POORNIMA

Experiment 1

1. Linked List
2. Stack
3. Queue
4. SET
5. map

Write all the Program in Java language.

1. Linked List :-

```
import java.util.*;
public class Test {
    public static void main (String
                    args[])
    {
        LinkedList<String> ll = new LinkedList
                <String>();
        ll.add("A");
        ll.add("B");
        ll.addLast("C");
        ll.addFirst("D");
        ll.add(2, "E");
        System.out.println(ll);
    }
}
```

POORNIMA

2. Stack :-

```
import java.io.*;
import java.util.*;
class Test
{
    Static void stack-push(Stack<Integer>Stack)
    {
        For (int i=0; i<5; i++)
        {
            Stack.push(i);
        }
    }
Static void Stack-pop(Stalk<Integer>Stack)
    {
        System.out.println("Pop operation");
        For( int i=0; i<5; i++)
        {
            integer y=(Integer)Stack.pop
            System.out.println(y);
        }
    }
Static void Stack-peek(Stack<Integer>Stack)
    {
        Integer element=(Integer)Stack.peek
```

Output :- [D, A, E, B, C]

[A]

Output :-

Pop operation:

4

3

2

1

0

Element on Stack Top : 4

Element is found at position : 3

Element not found

```
        System.out.println("Element on Stack
                top: " + element);
    }
    Static void stack_search (Stack<Integer>
                Stack, int element)
    {
        Integer pos = (Integer) stack.search
                (element);
        if (pos == -1)
            System.out.println("Element not
                found");
        else
            System.out.println("Element
    is found at position: " + pos);
    }
    public Static void main (String [] args)
    {
        Stack<Integer> Stack = new Stack<Integer>();
        Stack.push (Stack);
        Stack.pop (Stack);
        Stack.push (Stack);
        Stack.pop (Stack);
        Stack.search (Stack);
        Stack.search (Stack);
    }
}
```

3. Queue :-

```
import java.util.LinkedList;
import java.util.Queue;
public class QueueExample {
    public static void main(String[]
                             args)
    {
        Queue<Integer> q = new LinkedList<>();
        for(i=0; i<5; i++)
            q.add(i);

        System.out.println("Element of
                queue" + q);
        int removedele = q.remove();
        System.out.println("removed element" +
                removedele);
        System.out.println(q);

        int head = q.peek();
        System.out.println("Head of queue" + head);
        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

4. Set :-

```
import java.util.*;
public class setExample {
        public static void main(String[] args)
    {
        Set <String> data = new LinkedList
                            HashSet <String>();
        data.add("Implementing");
        data.add("Set");
            data.add("Example");
            System.out.println(data);
    }
}
```

5. Map :-

```
import java.util.*;
class HashMapDemo {
        public static void main(String
                    args[])
    {
        Map <String, Integer> hm = new HashMap
                    <String, Integer>();

        hm.put("a", new Integer(100));
        hm.put("b", new Integer(200));
        hm.put("c", new Integer(300));
```

Output :

[Implementing, Set, Example]

Output :

Map Elements :

a : 100
b : 200
c : 300
d : 400

```
hm.put("d", new Integer(400));

for(Map.Entry<String, Integer> me:
        hm.entrySet())
{
    System.out.println(me.getKey()+":");
    System.out.println(me.getValue());
}

}
}
```

POORNIMA

Experiment - 2

Objective : Perform setting up and Installing Hadoop in its three operating modes :

(i) Standalone (ii) Pseudo - distributed
(iii) Fully distributed

Requirements : Ubuntu Platform

prerequisite Test :

```
sudo apt update
sudo apt install openjdk-19-jdk headless -y

java -version ; javac -version
sudo apt install openssh-server openssh-client
sudo adduser hadoop
su - hadoop
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
ssh localhost
```

Downloading Hadoop

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.4/hadoop-3.3.4.tar.gz tar xzf hadoop-3.3.4.tar.gz
```

POORNIMA

Editing 6 important files

1st File =

sudo nano .bashrc – Here you might face issue saying HADOOP is not sudo user if this issue comes then

su -< Existing super User >

sudo adduser hadoop sudo

Sudo nano .bashrc

```
# Add Irelow lines in this file
# Hadoop Related options
export HADOOP_HOME =/ home/ hadoop / hadoop-3.3.4
export HADOOP_INSTALL = $ HADOOP_HOME
export HADOOP_MAPRED_HOME = $ HADOOP_HOME
export HADOOP_COMMON_HOME = $ HADOOP_HOME
export HADOOP_HDFS_HOME = $ HADOOP_HOME
export YARN_HOME = $ HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR = $ HADOOP_
    HOME / lib / native
export PATH = $ PATH. $ HADOOP_HOME / sbin: $ HADOOP
    _HOME/ bin
export HADOOP_OPTS " - Djava. library. path = $
    HADOOP_HOME / lib / native "
```

Source ~/. bashrc

POORNIMA

2nd File =>

sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

3rd File =>

sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml

Add below lines in this file (between "<configuration>" and "</configuration>")

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hadoop/tmpdata</value>
  <description>A base for other temporary directories</description>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
  <description>The name of the default File System</description>
```

POORNIMA

4th File :

sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml

Add below lines in this file (between "<configuration>" and "</configuration>")

<property>
<name> dfs.data.dir </name>
<value> /Home/hadoop/dfsdata/namenode </value>
</property>

<name> dfs.data.dir </name>
<value> /home/hadoop/dfsdata/datanode </value>
</property>

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

5th File :

sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml

<property>
<name> mapreduce.framework.name </name>
<value> yarn </value>
</property>

6th File :

sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

```
</property>
  <name> yarn.nodemanager.aux.service </name>
  <value> mapreduce_shuffle </value>
</property>
<property>
<name> yarn.nodemanager.aux.services.mapreduces
      .shuffle.class </name>
<value> org.apache.hadoop.mapred.ShuffleHandler
      </value>
</property>
<property>
  <name> yarn.resourcemanager.hostname </name>
<value> 127.0.0.1 </value>
</property>
<property>
   <name> yarn.acl.enable </name>
   <value> 0 </value>
 </property>
<property>
 <name> yarn.nodemanager.env.whitelist </
       name>
<value> JAVA_HOME, HADOOP_COMMON_HOME,
      HADOOP_HDFS_HOME, HADOOP_CONF_DIR,
CLASSPATH_PERPEND_DISTCACHE, HADOOP_YARN_
  HOME, HADOOP_MAPRED, HOME </value>
```

</property>

Launching Hadoop

hdfs namenode - Format

./start-dfs.sh

Lindha
20/2/24.