מכללת אורט ק. ביאליק

פ ר ו י י ק ט ג מ ר

למילוי חלקי של הדרישות לקבלת תואר

ה נ ד ס א י

ה נ ד ס ת - ת ו כ נ ה

בהתמחות : מ ח ש ב י ם

נושא הפרוייקט : פרויקט מעבדה –למידה מרחוק

שם הסטודנט/ית : נתנאל טנג'י

העבודה בוצעה בהנחיית : אודי מלכה, מוטי פניקשיוולי

שנה"ל תשע"ט 2019

מכללת אורט קרית ביאליק לטכנולוגיה מתקדמת ולמדעים

הצהרת סטודנט

אני הסטודנט נתנאל טנג'י מס' ת.ז . 209140581 החתום מטה, מצהיר בזאת שכל עבודת הגמר/ הפרוייקט המוגש/ת בחוברת זו היינו/ה פרי עבודתי בלבד.

על בסיס הנחייתו של המנחה ותוך הסתמכות על מקורות הידע והמידע האחרים המצויים בביליוגרפיה המובאת בחוברת זאת.

אני מודע לאחריות שהנני מקבל על עצמי ע"י חתימתי על הצהרה זו שכל הנאמר בה הינו אמת ורק אמת.

חתימת מגיש החוברת : _____

<u>אישור המנחה :</u>

הנני מאשר הגשת החוברת להערכה _____

תוכן עניינים

תוכן עניינים

מבוא

במעבדה זו מימשתי תוכנה פשוטה ללמידה מרחוק, ובדיקת התקדמות בלימודים או בדיקת נוכחות של התלמיד.

מטרת הפרויקט הייתה מימוש מערכת קליינט – שרת ומניעת deadlock.

מטרות

- יצירת מערכת למידה מרחוק בה מורה יכול לקבל עדכונים מהתלמידים שלו לגבי מה שהם עושים.
- שליחת תמונה/צילום מסך למורה על מנת לבדוק את התלמידים והתקדמותם.
- שליחת מסר מהמורה לכלל התלמידים על מנת לבקש מהם דבר מסוים.
- שמירה על רשימת תמונות מסודרת ועצירת deadlock על ידי מניעת גישה לרשימה על ידי יותר מתהליכון אחד.

עמדת פיתוח

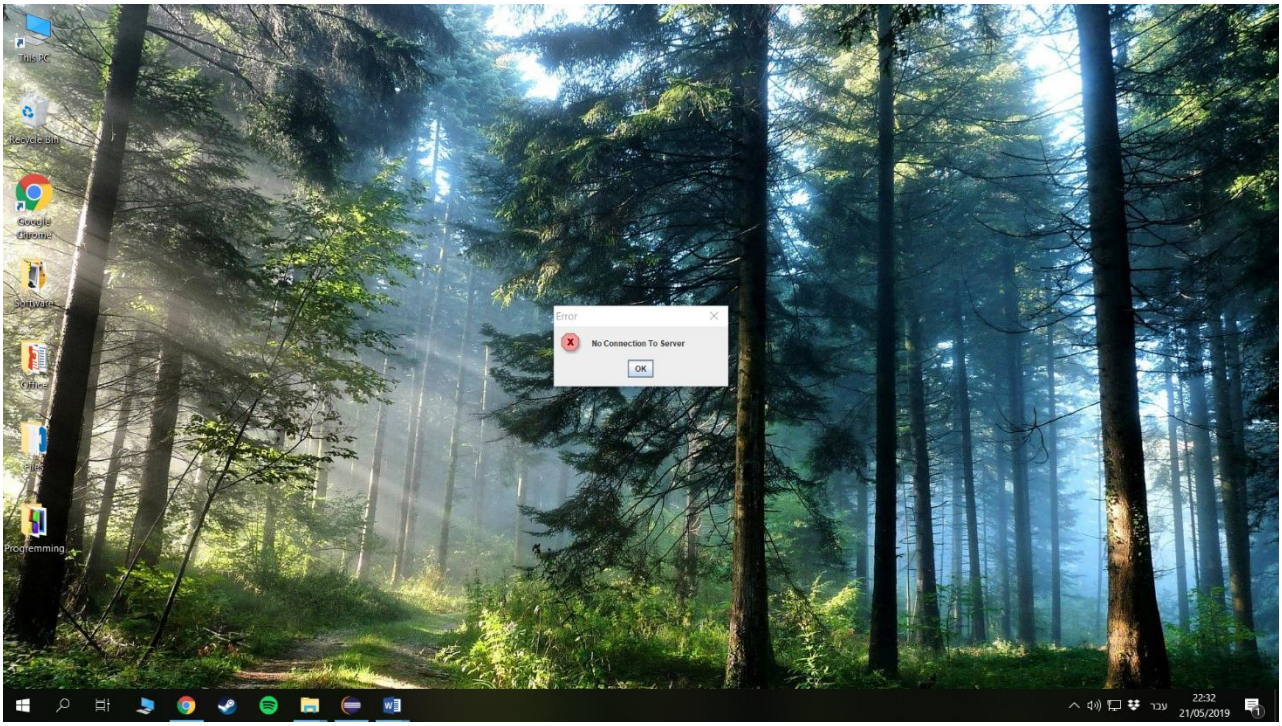
- Windows 7/10
- Eclipse Java Oxyjen

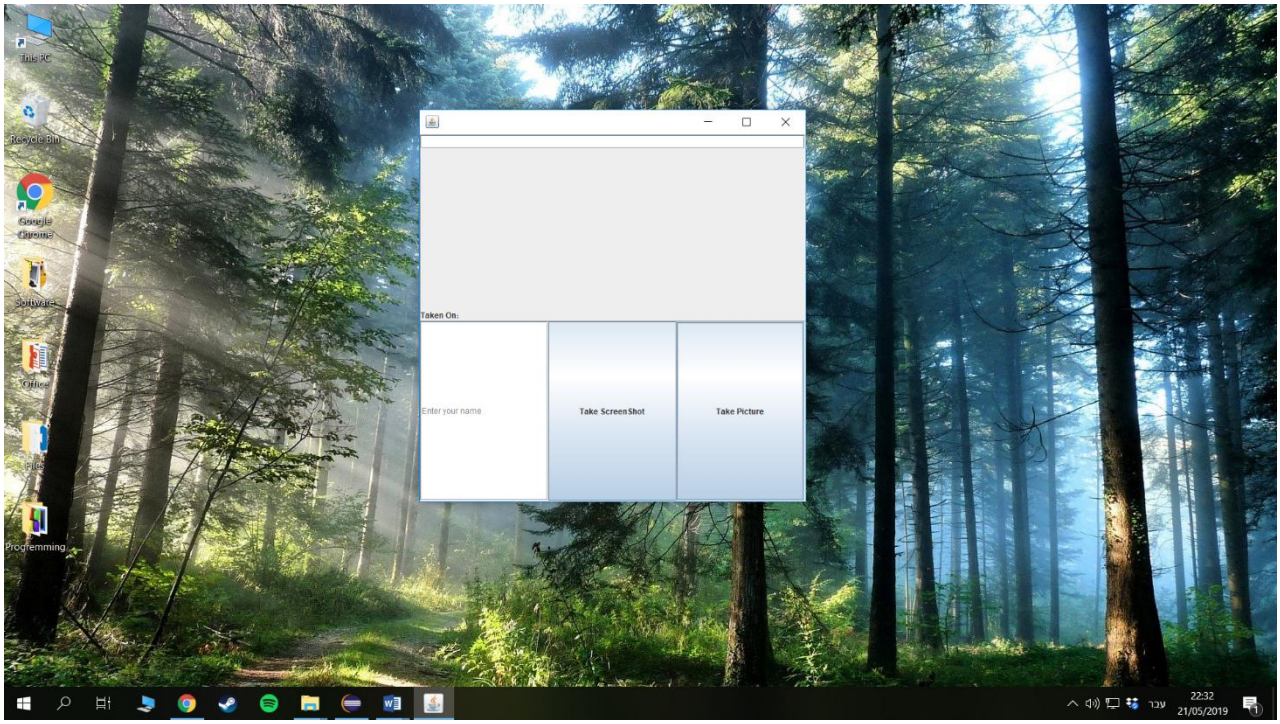
דרישות מערכת מהמשתמש

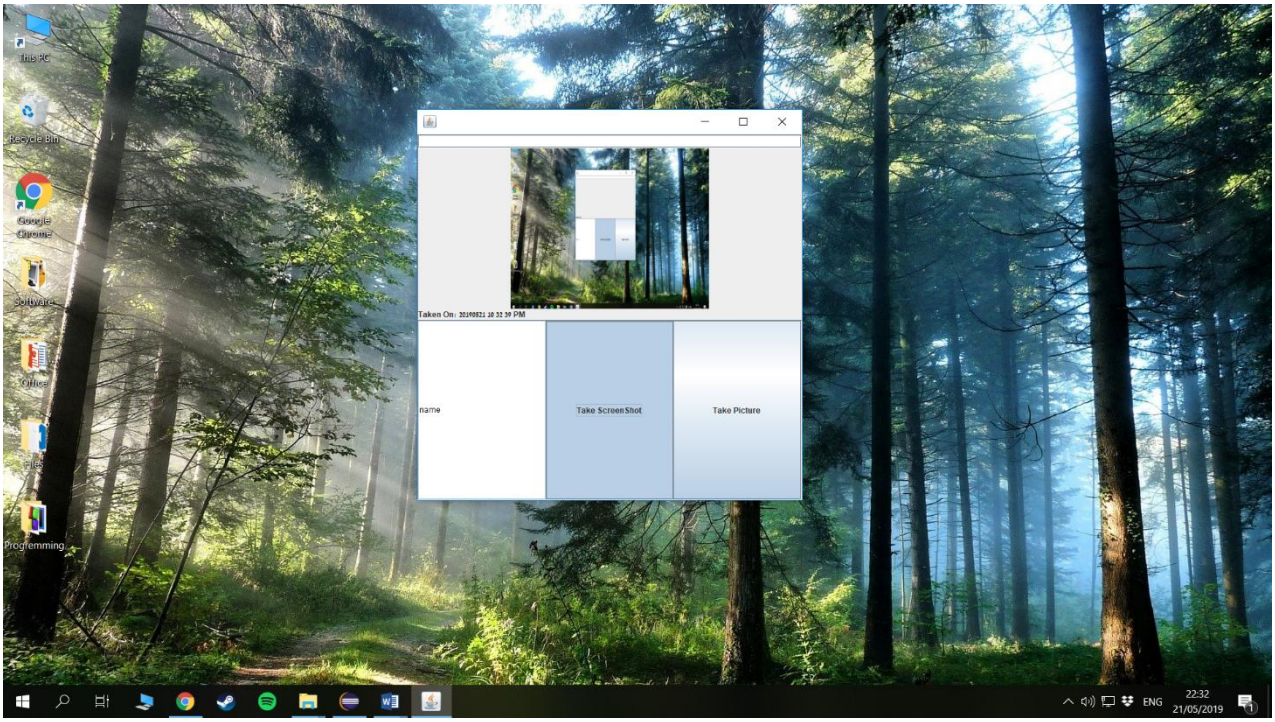
- Windows 7/10
- Java
- חיבור לאינטרנט

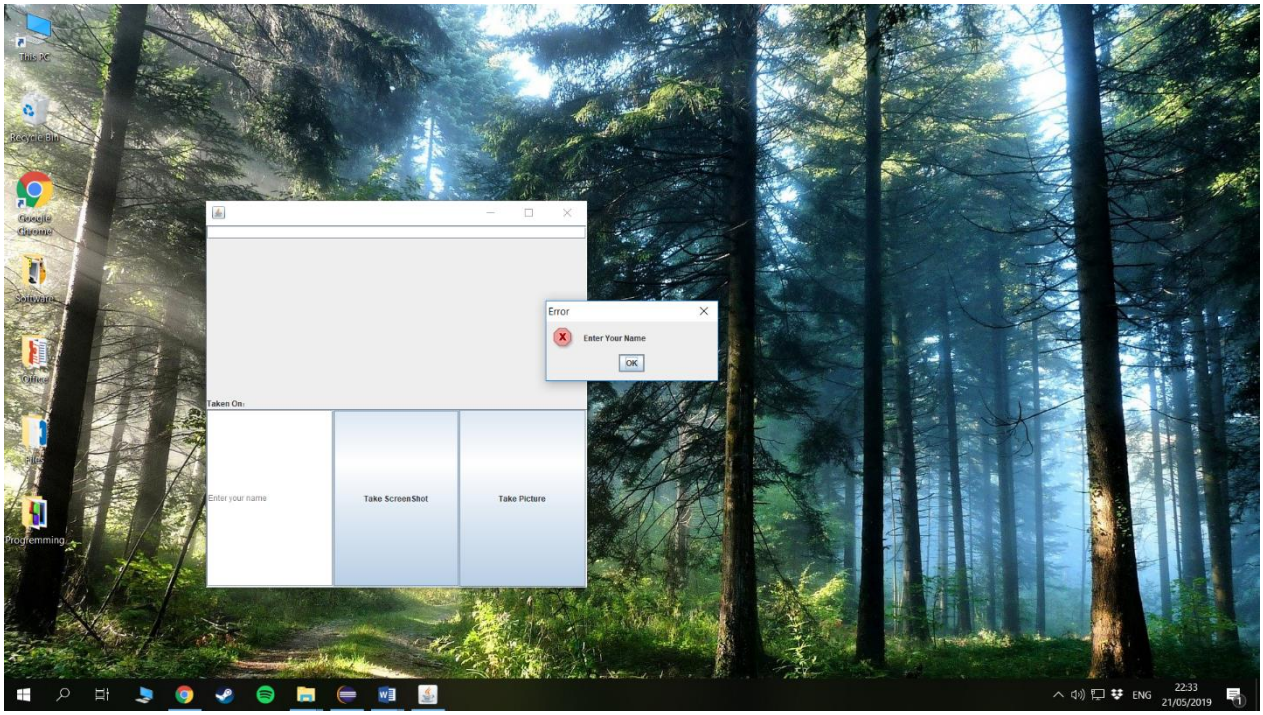
מדריך משתמש

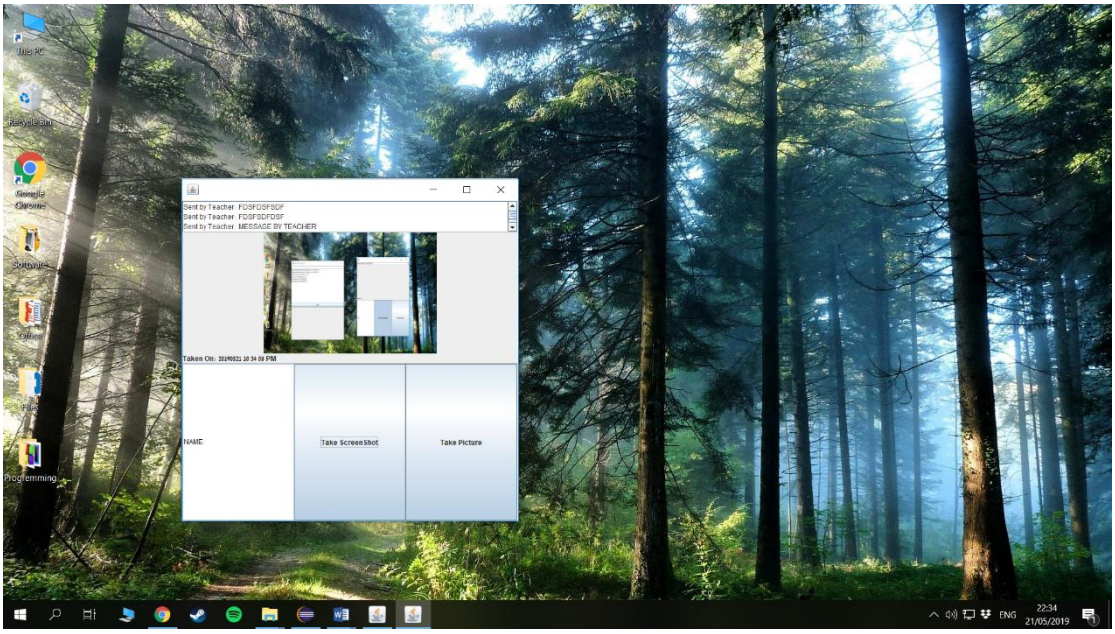
מסך משתמש





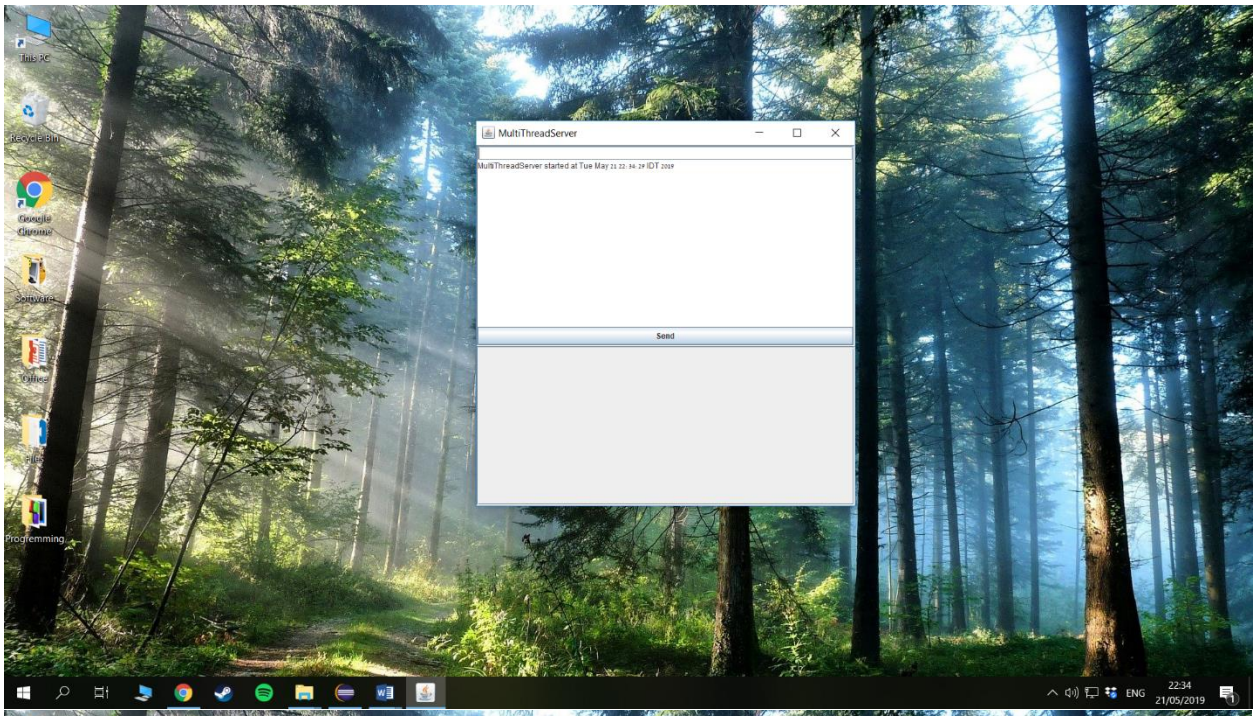

This PC
Recycle Bin
Google Chrome
Software
Office
Files
Programming
Taken On: 20190521 10 32 39 PM
name
Take ScreenShot
Take Picture
ENG
22:32
21/05/2019

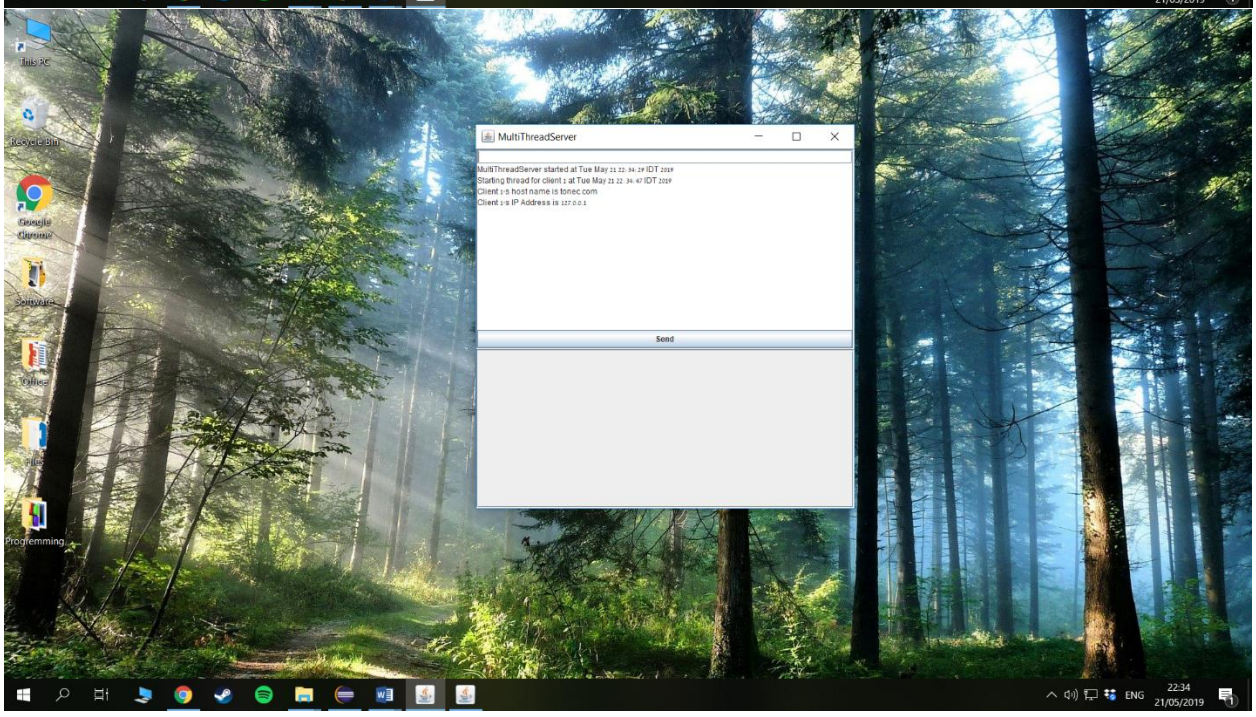

This PC
Recycle Bin
Google Chrome
Software
Office
Files
Programming
Error
Enter Your Name
OK
Taken On:
Enter your name
Take ScreenShot
Take Picture
ENG
22:33
21/05/2019

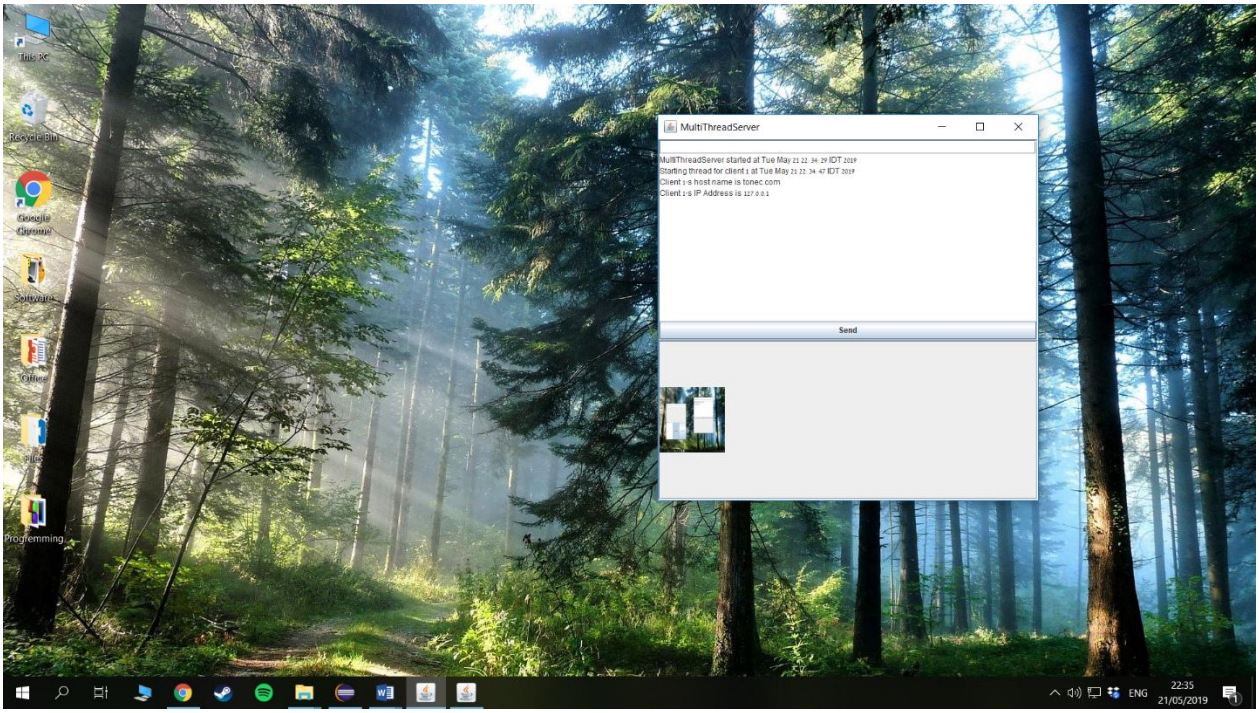


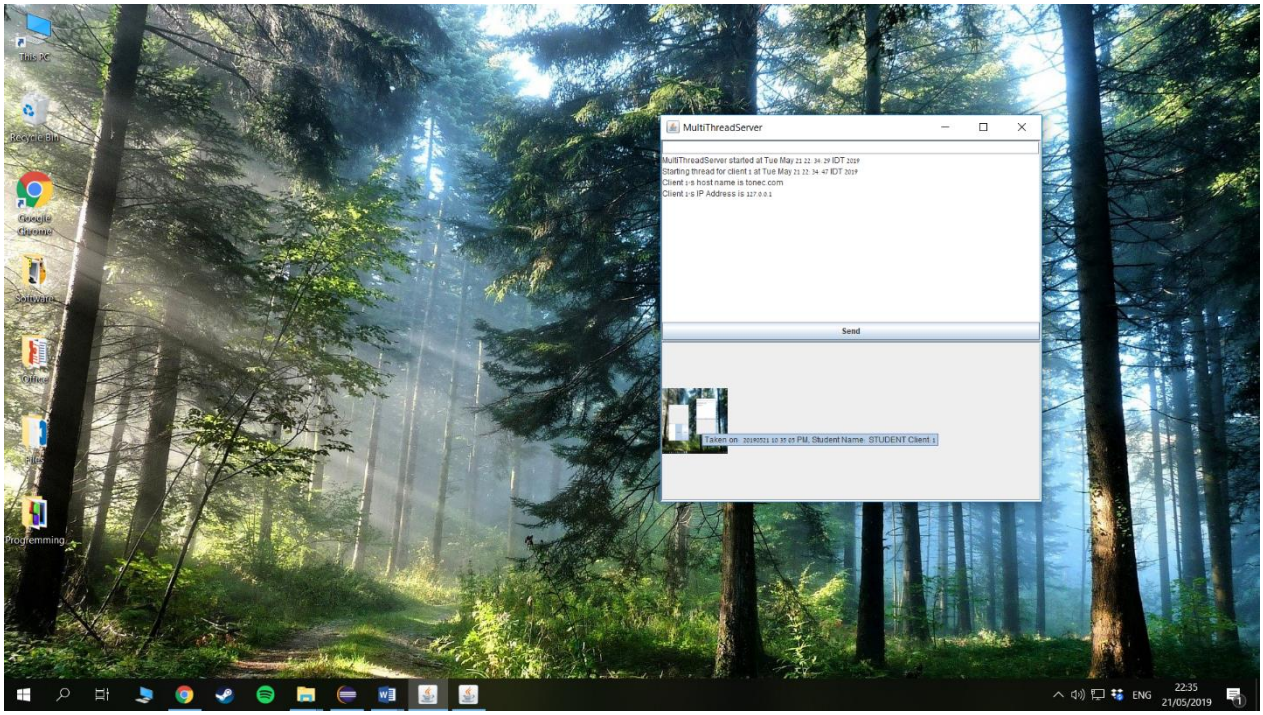
ממסך זה התלמיד יוכל למלא את שמו ולשלוח למורה תמונת שלו/צילום מסך.
בנוסף הוא יוכל לקבל הודעות מהמורה.

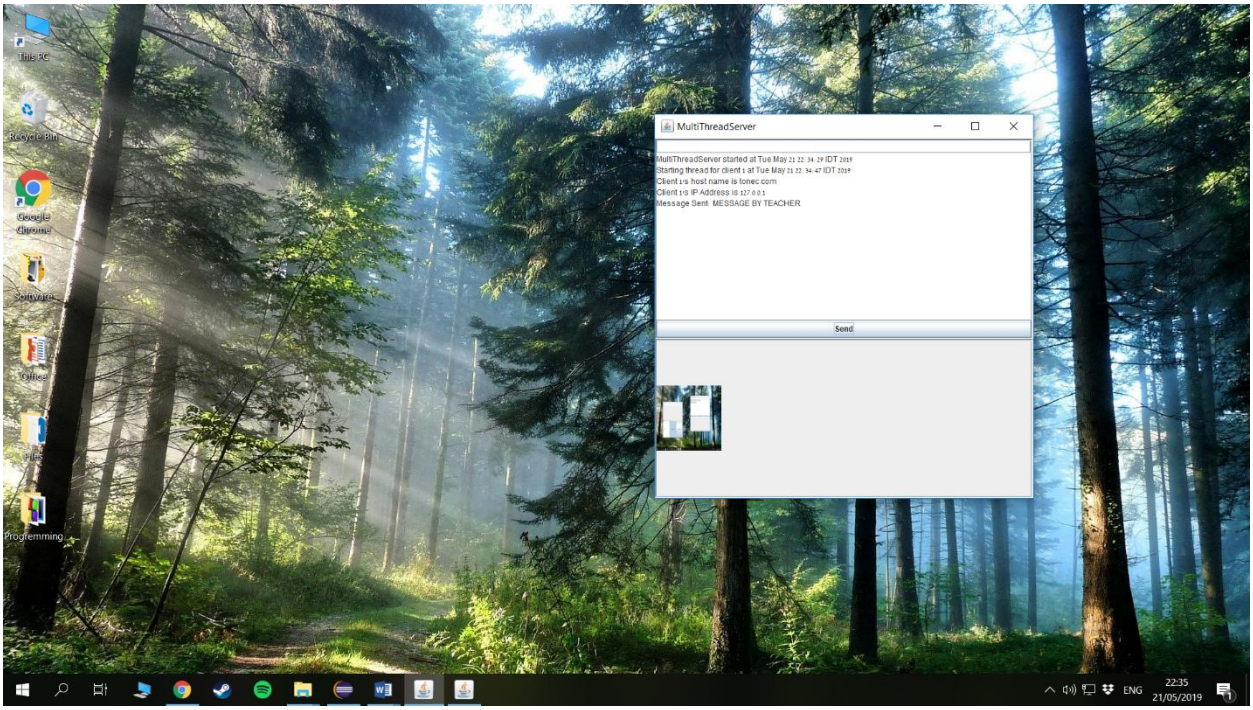
מסך מורה


MultiThreadServer
MultiThreadServer started at Tue May 21 22:34:29 IDT 2019
Send


MultiThreadServer
MultiThreadServer started at Tue May 21 22:34:29 IDT 2019
Starting thread for client 1 at Tue May 21 22:34:47 IDT 2019
Client 1 s host name is tonec.com
Client 1 s IP Address is 127.0.0.1
Send


MultiThreadServer
MultiThreadServer started at Tue May 21 22:34:29 IDT 2019
Starting thread for client 1 at Tue May 21 22:34:47 IDT 2019
Client 1 is host name is tonec.com
Client 1 is IP Address is 127.0.0.1
Send
This PC
Recycle Bin
Google Chrome
Software
Office
Programming
ENG
22:35
21/05/2019


MultiThreadServer
MultiThreadServer started at Tue May 21 22:34:29 IDT 2019
Starting thread for client 1 at Tue May 21 22:34:47 IDT 2019
Client 1 is host name is tonec.com
Client 1 is IP Address is 127.0.0.1
Send
Taken on: 20190521 10:35:05 PM, Student Name: STUDENT Client 1
This PC
Recycle Bin
Google Chrome
Software
Office
Programming
ENG
22:35
21/05/2019



במסך זה המורה יוכל לקבל תמונות מהתלמידים ולשלוח להם הודעות.
כאשר המורה מחזיק את העכבר מעל תמונה הוא יוכל לראות את שם התלמיד ששלח את התמונה ומתי היא נשלחה.

מושגים חשובים

תהליכון(Thread)

תהליכון (באנגלית: Thread of execution, או בקיצור: Thread) ולעיתים חוט, פתיל ריצה או נים הוא מושג במדעי המחשב המשמש במערכות הפעלה כדי לתאר הקשר ריצה במרחב כתובות.

מערכות הפעלה מודרניות מאפשרות לנהל במסגרת ריצה של תהליך (Process) מספר תהליכונים הרצים במקביל במרחב כתובות אחד. במערכות אלו כל תהליך חדש מתחיל את ביצועו באמצעות 'תהליכון ראשי' אשר עשוי בהמשך ליצור תהליכונים נוספים. מנגנון הריצה באמצעות תהליכונים מאפשר לספק למשתמש במערכת ההפעלה מהירות תגובה ורציפות פעולה כאשר התהליך (יישום) מבצע כמה משימות במקביל.

קיפאון(DeadLock)

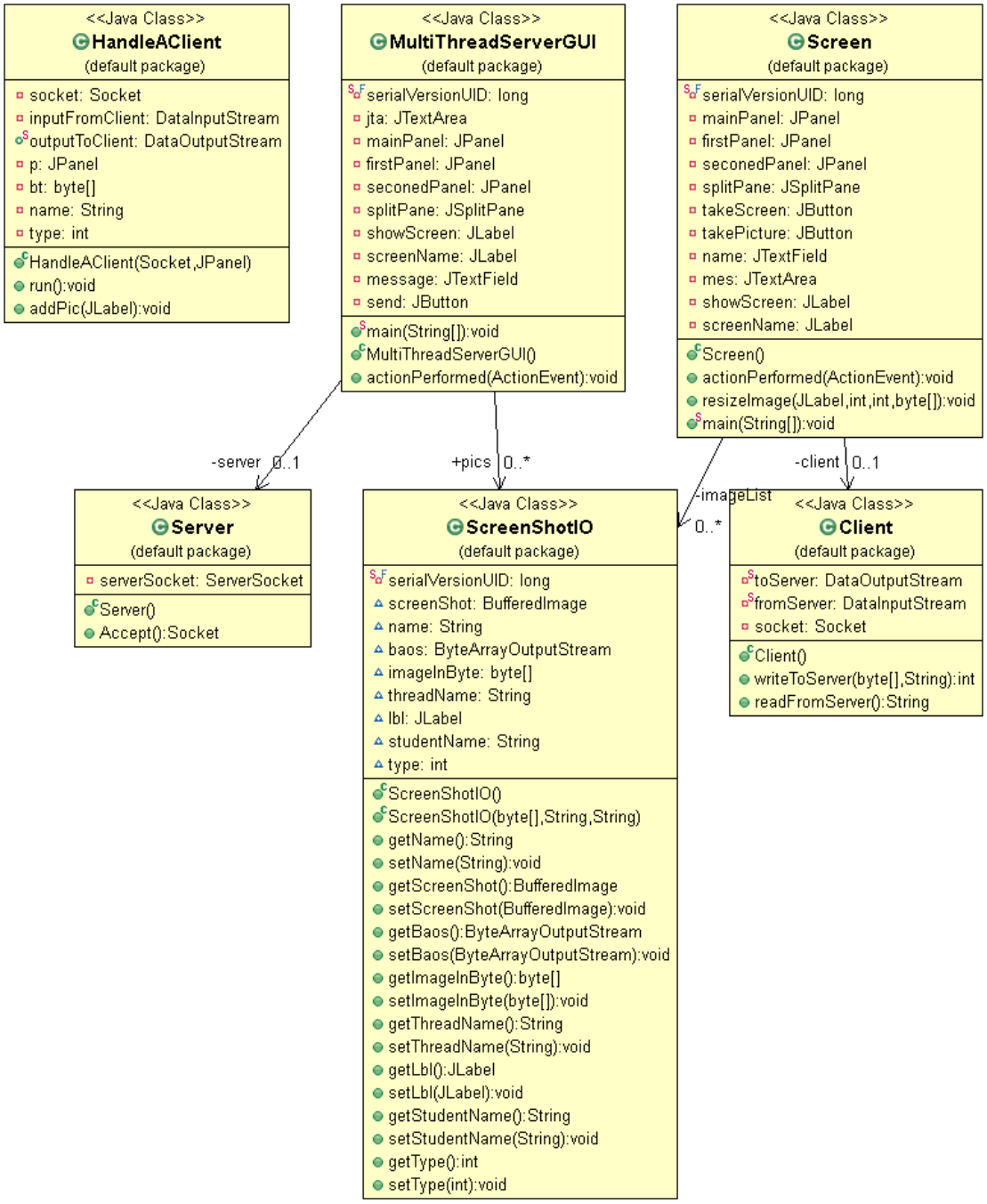
קיפאון (או תיקו, באנגלית: Deadlock) הוא מצב בו שתי פעולות מתחרות מחכות כל אחת לסיומה של האחרת, ומכיוון שכך, אף אחת מהן אינה מסתיימת. דוגמה: שני אנשים עומדים בפתחה של דלת, וכל אחד מהם מציע לרעהו את הזכות להיכנס ראשון. אם יתמידו בגישתם זו, לא יעברו לעולם בדלת.

בענף המחשבים, קיפאון מתייחס למצב בו שני תהליכים מחכים האחד לאחר לשחרורו של משאב, או למצב בו יותר משני תהליכים מחכים למשאבים בשרשרת מעגלית. דוגמה: תהליך א' נועל את משאב A וממתין לשחרורו של משאב B, משום שהוא זקוק לשני משאבים אלה יחדיו לשם השלמת פעולתו. באותו זמן תהליך ב' נועל את משאב B וממתין לשחרורו של משאב A, משום שגם שהוא זקוק לשני משאבים אלה יחדיו לשם השלמת פעולתו. בקשת הנעילה היא פעולה חוסמת ועל כן שני התהליכים הפעילו פעולות שלא יחזרו עד שישוחרר המשאב שביקשו לנעול. אך כיוון שהתהליך שמחזיק במשאב נמצא במצב דומה - הוא לא יכול לשחרר את המשאב ומכאן ששני התהליכים תקועים.

קיפאון הוא בעיה נפוצה בתחום העיבוד המקבילי שבו תהליכים רבים משתפים משאבים באמצעות מנגנון הידוע כ'מנעול תוכנה' או 'מנעול רך'. במערכות מחשב המיועדות לפעול תחת אילוצי זמן אמת, ישנו לרוב התקן חומרה הנקרא 'מנעול קשה' המבטיח גישה בלעדית לתהליכים ומניעת מצבי הרעבה, על ידי כפיית סדר עיבוד.

מצבי קיפאון הם מטרידים ביותר מכיוון שאין פתרון כללי למניעתם. על מתכנתים לנסות ולכתוב תוכניות שלא ייתכנו בהם כלל מצבי קיפאון. תוכניות כאלה נקראות תוכניות שיש להן חָיות (liveness).

תרשים מחלקות UML



אלגוריתמים השובים

קבלת תמונה

```
public void run() {
            try {
                  // Create data input and output streams
                  inputFromClient = new
DataInputStream(socket.getInputStream());
                  outputToClient = new
DataOutputStream(socket.getOutputStream());

                  // Continuously serve the client
                  while (true) {
                        // Receive radius from the client
                        //radius = inputFromClient.readDouble();
                        //type = inputFromClient.read();
                        name = inputFromClient.readUTF();
                        bt = new byte[inputFromClient.readInt()];
                        inputFromClient.readFully(bt, 0, bt.length);

                        int flag = 0;
                        synchronized (MultiThreadServerGUI.pics) {

                        if(MultiThreadServerGUI.pics.size() == 0) {
                              MultiThreadServerGUI.pics.add(new
ScreenShotIO(bt,Thread.currentThread().getName(),name));

      addPic(MultiThreadServerGUI.pics.get(0).getLbl());
                        } else {
                              for(int i = 0; i <
MultiThreadServerGUI.pics.size();i++) {

      if(MultiThreadServerGUI.pics.get(i).getThreadName().equals(Thre
ad.currentThread().getName())) {
                                          for(int j = 0; j <
p.getComponentCount(); j++) {

      if((JLabel)p.getComponent(j) ==
MultiThreadServerGUI.pics.get(i).getLbl()){
                                                      p.remove(j);
                                                      p.revalidate();
                                                      break;
                                                }
                                          }

      MultiThreadServerGUI.pics.remove(i);

      MultiThreadServerGUI.pics.add(i, new
ScreenShotIO(bt,Thread.currentThread().getName(),name));
                                          flag = 1;

      addPic(MultiThreadServerGUI.pics.get(i).getLbl());
                                          break;
                                    }
                              }
                              if(flag == 0) {
                                    MultiThreadServerGUI.pics.add(new
ScreenShotIO(bt,Thread.currentThread().getName(),name));
```

```
      addPic(MultiThreadServerGUI.pics.get(
MultiThreadServerGUI.pics.size() - 1).getLbl());
                                }
                        }
                        }

                }
            } catch (IOException e) {
                System.err.println(e);

}
public void addPic(JLabel lbl) {
            System.out.println(Thread.currentThread().getName());
            p.add(lbl);
            p.revalidate();
            System.out.println(p.getComponentCount());
}
```

קוד זה הוא קבלת התמונה מהקליינט וקליטת התמונה במערך התמונות שבסרבר.
כאשר מתקבלת תמונה המערך ננעל על ידי הקוד:

```
synchronized (MultiThreadServerGUI.pics)
```

שלמעשה סוגר את המערך בשביל תהליכונים אחרים עד שהתהליכון הנוכחי מסיים את קליטת התמונה, והצבתה במסך המורה.

כאשר התמונה מתקבלת נבדק האם התמונה היא ראשונה מתלמיד מסוים או תמונה חדשה מהתלמיד. אם היא חדשה התמונה מוצגת, אם התמונה היא תמונה נוספת מהתלמיד אז התמונה הקודמת שהתלמיד שלח נמחקת, והתמונה החדשה מוצגת.

קבלת הודעה

```
while(true) {
            String thing = client.readFromServer();
            if(!thing.equals("")) {
                mes.append("Sent by Teacher: " + thing + "\n");
            }
}
```

קוד במחלקת מסך התלמיד אשר מאזינה לקבלת הודעות מהמורה והצגתם לתלמיד. הקוד תמיד מאזין כל עוד התוכנית פועלת על מנת לקבל את הודעת המורה.

פירוט מחלקות

מחלקת Client

מחלקה זו אחראית על שליחת מידע מהקליינט ושליחת מידע לקליינט.

פירוט	פונקציה
פונקציה אשר אחראית על שליחת התמונה ושם התלמיד לסרבר.	`public int writeToServer(byte[] b,String name)`
פונקציה אשר אחראית על שליחת הודעת המורה לתלמיד	`public String readFromServer()`

מחלקת HandleAClient

מחלקה אשר מטפלת במידע שהקליינט שולח לסרבר.

פירוט	פונקציה
פונקציה אשר מתקבלת ממשק התהליכון. פונקציה זו תמיד מאזינה לבקשות הקליינט.	`public void run()`
פונקציה אשר מוסיפה את התמונה שהתקבלה למסך המורה.	`public void addPic(JLabel lbl)`

מחלקת MultiThreadServerGUI

מחלקה זו היא ממשק המשתמש של המורה אשר מפה המורה יראה את התמונות שהתקבלו מהתלמידים וישלח להם הודעות.

מחלקת Screen

מחלקה זו היא ממשק המשתמש של התלמיד, מפה התלמיד יוכל לשלוח תמונה/צילום מסך למורה.

פירוט	פונקציה
פונקציה זו מציגה את התמונה שנשלחה לתלמיד. היא מקבלת את התווית שעליה התמונה תמצא, גובה ורוחב של התמונה, והתמונה עצמה.	`public void resizeImage(JLabel label, int h, int w, byte[] b)`

מחלקת ScreenShotIO

מחלקה זו יוצרת אובייקט אשר מחזיק את שם התלמיד, התמונה שנשלחה, מתי היא נשלחה, ומספר הקליינט של התלמיד.

מחלקת Server

מחלקה אשר יוצרת את השרת.

בבליוגרפיה

קיפאון(Deadlock) –

https://he.wikipedia.org/wiki/%D7%A7%D7%99%D7%A4%D7%90%D7%95%D7%9F_(%D7%9E%D7%93%D7%A2%D7%99_%D7%94%D7%9E%D7%97%D7%A9%D7%91)

תהליכון(Thread) -
https://he.wikipedia.org/wiki/%D7%AA%D7%94%D7%9C%D7%99%D7%9B%D7%95%D7%9F

קוד הפרויקט

מחלקת Client

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;

import javax.swing.JOptionPane;

class Client {

	// IO streams
	private static DataOutputStream toServer;
	private static DataInputStream fromServer;

	private Socket socket;

	public Client() {

		try {
			// Create a socket to connect to the server
			socket = new Socket("localhost", 8000);

			// Create an output stream to send data
			// to the server
			toServer = new
DataOutputStream(socket.getOutputStream());

			// Create an input stream to receive data
			// from the server
			fromServer = new
DataInputStream(socket.getInputStream());


		} catch (IOException ex) {
			JOptionPane.showMessageDialog(null, "No Connection
To Server", "Error", 0);
		}
	}

	public int writeToServer(byte[] b,String name) {
		try {
			//toServer.writeInt(type);
			toServer.writeUTF(name);
			toServer.writeInt(b.length);
			toServer.write(b);
			toServer.flush();
			return 1;
		} catch (IOException e) {
			e.printStackTrace();
			//JOptionPane.showMessageDialog(null, "No
Connection To Server", "Error", 0);
			return 0;
		}
	}

	public String readFromServer() {
		try {
			return fromServer.readUTF();
```

```
            } catch (IOException e) {
                  e.printStackTrace();
                  //JOptionPane.showMessageDialog(null, "No
Connection To Server", "Error", 0);
            }
            //return 0;
            return "";
      }

}
```

מחלקת HandleAClient

```
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.IOException;
import java.net.Socket;
import java.util.ArrayList;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

@SuppressWarnings("unused")
class HandleAClient implements Runnable {
      private Socket socket; // A connected socket

      private DataInputStream inputFromClient;
      public static DataOutputStream outputToClient;
      private JPanel p;

      //private double radius;
      //private double area;
      private byte[] bt;
      private String name;
      private int type;

      public HandleAClient(Socket socket,JPanel p) {
            this.socket = socket;
            this.p = p;
      }

      public void run() {
            try {
                  // Create data input and output streams
                  inputFromClient = new
DataInputStream(socket.getInputStream());
                  outputToClient = new
DataOutputStream(socket.getOutputStream());

                  // Continuously serve the client
                  while (true) {
                        // Receive radius from the client
```

```
                        //radius = inputFromClient.readDouble();
                        //type = inputFromClient.read();
                        name = inputFromClient.readUTF();
                        bt = new byte[inputFromClient.readInt()];
                        inputFromClient.readFully(bt, 0, bt.length);

                        int flag = 0;
                        synchronized (MultiThreadServerGUI.pics) {

                        if(MultiThreadServerGUI.pics.size() == 0) {
                              MultiThreadServerGUI.pics.add(new
ScreenShotIO(bt,Thread.currentThread().getName(),name));

      addPic(MultiThreadServerGUI.pics.get(0).getLbl());
                        } else {
                              for(int i = 0; i <
MultiThreadServerGUI.pics.size();i++) {

      if(MultiThreadServerGUI.pics.get(i).getThreadName().equals(Thre
ad.currentThread().getName())) {
                                          for(int j = 0; j <
p.getComponentCount(); j++) {

      if((JLabel)p.getComponent(j) ==
MultiThreadServerGUI.pics.get(i).getLbl()){
                                                      p.remove(j);
                                                      p.revalidate();
                                                      break;
                                                }
                                          }

      MultiThreadServerGUI.pics.remove(i);

      MultiThreadServerGUI.pics.add(i, new
ScreenShotIO(bt,Thread.currentThread().getName(),name));
                                          flag = 1;

      addPic(MultiThreadServerGUI.pics.get(i).getLbl());
                                          break;
                                    }
                              }
                              if(flag == 0) {
                                    MultiThreadServerGUI.pics.add(new
ScreenShotIO(bt,Thread.currentThread().getName(),name));

      addPic(MultiThreadServerGUI.pics.get(
MultiThreadServerGUI.pics.size() - 1).getLbl());
                              }
                        }
                        }

                  }
            } catch (IOException e) {
                  System.err.println(e);
            }
      }

      public void addPic(JLabel lbl) {
            System.out.println(Thread.currentThread().getName());
            p.add(lbl);
            p.revalidate();
```

```
            System.out.println(p.getComponentCount());
      }

}
```

מחלקת MultiThreadServerGUI

```
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;

import javax.swing.*;

public class MultiThreadServerGUI extends JFrame implements
ActionListener {

      private static final long serialVersionUID = 1L;
      private JTextArea jta = new JTextArea();
      private Server server = new Server();

      private JPanel mainPanel, firstPanel, seconedPanel;
      private JSplitPane splitPane;
      private JLabel showScreen, screenName;
      private JTextField message;
      private JButton send;

      public static ArrayList<ScreenShotIO> pics = new
ArrayList<ScreenShotIO>();

      public static void main(String[] args) {
            new MultiThreadServerGUI();
            //users = new ArrayList<User>();
      }

      public MultiThreadServerGUI() {
            // Place text area on the frame
            //setLayout(new BorderLayout());

            jta.setEditable(false);
            mainPanel = new JPanel(new BorderLayout());
            firstPanel = new JPanel(new BorderLayout());
            seconedPanel = new JPanel(new GridLayout(0,4));

            showScreen = new JLabel();
            // showScreen.setSize(300,250);
            //showScreen.setHorizontalAlignment(JLabel.CENTER);
            //screenName = new JLabel("Taken On:");
            message = new JTextField();
            send = new JButton("Send");
            send.addActionListener(this);

            firstPanel.add(jta,BorderLayout.CENTER);
            firstPanel.add(message, BorderLayout.NORTH);
            firstPanel.add(send, BorderLayout.SOUTH);
            firstPanel.setOpaque(false);

            //seconedPanel.add(takeScreen, BorderLayout.SOUTH);
            seconedPanel.setOpaque(false);
```

```
            //add(new JScrollPane(seconedPanel),
BorderLayout.CENTER);

            splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
            splitPane.setResizeWeight(0.5);
            splitPane.setEnabled(false);
            splitPane.setDividerSize(1);
            splitPane.add(new JScrollPane(firstPanel));
            splitPane.add(new JScrollPane(seconedPanel));

            mainPanel.add(splitPane);
            setContentPane(mainPanel);

            setTitle("MultiThreadServer");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            pack();
            setLocation(300, 300);
            setSize(600, 600);
            setVisible(true);

            jta.append("MultiThreadServer started at " + new Date() +
'\n');

            int clientNo = 1;

            while (true) {
                  // Listen for a new connection request
                  Socket socket = server.Accept();

                  // Display the client number
                  jta.append("Starting thread for client " + clientNo
+ " at " + new Date() + '\n');

                  // Find the client's host name, and IP address
                  InetAddress inetAddress = socket.getInetAddress();
                  jta.append("Client " + clientNo + "'s host name is
" + inetAddress.getHostName() + "\n");
                  jta.append("Client " + clientNo + "'s IP Address is
" + inetAddress.getHostAddress() + "\n");


                  // Create a new task for the connection
                  Thread task = new Thread(new
HandleAClient(socket,seconedPanel),clientNo + "");
                  task.start();
                  clientNo++;
            }
      }

      @Override
      public void actionPerformed(ActionEvent arg0) {
            // TODO Auto-generated method stub
            if(arg0.getSource() == send &&
!message.getText().equals("")) {
                  try {

      HandleAClient.outputToClient.writeUTF(message.getText());
                        jta.append("Message Sent: " +
message.getText() +"\n");
                        message.setText("");
                  } catch (IOException e1) {
```

```
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                    }
        }
    }
}
```

מחלקת Screen

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import com.github.sarxos.webcam.Webcam;

public class Screen extends JFrame implements ActionListener {

        private static final long serialVersionUID = 1L;
        private JPanel mainPanel, firstPanel, seconedPanel;
        private JSplitPane splitPane;
        private JButton takeScreen,takePicture;
        private JTextField name;
        private JTextArea mes;
        private ArrayList<ScreenShotIO> imageList;
        private JLabel showScreen, screenName;
        private Client client = new Client();

        public Screen() {

                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                imageList = new ArrayList<ScreenShotIO>();

                mainPanel = new JPanel(new BorderLayout());
                firstPanel = new JPanel(new BorderLayout());
                seconedPanel = new JPanel(new GridLayout(1,3));

                showScreen = new JLabel();
                // showScreen.setSize(300,250);
                showScreen.setHorizontalAlignment(JLabel.CENTER);
                screenName = new JLabel("Taken On:");
```

```
            takeScreen = new JButton("Take ScreenShot");
            takeScreen.addActionListener(this);
            takePicture = new JButton("Take Picture");
            takePicture.addActionListener(this);

            name = new JTextField("Enter your name");
            name.setForeground(Color.GRAY);

            mes = new JTextArea();
            mes.setEditable(false);

            firstPanel.add(new JScrollPane(mes),BorderLayout.NORTH);
            firstPanel.add(showScreen, BorderLayout.CENTER);
            firstPanel.add(screenName, BorderLayout.SOUTH);
            firstPanel.setOpaque(false);

            seconedPanel.add(name);
            seconedPanel.add(takeScreen);
            seconedPanel.add(takePicture);
            seconedPanel.setOpaque(false);

            splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
            splitPane.setResizeWeight(0.5);
            splitPane.setEnabled(false);
            splitPane.setDividerSize(1);
            splitPane.add(firstPanel);
            splitPane.add(seconedPanel);

            mainPanel.add(splitPane);
            setContentPane(mainPanel);

            pack();
            setLocation(300, 300);
            setSize(600, 600);
            setVisible(true);

            while(true) {
                  String thing = client.readFromServer();
                  if(!thing.equals("")) {
                        mes.append("Sent by Teacher: " + thing +
"\n");
                  }
            }

      }

      @Override
      public void actionPerformed(ActionEvent arg0) {

            if (arg0.getSource() == takeScreen &&
!name.getText().toString().equals("Enter your name")) {

//                try {
//                      //Thread.sleep(10000);
//                } catch (InterruptedException e1) {
//                      e1.printStackTrace();
//                }

                  ScreenShotIO sc = null;
                  try {
                        sc = new ScreenShotIO();
```

```
                    } catch (Exception e) {
                          e.printStackTrace();
                    }
                    imageList.add(sc);

                    try {
                          resizeImage(showScreen, 300, 250,
sc.getImageInByte());
                          showScreen.revalidate();
                    } catch (IOException e) {
                          e.printStackTrace();
                    }

                    screenName.setText("Taken On: " + sc.getName());
                    screenName.revalidate();
                    this.revalidate();

                    System.out.println(showScreen.getIcon());
                    System.out.println(sc.getImageInByte().length);

                    name.setEnabled(false);
                    name.setDisabledTextColor(Color.black);
                    int check =
client.writeToServer(sc.getImageInByte(),name.getText());
                    System.out.println("Check = " + check);
                    if(check == 0) {
                          JOptionPane.showMessageDialog(null, "Server
Closed", "Error", 0);
                    }
              }

              else if(arg0.getSource() == takePicture &&
!name.getText().toString().equals("Enter your name")) {
                    Webcam webcam = Webcam.getDefault();
                    byte t[] = null;
                    if (webcam != null) {
                          System.out.println("Webcam: " +
webcam.getName());
                          //String path = "ReadyImages/" +
name.getText().toString() + ".png";
                          try {
                                //ImageIO.write(webcam.getImage(),
"PNG", new File(path));
                                ByteArrayOutputStream baos = new
ByteArrayOutputStream();
                                ImageIO.write(webcam.getImage(), "JPG",
baos);
                                baos.flush();
                                t = baos.toByteArray();
                                baos.close();
                          } catch (IOException e1) {
                                e1.printStackTrace();
                          }

                          try {
                                resizeImage(showScreen, 300, 250, t);
                                showScreen.revalidate();
                          } catch (IOException e) {
                                e.printStackTrace();
                          }
```

```
                        screenName.setText("Taken On: " +
Calendar.getInstance().toString());
                        screenName.revalidate();
                        this.revalidate();

                        int check =
client.writeToServer(t,name.getText());
                        name.setEnabled(false);
                        name.setForeground(Color.black);
                        System.out.println("Check = " + check);
                        if(check == 0) {
                              JOptionPane.showMessageDialog(null,
"Server Closed", "Error", 0);
                        }
                  } else {
                        System.out.println("No webcam detected");
                        JOptionPane.showMessageDialog(null, "No
Camera Found", "Error", 0);
                  }
            }

            else {
                  JOptionPane.showMessageDialog(null, "Enter Your
Name", "Error", 0);
            }

      }

      public void resizeImage(JLabel label, int h, int w, byte[] b)
throws IOException {
            System.out.println("Resizing Image");

            ImageIcon imageIcon = new ImageIcon(b);
            Image originalImage = imageIcon.getImage();
            originalImage = originalImage.getScaledInstance(h, w,
java.awt.Image.SCALE_SMOOTH);
            ImageIcon newimageIcon = new ImageIcon(originalImage);

            label.setBounds(new Rectangle(100, 100));
            label.setPreferredSize(new java.awt.Dimension(h, w));
            label.setIcon(newimageIcon);

      }

      public static void main(String[] args) {

            @SuppressWarnings("unused")
            Screen s = new Screen();
      }

}
```

מחלקת ScreenShotIO

```
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.Robot;
import java.awt.Toolkit;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.Serializable;
//import java.io.File;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;

public class ScreenShotIO implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	BufferedImage screenShot;
	String name;
	ByteArrayOutputStream baos;
	byte[] imageInByte;
	String threadName;
	JLabel lbl;
	String studentName;
	int type;

	public ScreenShotIO() throws Exception {
		Calendar now = Calendar.getInstance();
		Robot robot = new Robot();
		SimpleDateFormat formatter = new
SimpleDateFormat("yyyyMMdd hh mm ss a");
		screenShot = robot.createScreenCapture(new
Rectangle(Toolkit.getDefaultToolkit().getScreenSize()));
		baos = new ByteArrayOutputStream();
		ImageIO.write(screenShot, "JPG", baos);
		baos.flush();
		imageInByte = baos.toByteArray();
		baos.close();
		name = formatter.format(now.getTime());
		System.out.println(name);
		type = 0;
	}

	public ScreenShotIO(byte [] b,String tName,String nameX) {

		Calendar now = Calendar.getInstance();
		SimpleDateFormat formatter = new
SimpleDateFormat("yyyyMMdd hh mm ss a");
		imageInByte = b;
		name = formatter.format(now.getTime());
		threadName = tName;

		JLabel x = new JLabel();
		ImageIcon imageIcon = new ImageIcon(b);
		Image originalImage = imageIcon.getImage();
```

```
            originalImage = originalImage.getScaledInstance(100, 100,
java.awt.Image.SCALE_SMOOTH);
            ImageIcon newimageIcon = new ImageIcon(originalImage);

            x.setBounds(new Rectangle(100, 100));
            x.setPreferredSize(new java.awt.Dimension(100, 100));
            x.setIcon(newimageIcon);

            studentName = nameX;

            lbl = x;
            lbl.setToolTipText("Taken on: " + name + ", Student Name:
" + studentName + " Client:" + threadName);
      }

      public String getName() {
            return name;
      }

      public void setName(String name) {
            this.name = name;
      }

      public BufferedImage getScreenShot() {
            return screenShot;
      }

      public void setScreenShot(BufferedImage screenShot) {
            this.screenShot = screenShot;
      }

      public ByteArrayOutputStream getBaos() {
            return baos;
      }

      public void setBaos(ByteArrayOutputStream baos) {
            this.baos = baos;
      }

      public byte[] getImageInByte() {
            return imageInByte;
      }

      public void setImageInByte(byte[] imageInByte) {
            this.imageInByte = imageInByte;
      }

      public String getThreadName() {
            return threadName;
      }

      public void setThreadName(String threadName) {
            this.threadName = threadName;
      }

      public JLabel getLbl() {
            return lbl;
      }

      public void setLbl(JLabel lbl) {
            this.lbl = lbl;
```

```
        }

        public String getStudentName() {
                return studentName;
        }

        public void setStudentName(String studentName) {
                this.studentName = studentName;
        }

        public int getType() {
                return type;
        }

        public void setType(int type) {
                this.type = type;
        }

}
```

מחלקת Server

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

class Server {

        private ServerSocket serverSocket;

        public Server() {
                try {
                        serverSocket = new ServerSocket(8000);
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

        public Socket Accept() {
                try {
                        return serverSocket.accept();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                return null;
        }

}
```