**ASSIGNMENT 14**

1. **Implement a Queue using Arrays**

**Program:**

import java.util.Scanner;

public class Main {

static class Queue {

int front, rear;

int[] arr = new int[100];

Queue() {

front = -1;

rear = -1;

}

void enqueue(int value) {

if (rear == 99) {

System.out.println("Queue Overflow");

} else {

if (front == -1)

front = 0;

rear++;

arr[rear] = value;

System.out.println("Enqueued: " + value);

}

}

void dequeue() {

if (front == -1 || front > rear) {

System.out.println("Queue Underflow");

} else {

System.out.println("Dequeued: " + arr[front]);

front++;

}

}

void display() {

if (front == -1 || front > rear) {

System.out.println("Queue is empty");

} else {

System.out.print("Queue: ");

for (int i = front; i <= rear; i++) {

System.out.print(arr[i] + " ");

}

System.out.println();

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Queue q = new Queue();

while (true) {

System.out.println("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit");

System.out.print("Choose an operation: ");

int choice = sc.nextInt();

switch (choice) {

case 1:

System.out.print("Enter value to enqueue: ");

int val = sc.nextInt();

q.enqueue(val);

break;

case 2:

q.dequeue();

break;

case 3:

q.display();

break;

case 4:

System.out.println("Exiting...");

return;

default:

System.out.println("Invalid choice!");

}

}

}

}

**Output:**

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 1

Enter value to enqueue: 2

Enqueued: 2

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 1

Enter value to enqueue: 1

Enqueued: 1

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 2

Dequeued: 2

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 3

Queue: 1

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 1

Enter value to enqueue: 2

Enqueued: 2

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 2

Dequeued: 1

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 4

Exiting...

**2. Implement a Queue using Linked List**

**Program:**

import java.util.Scanner;

public class Main {

static class Node {

int data;

Node next;

Node(int value) {

data = value;

next = null;

}

}

static class Queue {

Node front, rear;

Queue() {

front = rear = null;

}

void enqueue(int value) {

Node newNode = new Node(value);

if (rear == null) {

front = rear = newNode;

System.out.println("Enqueued: " + value);

return;

}

rear.next = newNode;

rear = newNode;

System.out.println("Enqueued: " + value);

}

void dequeue() {

if (front == null) {

System.out.println("Queue Underflow");

return;

}

System.out.println("Dequeued: " + front.data);

front = front.next;

if (front == null)

rear = null;

}

void display() {

if (front == null) {

System.out.println("Queue is empty");

return;

}

System.out.print("Queue: ");

Node temp = front;

while (temp != null) {

System.out.print(temp.data + " ");

temp = temp.next;

}

System.out.println();

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Queue q = new Queue();

while (true) {

System.out.println("\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit");

System.out.print("Choose an operation: ");

int choice = sc.nextInt();

switch (choice) {

case 1:

System.out.print("Enter value to enqueue: ");

int val = sc.nextInt();

q.enqueue(val);

break;

case 2:

q.dequeue();

break;

case 3:

q.display();

break;

case 4:

System.out.println("Exiting...");

return;

default:

System.out.println("Invalid choice!");

}

}

}

}

**Output:**

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 1

Enter value to enqueue: 2

Enqueued: 2

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 1

Enter value to enqueue: 5

Enqueued: 5

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 1

Enter value to enqueue: 4

Enqueued: 4

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 2

Dequeued: 2

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 3

Queue: 5 4

1. Enqueue

2. Dequeue

3. Display

4. Exit

Choose an operation: 4

Exiting...

1. **Reverse First K Elements of Queue**

**Program:**

import java.util.\*;

public class Main {

public static void reverseFirstKElements(Queue<Integer> queue, int k) {

if (queue.isEmpty() || k > queue.size() || k < 0) {

System.out.println("Invalid value of k");

return;

}

Stack<Integer> stack = new Stack<>();

for (int i = 0; i < k; i++) {

stack.push(queue.poll());

}

while (!stack.isEmpty()) {

queue.add(stack.pop());

}

int size = queue.size();

for (int i = 0; i < size - k; i++) {

queue.add(queue.poll());

}

}

public static void displayQueue(Queue<Integer> queue) {

for (int val : queue) {

System.out.print(val + " ");

}

System.out.println();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Queue<Integer> queue = new LinkedList<>();

System.out.print("Enter number of elements in queue: ");

int n = sc.nextInt();

System.out.println("Enter " + n + " queue elements:");

for (int i = 0; i < n; i++) {

queue.add(sc.nextInt());

}

System.out.print("Enter value of k: ");

int k = sc.nextInt();

reverseFirstKElements(queue, k);

System.out.print("Modified Queue: ");

displayQueue(queue);

}

}

**Output:**

Enter number of elements in queue: 5

Enter 5 queue elements:

1

2

3

4

5

Enter value of k: 3

Modified Queue: 3 2 1 4 5