Experiment – 1

Aim: Implement the following Data structures in Java
i) Linked Lists ii) Stacks iii) Queues iv) Set v) Map

i) Linked List :

```
import java.util.*;
public class LinkedListDemo {
public static void main(String args[]) {
// create a linked list
LinkedList ll = new LinkedList();
// add elements to the linked list
ll.add("F");
ll.add("B");
ll.add("D");
ll.add("E");
ll.add("C");
ll.addLast("Z"):
ll.addFirst("A");
ll.add(1, "A2");
System.out.println("Original contents of ll: " +
ll); // remove elements from the linked list
ll.remove("F");
ll.remove(2);
System.out.println("Contents of ll after deletion: " +
ll); // remove first and last elements
  ll.removeFirst();
  ll.removeLast();
System.out.println("ll after deleting first and last: "+ ll);
// get and set a value
Object val = ll.get(2);
ll.set(2, (String) val + " Changed");
System.out.println("ll after change: " + ll);
}
} Output:
```

Original contents of ll: [A, A2, F, B, D, E, C, Z] Contents of ll after deletion: [A, A2, D, E, C, Z] ll after deleting first and last: [A2, D, E, C]
ll after change: [A2, D, E Changed, C]

ii) Stacks Program:

```
import java.util.*;
public class StackDemo {
  static void showpush(Stack st, int a) {
    st.push(new Integer(a));
    System.out.println("push(" + a + ")");
    System.out.println("stack: " + st);
  }
  static void showpop(Stack st) {
    System.out.print("pop -> ");
    Integer a = (Integer) st.pop();
    System.out.println(a);
    System.out.println("stack: " + st);
  }
  public static void main(String args[]) {
    Stack st = new Stack();
    System.out.println("stack: " + st);
    showpush(st, 42);
    showpush(st, 66);
    showpush(st, 99);
    showpop(st);
    showpop(st);
```

```
    showpop(st);
    try {
      showpop(st);
    }catch (EmptyStackException e) {
      System.out.println("empty stack");
    }
  }
}
```

output:

```
stack: [ ]
push(42)
stack: [42]
push(66)
stack: [42, 66]
push(99)
stack: [42, 66, 99]
pop -> 99
stack: [42, 66]
pop -> 66
stack: [42]
pop -> 42
stack: [ ]
pop -> empty stack
```

iii) Queues

```
// Java program to demonstrate working of Queue interface in Java
```

```
import java.util.LinkedList;
import java.util.Queue;
public class QueueExample
{
  public static void main(String[] args)
  {
    Queue<Integer> q = new LinkedList<>();
    // Adds elements {0, 1, 2, 3, 4} to queue
    for (int i=0; i<5; i++)
     q.add(i);
    // Display contents of the queue.
    System.out.println("Elements of queue-"+q);
     // To remove the head of queue.
    int removedele = q.remove();
    System.out.println("removed element-" + removedele);
    System.out.println(q);
    // To view the head of queue
    int head = q.peek();
System.out.println("head of queue-" + head);
    // Rest all methods of collection interface,
    // Like size and contains can be used with this
    // implementation.
    int size = q.size();
    System.out.println("Size of queue-" + size);
  }
}
```

Output:

Elements of queue-[0, 1, 2, 3, 4]

removed element-0

[1, 2, 3, 4]

head of queue-1

Size of queue-4

iv) Set

```
import java.util.*; public
class SetDemo {
public static void main(String args[]) {
int count[] = {34, 22,10,60,30,22};
Set<Integer> set = new HashSet<Integer>();
try{
for(int i = 0; i<5; i++){
set.add(count[i]);
}
System.out.println(set);
TreeSet sortedSet = new TreeSet<Integer>(set);
System.out.println("The sorted list is:");
System.out.println(sortedSet);
System.out.println("The First element of the set is: "+(Integer)sortedSet.first());
System.out.println("The last element of the set is: "+(Integer)sortedSet.last());
}
catch(Exception e){}
}
}
```

Output:

[34, 22, 10, 60, 30]

The sorted list is:

[10, 22, 30, 34, 60]

The First element of the set is: 10

The last element of the set is: 60

v) Map Program:

```
import java.awt.Color;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
public class MapDemo
{

public static void main(String[] args)
{
Map<String, Color> favoriteColors = new HashMap<String, Color>();
favoriteColors.put("sai", Color.BLUE); favoriteColors.put("Ram",
Color.GREEN); favoriteColors.put("krishna", Color.RED);
favoriteColors.put("narayana", Color.BLUE); // Print all keys and values in
the map
Set<String> keySet = favoriteColors.keySet(); for (String key :
keySet) {
Color value = favoriteColors.get(key);
System.out.println(key + " : " + value);
}
  }}
```

Output:

narayana : java.awt.Color[r=0,g=0,b=255]

sai : java.awt.Color[r=0,g=0,b=255]

krishna : java.awt.Color[r=255,g=0,b=0]

Ram : java.awt.Color[r=0,g=255,b=0]

Experiment – 2

Aim: Perform setting up and Installing Hadoop in its three operating modes:
Standalone, Pseudo distributed, Fully distributed.

1. **Installation of Hadoop:**

Hadoop software can be installed in three modes of operation:

- **Stand Alone Mode**: Hadoop is a distributed software and is designed to run on a commodity of machines. However, we can install it on a single node in stand-alone mode. In this mode, Hadoop software runs as a single monolithic java process. This mode is extremely useful for debugging purpose. You can first test run your Map-Reduce application in this mode on small data, before actually executing it on cluster with big data.

- **Pseudo Distributed Mode**: In this mode also, Hadoop software is installed on a Single Node. Various daemons of Hadoop will run on the same machine as separate java processes. Hence all the daemons namely NameNode, DataNode, SecondaryNameNode, JobTracker, TaskTracker run on single machine.

- **Fully Distributed Mode**: In Fully Distributed Mode, the daemons NameNode, JobTracker, SecondaryNameNode (Optional and can be run on a separate node) run on the Master Node. The daemons DataNode and TaskTracker run on the Slave Node.

Hadoop Installation: Ubuntu Operating System in stand-alone mode

Steps for Installation

1. sudo apt-get update
2. 2. In this step, we will install latest version of JDK (1.8) on the machine.
   The Oracle JDK is the official JDK; however, it is no longer provided by Oracle as a default installation. for Ubuntu. You can still install it using apt-get. To install any version, first execute the following
   commands:
   a. sudo apt-get install python-software properties
   b. sudo add-apt-repository ppa:webupd8team/

java

c. sudo apt-get update

Then, depending on the version you want to install, execute one of the following commands:

Oracle JDK 7: sudo apt-get install oraclejava7-installer

Oracle JDK 8: sudo apt-get install oraclejava8-installe

3. . Now, let us setup a new user account for Hadoop installation. This step is optional, but recommended because it gives you flexibility to have a separate account for Hadoop installation by separating this installation from other software installation

a. sudo adduser hadoop_dev (Upon executing this command, you will prompted to enter the new password for this user. Please enter the password and enter other details. Don't forget to save the details at the end)

b. su- hadoop_dev (Switches the user from current user to the new user created i.e Hadoop_dev)

4. Download the latest Hadoop distribution.

a. Visit this URL and choose one of the mirror sites. You can copy the download link and also use "wget" to download it from command prompt:

Wgethttp://
apache.mirrors.lucidnetworks.net/hadoop/
common/hadoop-2.7.0/hadoop-2.7.0.tar.gz

5. Untar the file :

```
tar xvzf hadoop-2.7.0.tar.gz
```

6. Rename the folder to hadoop2

```
mv hadoop-2.7.0 hadoop2
```

7. Edit configuration file /home/hadoop_dev/ hadoop2/etc/hadoop/hadoop-env.sh and set JAVA_HOME in that file.

a.

```
vim /home/hadoop_dev/hadoop2/etc/hadoop/
```

hadoop-env.sh

b. uncomment JAVA_HOME and update it following line:

export JAVA_HOME=/usr/lib/jvm/java-8-oracle (Please check for your relevant java installation and set this value accordingly. Latest versions of Hadoop require > JDK1.7)

8. Let us verify if the installation is successful or not (change to home directory cd /home/hadoop_dev/hadoop2/) :

a. bin/hadoop (running this command should prompt you with various options)

9. This finishes the Hadoop setup in stand-alone mode.

10. Let us run a sample hadoop programs that is provided to you in the download package:

$ mkdir input (create the input directory)

$ cp etc/hadoop/*.xml input (copy over all the xml files to input folder)

$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.0.jar grep input output 'dfs[a-z.]+' (grep/find all the files matching the pattern ‘dfs[a-z.]+’ and copy

those files to output directory)

$ cat output/* (look for the output in the output

directory that Hadoop creates for you).

Hadoop Installation: Psuedo Distributed Mode (Locally)

Steps for Installation

1. Edit the file /home/Hadoop_dev/hadoop2/etc/ hadoop/core-site.xml as below:

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

Note: This change sets the namenode ip and port.

2. Edit the file /home/Hadoop_dev/hadoop2/etc/ hadoop/hdfs-site.xml as below:

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
```

Note: This change sets the default replication count for blocks used by HDFS.

3. We need to setup password less login so that the master will be able to do a password-less ssh to start the daemons on all the slaves.

Check if ssh server is running on your host or not:

a. ssh localhost (enter your password and if you are able to login then ssh server is running)

b. In step a. if you are unable to login, then install ssh as follows:

sudo apt-get install ssh

c. Setup password less login as below:

i. ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa

ii. cat ~/.ssh/id_dsa.pub >> ~/.ssh/

authorized_key

4. We can run Hadoop jobs locally or on YARN in this mode. In this Post, we will focus on running the jobs **locally.**

5. Format the file system. When we format namenode it formats the meta-data related to data nodes. By doing that, all the information on the datanodes are lost and they can be reused for new data:

a. bin/hdfs namenode –format

6. Start the daemons

a. sbin/start-dfs.sh (Starts NameNode and DataNode)

You can check If NameNode has started successfully or not by using the following web interface: http://0.0.0.0:50070 . If you are unable to see this, try to check the logs in the /home/ hadoop_dev/hadoop2/logs folder.

7.You can check whether the daemons are running or not by issuing Jps command.

8. This finishes the installation of Hadoop in pseudo distributed mode.

9. Let us run the same example we can in the previous blog post:

i) Create a new directory on the hdfs

bin/hdfs dfs -mkdir –p /user/hadoop_dev

ii) Copy the input files for the program to hdfs:

bin/hdfs dfs -put etc/hadoop input

Run the program:

iv) View the output on hdfs:

bin/hdfs dfs -cat output/*

10. Stop the daemons when you are done executing the jobs, with the below command:

sbin/stop-dfs.sh

Hadoop Installation – Psuedo Distributed Mode (YARN)

Steps for Installation

1. Edit the file /home/hadoop_dev/hadoop2/etc/ hadoop/mapred-site.xml as below:

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

2. Edit the fie /home/hadoop_dev/hadoop2/etc/ hadoop/yarn-site.xml as below:

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

</configuration>

Note: This particular configuration tells MapReduce how to do its shuffle. In this case it uses the mapreduce_shuffle.

3. Format the NameNode:

```
bin/hdfs namenode –format
```

4. Start the daemons using the command:

```
sbin/start-yarn.sh
```

This starts the daemons ResourceManager and NodeManager.

Once this command is run, you can check if ResourceManager is running or not by visiting the following URL on browser : http://0.0.0.0:8088 . If you are unable to see this, check for the logs in the directory: /home/hadoop_dev/hadoop2/logs

5. To check whether the services are running, issue a jps command. The following shows all the services necessary to run YARN on a single server:

```
$ jps
15933 Jps
15567 ResourceManager
15785 NodeManager
```

6. Let us run the same example as we ran before:

i) Create a new directory on the hdfs

```
bin/hdfs dfs -mkdir –p /user/hadoop_dev
```

ii) Copy the input files for the program to hdfs:

```
bin/hdfs dfs -put etc/hadoop input
```

iii) Run the program:

```
bin/yarn jar share/hadoop/mapreduce/
hadoop-mapreduce-examples-2.6.0.jar grep
input output 'dfs[a-z.]+'
```

iv) View the output on hdfs:

```
bin/hdfs dfs -cat output/*
```

7. Stop the daemons when you are done executing the jobs, with the below command:

```
sbin/stop-yarn.sh
```

This completes the installation part of Hadoop.