Week 10

Subject	Data Structure and Algorithm
Lecturer	Imam Fahrur Rozi ST. MT.
Type	Assignment
Semester	Semester 2
Time	
Files & Media	

Activities #1

code

```
package JB10.src.Prac;

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

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

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

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

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

    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++;
        }
    }

    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);
                else front++;
            }
        }
        return data;
    }

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

    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);
        }
    }

    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");
    }
}
```

```
package JB10.src.Prac;

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. Print");
        System.out.println("3. Print");
        System.out.println("4. Peek");
        System.out.println("5. Clear");
        System.out.println("========================");
    }

    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("Inser 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);
    }
}
```

result

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

Question #1

1. because 0 represents as a first elements and -1 represents the queue is still empty and not filled with any elements
2. the code is used to cycle the rear position of the queue, so that if the rear is in the last position, it will cycle back to the first of the array and it won't be out of bounds

3. `data = Q[front];`
4. like the `enqueue()` method, the `dequeue()` is used to reset the position of the front element of the queue if the queue is reaches the end of the array, so it won't be out of bounds
5. sorry but there isn't any line of code that goes `int i = 0`, the `int i = front` is used because the beginning of the queue isn't always at the start of the array, because of that, if we use `int i = 0` we will repeat the queue in the middle of the queue
6. the code is used to go back at the start of the array if we reach the end of the array

Activities #2

code

```
package JB10.src.Prac;

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

    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;
    }
}
```

```
package JB10.src.Prac;

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

    PassengersQueue(int max)
    {
        this.max = max;
        create();
    }

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

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

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

    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++;
        }
    }

    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) rear = 0;
                else front++;
            }
        }
        return data;
    }

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

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

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

    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");
    }
}
```

```
package JB10.src.Prac;

import java.util.Scanner;

public class PassengersMain
{
    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("5. Clear queue");
```

```
        System.out.println("6. Check last queue");
        System.out.println("==========================");
    }

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

        int choose;
        do
        {
            menu();
            choose = sc.nextInt();
            switch (choose)
            {
                case 1:
                    System.out.print("Name: ");
                    String nm = sc.next();
                    sc.nextLine();
                    System.out.print("City origin: ");
                    String cOrg = sc.next();
                    sc.nextLine();
                    System.out.print("City Destination: ");
                    String cDes = sc.next();
                    sc.nextLine();
                    System.out.print("Ticket Amount: ");
                    int ticket = sc.nextInt();
                    System.out.print("Price: ");
                    int price = sc.nextInt();
                    Passengers p = new Passengers(nm, cOrg, cDes, price, ticket);
                    sc.nextLine();
                    queuePassenger.enqueue(p);
                    break;
                case 2:
                    Passengers data = queuePassenger.dequeue();
                    if (!"".equals(data.name) && !"".equals(data.cityOrigin) && !"".equals(data.cityDestination) && !"".equals(data.ticketA
                    break;
                case 3:
                    queuePassenger.peek();
                    break;
                case 4:
                    queuePassenger.print();
                    break;
                case 5:
                    queuePassenger.clear();
                    break;
                case 6:
                    queuePassenger.peekRear();
                    break;
            }
        }
        while(choose <= 6 && choose >= 1);
    }
}
```

result

```
Insert maximum queue : 5
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear queue
6. Check last queue
============================
1
Name: 123
City origin: 123
City Destination: 123
Ticket Amount: 123
Price: 123
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear queue
6. Check last queue
============================
1
Name: 234
City origin: 234
City Destination: 234
Ticket Amount: 234
Price: 234
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear queue
6. Check last queue
============================
```

```
1
Name: 345
City origin: 345
City Destination: 345
Ticket Amount: 345
Price: 345
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear queue
6. Check last queue
============================
4
123 123 123 123 123
234 234 234 234 234
345 345 345 345 345
Element amount : 3
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear queue
6. Check last queue
============================
```

Question #2

1. the line of code is used to ensure whether the queue contains any data or not, so we have to construct an empty dummy object and it will fill the initial data
2. the code will be error, since `Passenger()` requires 5 arguments
3. this line of code

```
Passengers data = queuePassenger.dequeue();
if (!"".equals(data.name) && !"".equals(data.cityOrigin) && !"".equals(data.cityDestination) && !"".equals(data.ticketAmount) && !"".equals
```

4. add this line of code in the `PassengersQueue` class

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

then add this line of code in the `PassengersMain` class

```
        System.out.println("4. Check all queue");
        System.out.println("5. Clear queue");
        System.out.println("6. Check last queue");
        System.out.println("============================");
    }
```

then add this line in the menu

```
                case 6:
                queuePassenger.peekRear();
                break;
            }
```

also change the last line into this

```
while(choose <= 6 && choose >= 1);
```

5. This

```
4. Check all queue
5. Clear queue
6. Check last queue
============================
4
123 123 123 123 123
234 234 234 234 234
345 345 345 345 345
Element amount : 3
Choose menu:
1. Queue
2. Dequeue
3. Check first queue
4. Check all queue
5. Clear queue
6. Check last queue
============================
6
The last element : 345 345 345 345 345
```

Assignment

1. uhhhhhhhhhhh where? method goes adios

2. code

```
package JB10.src.Asg;

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

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

```
        this.name = name;
        this.classNumber = classNumber;
        this.gpa = gpa;
    }

}
```

```
package JB10.src.Asg;

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

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

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

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

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

    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++;
            }
            Q[rear] = data;
            size++;
        }
    }

    Student dequeue()
    {
        Student data = new Student("", "",  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) rear = 0;
                else front++;
            }
        }
        return data;
    }

    void print()
    {
        if (isEmpty()) System.out.println("Queue is still empty");
        else
        {
            int i = front;
            while(i != rear)
```

```
            {
                System.out.println(Q[i].name + " " + Q[i].nim + " " + Q[i].classNumber + " " + Q[i].gpa);
                i = (i + 1) % max;
            }
            System.out.println(Q[rear].name + " " + Q[rear].nim + " " + Q[rear].classNumber + " " + Q[rear].gpa);
//            System.out.println(Q[i] + " ");
            System.out.println("Element amount : " + size);
        }
    }

    void peekRear()
    {
        if (!isEmpty()) System.out.println(Q[rear].name + " " + Q[rear].nim + " " + Q[rear].classNumber + " " + Q[rear].gpa);
        else System.out.println("Queue is still empty");
    }

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

    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");
    }

    void peekPosition(String nim)
    {
        Student student = null;
        for (int i = 0; i < Q.length; i++)
        {
            if (Q[i] == null) continue;
            if (nim.equals(Q[i].nim))
            {
                student = Q[i];
                break;
            }
        }

        if (student != null)
        {
            System.out.println("Selected Student : " + student.name + " " + student.nim + " " + student.classNumber + " " + student.gpa);
            return;
        }

        System.out.println("The Inputted NIM isn't available");
    }

    void printStudents(int pos)
    {
        int index = (pos + front) % max;
        if (Q[index] == null) System.out.println("There is no Student at that queue");
        else System.out.println("Selected Student : " + Q[index].name + " " + Q[index].nim + " " + Q[index].classNumber + " " + Q[index].gp
    }
}
```

```
package JB10.src.Asg;

import java.util.Scanner;

public class StudentMain
{
    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("5. Clear queue");
        System.out.println("6. Check last queue");
```

```
        System.out.println("7. Search Student data by inputting NIM");
        System.out.println("8. Search Student data by inputting queue number");
        System.out.println("============================");
    }

    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("Name: ");
                    String nm = sc.next();
                    sc.nextLine();
                    System.out.print("Nim: ");
                    String nim = sc.next();
                    sc.nextLine();
                    System.out.print("Class Number: ");
                    int classNum = sc.nextInt();
                    System.out.print("GPA: ");
                    int gpa = sc.nextInt();
                    Student p = new Student(nm, nim, classNum, gpa);
                    sc.nextLine();
                    queue.enqueue(p);
                    break;
                case 2:
                    Student data = queue.dequeue();
                    if (!"".equals(data.name) && !"".equals(data.nim)) System.out.println("Data removed : " + data.name + " " + data.nim +
                    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 pos = sc.next();
                    sc.nextLine();
                    queue.peekPosition(pos);
                    break;
                case 8:
                    System.out.print("Queue: ");
                    int select = sc.nextInt();
                    queue.printStudents(select);
                    break;
            }
        }
        while(choose <= 8 && choose >= 1);
    }
}
```