

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);
```

```
}
```

```
}
```

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(Stack<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 $\rightarrow$ [D, A, E, B, C]
[A]

AMIRKOT

1. Finding

10.2.2020

10.12

10.12

10.12

10.12

Output $\rightarrow$

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 remove dele = q.remove();  
        System.out.println(" removed element" +  
            remove dele);  
        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 $\Rightarrow$

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 LinkedHashSet<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]

(7 points) Write a Java program to implement a Set using an array.

Assume that the Set contains integers.

Write the code for the following methods:

(a) insert(int x): Inserts the element x into the Set.

(b) delete(int x): Deletes the element x from the Set.

(c) display(): Displays the elements of the Set.

(d) size(): Returns the number of elements in the Set.

(e) isEmpty(): Returns true if the Set is empty, otherwise false.

(f) contains(int x): Returns true if the Set contains the element x, otherwise false.

(g) remove(int x): Removes the element x from the Set and returns true if it was present, otherwise false.

(h) add(int x): Adds the element x to the Set and returns true if it was not already present, otherwise false.

(i) clear(): Removes all elements from the Set.

(j) toString(): Returns a string representation of the Set.

(k) equals(Object o): Returns true if the Set is equal to the object o, otherwise false.

(l) hashCode(): Returns the hash code of the Set.

(m) clone(): Returns a shallow copy of the Set.

(n) toArray(): Returns an array containing the elements of the Set.

(o) toArray(T[]): Returns an array containing the elements of the Set, cast to the type T.

(p) iterator(): Returns an iterator over the elements of the Set.

(q) Spliterator(): Returns a Spliterator over the elements of the Set.

(r) parallelStream(): Returns a parallel Stream over the elements of the Set.

(s) stream(): Returns a sequential Stream over the elements of the Set.

(t) streamSupport(): Returns a StreamSupport for the Set.

(u) spliteratorSupport(): Returns a SpliteratorSupport for the Set.

(v) ... (other methods)

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());
```

```
}
```

```
}
```

```
}
```

Experiment 32

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

Down loading 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

Editing 6 important files

1st File

`sudo nano /etc/sudoers` - 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 /etc/sudoers`

Add below 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`

2nd File

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

```
export JAVA_HOME=/usr/lib/jvm/java-19-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/tmp</value>
```

```
<description>A wash 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>
```

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/dfs data/namenode / </value>
```

```
</property>
```

```
<name> dfs.data.dir </name>
```

```
<value> /home / hadoop / dfs data / data node </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.mapreduels  
    .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.white list </  
    name>
```

```
  <value> JAVA_HOME, HADOOP_COMMON_HOME,  
    HADOOP_HDFS_HOME, HADOOP_CONF_DIR,  
    CLASSPATH_SEPARATOR, DISTCACHE, HADOOP_YARN_  
    HOME, HADOOP_MAPRED_HOME </value>
```


</properties>

Launching Hadoop

hdfs namenode - format

./start-dfs.sh

~~Airdh~~
~~20/2/24~~