JOBSHEET 10

Queue

Name
Sherly Lutfi Azkiah Sulistyawati

NIM
2341720241

Class
1I

Major
Information Technology

Study Program
D4 Informatics Engineering

Practicum 1

Practice > Week10 > Queue.java > Queue > Dequeue()

```java
package Week10;

public class Queue {
    int max, size, front, rear;
    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(x:"Queue is still empty");
        }
    }
```

```java
    public void print() {
        if (!IsEmpty()) {
            System.out.println(x:"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(x:"Queue has been cleared successfully");
        } else {
            System.out.println(x:"Queue is still empty");
        }
    }

    public void Enqueue(int data) {
        if (IsFull()) {
            System.out.println(x:"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 int Dequeue() {
        int data = 0;
        if (IsEmpty()) {
            System.out.println(x:"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;
    }
}

```

```java
package Week10;

import java.util.Scanner;

public class QueueMain {

    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("======================");
    }

    Run | Debug
    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.println("Data removed : " + removeData);
                        break;
                    }
                case 3:
                    Q.print();
                    break;
                case 4:
                    Q.peek();
                    break;
                case 5:
                    Q.clear();
            }
        } while (choose <= 5 && choose >=1);

        sc.close();
    }
}
```

```
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
Choose menu:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
======================
```

Question

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

   - front and rear are initialized to -1 to signify an empty queue. -1 indicates an empty queue, not 1.

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

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

   - if (rear == max - 1) checks if the rear pointer is at the end of the array. If so, it wraps around to the beginning by setting rear to 0.

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

   - New data is stored in the last position of the queue with 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 is removed from the first position of the queue with data = Q[front];.

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

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

   - if (front == max - 1) handles wrapping the front pointer to the beginning of the array if it reaches the end.

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

   - The loop process in the print() method starts from int i = front; to ensure printing starts from the first element of the queue.

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

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

   - The line i = (i + 1) % max; for correctly traversing the circular queue and printing its elements without going beyond the array bounds.

Practicum 2

```java
package Week10;

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

    public Passengers(String nm, String co, String cd, int ta, int pr) {
        name = nm;
        cityOrigin = co;
        cityDestination = cd;
        ticketAmount = ta;
        price = pr;
    }
}

```

```java
public class QueuePassengers {
    int max, size, front, rear;
    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(x:"Queue is still empty");
        }
    }

    public void print() {
        if (!IsEmpty()) {
            System.out.println(x:"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(x:"Queue has been cleared successfully");
        } else {
            System.out.println(x:"Queue is still empty");
        }
    }

    public void Enqueue(Passengers data) {
        if (IsFull()) {
            System.out.println(x:"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(nm:"", co:"", cd:"", ta:0, pr:0);
        if (IsEmpty()) {
            System.out.println(x:"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;
    }
```

```
package Week10;

import java.util.Scanner;

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

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

        QueuePassengers queuePassenger = new QueuePassengers(max);

        int choose;
        do {
            menu();
            choose = sc.nextInt();
            switch (choose) {
                case 1:
                    sc.nextLine();
                    System.out.print(s:"Name: ");
                    String nm = sc.nextLine();
                    System.out.print(s:"City Origin: ");
                    String cOrg = sc.nextLine();
                    System.out.print(s:"City Destination: ");
                    String cDes= sc.nextLine();
                    System.out.print(s:"Ticket Amount: ");
                    int ticket = sc.nextInt();
                    System.out.print(s:"Price: ");
                    int price = sc.nextInt();
                    Passengers p = new Passengers(nm, cOrg, cDes, ticket, price);
                    sc.nextLine();
                    queuePassenger.Enqueue(p);
                    break;
                case 2:
                    Passengers 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.peek();
                    break;
                case 4:
                    queuePassenger.print();
                    break;
            }
        } while (choose <= 4 && choose >=1);

        sc.close();
    }
}
```

```
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 Destination: 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 Destination: 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
```

Question

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

   - The purpose of the code in the Dequeue() method is to remove an element from the front of the queue and return the dequeued data.

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

   Passenger data = new Passenger()

   What will happen?

   - Changing Passenger data = new Passenger() would create a new instance of Passenger with default values, not representing any actual data from the queue.

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

```
case 2:
    Passengers 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;
    }
```

```
public Passengers Dequeue() {
    Passengers data = new Passengers(nm:"", co:"", cd:"", ta:0, pr:0);
    if (IsEmpty()) {
        System.out.println(x:"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

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

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

```
6       public static void menu(){
7           System.out.println(x:"Choose menu: ");
8           System.out.println(x:"1. Queue");
9           System.out.println(x:"2. Dequeue");
10          System.out.println(x:"3. Check first queue");
11          System.out.println(x:"4. Check all queue");
12          System.out.println(x:"5. Peek Rear");
13          System.out.println(x:"======================");
14      }
```

```
case 5:
    queuePassenger.peekRear();
    break;
```

Assignment

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

Practice > Week10 > Assignment > Student.java > ...

```java
package Week10.Assignment;

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

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

Practice > Week10 > Assignment > Queue.java > Queue > dequeue()

```java
package Week10.Assignment;

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

    public Queue(int max) {
        this.max = max;
        create();
    }

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

    public boolean isEmpty() {
        return size == 0;
    }

    public boolean isFull() {
        return size == max;
    }

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

    public Student dequeue() {
        if (isEmpty()) {
            System.out.println(x:"Queue is still empty");
            return null;
        } else {
                Student data = stdQueue[front];
                size--;
                if (isEmpty()) {
                    front = rear = -1;
                } else {
                    front = (front + 1) % max;
                }
                return data;
            }
        }

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

        public void peek() {
            if (!isEmpty()) {
                printStudents(front);
            } else {
                System.out.println(x:"Queue is still empty");
            }
        }

        public void peekRear() {
            if (!isEmpty()) {
                printStudents(rear);
            } else {
                System.out.println(x:"Queue is still empty");
            }
        }

```

```java
    public void peekPosition(String nim) {
        if (!isEmpty()) {
            int i = front;
            boolean found = false;
            while (i != rear) {
                if (stdQueue[i].nim.equals(nim)) {
                    printStudents(i);
                    found = true;
                    break;
                }
                i = (i + 1) % max;
            }
            if (stdQueue[i].nim.equals(nim)) {
                printStudents(i);
                found = true;
            }
            if (!found) {
                System.out.println("Student with NIM " + nim + " not found in the queue.");
            }
        } else {
            System.out.println(x:"Queue is still empty");
        }
    }

    public void printStudents(int position) {
        Student student = stdQueue[position];
        System.out.println("Student NIM: " + student.nim + ", Name: " + student.name +
                           ", Class Number: " + student.classNumber + ", GPA: " + student.gpa);
    }
}
```

Practice > Week10 > Assignment > J QueueMain.java > ...

```java
package Week10.Assignment;

import java.util.Scanner;

public class QueueMain {
    public static void menu() {
        System.out.println(x:"Choose menu: ");
        System.out.println(x:"1. Enqueue");
        System.out.println(x:"2. Dequeue");
        System.out.println(x:"3. Peek first");
        System.out.println(x:"4. Peek rear");
        System.out.println(x:"5. Peek by NIM");
        System.out.println(x:"6. Print all");
        System.out.println(x:"======================");
    }

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

        Queue queue = new Queue(max);

        int choice;
        do {
            menu();
            choice = sc.nextInt();
            switch (choice) {
                case 1:
                    sc.nextLine(); // consume newline
                    System.out.print(s:"NIM: ");
                    String nim = sc.nextLine();
                    System.out.print(s:"Name: ");
                    String name = sc.nextLine();
                    System.out.print(s:"Class Number: ");
                    int classNumber = sc.nextInt();
                    System.out.print(s:"GPA: ");
                    double gpa = sc.nextDouble();
                    Student student = new Student(nim, name, classNumber, gpa);
                    queue.enqueue(student);
                    break;
                case 2:
                    Student dequeuedStudent = queue.dequeue();
```

```
                    if (dequeuedStudent != null) {
                        System.out.println("Data removed: " + dequeuedStudent.nim + " " + dequeuedStudent.name
                                + " " + dequeuedStudent.classNumber + " " + dequeuedStudent.gpa);
                    }
                    break;
                case 3:
                    queue.peek();
                    break;
                case 4:
                    queue.peekRear();
                    break;
                case 5:
                    sc.nextLine(); // consume newline
                    System.out.print(s:"Enter NIM to peek: ");
                    String peekNim = sc.nextLine();
                    queue.peekPosition(peekNim);
                    break;
                case 6:
                    queue.print();
                    break;
            }
        } while (choice >= 1 && choice <= 6);

        sc.close();
    }
}

```