Job Sheet 10 Queue

From:

AL AZHAR RIZQI RIFA'I FIRDAUS

Class:

1 I

Absence:

01

Student Number Identity:

2241720263

Department:

Information Technology

Study Program:

Informatics Engineering

1.1. Learning Objective

After finishing this topic, students must be able to:

1. Understand the basic concept of Queue

2. Understand the basic operation of Queue

3. Implement the Queue concept as well as the operation in a program by using Java

1.2. Lab Activities #1

1.2.1. Steps

1. We will create a program based on this following class diagram

Queue
max: int size: int front: int rear: int Q: int[]
Queue(max: int) create(): void isFull(): boolean isEmpty(): boolean enqueue(data: int): void dequeue(): int peek: void print(): void clear(): void

2. Create a new project named Jobsheet8, create a new package with name practicum1.

After that, create a new class Queue

3. Add max, size, front,rear and Q as its attributes based on the class diagram above.

4. Add a constructor with parameter and method create as illustration below.

```
public Queue(int n) {
    max = n;
    Create();
}
```

Within the constructor, there is a code to execute create(). then we make the create

function

```
public void Create() {
    Q = new int[max];
    size = 0;
    front = rear = -1;
}
```

5. Create a method isEmpty with boolean as its return data type. We use this function to identify whether a queue is empty or not

```
public boolean IsEmpty() {
    if (size == 0) {
        return true;
    } else {
        return false;
    }
}
```

6. Create a method isFull with boolean as its return data type. We use this function to identify whether a queue is full or not

```
public boolean IsFull() {
    if (size == max) {
        return true;
    } else {
        return false;
    }
}
```

7. Create peek() with void as its return type to display the data at the beginning of the queue

```java
public void peek(){
    if(!isEmpty()){
        System.out.println("The first element : " + Q[front]);
    }else{
        System.out.println("Queue is still empty");
    }
}
```

8. Create print() with void as its return type to display the data from the beginning until the end of the queue

```java
public void print(){
    if(!isEmpty()){
        System.out.println("Queue is still empty");
    }else{
        int i = front;
        while(i != rear){
            System.out.println(Q[i] + " ");
            i = (i+1) % max;
        }
        System.out.println( Q[i] + " ");
        System.out.println("Element amount : " + size);
    }
}
```

9. Create clear() with void as its return type to remove all data in a queue

```java
public void clear(){
    if(!isEmpty()){
        front = rear = -1;
        size = 0;
        System.out.println("Queue has been cleared successfully");
    }else{
        System.out.println("Queue is still empty");
    }
}
```

10. Next up, we make a new method enqueue() to insert a new data in a queue that has integer datatype as its parameter

```java
public void Enqueue(int data){
    if(isFull()){
        System.out.println("Queue is already full");
    }else{
        if(isEmpty()){
            front = rear = 0;
        }else{
            if(rear == max - 1){
                rear = 0;
            }else{
                rear++;
            }
        }
        Q[rear] = data;
        size++;
    }
}
```

11. Create method dequeue() with integer as its return type. We will need this function whenever we want to remove the last data inside the queue

```java
public int Dequeue(){
    int data = 0;
    if(isEmpty()){
        System.out.println("Queue is still empty");
    }else{
        data = Q[front];
        size--;
        if(isEmpty()){
            front = rear = -1;
        }else{
            if(front == max - 1){
                front = 0;
            }else{
                front++;
            }
        }
    }
    return data;
}
```

12. Next, we create a new class named QueueMain still at the same package Practicum1. To

create a menu with void return type to allow user choose which function to be executed

when the program runs

```java
public static void menu(){
    System.out.println("Choose menu: ");
    System.out.println("1. Enqueue");
    System.out.println("2. Dequeue");
    System.out.println("3. Print");
    System.out.println("4. Peek");
    System.out.println("5. Clear");
    System.out.println("========================");
}
```

13. Create a main function, and declare the Scanner object with name sc

14. Create n variable to store input of how many elements that can be stored within the

queue

queue

```
Scanner sc = new Scanner(System.in);
System.out.print("Insert maximum queue : ");
int n = sc.nextInt();
```

15. Instantiate the Queue object with name Q and set the parameter n as its queue length

```
Queue Q = new Queue(n);
```

16. Declare the input of menu selected by user

17. Loop with do-while to run the program based on the given input. Inside the loop, there is a switch case condition to manipulate queue based on the given input

```java
int choose;
do {
    menu();
    choose = sc.nextInt();
    switch(choose){
        case 1:
            System.out.print("Insert new data: ");
            int newData = sc.nextInt();
            Q.Enqueue(newData);
            break;
        case 2:
            int removeData = Q.Dequeue();
            if(removeData != 0){
                System.out.println("Data removed : " + removeData);
                break;
            }
        case 3:
            Q.print();
            break;
        case 4:
            Q.peek();
            break;
        case 5:
            Q.clear();
            break;
    }
} while (choose <= 5 && choose >= 1);
```

18. Compile the program and run the QueueMain class. And observe the result

Result

Check if the result match with following image:

```
run:
Insert maximum queue : 4
Choose menu:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
=======================
1
Insert new data: 15
Choose menu:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
=======================
1
Insert new data: 31
Choose menu:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
=======================
4
The first element : 15
```

code :

main

```java
package practicum1;

import java.util.Scanner;

public class QueueMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Insert maximum queue : ");
        int n = sc.nextInt();
        Queue Q = new Queue(n);

        int choose;
        do {
            menu();
            choose = sc.nextInt();
            switch (choose) {
                case 1:
                    System.out.print("Insert new data : ");
                    int newData = sc.nextInt();
                    Q.Enqueue(newData);
                    break;
                case 2:
                    int removeData = Q.Dequeue();
                    if (removeData != 0) {
                        System.out.print("Data removed : " + removeData);
                        break;
                    }
                case 3:
                    Q.print();
                    break;
                case 4:
                    Q.peek();
                    break;
                case 5:
                    Q.clear();
                    break;
                default:
                    break;
            }
        } while (choose <= 5 && choose >= 1);

        sc.close();
    }

    public static void menu() {
        System.out.println("Choose Menu : ");
        System.out.println("1. Enqueue");
        System.out.println("2. Dequeue");
        System.out.println("3. Print");
        System.out.println("4. Peek");
        System.out.println("5. Clear");
        System.out.println("====================");
    }
}
```

Queue

```java
package practicum1;

public class Queue {
    public int max, size, front, rear;
    public int [] Q;

    public Queue(int n) {
        max = n;
        create();
    }

    public void create() {
        Q = new int[max];
        size = 0;
        front = rear = -1;
    }

    public boolean isEmpty() {
        if (size == 0) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isFull() {
        if (size == max) {
            return true;
        } else {
            return false;
        }
    }

    public void peek() {
        if (!isEmpty()) {
            System.out.println("The first element : " + Q[front]);
        } else {
            System.out.println("Queue is still empty");
        }
    }

    public void print() {
        if (!isEmpty()) {
            System.out.println("Queue is still empty");
        } else {
            int i = front;
            while (i != rear) {
                System.out.println(Q[i] + " ");
                i = (i+1) % max;
            }
            System.out.println(Q[i] + " ");
            System.out.println("Element amount : " + size);
        }
    }

    public void clear() {
        if (!isEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Queue has been cleared successfully");
        } else {
            System.out.println("Queue is still empty");
        }
    }

    public void Enqueue(int data) {
        if (isFull()) {
            System.out.println("Queue is already full");
        } else {
            if (isEmpty()) {
                front = rear = 0;
            } else {
                if (rear == max -1) {
                    rear = 0;
                } else {
                    rear++;
                }
            }
            Q[rear] = data;
            size++;
        }
    }

    public int Dequeue() {
        int data = 0;
        if (isEmpty()) {
            System.out.println("Queue is still empty");
        } else {
            data = Q[front];
            size--;
            if (isEmpty()) {
                front = rear = -1;
            } else {
                if (front == max -1) {
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return data;
    }
}
```

result

```
Insert maximum queue : 4
Choose Menu :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
====================
1
Insert new data : 15
Choose Menu :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
====================
1
Insert new data : 31
Choose Menu :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
====================
4
The first element : 15
```

Questions

1. In method create(), why is the front and rear attribute has an initial value with 1 and not 0?

- **The initial values of front and rear are set to -1, not 1. By setting front and rear to -1 initially, the code can indicate an empty queue. If front and rear were initialized to 0, it could mean a queue with one element.**

2. In method enqueue(), please explain the usage of this following code

```
if (rear == max - 1) {
    rear = 0;
```

- **The condition rear == max - 1 checks if the rear pointer is at the last index of the array (max - 1), which means it has reached the end of the array. If this condition is true, it means the next element should be inserted at the beginning of the array to maintain the circular nature of the queue. Therefore, the rear is set to 0 in this case.**

3. Observe enqueue() method, which line of code indicates that the new data will be stored in last position of the queue?

```
Q[rear] = data;
```

4. Observe dequeue() method, which line of code indicates that the data is removed in the first position of the queue?

```
data = Q[front];
```

5. In dequeue method(), explain the usage of these codes !

```
if (front == max - 1) {
    front = 0;
```

- **The condition front == max - 1 checks if the front pointer is at the last index of the array (max - 1), which indicates that it has reached the end of the array. By setting front to 0 in this case, it ensures that the next element to be dequeued will come from the start of the array.**

6. In method print(), why the loop process has int i = 0 instead of int i=front?

- **No line of code has value int i = 0, but by starting the loop with int i = front, the code correctly begins printing the elements from the front position, and then continues looping through the array in a circular manner until it reaches the rear position.**

7. In method print(), please explain why we insert this code in our program?

```
i = (i + 1) % max;
```

- **In the print() method, the line of code i = (i+1) % max is used to increment the i variable in a circular manner during the loop iteration. The purpose of this line is to ensure that the loop iterates through the circular queue correctly, wrapping around to the beginning of the array if it reaches the end.**

1.3. Lab Activities #2

In this lab activity, we will create a train ticket payment simple program with implementing the properties adjusted in railway stations environment

1.3.1. Steps

Passengers
name: String cityOrigin: String cityDestination: String ticketAmount: int price: int
Passengers(name: String, cityOrigin: String, cityDestination: String, ticketAmount: int, price: int)

1. Based on above class diagram, we will create a program written in Java
2. Create a new package named Practicum2 and create a new class named Passengers
3. Add attributes for Passengers based on above class diagram, add the constructor as well
4. Copy the program code written in Queue in 1st Lab Activities to be reused in this package. We will need to modify the class, since the stored value in 1st lab activity is in integer data type, but we need it to store an object
5. Modify the class Queue, we change the data type int[] Q into Passenger[] Q. Since this case we will need to store Passenger object in the queue. In addition, we will need

 to modify attributes, create(), enqueue(), and dequeue()

```
int max,size,front,rear;
Passenger[] Q;

public void Create(){
    Q = new Passenger[max];
    size = 0;
    front = rear = -1;
}
```

```java
public void Enqueue(Passenger data){
    if(isFull()){
        System.out.println("Queue is already full");
    }else{
        if(isEmpty()){
            front = rear = 0;
        }else{
            if(rear == max - 1){
                rear = 0;
            }else{
                rear++;
            }
        }
        Q[rear] = data;
        size++;
    }
}
```

```java
public Passenger Dequeue(){
    Passenger data = new Passenger("", "", "", 0, 0);
    if(isEmpty()){
        System.out.println("Queue is still empty");
    }else{
        data = Q[front];
        size--;
        if(isEmpty()){
            front = rear = -1;
        }else{
            if(front == max - 1){
                front = 0;
            }else{
                front++;
            }
        }
    }
    return data;
}
```

6.Because one element in queue holds some information (name, cityOrigin, cityDestination, ticketAmount, price), we are needed to display all of that information. This leads to modifying the peek() and print() as well

```java
public void peek(){
    if(!isEmpty()){
        System.out.println("The first element : " + Q[front].name + " "
        + Q[front].cityOrigin + " " + Q[front].cityDestination + " " +
        Q[front].ticketAmount + " " + Q[front].price);
    }else{
        System.out.println("Queue is still empty");
    }
}
```

```java
public void print(){
    if(!isEmpty()){
        System.out.println("Queue is still empty");
    }else{
        int i = front;
        while(i != rear){
            System.out.println("The first element : " + Q[front].name + " "
        + Q[front].cityOrigin + " " + Q[front].cityDestination + " " +
        Q[front].ticketAmount + " " + Q[front].price);
            i = (i+1) % max;
        }
        System.out.println( Q[i] + " ");
        System.out.println("Element amount : " + size);
    }
}
```

7.Next, create a new class named QueueMain within the Practicum2 package. Create menu() to provide menu options and allow the user to choose the menu when the program runs

```java
public static void menu(){
    System.out.println("Choose menu: ");
    System.out.println("1. Queue");
    System.out.println("2. Dequeue");
    System.out.println("3. Check first queue");
    System.out.println("4. Check all queue");
    System.out.println("=======================");
}
```

8. Create main method in the QueueMain, and declare the Scanner object with name sc

9. Create max variable to define the capacity of the queue. After that, instantiate queue object with name queuePassenger with its parameter is max

10. Declare a variable named choose with integer as its datatype to get which option did the user choose.

11. Add these following codes to loops menu options according to given input by the user.

```
int choose;
do {
    menu();
    choose = sc.nextInt();
    switch(choose){
        case 1:
            System.out.print("Name: ");
            String nm = sc.nextLine();
            System.out.print("City origin: ");
            String cOrg = sc.nextLine();
            System.out.print("City Desitnation: ");
            String cDes = sc.nextLine();
            System.out.print("Ticket Amount: ");
            int ticket = sc.nextInt();
            System.out.print("Price: ");
            int price = sc.nextInt();
            Passenger p = new Passenger(nm, cOrg, cDes, price, ticket);
            sc.nextLine();
            queuePassenger.Enqueue(p);
            break;
        case 2:
            Passenger data = queuePassenger.Dequeue();
            if(!"".equals(data.name) && !"".equals(data.cityOrigin) &&
                    !"".equals(data.cityDestination) && !"".equals(data.ticketAmount)
                    && !"".equals(data.price)){
                System.out.println("Data removed : " + data.name + " " + data.cityOrigin
                + " " + data.cityDestination + " " + data.ticketAmount + " "
                + data.price);
                break;
            }
        case 3:
            queuePassenger.print();
            break;
        case 4:
            queuePassenger.peek();
            break;
        case 5:
            queuePassenger.clear();
            break;
    }
} while (choose <= 4 && choose >= 1);
```

12. Compile the program and run the QueueMain class. And observe the result

1.3.2. Result

Check if the result match with following image:

```
run:
Insert maximum queue : 5
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
========================
1
Name: Angga
City origin: Solo
City Desitnation: Sidoarjo
Ticket Amount: 2
Price: 176000
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
========================
1
Name: Fadin
City origin: Banyuwangi
City Desitnation: Bandung
Ticket Amount: 1
Price: 65000
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
========================
3
The first element : Angga Solo Sidoarjo 2 176000
```

Code :

Queue Passengers

```java
package practicum2;

public class QueuePassengers {

    public int max, size, front, rear;
    public Passengers [] Q;

    public QueuePassengers(int n) {
        max = n;
        create();
    }

    public void create() {
        Q = new Passengers[max];
        size = 0;
        front = rear = -1;
    }

    public boolean isEmpty() {
        if (size == 0) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isFull() {
        if (size == max) {
            return true;
        } else {
            return false;
        }
    }

    public void peek() {
        if (!isEmpty()) {
            System.out.println("The first element : " + Q[front].name + " " + Q[front].cityOrigin + " " + Q[front].cityDestination + " " + Q[front].ticketAmount + " " + Q[front].price);
        } else {
            System.out.println("Queue is still empty");
        }
    }

    public void print() {
        if (!isEmpty()) {
            System.out.println("Queue is still empty");
        } else {
            int i = front;
            while (i != rear) {
                System.out.println("The first element : " + Q[front].name + " " + Q[front].cityOrigin + " " + Q[front].cityDestination + " " + Q[front].ticketAmount + " " + Q[front].price);
                i = (i+1) % max;
            }
            System.out.println(Q[i] + " ");
            System.out.println("Element amount : " + size);
        }
    }

    public void clear() {
        if (!isEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Queue has been cleared successfully");
        } else {
            System.out.println("Queue is still empty");
        }
    }

    public void Enqueue(Passengers data) {
        if (isFull()) {
            System.out.println("Queue is already full");
        } else {
            if (isEmpty()) {
                front = rear = 0;
            } else {
                if (rear == max -1) {
                    rear = 0;
                } else {
                    rear++;
                }
            }
            Q[rear] = data;
            size++;
        }
    }

    public Passengers Dequeue() {
        Passengers data = new Passengers("", "", "", 0, 0);
        if (isEmpty()) {
            System.out.println("Queue is still empty");
        } else {
            data = Q[front];
            size--;
            if (isEmpty()) {
                front = rear = -1;
            } else {
                if (front == max -1) {
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return data;
    }

}
```

Queue Main

```java
package practicum2;

import java.util.Scanner;

public class QueueMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Insert maximum queue : ");
        int max = sc.nextInt();
        QueuePassengers queuePassengers = new QueuePassengers(max);

        int choose;
        do {
            menu();
            choose = sc.nextInt();
            switch (choose) {
                case 1:
                    System.out.print("Name : ");
                    String name = sc.next();
                    sc.nextLine();
                    System.out.print("City origin : ");
                    String cityOrigin = sc.next();
                    sc.nextLine();
                    System.out.print("City destination : ");
                    String cityDestination = sc.next();
                    sc.nextLine();
                    System.out.print("Ticket amount : ");
                    int ticketAmount = sc.nextInt();
                    System.out.print("Price : ");
                    int price = sc.nextInt();
                    Passengers p = new Passengers(name, cityOrigin, cityDestination, ticketAmount, price);
                    sc.nextLine();
                    queuePassengers.Enqueue(p);
                    break;
                case 2:
                    Passengers data = queuePassengers.Dequeue();
                    if (!"".equals(data.name) && !"".equals(data.cityOrigin) && !"".equals(data.cityDestination) && !"".equals(data.ticketAmount) && !"".equals(data.price)) {
                        System.out.println("Data removed : " + data.name + " " + data.cityOrigin + " " + data.cityDestination + " " + data.ticketAmount + " " + data.price);
                        break;
                    }
                case 3:
                    queuePassengers.peek();
                    break;
                case 4:
                    queuePassengers.print();

                    break;
                case 5:
                    queuePassengers.clear();
                    break;
                default:
                    break;
            }
        } while (choose <= 4 && choose >= 1);

        sc.close();
    }

    public static void menu() {
        System.out.println("Choose Menu : ");
        System.out.println("1. Enqueue");
        System.out.println("2. Dequeue");
        System.out.println("3. Check first queue");
        System.out.println("4. Check all queue");
        System.out.println("5. Clear");
        System.out.println("====================");
    }
}
```

Passengers

```java
package practicum2;

public class Passengers {
    public String name, cityOrigin, cityDestination;
    public int ticketAmount, price;

    public Passengers(String name, String cityOrigin, String cityDestination, int ticketAmount, int price) {
        this.name = name;
        this.cityOrigin = cityOrigin;
        this.cityDestination = cityDestination;
        this.ticketAmount = ticketAmount;
        this.price = price;
    }
}
```

Result

```
1. Enqueue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear
====================
1
Name : Angga
City origin : Solo
City destination : Sidoarjo
Ticket amount : 2
Price : 176000
Choose Menu :
1. Enqueue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear
====================
1
Name : Fadin
City origin : Banyuwangi
City destination : Bandung
Ticket amount : 1
Price : 65000
Choose Menu :
1. Enqueue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear
====================
3
The first element : Angga Solo Sidoarjo 2 176000
```

Questions

1. In Queue Class, what's the function of this program code in method Dequeue?

```java
Passenger data = new Passenger("", "", "", 0, 0);
```

- **The line Passengers data = new Passengers("", "", "", 0, 0); creates a new instance of the Passengers class with empty or default values for its attributes. It initializes a Passengers object named data with an empty string for the passenger's name, an empty string for the passenger's destination, an empty string for the passenger's seat number, and 0 for both the passenger's age and ticket number.**

2. In previous number, if the program code changed to

Passenger data = new Passenger()

What will happen?

- **If the program code is changed to Passenger data = new Passenger();, it would result in a compilation error unless there is a parameterless constructor defined in the Passenger class.**

3. Show the program code used for displaying the data retrieved / removed from the queue!

```java
public Passengers Dequeue() {
    Passengers data = new Passengers("", "", "", 0, 0);
    if (isEmpty()) {
        System.out.println("Queue is still empty");
    } else {
        data = Q[front];
        size--;
        if (isEmpty()) {
            front = rear = -1;
        } else {
            if (front == max -1) {
                front = 0;
            } else {
                front++;
            }
        }
    }
    return data;
}
```

4. Modify the program by adding a method named peekRear() in Queue class to check the last position within the queue. Add a menu for the user to perform and explore your program as well

```java
public void peekRear() {
    if (!isEmpty()) {
        System.out.println("Last element: " + Q[rear].name + " " + Q[rear].cityOrigin + " " + Q[rear].cityDestination + " " + Q[rear].ticketAmount + " " + Q[rear].price);
    } else {
        System.out.println("Queue is empty");
    }
}
```

5. Ensure that the peekRear() function can be executed inside the program

```java
case 5:
    queuePassengers.peekRear();
    break;
```

```java
public static void menu() {
    System.out.println("Choose Menu : ");
    System.out.println("1. Enqueue");
    System.out.println("2. Dequeue");
    System.out.println("3. Check first queue");
    System.out.println("4. Check all queue");
    System.out.println("5. Peek rear");
    System.out.println("====================");
}
```

Assignments

1. Add these 2 methods in Queue class in 1st practicum

- **There is no method that is given.**

2. Make a queue program for students when they need the signs for their KRS by the DPA. If the student is in queue, they will be required to fill in some information as follows:

Student
nim: String name: String classNumber: int gpa: double
Student (nim: String, name: String, classNumber: int,gpa: double)

Queue Class diagram:

Queue
max: int front: int rear: int size: int stdQueue: Student[]
Queue(max: int) create(): void isEmpty(): boolean isFull(): boolean enqueue(stdQueue: Student): void dequeue(): int print(): void peek(): void peekRear(): void peekPosition(nim: String): void printStudents(position: int): void

Notes:

The implementation of Create(), isEmpty(), isFull(), enqueue(), dequeue() and print() functions are similar with we've built in practicum

Peek() method is used for displaying students data in the first queue

peekRead() method is used for displaying students data in the last queue

peekPosition() method is used for displaying students data in the queue by their NIM

printStudents() method is used for displaying a student data in specified position in a queue

- Student

```java
package Assignment;

public class Student {
    public String nim, name;
    public int classNumber;
    public double gpa;

    public Student(String nim, String name, int classNumber, double gpa) {
        this.nim = nim;
        this.name = name;
        this.classNumber = classNumber;
        this.gpa = gpa;
    }
}

```

- Queue

```java
package Assignment;

public class Queue {
    public int max, size, front, rear;
    public Student [] stdQueue;

    public Queue(int n) {
        max = n;
        create();
    }

    public void create() {
        stdQueue = new Student[max];
        size = 0;
        front = rear = -1;
    }

    public boolean isEmpty() {
        if (size == 0) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isFull() {
        if (size == max) {
            return true;
        } else {
            return false;
        }
    }

    public void peek() {
        if (!isEmpty()) {
            System.out.println("The first element : " + stdQueue[front].nim + " " + stdQueue[front].nim + " " + stdQueue[front].classNumber + " " + stdQueue[front].gpa);
        } else {
            System.out.println("stdQueueueue is still empty");
        }
    }

    public void peekRear() {
        if (!isEmpty()) {
            System.out.println("The last element : " + stdQueue[rear].nim + " " + stdQueue[rear].nim + " " + stdQueue[rear].classNumber + " " + stdQueue[rear].gpa);
        } else {
            System.out.println("Queue is empty");
        }
    }

    public void print() {
        if (!isEmpty()) {
            System.out.println("Queue is still empty");
        } else {
            int i = front;
            while (i != rear) {
                System.out.println("The first element : " + stdQueue[front].nim + " " + stdQueue[front].nim + " " + stdQueue[front].classNumber + " " + stdQueue[front].gpa);
                i = (i+1) % max;
            }
            System.out.println(stdQueue[i] + " ");
            System.out.println("Element amount : " + size);
        }
    }

    public void clear() {
        if (!isEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Queue has been cleared successfully");
        } else {
            System.out.println("Queue is still empty");
        }
    }

    public void Enqueue(Student data) {
        if (isFull()) {
            System.out.println("Queue is already full");
        } else {
            if (isEmpty()) {
                front = rear = 0;
            } else {
                if (rear == max -1) {
                    rear = 0;
                } else {
                    rear++;
                }
            }
            stdQueue[rear] = data;
            size++;
        }
    }

    public Student Dequeue() {
        Student data = new Student("", "", 0, 0);
        if (isEmpty()) {
            System.out.println("Queue is still empty");
        } else {
            data = stdQueue[front];
            size--;
            if (isEmpty()) {
                front = rear = -1;
            } else {
                if (front == max -1) {
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return data;
    }

    public void peekPosition(String nim) {
        if (!isEmpty()) {
            for (int i = front; i != rear; i = (i + 1) % max) {
                if (stdQueue[i].nim.equals(nim)) {
                    System.out.println("Student found at position " + (i - front + 1));
                    return;
                }
            }
            if (stdQueue[rear].nim.equals(nim)) {
                System.out.println("Student found at position " + (rear - front + 1));
                return;
            }
            System.out.println("Student not found in the queue");
        } else {
            System.out.println("Queue is empty");
        }
    }

    public void printStudent(int position) {
        if (!isEmpty() && position > 0 && position <= size) {
            int index = (front + position - 1) % max;
            System.out.println("Student at position " + position + ": " + stdQueue[index].nim + " " + stdQueue[index].name + " " + stdQueue[index].classNumber + " " + stdQueue[index].gpa);
        } else {
            System.out.println("Invalid position or queue is empty");
        }
    }

}
```

- Main

```java
package Assignment;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Insert maximum queue : ");
        int max = sc.nextInt();
        Queue queue = new Queue(max);

        int choose;
        do {
            menu();
            choose = sc.nextInt();
            switch (choose) {
                case 1:
                    System.out.print("nim : ");
                    String nim = sc.next();
                    sc.nextLine();
                    System.out.print("Name : ");
                    String name = sc.next();
                    sc.nextLine();
                    System.out.print("Class number : ");
                    int classNumber = sc.nextInt();
                    System.out.print("GPA : ");
                    double gpa = sc.nextDouble();
                    Student student = new Student(nim, name, classNumber, gpa);
                    sc.nextLine();
                    queue.Enqueue(student);
                    break;
                case 2:
                    Student data = queue.Dequeue();
                    if (!"".equals(data.nim) && !"".equals(data.name) && !"".equals(data.classNumber) && !"".equals(data.gpa)) {
                        System.out.println("Data removed : " + data.nim + " " + data.name + " " + data.classNumber + " " + data.gpa);
                        break;
                    }
                case 3:
                    queue.peek();
                    break;
                case 4:
                    queue.print();
                    break;
                case 5:
                    queue.clear();
                    break;
                case 6:
                    queue.peekRear();
                    break;
                case 7:
                    System.out.print("NIM : ");
                    String searchNim = sc.next();
                    sc.nextLine();
                    queue.peekPosition(searchNim);
                    break;
                case 8:
                    System.out.print("Position : ");
                    int position = sc.nextInt();
                    queue.printStudent(position);
                    break;
                default:
                    break;
            }
        } while (choose <= 8 && choose >= 1);

        sc.close();
    }

    public static void menu() {
        System.out.println("Choose Menu : ");
        System.out.println("1. Enqueue");
        System.out.println("2. Dequeue");
        System.out.println("3. Peek");
        System.out.println("4. Print");
        System.out.println("5. Clear");
        System.out.println("6. Rear peek");
        System.out.println("7. Peek position");
        System.out.println("8. Print student");
        System.out.println("====================");
    }
}
```