פרויקט גמר

למילוי חלקי של הדרישות לקבלת תואר הנדסאי

הנדסת תוכנה

בהתמחות: מחשבים

נושא הפרויקט: לחצן מצוקה לקריאה לאחות בבית חולים

שם הסטודנטית: אילה רינמן

מספר זהות:212220552

העבודה בוצעה בהנחיית : נחמה היימן

תוכן עניינים

הקדמה: 3
צילומי מסך 4
תרשים UML 8
פירוט מחלקות 9
Server Side: 9
Client Side 10
קטעי קוד 11
קטעי קוד בצרת שרת 11
קטעי קוד בצד לקוח 14

הקדמה:

בפרויקט המעבדה מוצג קשר בין שרת ללקוח, בפרויקט שלי אעסוק בקריאה לשרות רפואי דחוף בעזרת לחצן מצוקה.

בפרויקט זה, התוכנה מדמה לחצן מצוקה בבית חולים, כאשר מאושפז זקוק לעזרה דחופה של האחות הוא לוחץ על הלחצן ונכנס לתור הממתינים לאחות. האחות נגשת לחולה אחר חולה לפי סדר הקריאות. כל חולה שהיא מסיימת את הטיפול בו היא מסמנת בלחצן שלו שהטיפול הסתיים-משחררת את התור.

התור מתנהל בצד שרת על ידי מבנה נתונים של רשימה מקושרת. ובצד לקוח ניתן ללחוץ על קריאה לאחות.

בפרויקט השתמשי ב- threads על מנת שכמה מטופלים יוכלו לבצע קריאה בו זמנית לאחות.

אך כדי למנוע מצב ש-2 אחיות יתפנו לאותה קריאה ותתרחש הוצאה מהתור של 2 מטופלים בו זמנית מהתור עשיתי שימוש גם במנעולים.

צילומי מסך

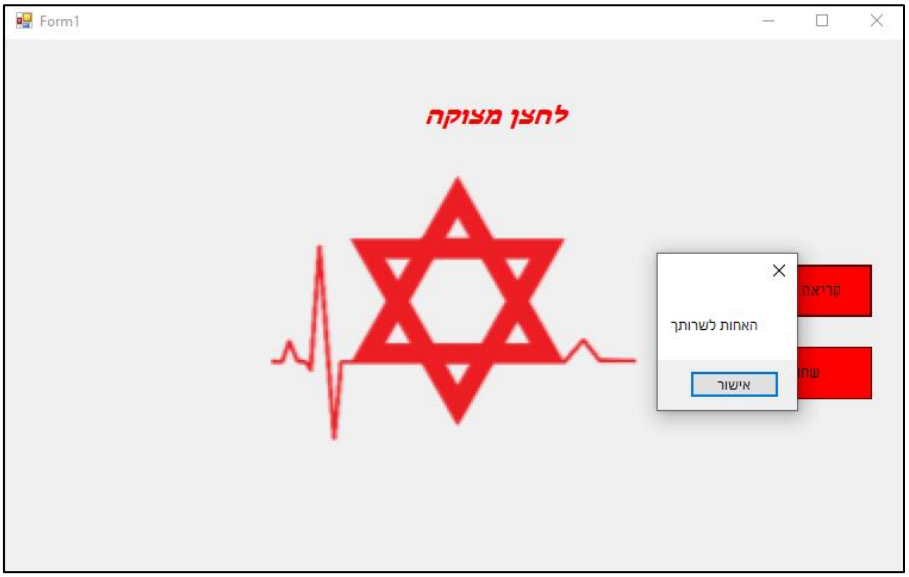
ראשית הפעל את ה-server:

בעקבות זאת יופיע החלון הבא:

לאחר מכן נריץ את את ה-client בו יש למשתמש 2 אפשרויות.

1.המשתמש יכול ללחוץ על לחצן המצוקה לקריאה לאחות.

אם האחות פנויה ויכולה לגשת למטופל הוא יקבל הודעה על כך:

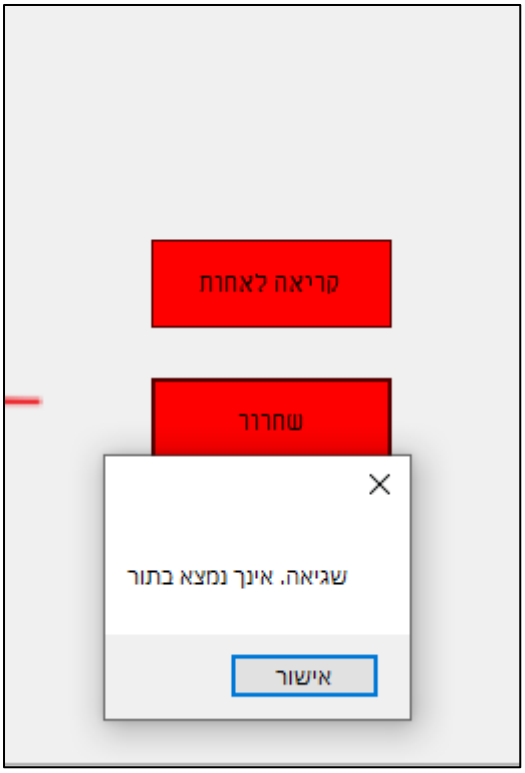


אם האחות עסוקה הוא נכנס לתור המתינים

לאחר שהמטופל ביקש טיפול תחסם לו ההאפשרות לקריאה חוזרת עד שהאחות תתפנה אליו(בשביל למנוע את הוספתו לתור שוב ושוב)

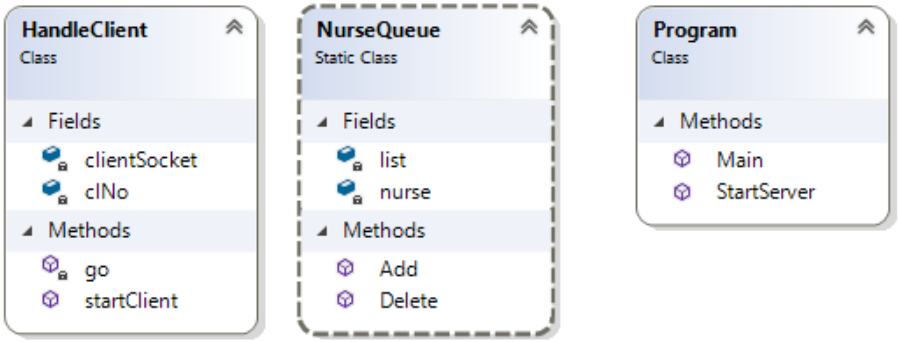
כאשר האחות מסיימת את הטיפול היא לוחצת על הכפתור שחרור ובכך המטופל יוצא מתור הממתינים. או כאשר מטופל רוצה לבטל את הקריאה הוא לוחץ על הכפתור ובכך יוצא מהרשימה.

אם מטופל לחץ על כפתור שחרור לפני שבכלל לחץ על קריאה לאחות הוא יקבל הודעה שגיאה.

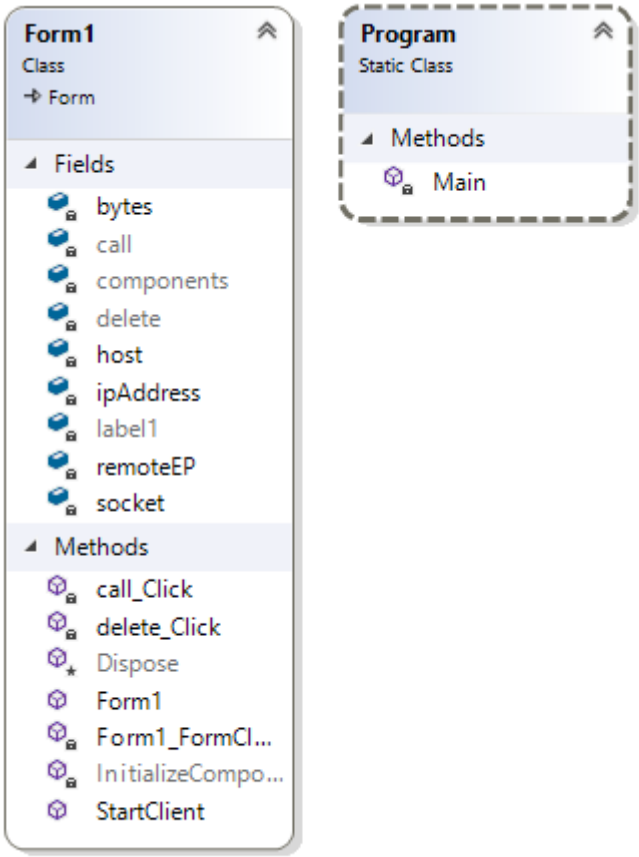


תרשים UML

Server Side



Client Side



פירוט מחלקות

Server Side:

מחלקת Program-

פירוט	מאפיינים
-	-

פירוט	פונקציות
הפונקציה הראשית	`public static int Main(String[] args)`
פונקציה המפעילה את צד השרת	`public static void StartServer()`

מחלקת HandleClient-

פירוט	מאפיינים
הסוקט שנוצר כאשר מתקבלת קריאה מצד לקוח	`Socket clientSocket`
מספר מזהה של הסוקט	`string clNo`

פירוט	פונקציות
פותחת סוקט עבור Client חדש שמתחבר לשרת ומפעילה thread חדש	`public void startClient(Socket inClientSocket, string clineNo)`
מפעילה את הסוקט	`private void go()`

מחלקת NurseQueue-

פירוט	מאפיינים
רשימה מקושרת לשמירת הממתינים לקבלת טיפול מהאחות	`static LinkedList<Socket> list`
משתנה המציין האם האחות פנויה או תפוסה	`static bool nurse = false`

פירוט	פונקציות
מוסיפה משתמש לתור הממתינים	`public static void Add(Socket s)`
שולחת את האחות למטופל הבא	`public static void Delete(Socket s)`

Client Side

לחצן מצוקה לקריאה לאחות

מחלקת Program-

פירוט	מאפיינים
-	-

פירוט	פונקציות
הפונקציה הראשית	`static void Main()`

מחלקת Form1-

פירוט	מאפיינים
חיבור לשרת מרוחק	`IPHostEntry host`
כתובת הIP של השרת	`IPAddress ipAddress`
חיבור בין השרת והלקוח בשימוש ב Port	`IPEndPoint remoteEP`
סוקט של הלקוח	`Socket socket`
מערך עזר של בתים לקבלת ושליחת הודעות מקודדות	`byte[] bytes = new byte[1024]`

פירוט	פונקציות
פותחת חיבור בין הלקוח לשרת	`public void StartClient()`
בנאי המאתחל את הForm	`public Form1()`
בלחיצה על הכפתור, תשלח הודעה לשרת על מטופל חדש הזקוק לטיפול והוא יכנס לתור הממתינים לאחות.	`private void call_Click(object sender, EventArgs e)`
בלחיצה על הכפתור, תשלח הודעה לשרת על סיום הטיפול בכדי שהאחות תוכל להתפנות למטופל הבא.(אם המטופל עוד לא קיבל שירות מהאחות אך רוצה לבטל את הקריאה בלחיצה על הכפתור הוא ייצא מתור הממתינים*	`private void delete_Click(object sender, EventArgs e)`
בסגירת הטופס הפונקציה סוגרת את הסוקט	`private void Form1_FormClosing(object sender, FormClosingEventArgs e)`

קטעי קוד

קטעי קוד בצרת שרת

מחלקת Program

```
    class Program
    {
        public static int Main(String[] args)
        {
            StartServer();
            return 0;
        }
        public static void StartServer()
        {
            // Get Host IP Address that is used to establish a connection
            // In this case, we get one IP address of localhost that is IP :
127.0.0.1
            // If a host has multiple addresses, you will get a list of
addresses
            int counter = 0;
            IPHostEntry host = Dns.GetHostEntry("localhost");
            IPAddress ipAddress = host.AddressList[0];
            IPEndPoint localEndPoint = new IPEndPoint(ipAddress, 3000);
            // Create a Socket that will use Tcp protocol

            Socket listener = new Socket(ipAddress.AddressFamily,
SocketType.Stream, ProtocolType.Tcp);
            // A Socket must be associated with an endpoint using the Bind
method
            listener.Bind(localEndPoint);
            // Specify how many requests a Socket can listen before it gives
Server busy response.
            // We will listen 10 requests at a time
            listener.Listen(10);

            while (true)
            {
                try
                {
                    Console.WriteLine("Waiting for a connection...");
                    Socket handler = listener.Accept();
                    counter++;
                    HandleClient client = new HandleClient();
                    client.startClient(handler, Convert.ToString(counter));
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.ToString());
                }

               //Console.WriteLine("\n Press any key to continue...");
               //Console.ReadKey();
            }
        }
    }
```

מחלקת HandeleClient

```
class HandleClient
  {
      Socket clientSocket;
      string clNo;
      public void startClient(Socket inClientSocket, string clineNo)
      {
          this.clientSocket = inClientSocket;
          this.clNo = clineNo;
          Thread ctThread = new Thread(go);
          ctThread.Start();
      }
      private void go()
      {
          // Incoming data from the client.
          string data = null;
          byte[] bytes = null;
          bytes = new byte[1024];

          try
          {
              while (true)
              {
                  bytes.Clone();
                  int bytesRec = clientSocket.Receive(bytes);
                  data = Encoding.UTF8.GetString(bytes, 0, bytesRec);
                  if (data.IndexOf("<EOF>") > -1)
                  {
                      break;
                  }
                  switch (data)
                  {
                      case "1":
                          NurseQueue.Add(clientSocket);
                          break;
                      case "0":
                          NurseQueue.Delete(clientSocket);
                          break;
                  }
              }
          }
          catch (Exception ex)
          {
              Console.WriteLine(" >> " + ex.ToString());
          }
          finally
          {
              clientSocket.Shutdown(SocketShutdown.Both);
              clientSocket.Close();
          }
      }
```

מחלקת NurseQueue

```
static class NurseQueue
    {
        static LinkedList<Socket> list = new LinkedList<Socket>();// רשימת ממתינים
לאחות
        static bool nurse = false;

        public static void Add(Socket s)//הוספת ממתין לתור
        {
            lock ("key")
            {
                byte[] msg;
                list.AddLast(s);
                if (nurse == false && list.First.Value == s)
                {
                    nurse = true;
                    msg = Encoding.UTF8.GetBytes("האחות לשרותך");
                }
                else
                    msg = Encoding.UTF8.GetBytes("האחות עסוקה כרגע. המתן בסבלנות");

                // Send the data through the socket.
                int bytesSent = s.Send(msg);
            }
        }
        public static void Delete(Socket s)//סיום טיפול במטופל ושחרור האחות לצטופל הבא
        {
            lock ("key")
            {
                byte[] msg;
                if (list.Find(s) != null)
                {
                    if (list.First.Value == s) { nurse = false;  }
                    list.Remove(s);
                    msg = Encoding.UTF8.GetBytes("שמחנו לעזור לך");
                }
                else msg = Encoding.UTF8.GetBytes("שגיאה. אינך נמצא בתור");

                // Send the data through the socket.
                int bytesSent = s.Send(msg);
            }
        }
    }
```

קטעי קוד בצד לקוח

מחלקת Program

```
static class Program
    {
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
```

מחלקת Form1

```
    public partial class Form1 : Form
    {
        IPHostEntry host;
        IPAddress ipAddress;
        IPEndPoint remoteEP;
        Socket socket;
        //משתנה אסקי ששולח את הנתונים בסוקט
        byte[] bytes = new byte[1024];

        public void StartClient()
        {
            try
            {
                // Connect to a Remote server
                // Get Host IP Address that is used to establish a connection
                // In this case, we get one IP address of localhost that is IP
: 127.0.0.1
                // If a host has multiple addresses, you will get a list of
addresses
                host = Dns.GetHostEntry("localhost");
                ipAddress = host.AddressList[0];
                remoteEP = new IPEndPoint(ipAddress, 3000);

                // Create a TCP/IP  socket.
                socket = new Socket(ipAddress.AddressFamily,
                    SocketType.Stream, ProtocolType.Tcp);

                // Connect the socket to the remote endpoint. Catch any errors.
                try
                {
```

```
                    // Connect to Remote EndPoint
                    socket.Connect(remoteEP);
                    Console.WriteLine("Socket connected to {0}",
                        socket.RemoteEndPoint.ToString());
                }
                catch (ArgumentNullException ane)
                {
                    Console.WriteLine("ArgumentNullException : {0}",
ane.ToString());
                }
                catch (SocketException se)
                {
                    Console.WriteLine("SocketException : {0}", se.ToString());
                }
                catch (Exception e)
                {
                    Console.WriteLine("Unexpected exception : {0}",
e.ToString());
                }

            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }



        public Form1()
        {
            InitializeComponent();
            StartClient();
        }


        private void delete_Click(object sender, EventArgs e)
        {
            byte[] msg = Encoding.UTF8.GetBytes("0");

            // Send the data through the socket.
            int bytesSent = socket.Send(msg);

            // Receive the response from the remote device.
            int bytesRec = socket.Receive(bytes);
            String s = Encoding.UTF8.GetString(bytes, 0, bytesRec);

            // Displays the MessageBox.
            MessageBox.Show(s);
            call.Enabled = true;
            call.BackColor = Color.FromName("Red");
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
             socket.Shutdown(SocketShutdown.Both);
             socket.Close();
        }
```

```
        private void call_Click(object sender, EventArgs e)
        {
            byte[] msg = Encoding.UTF8.GetBytes("1");

            // Send the data through the socket.
            int bytesSent = socket.Send(msg);

            // Receive the response from the remote device.
            int bytesRec = socket.Receive(bytes);
            String s = Encoding.UTF8.GetString(bytes, 0, bytesRec);

            // Displays the MessageBox.
            MessageBox.Show(s);
            call.Enabled = false;
            call.BackColor = Color.FromName("LightGray");
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
```