בס"ד

פרויקט גמר

למילוי חלקי של הדרישות לקבלת תואר הנדסאי

הנדסת תוכנה

בהתמחות: מחשבים

<u>נושא הפרויקט:</u> Business calculator

<u>שם הסטודנטית:</u> דסי שפירא

<u>העבודה בוצעה בהנחיית:</u> הגב' כהן

תוכן

מבוא

על מנת להבין ולהתעמק בנושא המהותי של חישוב מקבילי ומבוזר ניסיתי לחשוב על נושא שישלב את שתי הבחינות – חישוב מקבילי ומבוזר. ולכן שילבתי בפרויקט את הבחינה של מקביליות, ואפשרתי על ידי threads לכמה חישובים בו זמנית. ובנוסף שימוש במערכות מבוזרות, על ידי

שימוש בחיבור של sockets.

פרויקט זה מחשב בדך יעילה הוצאות והכנסות של לקוח למשך שנה, בכל אחד מחדשי השנה, ע"י הזנת הנתונים לשנה הקרובה, וסכום השקעה ראשוני.

הפרויקט מתבסס על אלגוריתם prefix.

בהינתן סדרת מספרים, מערך ה- prefix sum שלה הוא מערך בו כל תא מכיל את סכום כל המספרים הנמצאים בסדרת המספרים לפניו, כולל הוא בעצמו.

לדוגמה:

סדרת מספרים	1	2	3	4	5	6	7	8
Prefix sum	1	3	6	10	15	21	28	36

תהליך המחקר

ראשית, מצאתי את האלגוריתם הסדרתי לחישוב ה- prefix sum. הוא פשוט ביותר:

```
for (int i = 1; i < arrSidrati.Length; i++)
{
    arrSidrati[i] = arrSidrati[i - 1] + arrSidrati[i];
}
```

כעת ניתן לחשוב, כיצד ניתן למקבל את האלגוריתם.

מטרות

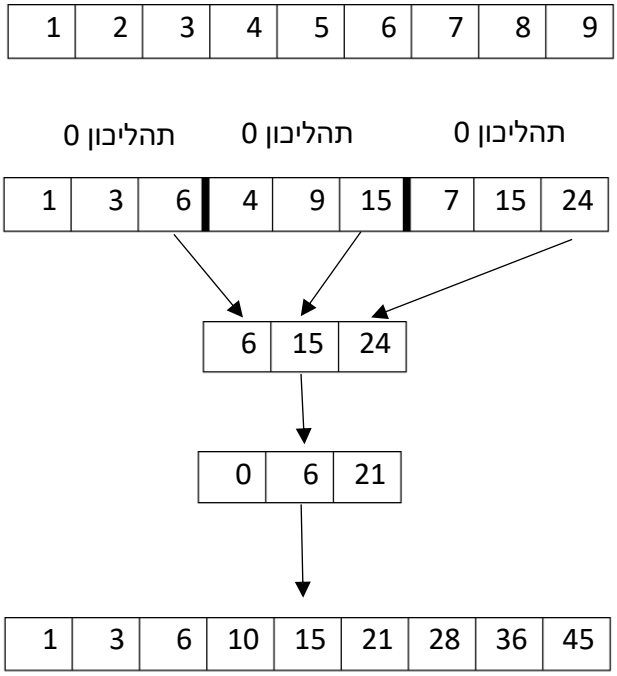
- לימוד והתנסות בכתיבת אלגוריתם ממוקבל ומבוזר.
- כתיבת תוכנית המבצעת באופן יעיל, חישוב prefix sum.

תיאור האלגוריתם הנבחר

חישוב האלגוריתם הממוקבל יתבצע בשלושה מהלכים:

- כל ליבה / תהליכון מחשב את ה- prefix sum של האיברים במחיצה שלו באופן מקומי.
- האיברים האחרונים, שהם גם הגדולים יותר, של כל תהליכון- מועברים למערך עזר, כאשר המקום הראשון במערך מאותחל ל-0.
- כל תהליכון מעדכן את האיברים במחיצה שלו ע"י הוספת הערך המצוי במערך העזרת בהתאמה.

תרשים זרימה של האלגוריתם:



אפיון פונקציונלי

- Prefix_sum_inPlace()
- Sum()
- Insert_prefix()
- End()

פירוט מחלקות

Server Side:

מחלקת Program-

פונקציות המחלקה-

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

מחלקת HandleClient-

משתני המחלקה-

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

פונקציות המחלקה-

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

Client Side

מחלקת Program-

פונקציות המחלקה

פירוט	פונקציות
הפונקציה הראשית	static void Main()

מחלקת Index-

משתני המחלקה-

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

פונקציות המחלקה

פירוט	פונקציות
פותחת חיבור בין הלקוח לשרת	public void StartClient()
בנאי המאתחל את ה Index	Public Index()
בלחיצה על הכפתור תשלח הודעה לשרת האם המספר משתמש תקין ותחזור הודעה מתאימה	button1_Click(object sender, EventArgs e)
בסגירת הטופס הפונקציה סוגרת את הסוקט	private void Form1_FormClosing(object (sender, FormClosingEventArgs e

מחלקת Forme

משתני המחלקה-

שם	סוג	פרוט
n	static int	משתנה המגדיר את גודל המערכים - לשנה
nThread	static int	משתנה המגדיר את מספר ה-threads

partition	static int	משתנה השווה לגודל המערכים לחלק למספר ה-threads
arrString	static int[]	מערך ההכנסות
arrString2	static int[]	מערך ההוצאות
Arrat2	static int[]	מערך ההכנסות וההוצאות
temp	static int[]	מערך עזר
prefix	static int[]	מערך ה-prefix

פונקציות המחלקה-

פרוט	פונקציות
חישוב הכנסות, הוצאות ושניהם.	private void button1_Click_1(object sender, EventArgs e)

פרוט	פונקציות
חלוקת העבודה ל threads	public void prefix_sum_inPlace (int[]array)
חישוב המערך הזמני	public static void sum(int []array,int indexStart, int indexEnd)
סכימת המערך הזמני	public static void insert_prefix()
הצעד האחרון	public static void end
סכימת המערכים הכנסות והוצאות	public static int[] arrto()

קוד

קטעי קוד 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());
                }
            }
```

מחלקת HandleClient

```
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;
            byte[] by = null;
            string msg = 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;
                    }
                    string text
=File.ReadAllText(@"D:\Supply_Calculate\server\ConsoleApp2\castemers.txt");
                   string a = ","+data + ",";
```

```
                if (text.Contains(a))
                {
                    msg = "true";
                    by = Encoding.UTF8.GetBytes(msg);
                }
                else
                {
                    msg = "false";
                    by = Encoding.UTF8.GetBytes(msg);
                }
                int bytesSent = clientSocket.Send(by);

            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(" >> " + ex.ToString());
        }
        finally
        {
            clientSocket.Shutdown(SocketShutdown.Both);
            clientSocket.Close();
        }

    }
```

קטעי קוד לקוח

מחלקת Index

```
public partial class Index : 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 Index()
        {
            InitializeComponent();
            StartClient();

        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            byte[] msg = Encoding.UTF8.GetBytes(textBox1.Text);

            // 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);
            if (s=="true")
            {
                Form1 f1 = new Form1();
                f1.Show();
            }
            else
            {
                MessageBox.Show("הכנס קוד משתמש תקין");
            }

                                                                    }
```

מחלקת Form1

```
  public partial class Form1 : Form
    {
```

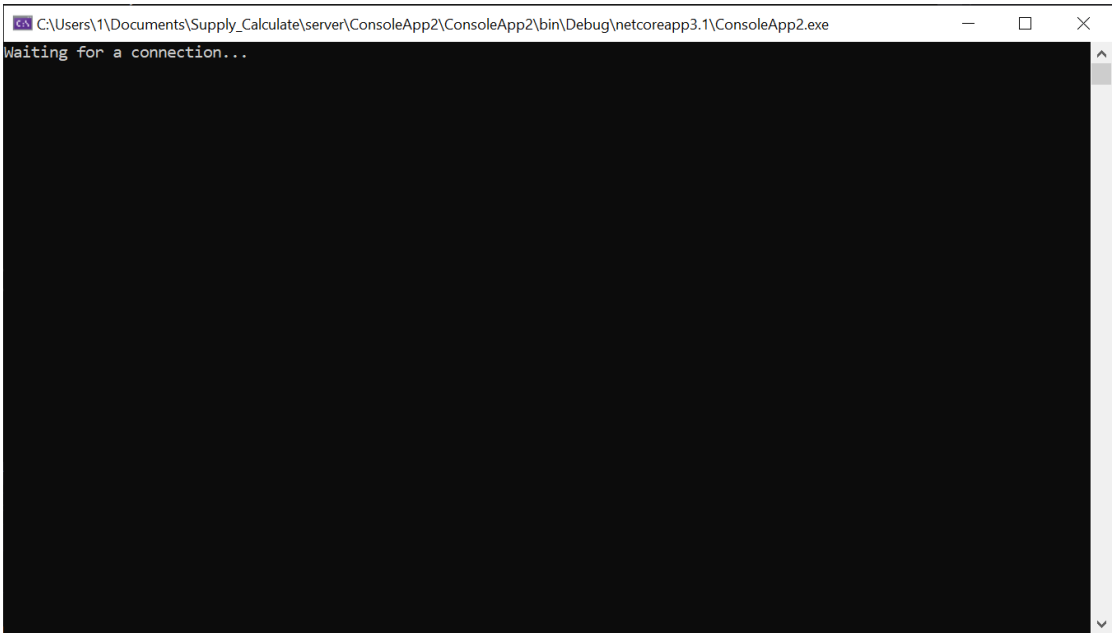
```
        public Form1()
        {
            InitializeComponent();
        }

        static int n = 12;
        static int nThread = 3;
        static int partition = n / nThread;
        static int[] arrString = new int[n];
        static int[] arrString2 = new int[n];
        static int[] arrayt = new int[n];

        static int[] temp = new int[nThread + 1];
        static int[] prefix = new int[nThread];
        private void button1_Click_1(object sender, EventArgs e)
        {
            arrString[0] = Convert.ToInt32(textBox1.Text);
            arrString[1] = Convert.ToInt32(textBoxE2.Text);
            arrString[2] = Convert.ToInt32(textBoxE3.Text);
            arrString[3] = Convert.ToInt32(textBoxE4.Text);
            arrString[4] = Convert.ToInt32(textBoxE5.Text);
            arrString[5] = Convert.ToInt32(textBoxE6.Text);
            arrString[6] = Convert.ToInt32(textBoxE7.Text);
            arrString[7] = Convert.ToInt32(textBoxE8.Text);
            arrString[8] = Convert.ToInt32(textBoxE9.Text);
            arrString[9] = Convert.ToInt32(textBoxE10.Text);
            arrString[10] = Convert.ToInt32(textBoxE11.Text);
            arrString[11] = Convert.ToInt32(textBoxE12.Text);

            arrString2[0] = Convert.ToInt32(textBox1.Text);
            arrString2[1] = Convert.ToInt32(textBoxR2.Text);
            arrString2[2] = Convert.ToInt32(textBoxR3.Text);
            arrString2[3] = Convert.ToInt32(textBoxR4.Text);
            arrString2[4] = Convert.ToInt32(textBoxR5.Text);
            arrString2[5] = Convert.ToInt32(textBoxR6.Text);
            arrString2[6] = Convert.ToInt32(textBoxR7.Text);
            arrString2[7] = Convert.ToInt32(textBoxR8.Text);
            arrString2[8] = Convert.ToInt32(textBoxR9.Text);
            arrString2[9] = Convert.ToInt32(textBoxR10.Text);
            arrString2[10] = Convert.ToInt32(textBoxR11.Text);
            arrString2[11] = Convert.ToInt32(textBoxR12.Text);

            temp[0] = 0;
            //to
            arrayt = arrto();

            //enter
            prefix_sum_inPlace(arrString);
            int j;
            for (j = 0; j < arrString.Length; j++)
            {
                label11.Text = label11.Text + arrString[j].ToString() + '\n';
            }
            label11.Text = label11.Text + (Convert.ToInt32(textBoxE13.Text) +
arrString[--j]).ToString() + '\n';

           //remove
            prefix_sum_inPlace(arrString2);

            for (j = 0; j < arrString2.Length; j++)
            {
                label96.Text = label96.Text + arrString2[j].ToString() + '\n';
```

```
            }
            label96.Text = label96.Text + (arrString2[--j] +
Convert.ToInt32(textBoxR13.Text)).ToString() + '\n';
            //to
            prefix_sum_inPlace(arrayt);
            for (j = 0; j < arrayt.Length; j++)
            {
                labelM.Text = labelM.Text + arrayt[j].ToString() + '\n';
            }
            labelM.Text = labelM.Text + (arrayt[--j] +
Convert.ToInt32(textBoxR13.Text)+ Convert.ToInt32(textBoxE13.Text)).ToString()
+ '\n';


        }

        public void prefix_sum_inPlace(int[]array)
        {

            Parallel.For(0, nThread, i =>
            {
                sum(array,i * partition, (i * partition) + partition);
            });
            insert_prefix();

            Parallel.For(0, nThread, i =>
            {
                end(array,i * partition, (i * partition) + partition);
            });

        }
        public static void sum(int []array,int indexStart, int indexEnd)// חישוב
המערך הזמני
        {
            int sum = 0;
            int local = indexEnd / partition - 1;
            for (int i = indexStart; i < indexEnd; i++)
            {
                sum += array[i];
                array[i] = sum;
            }
            temp[local + 1] = array[indexEnd - 1];
        }
        public static void insert_prefix()//סכמת המערך הזמני
        {
            prefix[0] = 0;
            for (int i = 1; i < nThread; i++)
            {
                prefix[i] = temp[i - 1] + temp[i];
            }
        }
        public static void end(int[]array ,int indexStart, int indexEnd)// הצעד
האחרון
        {
            int local = indexEnd / partition - 1;
            for (int i = indexStart; i < indexEnd; i++)
            {
                array[i] += prefix[local];
            }

        }
        //to
```

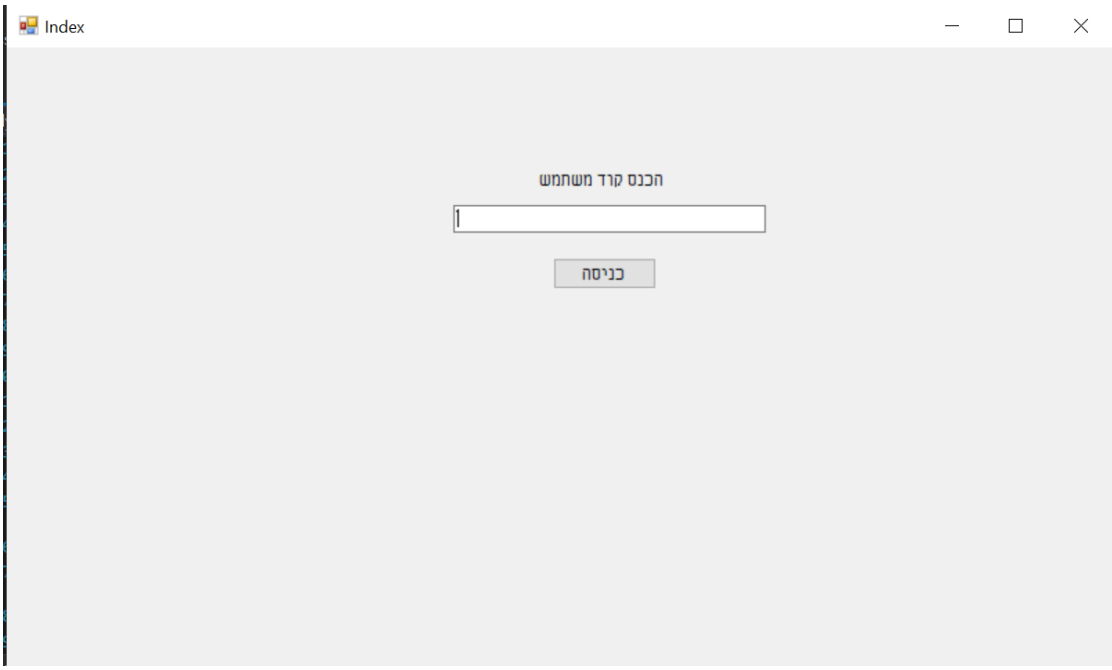
```
public static int[] arrto()
{
    arrayt[0] = arrString[0];
    for (int i = 1; i < arrString.Length; i++)
    {
        arrayt[i] = arrString[i] + arrString2[i];
    }
    return arrayt;
}
```

מסכים

לאחר הרצת השרת יופיע המסך הזה:

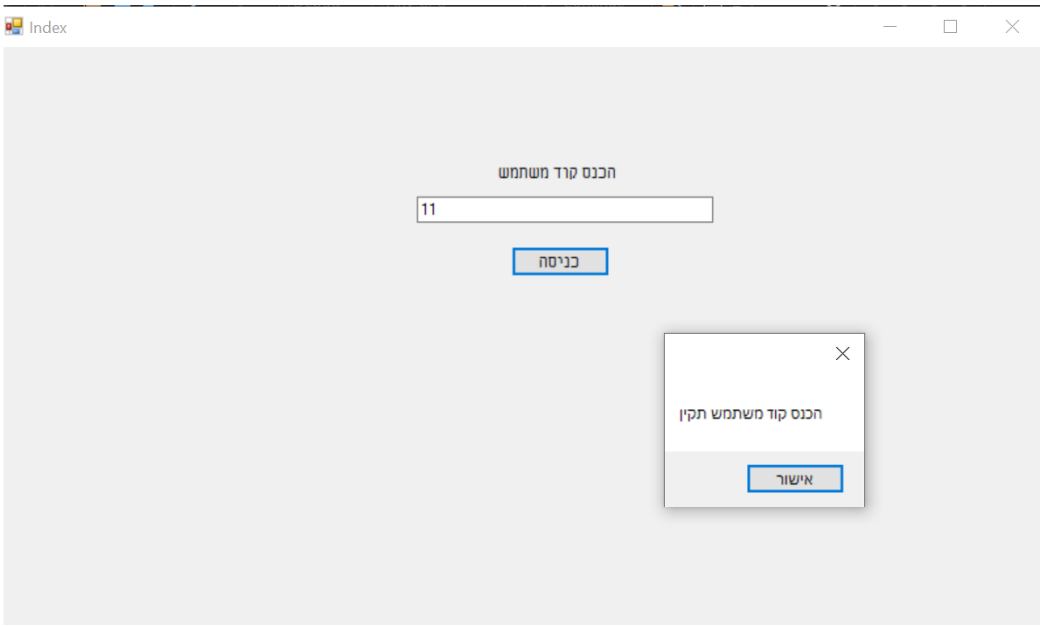


אחרי הרצת הלקוח יפיע מסך הכניסה המשתמש יכניס קוד משתמש:

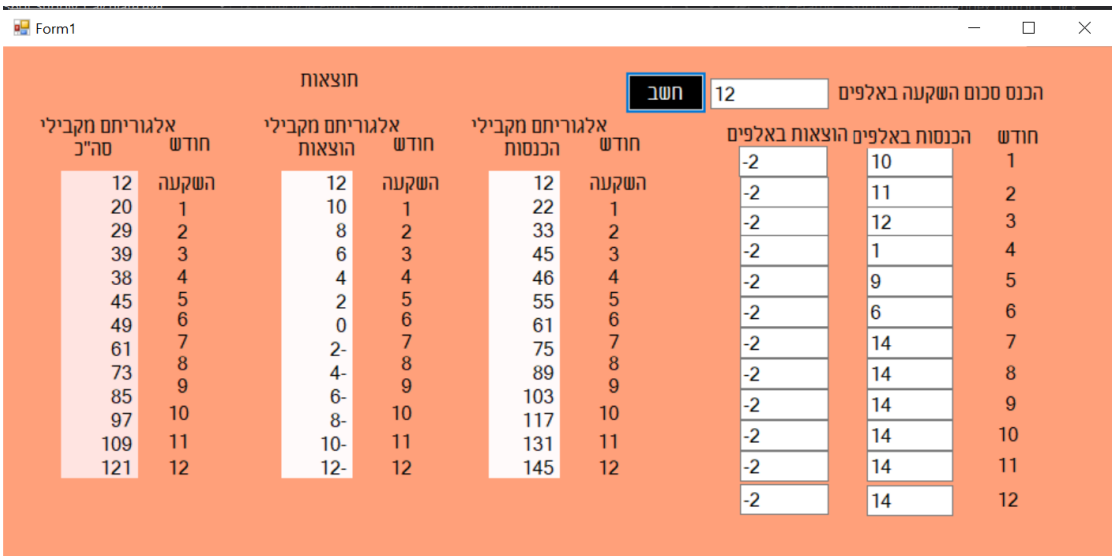


השרת יבדוק אם הקוד משתמש תקין וישלח הודעה מתאימה או יעביר למסך החישוב

קוד משתמש שגוי:

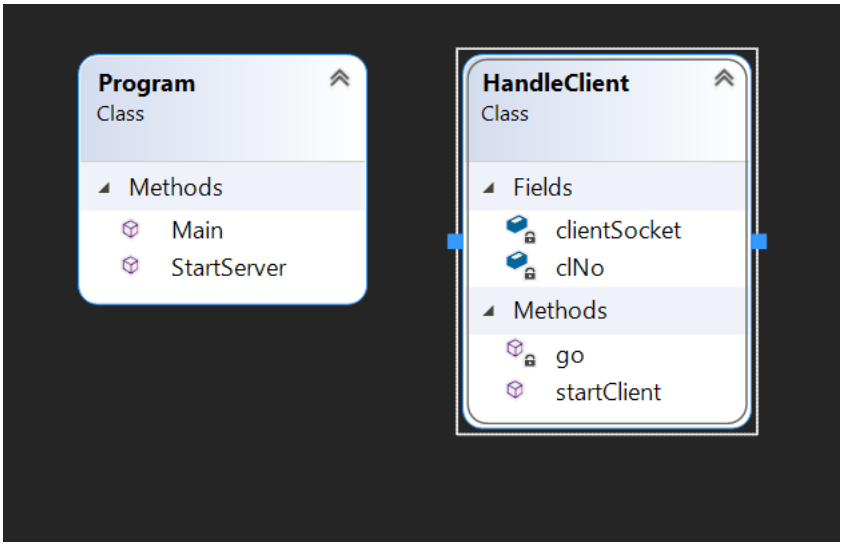


קוד משתמש תקין יעביר למסך החישוב

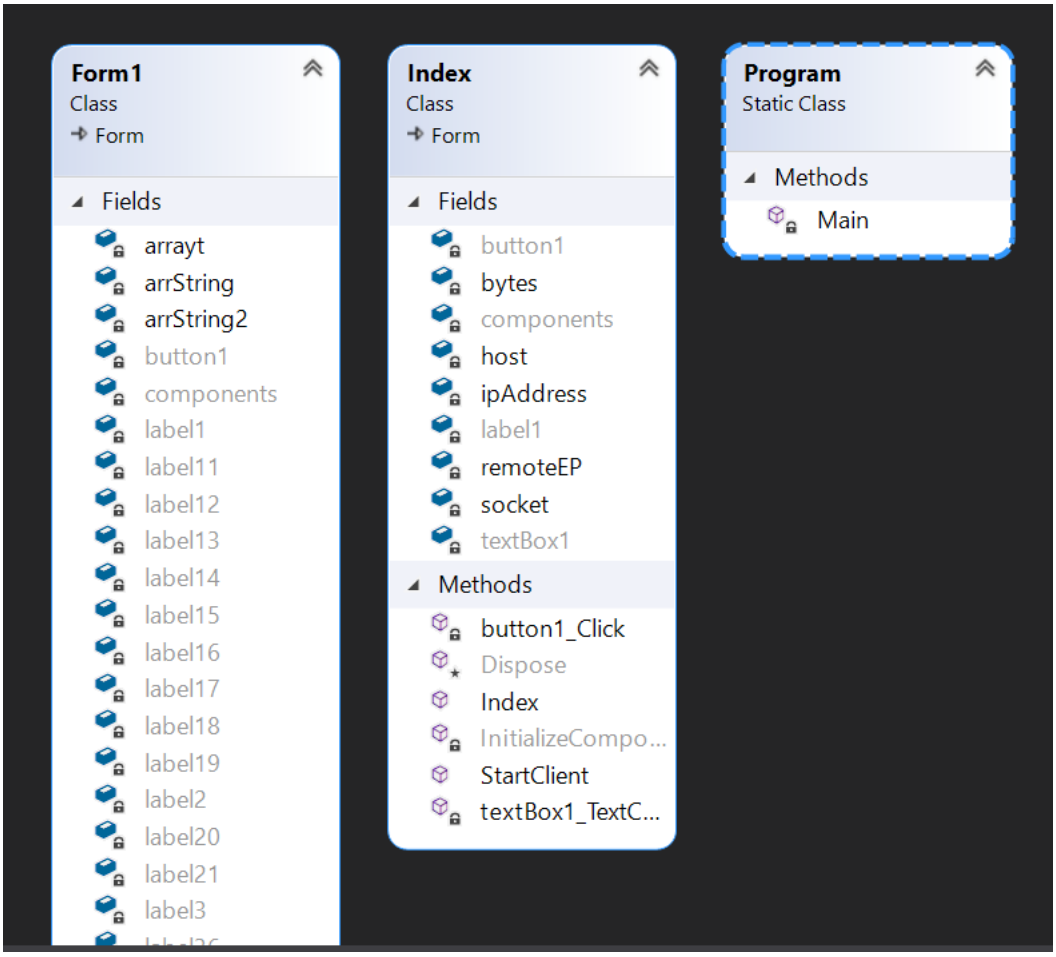


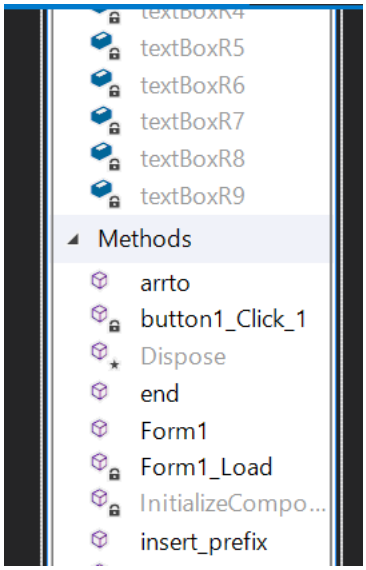
תרשים UML

מחלקת שרת

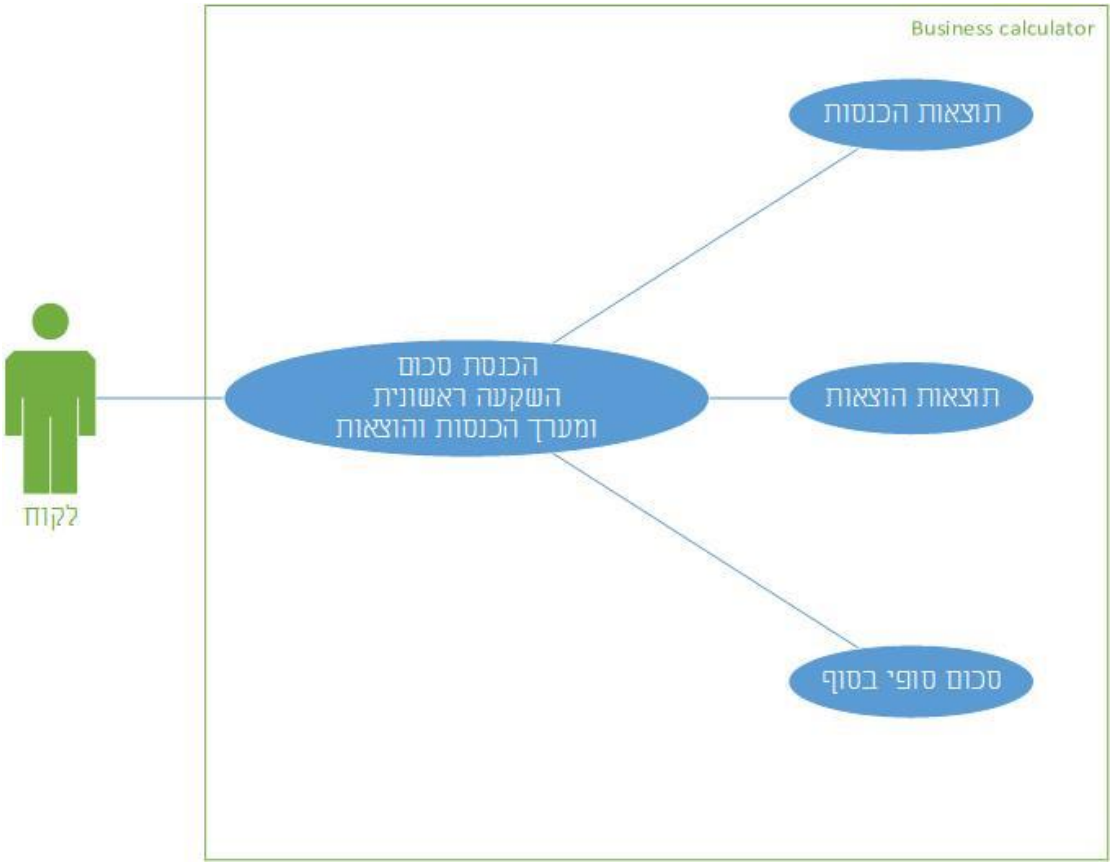


מחלקת לקוח





תרשים US



הוראות למשתמש

הלקוח מכניס שם משתמש תקין ולאחר מכן עובר למסך החישוב

הלקוח מכניס סכום השקעה ראשוני, הכנסות והוצאות של בית ההעסק לשנה זו.

בחירה על "תוצאות הכנסות" תראה למשתמש מה יהיו ההכנסות שלו בכל אותה השנה.

בחירה על "תוצאות הוצאות" תראה למשתמש מה יהיו ההוצאות שלו בכל אותה השנה.

בחירה על "סכום סופי בסוף" תראה למשתמש מה יהיו הכנסות והוצאות שלו באותה השנה.

תוצאות ההרצה:

Number of threads	1	2	4
Time	0.48436	0.42488	**0.43332**
Speedup		1.13999	**1.11778**

השיפור שמושג עם שתי ליבות הוא רק 13%, ואילו עם ארבע ליבות- 11% בלבד, עקב התקורה הכרוכה במקביליות.

למעשה, במקרה שלפנינו רוב הזמן שנמדד הוא תקורה מקבילית, ורק חלקו הקטן הוא זמן חישוב נטו.

מסד נתונים

קובץ טקסט המכיל את קודי המשתמשים

קובץ עריכה עיצוב תצוגה עזרה

kjhu2,9887hu,tttt,111,222,333,444,555,666,333,909,152687,888

מבני נתונים בהם משתמשים בפרויקט

מערך אחד המייצג את הסדרה החשבונית, ומערך נוסף- המכיל את ה- prefix sumses.

מסקנות

עם מעט מחשבה, ניתן לייעל חישובים מסובכים- ולהופכם לקלים וקצרים הרבה יותר. גם אם המאמץ גדול יותר, התוצאה- שווה.

פיתוחים עתידיים

אם נוסיף פונקציה המבצעת עבודת סרק ובכך מגדילה את זמן החישוב, יעילות המקבול תגדל כפי שנראה בטבלה הבאה:

Dummy time	31.13009	15.56909	**9.00891**
Dummy speedup		1.99883	**3.45548**