <class\_name> <input\_hdfs\_file\_path>”
* In our case : “hadoop jar target/BDDS3A1.jar list\_all\_files\_in\_hdfs\_recursively hdfs://pavan/tmp/dummy”
* **Note: If port\_no not specified, it takes the default port\_no : 8020 specified in core-site.xml**
* Output gets displayed on the screen



**Task 3:**

**Q) Modify the previous program to take multiple HDFS paths (separated by space) and list all the files and sub-directories in those HDFS paths recursively.**

A) **Java Program**: Takes multiple HDFS Paths, display all files and sub-directories in those HDFS path **recursively**

(**Note:** Assuming the HDFS path along with its sub-directories have Read permission for displaying the content)

import java.io.FileNotFoundException;

import java.io.IOException;

import java.net.URISyntaxException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileStatus;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.FileUtil;

import org.apache.hadoop.fs.Path;

public class list\_all\_files\_in\_hdfs\_recursively{

public static void main(String[] args) throws IOException, URISyntaxException

{

for(int i=0; i < args.length ; i++) {

//1. Get the file path instance

Path filepath = new Path(args[i]);

//2. Get the Configuration instance

Configuration configuration = new Configuration();

//3. Get the instance of the HDFS - file system

FileSystem filesystem = FileSystem.get(filepath.toUri(), configuration);

//4. Get the metadata of the desired directory

FileStatus[] fileStatus = filesystem.listStatus(filepath);

//5. Call the function which displays the content of the directory

displayDirectoryContents(fileStatus);

System.out.println(""); // Just separating the display contents for each HDF input path

}

}

public static void displayDirectoryContents(FileStatus[] status) throws FileNotFoundException, IOException {

for (int i = 0; i < status.length; i++) {

FileStatus fileStatus = status[i];

if (fileStatus.isDirectory()) {

Configuration conf = new Configuration();

FileSystem filesystem = FileSystem.get(fileStatus.getPath().toUri(), conf);

FileStatus[] subStatus = filesystem.listStatus(fileStatus.getPath());

System.out.println("directory:" + fileStatus.getPath());

displayDirectoryContents(subStatus);

} else {

System.out.println(" file:" + fileStatus.getPath());

}

}

}

}

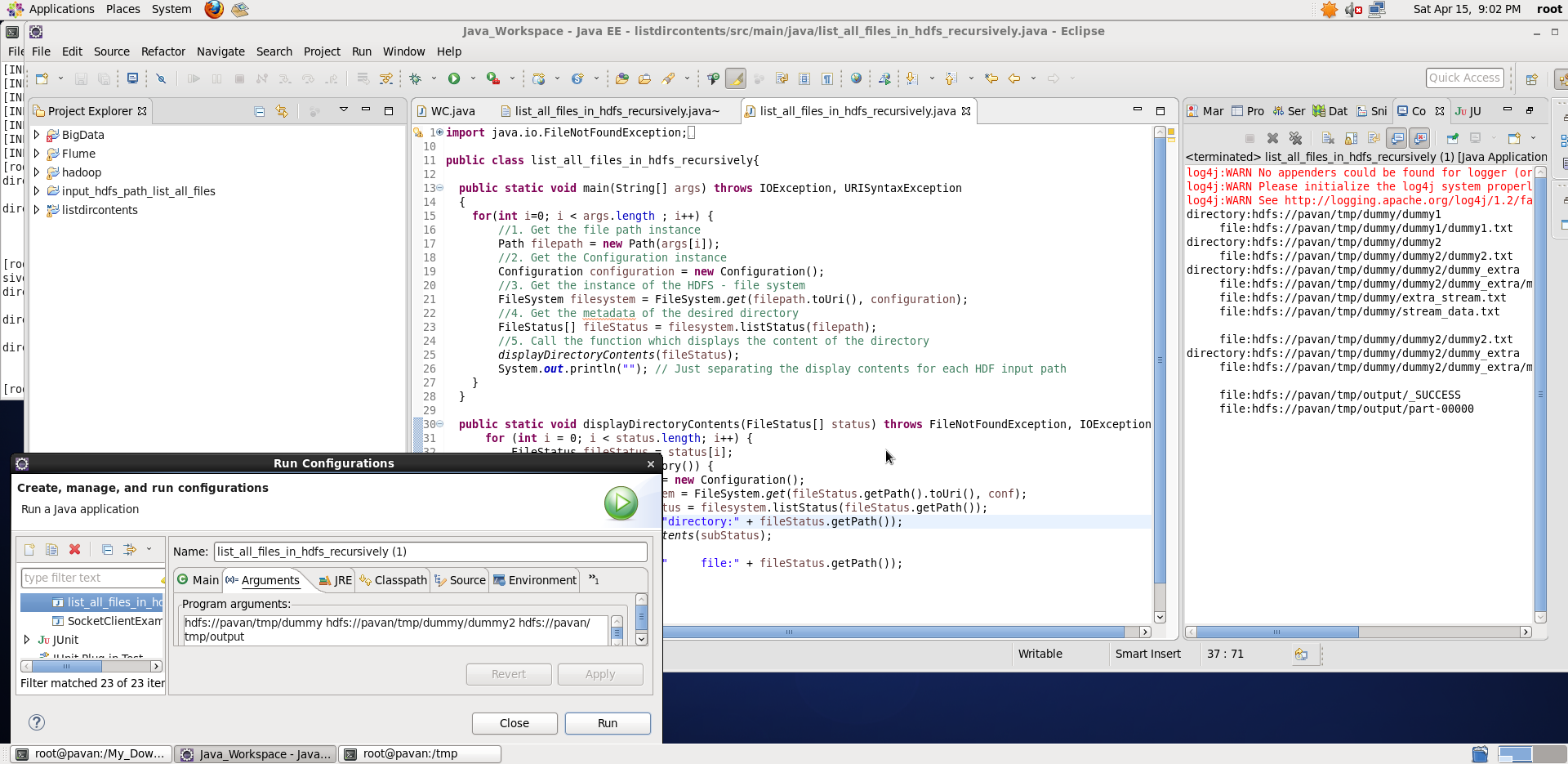
**Input format: “**hdfs://<hostname>:<port\_no>/<HDFS\_File\_Path1> hdfs://<hostname>:<port\_no>/<HDFS\_File\_Path2> hdfs://<hostname>:<port\_no>/<HDFS\_File\_Path3> ..**”**

**Ex: “**hdfs://pavan/tmp/dummy hdfs://pavan/tmp/output**” here, hostname :** pavan , **HDFS\_File\_Path1 :** tmp/dummy , **HDFS\_File\_Path2 :** tmp/output

**Note: If port\_no not specified, it takes the default port\_no : 8020 specified in core-site.xml**

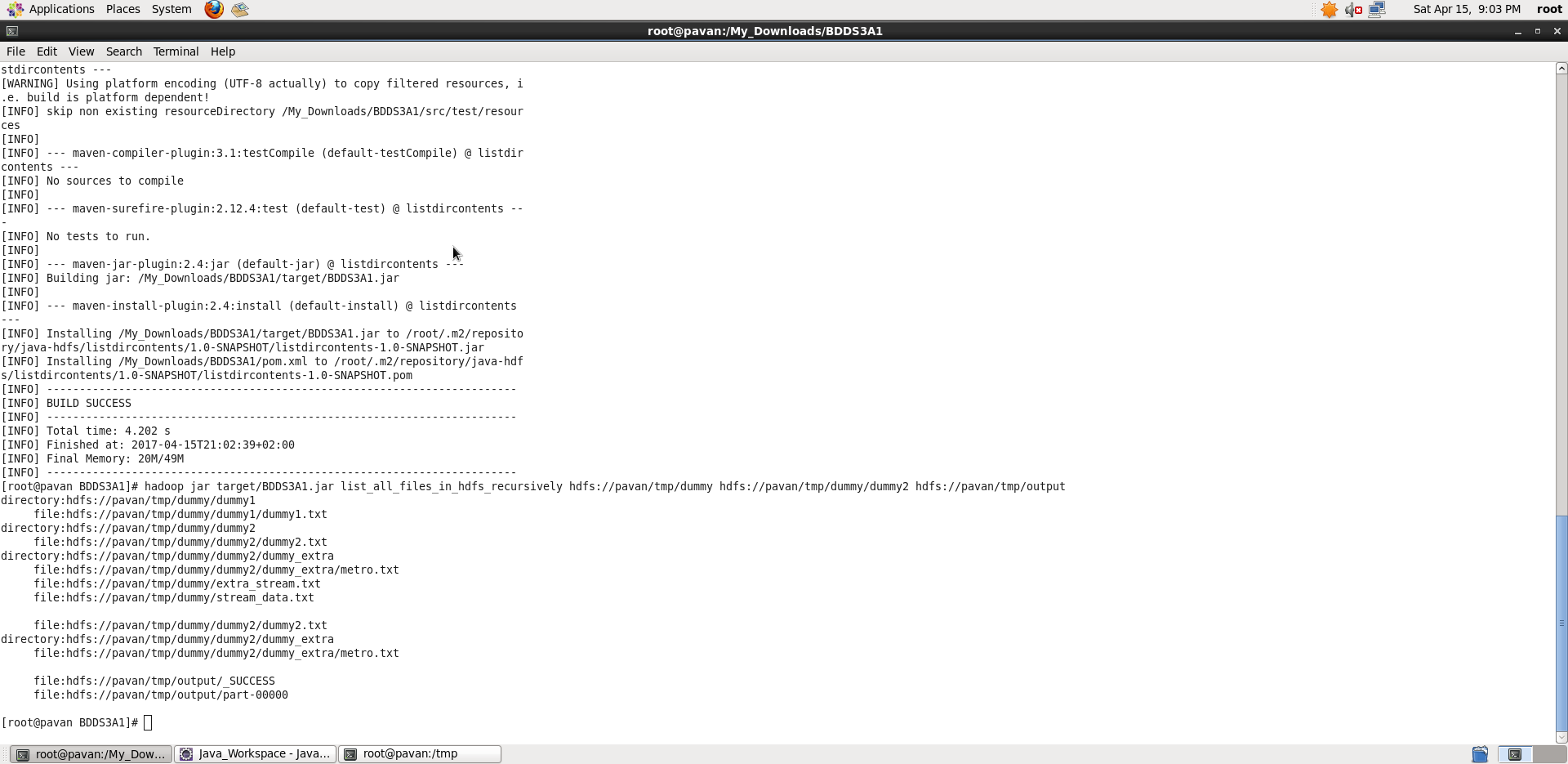
**Running in Eclipse:**

* Input given in Run Configurations Argument Tab as : “hdfs://pavan/tmp/dummy hdfs://pavan/tmp/dummy/dummy2 hdfs://pavan/tmp/output ”
* Output in Console (Right side) - displays all the files and sub-directories recursively in all those HDFS paths

****

**Running in Command Line:**

* Make the folder structure BDDS3A1/src/main/java/list\_all\_files\_in\_hdfs\_recursively.java
* Then in BDDS3A1 folder, open terminal
* Create the Jar file target/BDDS3A1.jar using Maven as **root user** # mvn clean install
* Run the Hadoop Job : “hadoop jar <jar\_file\_path> <class\_name> <input\_hdfs\_file\_path1> <input\_hdfs\_file\_path2> <input\_hdfs\_file\_path3> …”
* In our case : “hadoop jar target/BDDS3A1.jar list\_all\_files\_in\_hdfs\_recursively hdfs://pavan/tmp/dummy hdfs://pavan/tmp/dummy/dummy2 hdfs://pavan/tmp/output”
* **Note: If port\_no not specified, it takes the default port\_no : 8020 specified in core-site.xml**
* Output gets displayed on the screen

****

**Thus, we achieved the task of taking multiple HDFS paths (separated by space) and list all the files and sub-directories in those HDFS paths recursively**.

**Please Note: I didn’t mention port\_no while taking screenshots, internally if port\_no not specified, it takes the default port\_no : 8020 specified in core-site.xml.**