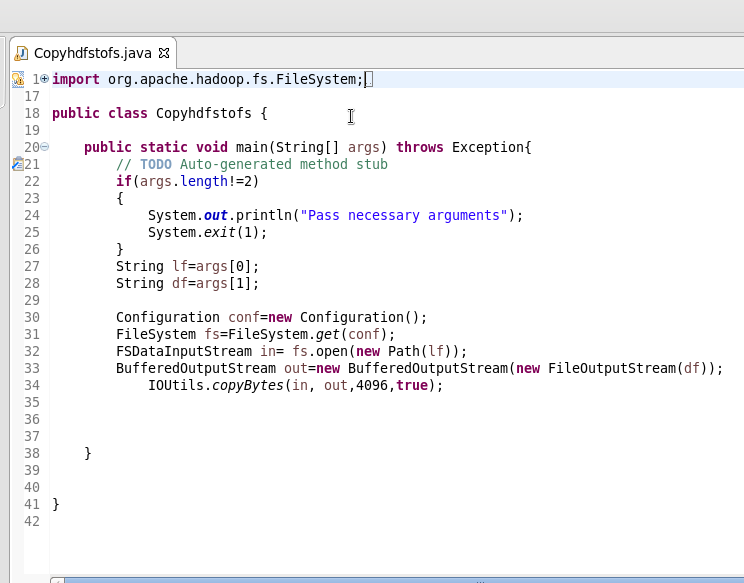
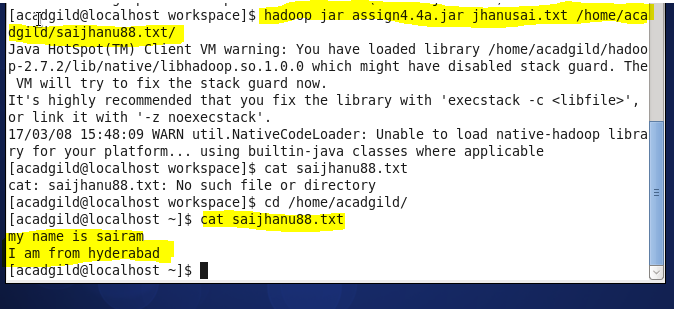
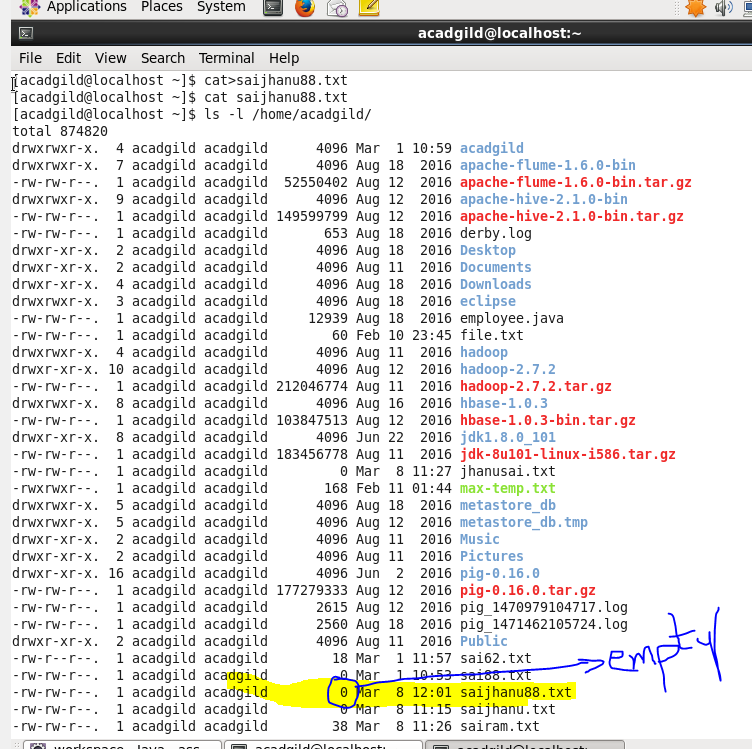
1. Write a Java program to copy a file from HDFS to LFS





2. Explain the importance and usage of the below terms in details

● DFSInputStream

● DFSOutputStream

● FSDataInputStream

● FSDataOutputStream

**DFSInputStream:**

In read operation in HDFS, client calls read() on DFSInputStream. DFSInputStream holds the list of address of block locations on the Datanode for the first few blocks of the file. It then locates the first block on closest datanode and connects to it. Block reader gets initialized on target Block/datanode along with the following information:

Block Id, Data start offset to read from, Length of data to read, Client name.

Data is streamed from the Datanode back to the client in form of packets, this data is copied directly to input buffer provided by the client. DFS client is reading and performing checksum operation and updating the client buffer. read() is called repeatedly on stream till the end of the block is reached. When end of the block is reached DFSInputStream will close the connection to Datanode and search next closest Datanode to read the block from it. Blocks are read in order, once DFSInputStream done through the reading of the first few blocks, it calls the Namenode to retrieve Datanode locations for the next batch of blocks. If Datanode is down reading or DFSInputStream will switch to next available Datanode where replica can be found. DFSInputStream remembers the Datanode which encountered an error so that it does not try them for later blocks.

**DFSOuputStream:**

In write operations in HDFS just as in the read case, FSDataOutputStream wraps a DFSOutputStream, which handles communication with the datanodes and namenode. As the client writes data, DFSOutputStream splits into packets, which it writes to an internal queue, called the data queue. DFSOutputStream also maintains an internal queue of packets that are waiting to be acknowledged by datanodes, called the ack queue.

**FSDataInputStream:**

FSDataInputStream wraps the DataInputStream and implements Seekable, PositionedReadable interfaces which provide method like getPos(), seek() method to provide Random Access on HDFS file.  
FileSystem have open() method which return FSDataInputStream as below:

URI uri = URI.create (“hdfs://host: port/file path”);

Configuration conf = new Configuration ();

FileSystem file = FileSystem.get (uri, conf);

FSDataInputStream in = file.open(new Path(uri));

**FSDataOuputStream:**

Filesystem’s create () method return FSDataOutputStream, which use to create new HDFS file or write the content at the EOF. It doesn’t provide seek because of HDFS limitation to write to content at the EOF only. It wrap Java IO’s DataOutputStream and add method such as getPos() to get the position of the file and write() to write the content at the last position.

Below method signature provide FSDataOutputStream:  
Create method on FileSystem create file e.g.

public FSDataOutputStream create(Path f) create empty file.

public FSDataOutputStream append(Path f) will append existing file