

פרויקט סיכום

תכנות מקבילי ומבוזר

נושא הפרויקט: מערכת לניהול תורים בקופת חולים

מגישה: חני זילברברג
מ"ז : 325150019
מורה מנחה: המורה שנהב
סמינר "דרכי חנה"

2הקדמה

5הפעלת התוכנה:

8תאור המחלקות:

8..... מחלקת HospitalCustomerService_Server

9..... מחלקת ClientHandler

9..... מחלקת Doctor

10..... מחלקת Nurse

10..... מחלקת Secretary

11..... מחלקת ScreenDisplay

11..... מחלקת HospitalCustomerService_Client

12..... מחלקת MakingAnAppointment

12..... מחלקת QueueLocation

13קטעי קוד:

13..... מחלקת HospitalCustomerService_Server

14..... מחלקת ClientHandler

16..... מחלקת Doctor

18..... מחלקת Nurse

20..... מחלקת Secretary

21..... מחלקת ScreenDisplay

24..... מחלקת HospitalCustomerService_Client

25..... מחלקת MakingAnAppointment

27..... מחלקת QueueLocation

הקדמה

אז למה במקביל?

- עולם התכנות המקבילי התפתח בצורה מואצת בשנים האחרונות. עוד ועוד מגלים כי כאשר ממקבילים תוכניות מחשב שונות ניתן להכפיל ואף לשלש את יעילות ביצוען.
- לאחר מכן בעקבות השימוש המוגבר ברשת ומשתמשים רבים בשירותים שונים בו זמנית, התברר שהשימוש במקבילי הוא אמנם מוסיף יעילות ומגביר מהירות זמני ריצה, אך גם גורר לבעיות שהתעוררו כמו DATA-RACE ו- RACE-CONDITION שפגעו בנכונות הפעולות ויצרו פגיעה בנתונים.
- כל זאת הביא למציאת פתרונות טכנולוגים רבים למקבול תוכניות באופן שישמור על יתרונות המקבול אך גם יגן כראוי על משתנים משותפים למספר משתמשים באופן שנכונות החישוב תצא נכונה גם כאשר המשתנה מתעדכן בו זמנית.
- ניתן לראות מנגנוני סנכרון רבים במגוון שפות המאפשרים לנו שימוש בטכנולוגיה מקבילית באופן בטוח.

בואו נראה את זה בפועל...

בפרויקט זה מוצג קשר בין שרת למספר לקוחות.

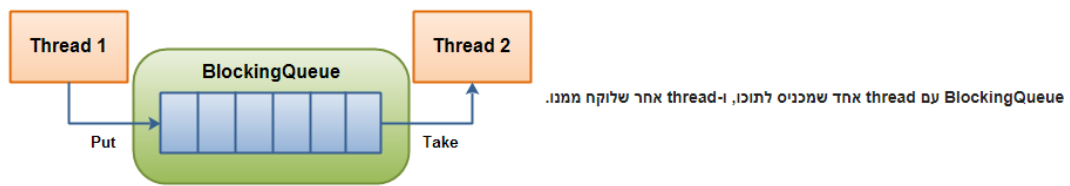
בפרויקט אעסוק בתוכנה המאפשרת ניהול תורים במרפאה.

-הלקוח נכנס לתור ע"י הקשת מ.ז והרופא אליו הוא ממתין, ומקבל מהשרת את מספרו בתור.

-כמו כן השרת מציג בכל עת את מצב התורים על המסך המרכזי.

בפרויקט זה נעשה שימוש ב Blocking Queues (התורים של הרופא המזכירה והאחות) -- blocking queue - הוא תור שחוסם כאשר מנסים להוציא ממנו והתור ריק, או אם מנסים להכניס אליו והתור כבר מלא. Thread שמנסה להוציא מתור ריק נחסם עד ש- thread אחר מכניס פריט אל התור. thread שמנסה להכניס פריט אל תור מלא נחסם עד ש- thread אחר מפנה מקום בתור, או על ידי הוצאת פריט אחד או יותר או על ידי ביקוי התור לחלוטין.

להלן תרשים המציג שני threads המשתפים פעולה דרך blocking queue :



המימוש של blocking queue :

```
public class BlockingQueue {

    private List queue = new LinkedList();
    private int limit = 10;

    public BlockingQueue(int limit) {
        this.limit = limit;
    }

    public synchronized void enqueue(Object item)
    throws InterruptedException {
        while(this.queue.size() == this.limit) {
            wait();
        }
        if(this.queue.size() == 0) {
            notifyAll();
        }
        this.queue.add(item);
    }

    public synchronized Object dequeue()
    throws InterruptedException{
        while(this.queue.size() == 0){
            wait();
        }
        if(this.queue.size() == this.limit){
            notifyAll();
        }

        return this.queue.remove(0);
    }
}
```

}

כמו כן נעשה שימוש בפעולה synchronize כדי לוודא שכאשר מס לקוחות נכנסים
בו זמנית לתור לא יקבלו את אותו מס' בתור אלא הכניסה לתור תתבצע כראוי ובצורה
מסודרת ללא הרס של נתונים.

הפעלת התוכנה:

1. ראשית הפעל את השרת-

מחלקת HospitalCustomerService_Server

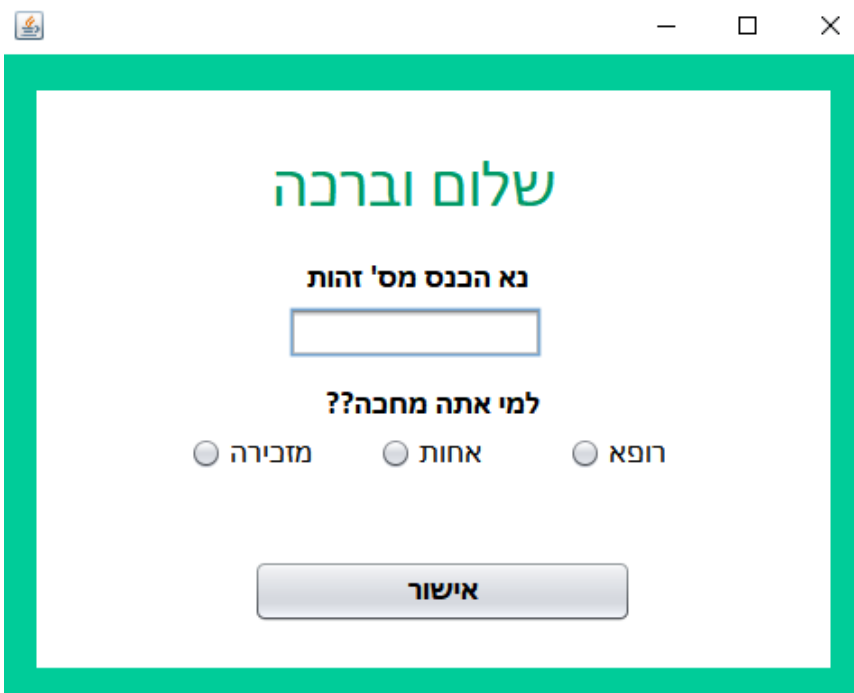
ובעיקובותיה יפתח החלון הבא-מסך מרכזי:



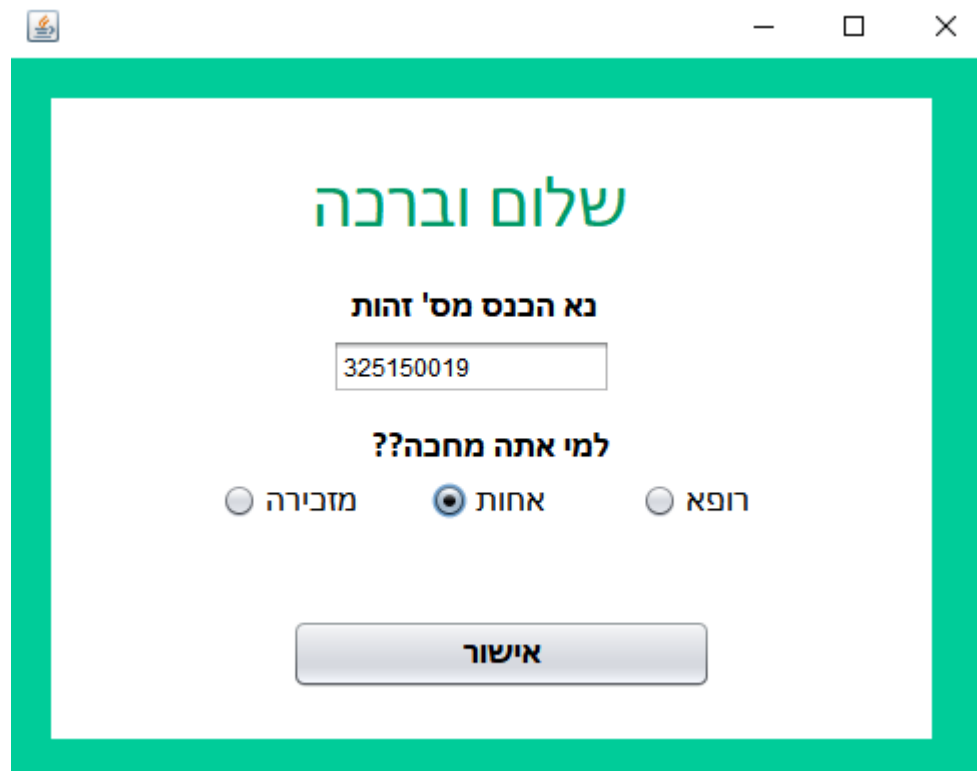
המסך יתמלא כשיגיעו לקוחות....

2. הרץ צד לקוח-מחלקת MakingAnAppointment:

ויפתח החלון הבא:

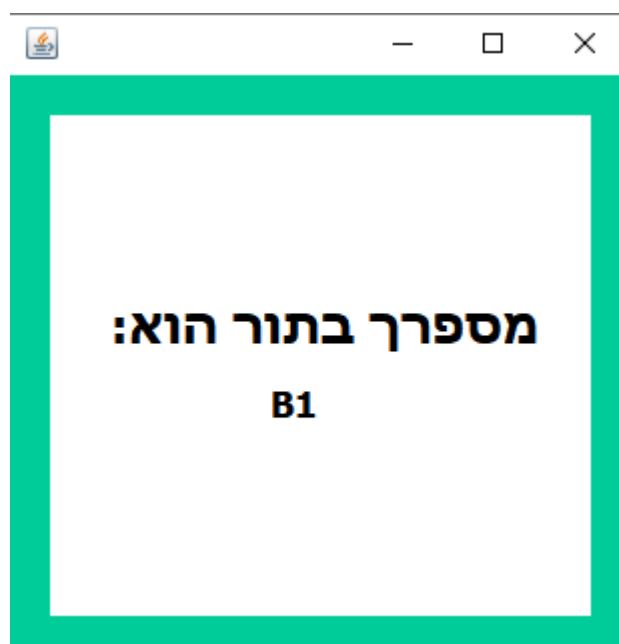


3. לקוח מזין את המזעור נקבע התור ובוחר את סוג הרפואה שהוא צריך. לאחר מכן הוא לוחץ על כפתור האישור.

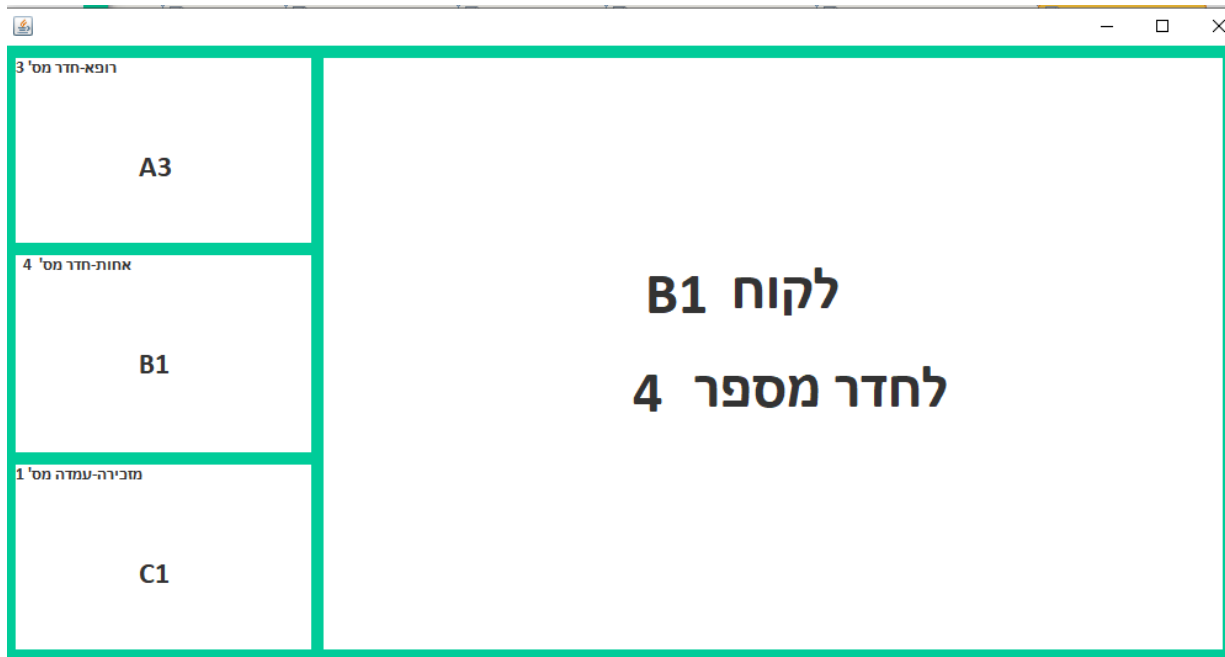


בעת הלחיצה על אישור הלקוח מתחבר לשרת ע"י יצירת סוקט שמעורר בשרת היענות לפקודה ACCEPT ובחירתו של הלקוח נשלחת לשרת.

4. השרת כתגובה יוצר אובייקט מסוג ClientHandler היורש ממחלקת Thread כדי לאפשר למספר לקוחות לתקשר בו זמנית מול השרת, ודרכה הוא קורא את בחירתו של הלקוח מכניס את לקוח זה לתור הממתינים של השרות אותו הוא בחר, מעלה את מספר הממתינים ומחזיר ללקוח את מספרו בתור.



5. כמו כן השרת מציג על המסך המרכזי את המספר הנוכחי בכל חדר במרפאה. השרת מוציא מספר מהתורים פעם בכמה דקות באופן רנדומלי ומציג את המספר בגדול על המסך.



תאור המחלקות:

Server:

- HospitalCustomerService_Server •
- ClientHandler •
- Doctor •
- Nurse •
- Secretary •
- ScreenDisplay •

Client:

- HospitalCustomerService_Client •
- MakingAnAppointment •
- QueueLocation •

Server

מחלקת HospitalCustomerService_Server

שדה	פירוט
ServerSocket main;	סוקט לחיבורי משתמשים חדשים
Socket socket;	Socket שקע לחיבורים
public static Doctor doctor;	אובייקט מסוג מחלקת רופא
public static Nurse nurse;	אובייקט מסוג מחלקת אחות
public static Secretary secretary;	אובייקט מסוג מחלקת מזכירה
int port;	מספר הפורט של השרת-שאליו מאזין

פעולה	פירוט
HospitalCustomerService_Server(int port,ScreenDisplay s)	פעולה בונה+המתנה לחיבור שיבוצע לסוקט וקבלתו.

מחלקת ClientHandler

שדה	פירוט
<code>DataInputStream din;</code>	המידע שמתקבל מהלקוח
<code>DataOutputStream dout;</code>	המידע שישלח ללקוח
<code>Socket socket;</code>	Socket שקע לחיבורים

פעולה	פירוט
<code>public ClientHandler(Socket socket)</code>	פעולה בונה, מאתחלת את התקשורת עם הלקוח.
<code>public int ReadTypeFromClient()</code>	מקבלת את מספר בחירתו של הלקוח (רופא / אחות / מזכירה)
<code>public void WriteTypeToClient(int numInQueue)</code>	שולחת ללקוח את מספרו בתור.
<code>public void run()</code>	רצה ברקע מקבלת מהלקוח את בחירתו מכניסה את הלקוח לתור הממתינים של בחירתו ומחזירה ללקוח את מספרו בתור
<code>private int insertToQueue(int type)</code>	מכניסה את הלקוח לתור הממתינים של בחירתו

מחלקת Doctor

שדה	פירוט
<code>static BlockingQueue<Integer> qDoctor;</code>	תור הממתינים לרופא
<code>static int counterD=0;</code>	משתנה סטטי הגדל עם כניסתו של לקוח לתור לרופא.
<code>ScreenDisplay s;</code>	אובייקט מסוג הטופס (ע"מ להשתמש בפונקציות שלו)

פעולה	פירוט
<code>public Doctor(ScreenDisplay s)</code>	פעולה בונה, מאתחלת את תור הממתינים ומפעילה את פונקציה <code>run()</code>
<code>public void run()</code>	רצה ברקע מוציאה מספרים מתור הממתינים ומציגה אותם ע"ג המסך
<code>synchronized void insertToQ()</code>	מכניסה מספר לקוח לתור הממתינים של הרופא ומעלה את מס' הממתינים.
<code>int popFromQ()</code>	מוציאה מהתור מספר

מחלקת Nurse

שדה	פירוט
<code>static BlockingQueue<Integer> qNurse;</code>	תור הממתינים לאחות
<code>static int counterN=0;</code>	משתנה סטטי הגדל עם כניסתו של לקוח לתור לאחות.
<code>ScreenDisplay s;</code>	אובייקט מסוג הטופס(ע"מ להשתמש בפונקציות שלו)

פעולה	פירוט
<code>public Nurse(ScreenDisplay s)</code>	פעולה בונה, מאתחלת את תור הממתינים ומפעילה את פונקציה <code>run()</code>
<code>public void run()</code>	רצה ברקע מוציאה מספרים מתור הממתינים ומציגה אותם ע"ג המסך
<code>synchronized void insertToQ()</code>	מכניסה מספר לקוח לתור הממתינים של האחות ומעלה את מס' הממתינים.
<code>int popFromQ()</code>	מוציאה מהתור מספר

מחלקת Secretary

שדה	פירוט
<code>static BlockingQueue<Integer> qSecretary;</code>	תור הממתינים למזכירה
<code>static int counterN=0;</code>	משתנה סטטי הגדל עם כניסתו של לקוח לתור למזכירה.
<code>ScreenDisplay s;</code>	אובייקט מסוג הטופס(ע"מ להשתמש בפונקציות שלו)

פעולה	פירוט
<code>public Secretary(ScreenDisplay s)</code>	פעולה בונה, מאתחלת את תור הממתינים ומפעילה את פונקציה <code>run()</code>
<code>public void run()</code>	רצה ברקע מוציאה מספרים מתור הממתינים ומציגה אותם ע"ג המסך
<code>synchronized void insertToQ()</code>	מכניסה מספר לקוח לתור הממתינים של המזכירה ומעלה את מס' הממתינים.
<code>int popFromQ()</code>	מוציאה מהתור מספר

מחלקת ScreenDisplay

פירוט	פעולה
פעולה בונה המאתחלת את הטופס	<code>public ScreenDisplay()</code>
פונקציה הקוראת ללקוח מסוים להיכנס לחדר כלשהו	<code>public void CallToClient(int num,int type)</code>
מציגה את מספר הלקוח הנמצא בחדר האחות	<code>public void SetTextNurse(int num)</code>
מציגה את מספר הלקוח הנמצא בחדר הרופא	<code>public void SetTextDoctor(int num)</code>
מציגה את מספר הלקוח הנמצא בעמדת המזכירה	<code>public void SetTextSecretary(int num)</code>

Client:

מחלקת HospitalCustomerService_Client

פירוט	שדה
המידע שמתקבל מהלקוח	<code>DataInputStream din;</code>
המידע שישלח ללקוח	<code>DataOutputStream dout;</code>
Socket שקע לחיבורים	<code>Socket socket;</code>

פירוט	פעולה
פעולה בונה+התחברות עם השרת	<code>public HospitalCustomerService_Client()</code>
שולחת לשרת את סוג השרות שהוא צריך	<code>public void WriteToServer(int type)</code>
מקבלת מהשרת את המספר בתור	<code>public int ReadFromServer()</code>

מחלקת MakingAnAppointment

פירוט	פעולה
פעולה בונה המאתחלת את הטופס	public MakingAnAppointment()
יוצרת קשר עם השרת, שולחת לו את הבחירה של הלקוח מקבלת חזרה את מספרו בתור הממתינים ושולחת זאת לטופס המציג את מספרו.	private void jButtonOKActionPerformed

מחלקת QueueLocation

פירוט	פעולה
פעולה בונה המקבלת את המספר בתור של הלקוח ולמי הוא ממתין	public QueueLocation(int numInQ,int select)
מציגה ללקוח את מספרו בתור	public void FillNumInQueue(int numInQ,int select)

קטעי קוד: Server

מחלקת HospitalCustomerService_Server

```
public class HospitalCustomerService_Server {  
    ServerSocket main;  
    Socket socket;  
    public static Doctor doctor;  
    public static Nurse nurse;  
    public static Secretary secretary;  
    int port;  
  
    HospitalCustomerService_Server(int port,ScreenDisplay s) throws IOException{  
        this.port=port;  
        main = new ServerSocket(port);  
        nurse=new Nurse(s);  
        doctor=new Doctor(s);  
        secretary=new Secretary(s);  
        while(true){  
            socket = main.accept();  
            ClientHandler ch=new ClientHandler(socket);  
            ch.start();  
        }  
    }  
  
    public static void main(String[] args) throws IOException {  
        ScreenDisplay s=new ScreenDisplay();
```

```

s.setVisible(true);
new HospitalCustomerService_Server(1800,s);
}
}

```

מחלקת ClientHandler

```

public class ClientHandler extends Thread{
    DataInputStream din;
    DataOutputStream dout;
    Socket socket;

    public ClientHandler(Socket socket) throws IOException{
        System.out.println("לקוח התחבר בהצלחה");
        this.socket = socket;
        din=new DataInputStream(socket.getInputStream());
        dout=new DataOutputStream(this.socket.getOutputStream());
    }

    //רופא/מזכירה/אחות)הפונקציה מקבלת את הסוג אליו מחכה הלקוח//
    public int ReadTypeFromClient() {
        try {
            String s =din.readUTF();
            return Integer.parseInt(s);
        } catch (IOException e) {
        }
        return 0;
    }
}

```

```

public void WriteTypeToClient(int numInQueue) {
    try {
        dout.writeUTF(String.valueOf(numInQueue));
    } catch (IOException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE, null,
ex);

    }
}

@Override
public void run() {
    try {
        System.out.println(din.readUTF());
        int type=ReadTypeFromClient();
        int numInQueue=insertToQueue(type); //הכנסה לתור הממתינים
        WriteTypeToClient(numInQueue);
    } catch (IOException | InterruptedException ex) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE, null,
ex);

    }
}

private int insertToQueue(int type) throws InterruptedException {
    int numInQueue = 0;
    if(type==0)//הלקוח מחכה לרופא
        try {

```



```

        HospitalCustomerService_Server.doctor.insertToQ();
        numInQueue = Doctor.counterD;
    } catch (InterruptedException ex) {

        Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE, null,
        ex);

    }
    else{
        if(type==1)//הלקוח מחכה לאחות
        {
            HospitalCustomerService_Server.nurse.insertToQ();
            numInQueue = Nurse.counterN;
        }
        else//הלקוח מחכה למזכירה
        {
            HospitalCustomerService_Server.secretary.insertToQ();
            numInQueue = Secretary.counterS;
        }
    }
    return numInQueue;
}
}

```

מחלקת Doctor

```

public class Doctor extends Thread{
    static BlockingQueue<Integer> qDoctor;//תור לרופא

```

```

static int counterD=0;

ScreenDisplay s;

public Doctor(ScreenDisplay s){
    qDoctor = new ArrayBlockingQueue<>(100);
    this.s=s;
    start();
}

@Override
public void run() {
    Random r=new Random();
    int num;
    while(true){
        try {
            num=popFromQ();
            s.SetTextDoctor(num);
            s.CallToClient(num, 1);
        } catch (InterruptedException ex) {
            Logger.getLogger(Doctor.class.getName()).log(Level.SEVERE, null, ex);
        }
        try {
            sleep(r.nextInt(7000)+2000);
        } catch (InterruptedException ex) {
            Logger.getLogger(Secretary.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

```

synchronized void insertToQ() throws InterruptedException {
    qDoctor.put(++counterD);
}

int popFromQ() throws InterruptedException {
    Integer remove = qDoctor.take();
    return remove;
}
}

```

מחלקת Nurse

```

public class Nurse extends Thread {
    static BlockingQueue<Integer> qNurse;//תור לאחות
    static int counterN=0;
    ScreenDisplay s;

    public Nurse(ScreenDisplay s){
        qNurse= new ArrayBlockingQueue<>(100);
        this.s=s;
        start();
    }

    @Override
    public void run() {
        Random r=new Random();
        int num;
    }
}

```

```

while(true){
    try {
        num=popFromQ();
        s.SetTextNurse(num);
        s.CallToClient(num, 2);
    } catch (InterruptedException ex) {
        Logger.getLogger(Nurse.class.getName()).log(Level.SEVERE,
null, ex);
    }
    try {
        sleep(r.nextInt(7000)+2000);
    } catch (InterruptedException ex) {
        Logger.getLogger(Secretary.class.getName()).log(Level.SEVERE,
null, ex);
    }
}

synchronized void insertToQ() throws InterruptedException {
    qNurse.put(++counterN);
}

int popFromQ() throws InterruptedException {
    Integer remove = qNurse.take();
    return remove;
}
}

```

מחלקת Secretary

```
public class Secretary extends Thread{

    static BlockingQueue<Integer> qSecretary;//תור למזכירה
    static int counterS=0;

    int num;

    ScreenDisplay s;

    public Secretary(ScreenDisplay s){

        qSecretary = new ArrayBlockingQueue<>(100);

        this.s=s;

        start();

    }

    @Override

    public void run() {

        Random r=new Random();

        while(true){

            try {

                num=popFromQ();

                s.SetTextSecretary(num);

                s.CallToClient(num,3);

            } catch (InterruptedException ex) {

                Logger.getLogger(Secretary.class.getName()).log(Level.SEVERE, null, ex);

            }

            try {

                sleep(r.nextInt(7000)+2000);

            } catch (InterruptedException ex) {
```

```

        Logger.getLogger(Secretary.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

```

synchronized void insertToQ() throws InterruptedException {
    qSecretary.put(++counterS);
}

```

```

int popFromQ() throws InterruptedException {
    Integer remove = qSecretary.take();
    return remove;
}
}

```

מחלקת ScreenDisplay

```

public class ScreenDisplay extends javax.swing.JFrame {

```

```

    public ScreenDisplay() {
        initComponents();
        jTextField3.setVisible(false);
        numClient.setVisible(false);
        numRoom.setVisible(false);
        jTextField1.setVisible(false);
    }

```

```

    public void CallToClient(int num,int type){
        jLabel4.setVisible(false);
    }
}

```

```

if(type==3){
    jTextField3.setText("לעמדה מספר");
    numRoom.setText("1");
    numClient.setText('C'+String.valueOf(num));
}
else{
    if(type==1){
        jTextField3.setText("לחדר מספר");
        numRoom.setText("3");
        numClient.setText('A'+String.valueOf(num));
    }
    else{
        if(type==2){
            jTextField3.setText("לחדר מספר");
            numRoom.setText("4");
            numClient.setText('B'+String.valueOf(num));}
        }
    }
    jTextField3.setVisible(true);
    numClient.setVisible(true);
    numRoom.setVisible(true);
    jTextField1.setVisible(true);
    try {
        sleep(5000);
    } catch (InterruptedException ex) {
        Logger.getLogger(ScreenDisplay.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

```

jTextField3.setVisible(false);
numClient.setVisible(false);
numRoom.setVisible(false);
jTextField1.setVisible(false);
jLabel4.setVisible(true);
}

public void SetTextNurse(int num){
    numInNurse.setText('B'+String.valueOf(num));
}

public void SetTextDoctor(int num){
    numInDoctor.setText('A'+String.valueOf(num));
}

public void SetTextSecretary(int num){
    numInSecretary.setText('C'+String.valueOf(num));
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ScreenDisplay().setVisible(true);
        }
    });
}

```

Client

מחלקת HospitalCustomerService_Client

```
public class HospitalCustomerService_Client {  
    Socket socket;  
    DataInputStream din;  
    DataOutputStream dout;  
  
    public HospitalCustomerService_Client() {  
        try {  
            socket = new Socket("127.0.0.1", 1800);//התחברות לשרת  
            din=new DataInputStream(socket.getInputStream());  
            dout=new DataOutputStream(socket.getOutputStream());  
            dout.writeUTF("הצלחתי להתחבר");  
        } catch (IOException ex) {  
  
            Logger.getLogger(HospitalCustomerService_Client.class.getName()).log(Level.SEVERE, null,  
ex);  
        }  
    }  
  
    public void WriteToServer(int type) {  
        try {  
            dout.writeUTF(String.valueOf(type));  
        } catch (IOException e) {  
        }  
    }  
  
    public int ReadFromServer() {  
        int i=0 ;
```

```

try {
    String readUTF = din.readUTF();
    i = Integer.parseInt(readUTF);

} catch (IOException ex) {

Logger.getLogger(HospitalCustomerService_Client.class.getName()).log(Level.SEVERE, null,
ex);

    }
    return i;
}
}

```

מחלקת MakingAnAppointment

```

public class MakingAnAppointment extends javax.swing.JFrame {

    public MakingAnAppointment() {
        initComponents();
    }

    private void jButtonOKActionPerformed(java.awt.event.ActionEvent evt) {

        int select;

        try{
            if (TzTextField.getText().isEmpty()) {
                JOptionPane.showMessageDialog(this, "יש להזין מספר זהות");
                return;
            }
            else{
                if (!TzTextField.getText().matches("[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]"))
                {
                    JOptionPane.showMessageDialog(this, "יש להכניס תשע ספרות");

```

```

        return;
    }
    else
    {
        if(!jRadio0.isSelected() && !jRadio1.isSelected() && !jRadio2.isSelected())
        {
            JOptionPane.showMessageDialog(this, "* לא הוקשה בחירה  
(אחות/רופא/מזכירה)");
            return;
        }
        else{
            if(jRadio0.isSelected() && jRadio1.isSelected() || jRadio0.isSelected() && jRadio2.isSelected() ||
            jRadio1.isSelected() && jRadio2.isSelected())
            {
                JOptionPane.showMessageDialog(this, "* עליו לבחור רק באחת מהאפשרויות  
(אחות/רופא/מזכירה)");
                return;
            }
            else{
                if(jRadio0.isSelected())
                    select=0;
                else{
                    select=jRadio1.isSelected()?1:2;
                }

                HospitalCustomerService_Client hospitalCustomerService_Client=new
                HospitalCustomerService_Client();

                hospitalCustomerService_Client.WriteToServer(select);
            }
        }
    }
}

```



```
if(select==1)//אחות
{
    locationTextField.setText('B'+String.valueOf(numInQ));
}
else//מזכירה
{
    locationTextField.setText('C'+String.valueOf(numInQ));
}
}
```