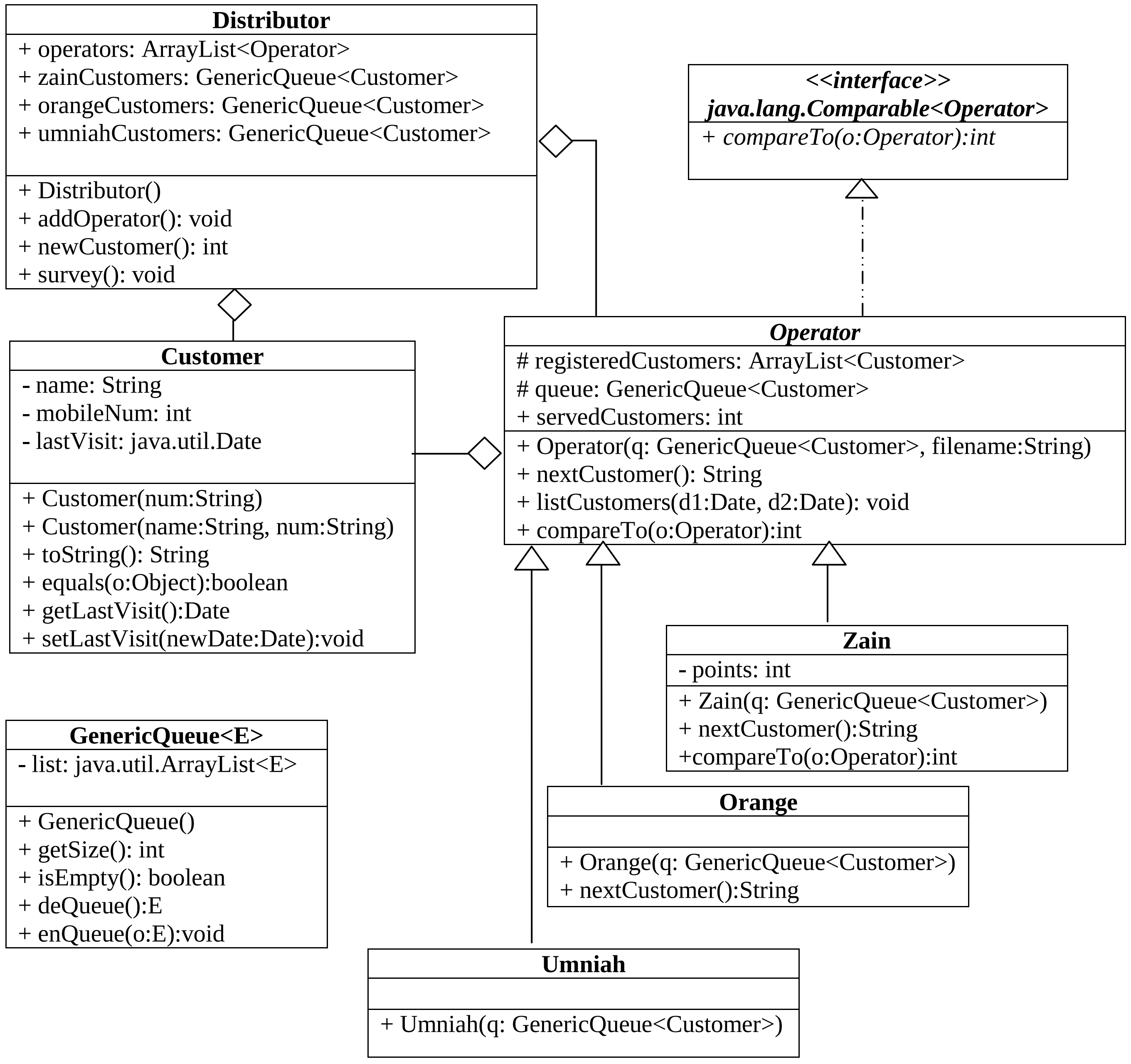
**University Of Jordan**
**Computer Engineering Department**

# Object-Oriented Problem Solving Lab
**Final Exam – Fall 2017**
**Eng. Asma Abdel Karim**

**Name: ……………………………..…. University ID: ……..……….. Computer Number: ………**

---

❖ **Write a JAVA program to build an appropriate implementation for the UML diagram below, which models a mobile service center.**


Distributor
+ operators: ArrayList<Operator>
+ zainCustomers: GenericQueue<Customer>
+ orangeCustomers: GenericQueue<Customer>
+ umniahCustomers: GenericQueue<Customer>
+ Distributor()
+ addOperator(): void
+ newCustomer(): int
+ survey(): void
<<interface>>
java.lang.Comparable<Operator>
+ compareTo(o:Operator):int
Customer
- name: String
- mobileNum: int
- lastVisit: java.util.Date
+ Customer(num:String)
+ Customer(name:String, num:String)
+ toString(): String
+ equals(o:Object):boolean
+ getLastVisit():Date
+ setLastVisit(newDate:Date):void
Operator
# registeredCustomers: ArrayList<Customer>
# queue: GenericQueue<Customer>
+ servedCustomers: int
+ Operator(q: GenericQueue<Customer>, filename:String)
+ nextCustomer(): String
+ listCustomers(d1:Date, d2:Date): void
+ compareTo(o:Operator):int
Zain
- points: int
+ Zain(q: GenericQueue<Customer>)
+ nextCustomer():String
+compareTo(o:Operator):int
GenericQueue<E>
- list: java.util.ArrayList<E>
+ GenericQueue()
+ getSize(): int
+ isEmpty(): boolean
+ deQueue():E
+ enQueue(o:E):void
Orange
+ Orange(q: GenericQueue<Customer>)
+ nextCustomer():String
Umniah
+ Umniah(q: GenericQueue<Customer>)

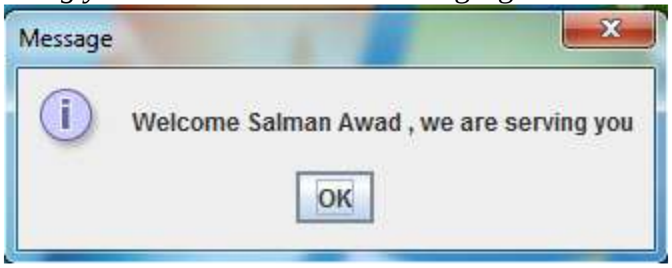
1. ***GenericQueue*:**
   A queue is a data structure in which the first element that enters the queue is the first element that will be removed (FIFO).
   a. *enQueue*: is a method that adds an element of generic type *E* to the queue.
   b. *deQueue*: is a method that removes an element from the queue and returns it.

2. ***Customer*:**
   a. The *lastVisit* data field should be initialized to the current date.
   b. The *toString* method should be overridden to return the customer data fields values separated by dashes (-).
   (e.g. Asma Abdelkarim – 079xxxxxxx - Wed Dec 20 22:12:57 EET 2017)
   c. The *equals* method should be overridden such that two customers are equal if they have the same mobile number.

3. ***Operator*:**
   a. Data fields:
      - *registeredCustomers*: represents the list of all customers registered in the mobile network.
      - *queue*: represents the queue of customers waiting to be served.
      - *servedCustomers*: represents the number of customers served so far.

   b. The constructor should:
      - Initialize the *queue* data field with the passed parameter *q*. Note that it should copy the reference such that they both point to the same queue.
      - Initialize the *registeredCustomers* array list by reading from the file with the passed *fileName* by using the method *readFile* provided in the *FileInput* class. Note that the method *readFile* has the following header:
        *public static ArrayList readFile(String fileName)*

   c. The *nextCustomer* method should serve the next customer in the queue. If there are customers waiting in the queue, it should de-queue the next customer and search the *registeredCustomers* array list for that customer (based on mobile number) and display a message dialog box with the following message: *Welcome <customer name>, we are serving you* as shown in the following figure.



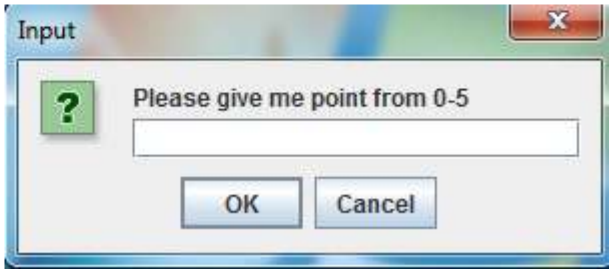
Welcome Salman Awad , we are serving you

   Note that this method should rewrite the *lastVisit* data field of the customer with the current date and increment the number of served customers.

   d. The *listCustomers* method should search the *registeredCustomers* array list for all customers that visited in the time period between date d1 and date d2 (after d1 and before d2), then list them in a message dialog box each customer's info on a separate line by invoking their *toString* method.

   e. The *compareTo* method should be overridden to return the difference in the number of served customers of the two operators.

4. **Zain:**
    a. This class has an additional data field named *points*.
    b. The constructor should invoke the super class constructor with the passed parameter *q* and the file name *"zain.out"*.
    c. The *nextCustomer* method should be overridden to first invoke the super class *nextCustomer* method then ask the customer to give the operator points from 0-5 in an input dialog box. The entered *points* should be added to the operator *points*.

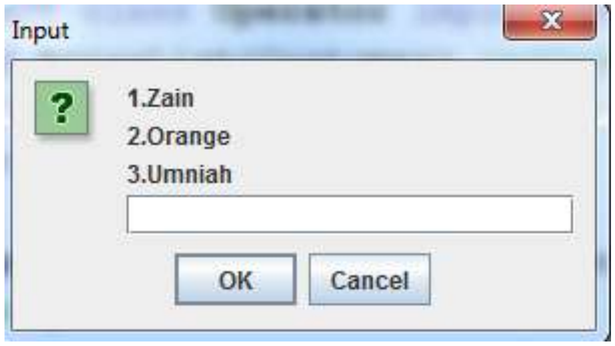


    d. The *compareTo* method should be overridden such that if the operator passed as an argument is a *Zain* operator it should return the difference between the two operators *points*. Otherwise, if the passed operator is not a *Zain* operator it should invoke the *compareTo* method of the super class to compare the two operators.
5. **Orange**
    a. The constructor should invoke the super class constructor with the passed parameter *q* and the file name *"orange.out"*.
    b. The *nextCustomer* method should be overridden to first invoke the super class *nextCustomer* method then show the following message in a message dialog box: "*Visit us again soon @ Orange Jordan*".

6. **Umniah**
    a. The constructor should invoke the super class constructor with the passed parameter *q* and the file name *"umniah.out"*.

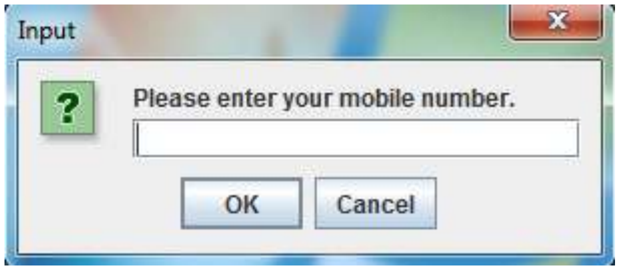
7. **Distributer.**
    a. Data fields:
        - An array list of *Operator* objects which should be initialized to an empty array list.
        - Three *GenericQueues* of *Customer* objects one for each type of operators: *Zain*, *Orange*, *Umniah*. These should be initialized to empty queues.

    b. The *addOperator* method should ask the user in an input dialog box about the type of the operator to be added as in the following input dialog box.



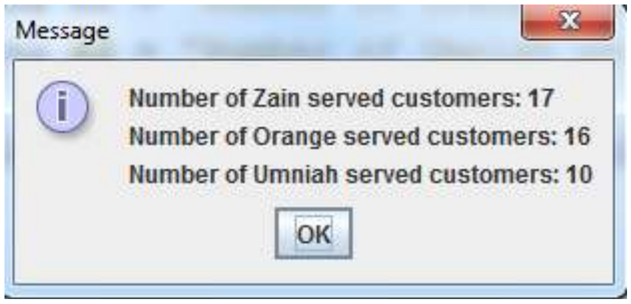
    Then, it should add a new operator according to the selected type and pass the correct queue reference to the operator. (i.e. *zainCustomers* if it is a *Zain* operator, *orangeCustomers* if it is an *Orange* operator, and *UmniahCustomers* if it is an *Umniah* operator)

c. The *newCustomer* method should:
   - Ask the customer to enter his mobile number in an input dialog box. If the entered mobile number does not include exactly 10 digits, it should re-ask him to enter his mobile number (until entered correctly).



   - Once the mobile number is entered, it should create a new *Customer* object with the entered mobile number.
   - Then, it should add the customer to the *zainCustomers*, *orangeCustomers*, or *UmniahCustomers* queue based on the mobile number (i.e. 079 for Zain, 077 for Orange and 078 for Umniah).

d. The *survey* method should show a message dialog box with the total number of customers served by each operator's type as in the following message dialog box.



```
Number of Zain served customers: 17
Number of Orange served customers: 16
Number of Umniah served customers: 10
```

**Good Luck ☺**