POORNIMA

# Experiment - 11

Objective :- Implement the following data structures of java :-

(1) Linked list
(2) Queue
(3) stack

(1) Linked list :—

```
class Node {
        int data;
        Node next;
        public Node (int data) {
            this.data = data;
            this.next = null;
        }
    }
class LinkedList {
    private Node head;
    public LinkedList () {
        this.head = null;
    }
// Function to insert a new node at the end of linked list
public void insertAtEnd (int data) {
  if (head == null) {
      head == newNode;
  }
```

```
else {
    Node current = head;
    while (current.next != null) {
        current = current.next;
    }
    current.next = newNode;
}
}
//function to insert a new node at the beginning of the linked list
public void insertAtBeginning (int data) {
    Node newNode = new Node (data);
    newNode.next = head;
    head = newNode;
}

// function to delete a node with a specific value from the
linked list
public void deleteNode (int data) {
    if (head == null) {
        return;
    }
    if (head.data == data) {
        head = head.next;
        return;
    }
Node current = head;
while (current.next != null && current.next.data !=
                                        data) {
```

```
        current = current.next;
    }
    if (current.next != null) {
        current.next = current.next.next;
    }
}
// Function to display the linked list
public void display() {
    Node current = head;
    while (current != null) {
        System.out.print(current.data + " ");
        current = current.next;
    }
    System.out.println();
}
}

public class Main {
    public static void main(String[] args) {
        LinkedList linkedList = new LinkedList();

        linkedList.insertAtEnd(1);
        linkedList.insertAtEnd(2);
        linkedList.insertAtEnd(3);

        System.out.println("Linked List:");
        linkedList.display();

        linkedList.insertAtBeginning(0);
```

Output:-

Linked List:

1 2 3

Linked List after inserting at the beginning:

0 1 2 3

Linked List after deleting node with value 2:

0 1 3

```
System.out.println("Linked List after inserting at the
    beginning :");
linkedList.display();

linkedList.deleteNode(2);

System.out.println("Linked List after deleting node with
    value 2:");
    linkedList.display();
}
}
```

(ii) Queue

```
class Queue {
    private static final int max-size = 5;
    private int front, rear, size;
    private int[] elements;

public Queue() {
    this.front = -1;
    this.rear = -1;
    this.size = 0;
    this.elements = new int[MAX-SIZE];
}
// Function to check if the queue is empty
public boolean isEmpty() {
    return size == 0;
}
```

```
// function to check if queue is full
    public boolean isFull () {
            return size == MAX_INT SIZE ;
}
// Function to add an element to the rear of the queue (enqueue)
    public void enqueue (int data ) {
        if (isFull () ) {
        System.out.println ("Queue is full. Cannot engue" +da
          return ;
      }
       if (isEmpty () ) {
            front = rear = 0 ;
        } else {
            rear = (rear + 1) % MAX-SIZE ;
        }
       element [rear ] = data ;
         size++;

     System.out.println (" Enqueued : " + data );
  }
// Function to remove and return the element from the front
   of the queue (dequeue)

  Public Void dequeue () {
        if (is Empty () ) {
         System. out. println ("Queue is empty: Cannot dequeue
           return ;
      }
```

```
int removedElement = elements[front];
  size--;
if (isEmpty()){
    front = rear = -1;
  }
  else {
    front = (front + 1) % MAX-SIZE;
  }
  System.out.println("Dequeued :" + removedElements);
}

//function to display the elements of the queue:
public void display() {
    if(isEmpty()) {
      System.out.println("Queue is empty:");
      return;
  }
    System.out.print("Queue :");
      int count = 0;
      int index = front;

    while (count < size) {
      System.out.print(elements[index] + " ");
      index = (index + 1) % MAX-SIZE;
      count++;
  }
    System.out.println();
}
```

Output:—

Enqueued : 1
Enqueued : 2
Enqueued : 3

Queue : 1 2 3

Dequeued : 1

Queue : 2 3

Enqueued : 4
Enqueued : 5

Queue is Full . Cannot enqueue 6

Dequeued : 2

Dequed : 3
Dequed : 4
~~Queue is empty~~

```
}

public class main {{
    public static void main ( String [] args ) {
    Queue queue = new Queue ();

    queue.enqueue (1);
    queue.enqueue (2);
    queue.enqueue (3);

    queue.display ();

    queue.dequeue ();
    queue.display ();

    queue.enqueue (4);
    queue.enqueue (5);
    queue.enqueue (6);

    queue.dequeue ();
    queue.dequeue ();
    queue.dequeue ();
    queue.display ();

  }
}
```

Output :-

```
pushed : 1
pushed : 2
pushed : 3

Stack : 1 2 3
Pop : 3

Stack : 1 2

pushed : 4
pushed : 5

Stack is full cannot push !

Popped : 2
popped : 1
popped : 4
```

(iii) Stack —

```
class stack {
    private static final int MAX-SIZE = 5;
    private int top;
    private int [] elements;

    public stack () {
        this.top = -1;
        this.elements = new int [MAX-SIZE];
    }

//function to check if stack is empty
    public boolean isEmpty () {
        return top == -1;
    }

//function to check if stack is full
    public boolean isFull () {
        return top == MAX-SIZE -1;
    }

//function to add an element to the top of the stack (push)
    public void push (int data) {
        if (isFull ()) {
            System.out.println (" stack is full. Cannot push "+data)
            return;
    }
```

```
        elements[++top] = data;
        System.out.println("Pushed: " + data);
    }

    // Function to remove and return the element from the top of
    the stack (pop)
    public void pop() {
        if (isEmpty()) {
            System.out.println("Stack is empty. Cannot pop");
            return;
        }
        int poppedElement = elements[top--];
        System.out.println("Popped: " + poppedElement);
    }

    // Function to display the element of the stack
    public void display() {
        if (isEmpty()) {
            System.out.println("Stack is empty.");
            return;
        }
        System.out.print("Stack: ");
        for(int i = 0; i <= top; i++) {
            System.out.print(elements[i] + " ");
        }
        System.out.println();
    }
}
```

```
public class main {
    public static void main (String[] args) {
      Stack stack = new Stack ();

    stack.push (1);
    stack.push (2);
    stack.push (3);

    stack.display ();

    stack.pop ();
    stack.display ();

    stack.push (4);
    stack.push (5);
    stack.push (6)

    stack.pop ();
    stack.pop ();
    stack.pop ();

    stack.display ();

  }
}
```

POORNIMA

# Experiment - 02

Objective: Implement the following data structures in Java.

(i) Set

(ii) Map

(i) Set -

```
import java.util.HashSet;
import java.util.Iterator;

Class CustomSet {
 private HashSet<Integer> set;

   public CustomSet() {
      this.set = new HashSet<>();
   }
// Function to add an element to the set
   public void addElement (int element) {
      set.add (element);
    System.out.println ("Added" + element );
  }

// Function to Remove an element from the set
  public void RemovElement (int element) {
     if ( set.contains ( elements )) {
        set.remove ( elements);
     System.out.println ("Removed:" + element);
```

```
} else {
    System.out.println("Element " + element + " not found in
        the set");
    }
}
// Function to check if an element is present in the set
public void containsElement(int element) {
        if (set.contains(element)) {
        System.out.println("set contains element " +
                                element);
}else {
    System.out.println("Set does not contain element " +
                                element);
    }
}
// Function to display the element of the set
public void display() {
    if (set.isEmpty()) {
System.out.println("set is empty");
}
    else {
        System.out.print("set: ");
Iterator<Integer> iterator = set.iterator();
while (iterator.hasNext()) {
System.out.print(iterator.next() + " ");
}
System.out.println();
```

```
    }
  }
}

public class Main {
    public static void main (String[] args) {
      CustomSet customset = new CustomSet ();

      customset.addElement (1);
      customset.addElement (2);
      customset.addElement (3);

      customset.display ();

      customset.removeElement (2);
      customset.display ();

      customset.containsElements (3);
      customset.containsElement (1);

    }
}
```

(ii) Map—

```
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

class CustomMap {
```

Output:-

```
Added : 1
Added : 2
Added : 3
Set : 1 2 3
Removed: 2
Set : 1 3
Set Contains element 3
Set does not contain element 4
```

```
private Map<String, Integer> map;

public CustomMap() {
    this.map = new HashMap<>();
}

// Function to add a key-value pair to the map
public void addKeyValuePair(String key, int value) {
    map.put(key, value);
    System.out.println("Added " + key + " => " + value);
}

// Function to remove a key-value pair with from the
map
public void removeKeyValuePair(String Key) {
    if (map.containsKey(Key)) {
        int removedValue = map.remove(key);
        System.out.println("Removed: " + key + " -> " +
                removedValue);
    } else {
        System.out.println("Key " + key + " not found in
            the map");
    }
}

// Function to check if a key is present in the map
public void containsKey(String key) {
    if (map.containsKey(key)) {
        System.out.println("Map contains key: " + key);
```

```
} else {
    System.out.println("Map does not contain key :" + key);
  }
}

// function to display the key-value pairs in map
public void display() {
        if(map.isEmpty()) {
            system.out.println("Map is empty");
    } else {
    System.out.println("Map:");
 Set<Map.Entry<String, Integer>> entrySet = map.
                        entrySet();
 for(Map.Entry<String, Integer> entry : entrySet) {
    system.out.println(entry.getKey() + " ->" +
            entry.getValue());
  }
 }
}
}

public class Main {
    public static void main(String[] args) {
      CustomMap customMap = new CustomMap();

    customMap.addKeyValuePair("A", 1);
    customMap.addKeyValuePair("B", 2);
    customMap.addKeyValuePair("C", 3);
     customMap.display();
```

```
customMap.removeKeyValuePair("B");
customMap.display();

customMap.containsKey("C");
customMap.containsKey("D");
```

Output :-

Added : A -> 1
Added : B -> 2
Added : C -> 3

Map :
A -> 1
B -> 2
C -> 3

Removed : B -> 2

Map :
A -> 1
C -> 3

Map contains key : C

Map does not contain key : D

POORNIMA

# Experiment - 03

Objective: Introduction to Hadoop Installation

Theory –

↳ Hadoop is an open source framework based on Java that manages the storage and processing of large amount of data for applications.

↳ Hadoop used distributed storage and parallel programming to handle big data and analytics jobs. breaking workloads down into smaller workloads that can be run at the same time

↳ four modules comprise the primary Hadoop framework and work collectively to form the hadoop ecosystem.

1. HDFS – Individual hadoop nodes operates on data that resides in their local storage this removes network later providing high throughput access to application.

2. YARN – It is a resource management platform, responsible for managing computer resources in cluster and using them to schedule their applications.

3. Map Reducer – Programming model for large scale data processing in the model subset of laeger data sets and instruction for processing the subset is processed by a node in parallel with other processing

jobs.

4. Hadoop Common- It includes the libraries and utilities used and shared by other modules.

↳ A part of above four tabls and app used for collect store, Process, analyse and manage big data.

- Apache Pig
- Apache Hive
- Apache HBase
- Apache Spark
- Presto
- Appache zepplin

## Setting up Hadoop in stand alone Mode:-

↳ Install JDK
↳ Download Hadoop Package
↳ Verify the Jare installed.
↳ Extract Hadoop at C:\Hadoop
↳ setting up the Hadoop Home variable.
↳ Setting Hadoop and java in directory
↳ Hadoop Configuration of following file:-

(i) core-site.xml.
(ii) mapred-site.xml
(iii) HDFS-Site.xml

(iv) Yarn-site.xml
(v) Hadoop-env.cmd
(vi) Create two folders 'datanode and namenode

- Format the namenode folder
- Test the setup by entering start-all.cmd in command prompt.
- Open https://localhost:50070

## Setting up Hadoop in Pseduo Distributed Mode:-

- Download binnary package
- Save file to c:\big data
- Unzip the binary package
- Create folder for datanode & namenode
- make short name of Java Home Path.
- Configure Hadoop
- Edit Hdfs-site.xml
- Edit Core-site.xml
- Yarn Configuration
- Edit Mapred-site.xml
- Format namenode
- Launch Hadoop
- Hadoop Web UI
- Resource manager - open localhost:8088
- Node Manager - open localhost:8042
- Name Node - open localhost:9870
- Data Node - open localhost:9864

## Setting up Hadoop in Fully Distributed Mode –

- Create Hadoop user
- Add domain name and IP address mappings.
- Install a SSH client and server
- Install Java (version 7 or 8)
- Download Hadoop
- Setup Hadoop user as owner
- Configure Hadoop files.

(i) core-site.xml – one of core configuration file used to configure the Hadoop Distribution file System (HDFS) and other core component. It contains properties that define the basic configuration setting for Hadoop, such as the file system URI, default block size, and various I/O setting

(ii) HDFS-site.xml – a configuration file used to specify various settings related to Hadoop Distributed file system (HDFS). It contains parameter that define the behaviour and characteristics of HDFS components, including the NameNode and DataNodes.

(iii) mapred-site.xml – file is used to configure the Hadoop Distributed file system (HDFS) setting.

it contains that define the behavior and characteristics of the HDFS components such as NameNode and DataNode.

(iv) YARN-site.xml — file is used to configure setting for Apache Hadoop YARN (Yet Another Resource Negotiator) resource management layer. Yarn is responsible for managing resources and scheduling application in Hadoop cluster.

- Update user's bash profile.
- Launch yarn and HDFS by typing the following commands on your master node —

```
hdfs namenode -format
start-dfs.sh
start-yarn.sh
```