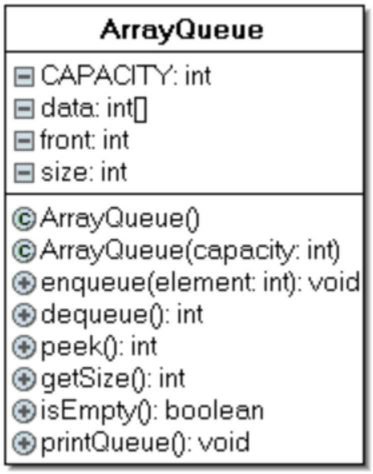
Data Structures and Algorithms Laboratory		
Laboratory 7: Queues	**School of Information Technology**	
Name:Sai Hae Naing Lay	**ID:6531501208**	**Section:4**
Date: 25 March 2023	**Due date: on the LMS**	

Objective

- To create the queue program
- To implement the queue solution to solve the problem.
- To implement the circular-array queue.

Exercise 1: (in-class) From the given class diagram, create an array-based queue and complete the program to get the results as shown.

Class Diagram:

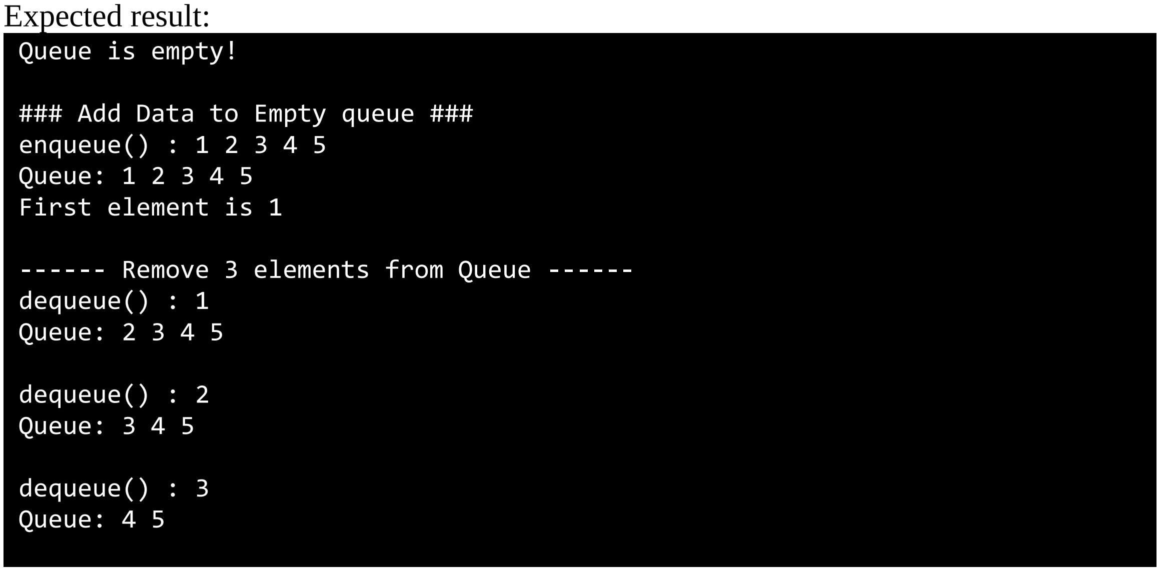


ArrayQueue

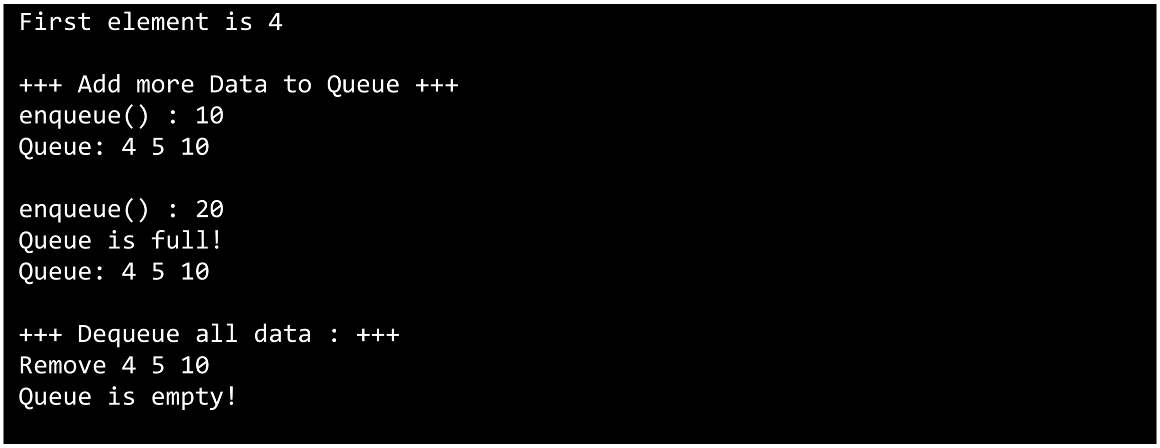
CAPACITY: int
data: int[]
front: int
size: int

ArrayQueue()
ArrayQueue(capacity: int)
enqueue(element: int): void
dequeue(): int
peek(): int
getSize(): int
isEmpty(): boolean
printQueue(): void

Expected result:



```
Queue is empty!

### Add Data to Empty queue ###
enqueue() : 1 2 3 4 5
Queue: 1 2 3 4 5
First element is 1

------ Remove 3 elements from Queue ------
dequeue() : 1
Queue: 2 3 4 5

dequeue() : 2
Queue: 3 4 5

dequeue() : 3
Queue: 4 5
```



```
First element is 4

+++ Add more Data to Queue +++
enqueue() : 10
Queue: 4 5 10

enqueue() : 20
Queue is full!
Queue: 4 5 10

+++ Dequeue all data : +++
Remove 4 5 10
Queue is empty!
```

Complete source codes below.

Class ArrayQueue

```
package Queuing;

public class ArrayQueue

{

private final int CAPACITY = 10;

private int[] data; //array to store the data of queue

private int front = 0; //pointer for the first element

private int size = 0; //size of queue(no. of the element)

public ArrayQueue()

{

data = new int[CAPACITY];

}

public ArrayQueue(int capacity)

{

data = new int[capacity];

}

public void enqueue(int element)
```

```
{
if((front + size) == data.length) //check the size of the queue
System.out.println("Queue is full!");
else
{
//add the element to the rear
data[front+size] = element;
size++;
}
}
public int dequeue()
{
if(size == 0)
return -999;//null
else
{
int element = data[front];
front++;
size--;
return element;
//note:
/*
size--;
return data[front++];
*/
}
}
public int peek()
{
if(size == 0)
return -999; //null
else
return data[front];
}
public int getSize() //get the current size (no. of the element)
{
return size;
}
public boolean isEmpty() //is the queue empty?
{
if(size == 0)
return true;
else
return false;
}
public void printQueue()//print all the members in the queue
{
if(size == 0)
System.out.println("Queue is empty!");//if size is empty
else
```

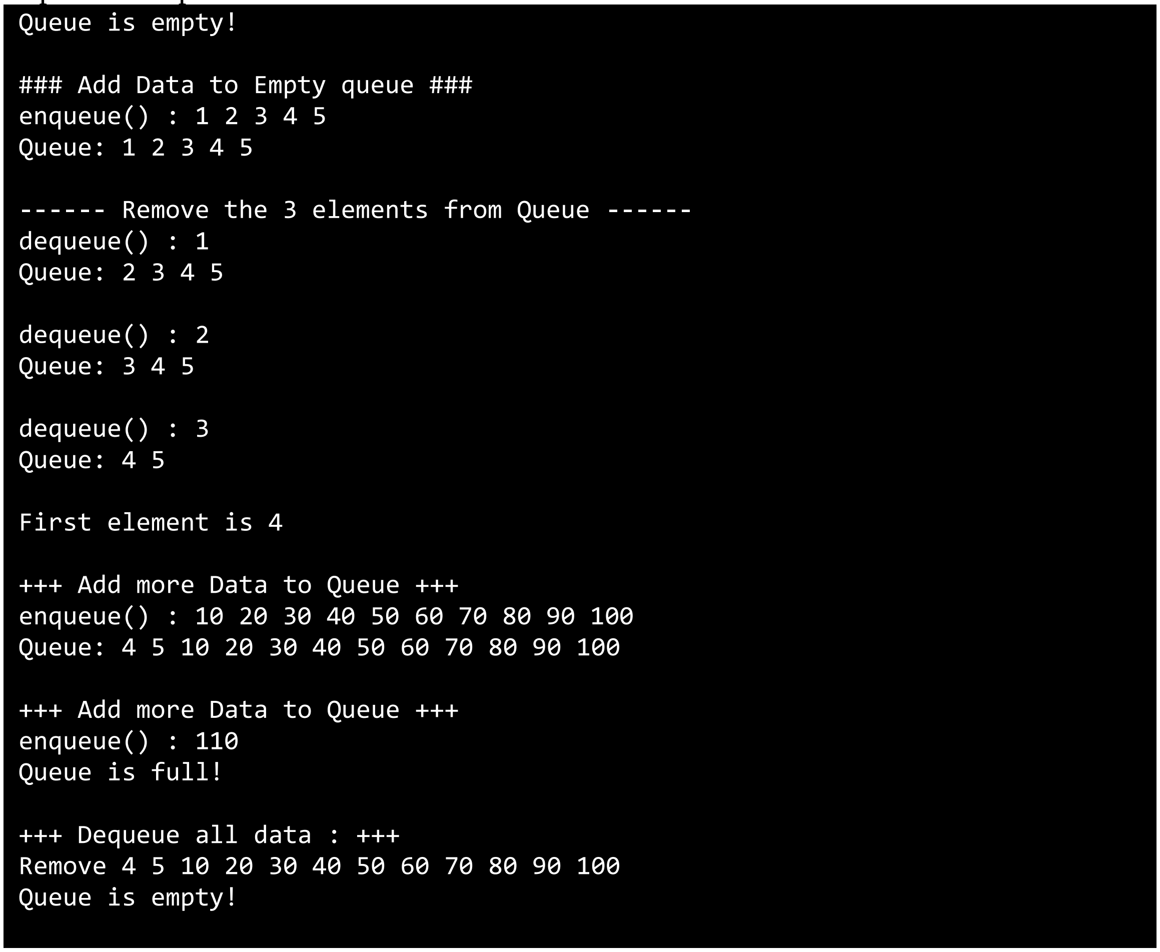
```
{

System.out.print("Queue: "); // if no print all members

for(int i = front; i < (front + size); i++)

{

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

}

System.out.println();

}

}

}
```

Class MainArrayQueue

```java
public class MainArrayQueue {

    public static void main(String[] args) {
        ArrayQueue queue = new ArrayQueue(6);
        queue.printQueue();
        System.out.println();

        // Enqueue 5 elements
        System.out.println("### Add Data to Empty queue ###");
        System.out.print("enqueue() :");
        for (int i = 1; i <= 5; i++) {
            queue.enqueue(i);
            System.out.print(" " + i);
        }
        System.out.println();
        queue.printQueue();

        //Peek
        System.out.println("First element is " + queue.peek());
        System.out.println();

        // Remove 3 elements
        System.out.println("------ Remove 3 elements from Queue ------");
        for (int i = 1; i <= 3; i++) {
            System.out.println("dequeue() : " + queue.dequeue());
            queue.printQueue();
            System.out.println();
        }

        //Peek
        System.out.println("First element is " + queue.peek());
        System.out.println();

        // Add 2 more elements
        System.out.println("+++ Add more Data to Queue +++");
        System.out.println("enqueue() : 10");
        queue.enqueue(10);
        queue.printQueue();
        System.out.println();

        System.out.println("enqueue() : 20");
        queue.enqueue(20);
        queue.printQueue();
        System.out.println();

        System.out.println("+++ Dequeue all data : +++");
        System.out.print("Remove ");
        while (!queue.isEmpty()) {
            System.out.print(queue.dequeue() + " ");
        }
        System.out.println();
        queue.printQueue();
        }
}
```

<u>Exercise 2</u>: (in-class) Apply the concept of "Circular Array (-Based) Queues" to modify the source code in 'Exercise 1'.

Expected Output:



```
Queue is empty!

### Add Data to Empty queue ###
enqueue() : 1 2 3 4 5
Queue: 1 2 3 4 5

------ Remove the 3 elements from Queue ------
dequeue() : 1
Queue: 2 3 4 5

dequeue() : 2
Queue: 3 4 5

dequeue() : 3
Queue: 4 5

First element is 4

+++ Add more Data to Queue +++
enqueue() : 10 20 30 40 50 60 70 80 90 100
Queue: 4 5 10 20 30 40 50 60 70 80 90 100

+++ Add more Data to Queue +++
enqueue() : 110
Queue is full!

+++ Dequeue all data : +++
Remove 4 5 10 20 30 40 50 60 70 80 90 100
Queue is empty!
```

Fill in the missing codes.

Class CircularArrayQueue

```java
package Queuing;

public class CircularArrayQueue

{

//----------------- data --------------

private final int CAPACITY = 10;

private int[] data; //array to store queue data
private int front = 0; //pointer for first queue element
private int size = 0;//size of queue (no. of elements)
//----------------- method --------------
public CircularArrayQueue()
{
data = new int[CAPACITY];
}
public CircularArrayQueue(int capacity)
{
data = new int[capacity];
}
public void enqueue(int element)
{
if( size == data.length)
System.out.println("Queue is full!");
else
{
//add element to the rear
data[(front + size) % data.length] = element;
size++;
}
}
public int dequeue()
{
if(size == 0)
```

```
return -999;
else
{
int element = data[front];
front = (front + 1) % data.length;
size--;
return element;
}
}
public int peek()
{
if(size == 0)
return -999; //null
else
return data[front];
}
public int getSize() //get the current size (no. of the element)
{
return size;
}
public boolean isEmpty() //is the queue empty?
{
if(size == 0)
return true;
else
```

```
return false;
}
public void printQueue()
{
if(size == 0)
System.out.println("Queue is empty!");
else
{
for(int i = front; i <((front + size) % data.length) ; i++)
{
System.out.print(data[i] + " ");
}
System.out.println();
}
}
}
}
```

Class MainCircularArrayQueue

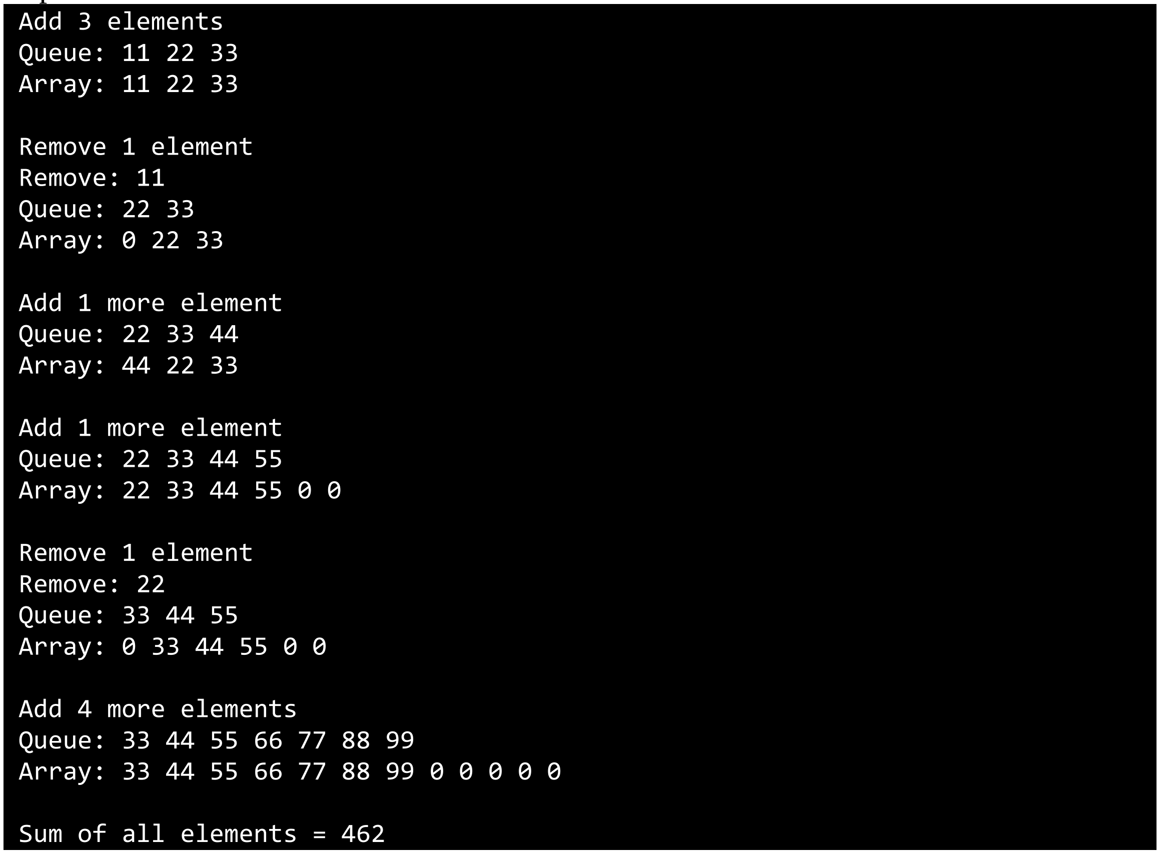
```
public class MainCircularArrayQueue {
	public static void main(String[] args) {
		CircularArrayQueue queue = new CircularArrayQueue(12);
		queue.printQueue();
		System.out.println();

		// Enqueue 5 elements
		System.out.println("### Add Data to Empty queue ###");
		System.out.print("enqueue() :");
		for (int i = 1; i <= 5; i++) {
			queue.enqueue(i);
			System.out.print(" " + i);
		}
		System.out.println();
		queue.printQueue();
		System.out.println();
```

```
        // Dequeue 3 elements
        System.out.println("------ Remove the 3 elements from Queue ------");
        for (int i = 1; i <= 3; i++) {
            System.out.println("dequeue() : " + queue.dequeue());
            queue.printQueue();
            System.out.println();
        }

        // Peek
        System.out.println("First element is " + queue.peek());
        System.out.println();

        // Add 10 more elements
        System.out.println("+++ Add more Data to Queue +++");
        System.out.print("enqueue() :");
        for (int i = 10; i <= 100; i += 10) {
            queue.enqueue(i);
            System.out.print(" " + i);
        }
        System.out.println();
        queue.printQueue();
        System.out.println();

        // Test queue full
        System.out.println("+++ Add more Data to Queue +++");
        System.out.println("enqueue() : 110");
        queue.enqueue(110);

        System.out.println("\n+++ Dequeue all data : +++");
        System.out.print("Remove ");
        while (!queue.isEmpty()) {
            System.out.print(queue.dequeue() + " ");
        }
        System.out.println();
        queue.printQueue();
    }
}
```

Exercise 3: (Homework) Modify the circular array-based queue in the previous exercise so that

- When dequeue, the removed array element becomes 0.
- It has a new function called “sum()” to find summation of all integer members in the queue.
- It has a new function called “printArray()” to print the whole data array from the first to last element.
- It can be expanded by two times of the previous capacity when it is full. This could be done by creating a new function “resize()” that will be called whenever the “enqueue()” method found that the queue is full.

Expected result



```
Add 3 elements
Queue: 11 22 33
Array: 11 22 33

Remove 1 element
Remove: 11
Queue: 22 33
Array: 0 22 33

Add 1 more element
Queue: 22 33 44
Array: 44 22 33

Add 1 more element
Queue: 22 33 44 55
Array: 22 33 44 55 0 0

Remove 1 element
Remove: 22
Queue: 33 44 55
Array: 0 33 44 55 0 0

Add 4 more elements
Queue: 33 44 55 66 77 88 99
Array: 33 44 55 66 77 88 99 0 0 0 0 0

Sum of all elements = 462
```

Class CircularArrayQueue2

```
//Extended version with sum() and auto double-sized capacity


//------------------ data --------------

private final int CAPACITY = 10;
private int[] data; //array to store queue data
private int front = 0; //pointer for first queue element
private int size = 0;//size of queue (no. of elements)
//----------------- method --------------
public CircularArrayQueue()
{
data = new int[CAPACITY];
}
public CircularArrayQueue(int capacity)
{
data = new int[capacity];
}
public void enqueue(int element)
{
if(size == data.length)
{
resize();
}
//add element to the rear
data[(front + size) % data.length] = element;
size++;
}
public int dequeue()
{
if(size == 0)
```

```
{

return -999;

}

else

{

int element = data[front];

data[front] = 0;

front = (front + 1) % data.length;

size--;

return element;

}

}

public int peek()

{

if(size == 0)

return -999; //null

else

return data[front];

}

public int getSize() //get the current size (no. of the element)

{

return size;

}

public boolean isEmpty() //is the queue empty?

{
```

```
if(size == 0)
return true;
else
return false;
}
public void printQueue()
{
if(size == 0)
System.out.println("Queue is empty!");
else
{
System.out.print("Queue: ");
for(int i = 0; i < size; i++)
{
int j = (front + i) % data.length;
System.out.print(data[j] + " ");
}
System.out.println();
}
}
//print real array
public void printArray()
{
System.out.print("Array: ");
for(int i = 0; i < data.length; i++)
```

```
{

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

}

System.out.println();

}

//find all sum elements

public int sum()

{

int sum = 0;

for(int i = front; i <((front + size) % data.length); i++)

{

int j = (front + i) % data.length;

sum += data[j];

}

return sum;

}

public void resize()

{

int[] newData = new int[data.length * 2];

for (int i = 0; i < size; i++)

{

newData[i] = data[front];

front = (front + 1) % data.length;

}

data = newData;
```

```
front = 0;

}

}
```

Class MainCircularArrayQueue2

```
public class MainCircularArrayQueue2 {

        public static void main(String[] args) {
                CircularArrayQueue2 queue = new CircularArrayQueue2(3);
                System.out.println("Add 3 elements");
                queue.enqueue(11);
                queue.enqueue(22);
                queue.enqueue(33);
                queue.printQueue();
                queue.printArray();
                System.out.println();

                System.out.println("Remove 1 element");
                System.out.println("Remove: " + queue.dequeue());
                queue.printQueue();
                queue.printArray();
                System.out.println();

                System.out.println("Add 1 more element");
                queue.enqueue(44);
                queue.printQueue();
                queue.printArray();
                System.out.println();

                System.out.println("Add 1 more element");
                queue.enqueue(55);
                queue.printQueue();
                queue.printArray();
                System.out.println();

                System.out.println("Remove 1 element");
                System.out.println("Remove: " + queue.dequeue());
                queue.printQueue();
                queue.printArray();
                System.out.println();

                System.out.println("Add 4 more elements");
                queue.enqueue(66);
                queue.enqueue(77);
                queue.enqueue(88);
                queue.enqueue(99);
                queue.printQueue();
                queue.printArray();
                System.out.println();
                System.out.println("Sum of all elements = " + queue.sum());
        }
}
```