**Practical 7**

**Name: Shantanu Sethi**

**Roll no: 163**

**Aim: To implement circular queue using array**

1. enqueue

2. dequeue

3. display

**Objectives:**

1. Learn how to implement different operations on circular queue using array

**Theory:**

**Algorithm : enqueue**

**CQINSERT (Q,F,R,N,Y) : Given F and R , pointers to the front and rear elements of circular queue , a queue Q consisting of N elements and y is element which is inserted by this procedure at rear of queue . Initially F and R are set to -1**

1. If (R = max-1 && F =0 )|| R+1= F )

then write( “ overflow” )

return

1. [ Reset rear pointer ]

If (R= max-1)

then R= 0

else R= R + 1

1. [Insert element]

Q[R] = Y

1. [ Is front pointer properly set]

if F= -1

Then F= 0

Return

**Algorithm : Dequeue**

CQDELETE (Q,F,R) : Given F and R , pointers to the front and rear elements of circular queue , This procedure delete element at front of the queue . Y is temporary variable.

1. [ checking for underflow]

If ( F== -1)

then write ( “ underflow”)

return

1. [ Delete element ]

Y= Q[F]

1. [ Queue empty]

if F=R

then F= R = -1

return ( Y)

1. [ Increment front pointer ]

if ( F = max -1 )

then F= 0

else F= F +1

Return( Y )

**● enqueue**

**program:**

package linear;

public class enqueue {

public static void main(String[] args) {

int max = 5;

int[] Q = new int[max];

int F = -1;

int R = -1;

int N = 0;

int Y = 10;

System.*out*.println("184-ABHINAV SINGH");

if ((R == max - 1 && F == 0) || (R + 1 == F)) {

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

} else {

if (R == max - 1) {

R = 0;

} else {

R = R + 1;

}

Q[R] = Y;

if (F == -1) {

F = 0;

}

N++;

System.*out*.println("Inserted element: " + Y);

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

int i = F;

while (i != R) {

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

i = (i + 1) % max;

}

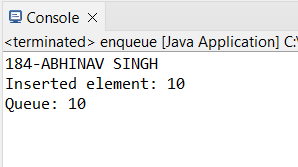
System.*out*.println(Q[R]);

}

}

}

**OUTPUT:-**

****

**● Dequeue**

**program:**

**package** linear;

**import** java.util.Scanner;

**public** **class** dequeue {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("184-ABHINAV SINGH");

System.***out***.print("Enter the maximum size of the circular queue: ");

**int** max = sc.nextInt();

**int**[] Q = **new** **int**[max];

**int** F = -1;

**int** R = -1;

**int** N = 0;

System.***out***.println("\nQueue is ready. Please enter elements to insert into the queue:");

**while** (**true**) {

System.***out***.print("\nEnter an element to insert into the queue (or type -1 to stop adding): ");

**int** Y = sc.nextInt();

**if** (Y == -1) {

**break**;

}

**if** ((R == max - 1 && F == 0) || (R + 1 == F)) {

System.***out***.println("Overflow! Cannot insert element.");

} **else** {

**if** (R == max - 1) {

R = 0;

} **else** {

R = R + 1;

}

Q[R] = Y;

**if** (F == -1) {

F = 0;

}

N++;

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

**int** i = F;

**while** (i != R) {

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

i = (i + 1) % max;

}

System.***out***.println(Q[R]);

}

}

// Now perform the dequeue operation

**while** (**true**) {

System.***out***.print("\nEnter 1 to delete an element from the queue, or type -1 to exit: ");

**int** choice = sc.nextInt();

**if** (choice == -1) {

System.***out***.println("Exiting the program.");

**break**;

}

// Dequeue operation

**if** (F == -1) {

System.***out***.println("Underflow! The queue is empty.");

} **else** {

**int** Y = Q[F];

**if** (F == R) {

F = R = -1;

System.***out***.println("Deleted element: " + Y);

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

} **else** {

**if** (F == max - 1) {

F = 0;

} **else** {

F = F + 1;

}

System.***out***.println("Deleted element: " + Y);

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

**int** i = F;

**while** (i != R) {

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

i = (i + 1) % max; }

System.***out***.println(Q[R]);

}

}

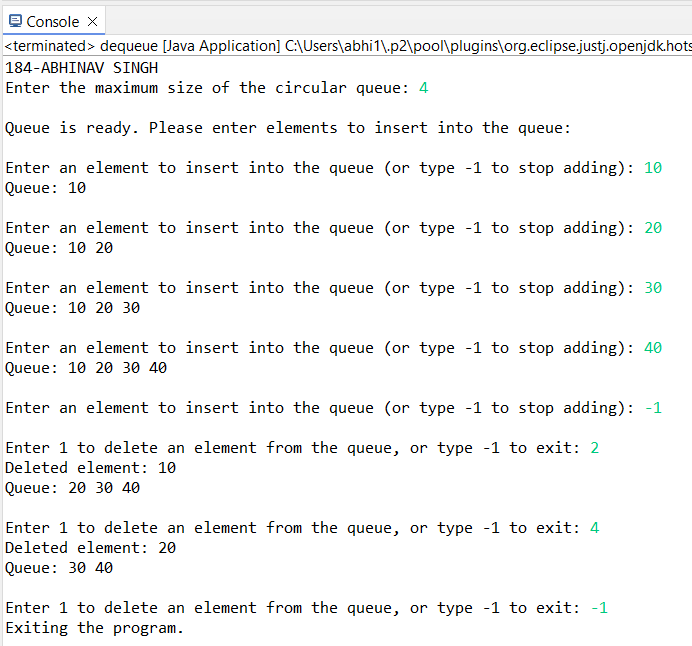
}

sc.close();

}

}

**OUTPUT:-**

****

**● Display**

**program:**

package linear;

import java.util.Scanner;

public class display {

public static void main(String[] args) {

// Create a scanner object for user input

Scanner sc = new Scanner(System.*in*);

System.*out*.println("184-ABHINAV SINGH");

System.*out*.print("Enter the maximum size of the circular queue: ");

int max = sc.nextInt();

int[] queue = new int[max];

int front = -1, rear = -1;

while (true) {

// Menu

System.*out*.println("\n1. Enqueue");

System.*out*.println("2. Dequeue");

System.*out*.println("3. Display");

System.*out*.println("4. Exit");

System.*out*.print("Enter your choice: ");

int choice = sc.nextInt();

switch (choice) {

case 1:

System.*out*.print("Enter the element to enqueue: ");

int item = sc.nextInt();

if ((rear + 1) % max == front) {

System.*out*.println("Overflow! Queue is full.");

} else {

if (front == -1) front = 0;

rear = (rear + 1) % max;

queue[rear] = item;

System.*out*.println(item + " enqueued.");

}

break;

case 2:

if (front == -1) {

System.*out*.println("Underflow! Queue is empty.");

} else {

System.*out*.println(queue[front] + " dequeued.");

if (front == rear) {

front = rear = -1;

} else {

front = (front + 1) % max;

}

}

break;

case 3:

if (front == -1) {

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

} else {

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

int i = front;

while (i != rear) {

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

i = (i + 1) % max;

}

System.*out*.println(queue[rear]);

}

break;

case 4:

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

sc.close();

return;

default:

System.*out*.println("Invalid choice! Try again.");

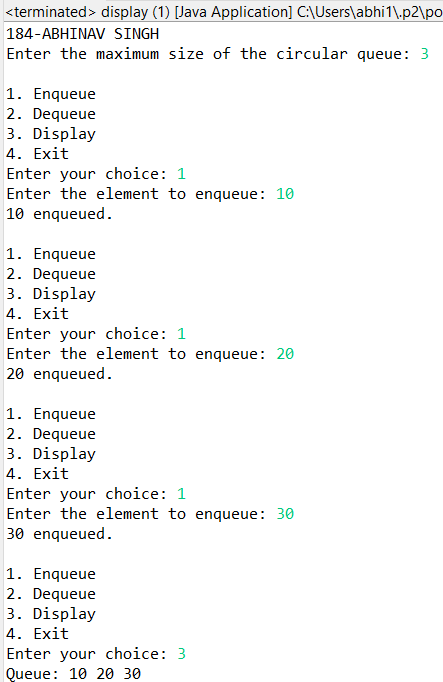
}

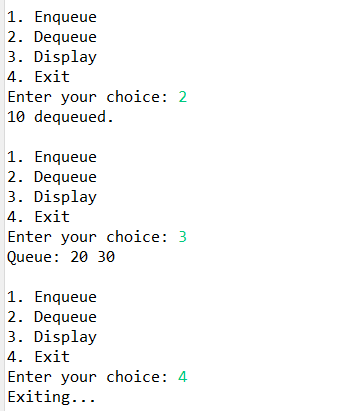
}

}

}

**OUTPUT:-**

****

****

**Conclusion**: Circular queue is linear data structure which work on First In First Out principle.