



# **פרויקט גמר**

## **למילוי חלקי של דרישות 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 הפרויקט: מערכת ליצירת סקר**  
**שם הסטודנט: זיו בן סימון**

**העבודה בוצעה בהנחיית : אודי מלכה , מוטי פניקשיוולי**

**שנה"ל תש"ף 2020**



מכללת אורט קרית ביאליק לטכנולוגיה מתקדמת ולמדעים

## הצהרת סטודנט

אני הסטודנט זיו בן סימון מס' ת.ז. [REDACTED] החתום מטה, מצהיר בזאת שכל עבודת הגמר/ הפרויקט 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: \_\_\_\_\_

אישור המנחה:

הנני מאשר הגשת החוברת להערכה \_\_\_\_\_

## תוכן עניינים

1.....	פתיח
3.....	תוכן עניינים
4.....	מבוא
4.....	עמדת פיתוח
4.....	דרישות מערכת מהמשתמש
5.....	מדריך למשתמש
8.....	תרשים UML
9.....	פירוט מחלקות
15.....	קוד מקור

## **מבוא**

במעבדה זו מימשתי תוכנה פשוטה ליצירת סקר\שאלון בכך שלמנהל הסקר יש אפשרות לבחור שאלה, ואפשרויות תשובה לשאלה בכל נושא שירצה ושלוח אותה דרך השרת לכל מי שמשתתף בסקר, התוכנה אוספת את התוצאות של הסקר ונותנת למנהל הסקר את התוצאה שנבחרה הכי הרבה.  
מטרת הפרויקט הייתה מימוש של מערכת שרת – לקוח ותרגול סנכרון תהליכונים.

## **עמדת פיתוח**

Windows 10

Eclipse Java IDE

## **דרישות מערכת מהמשתמש**

Windows 10

Java

חיבור לאינטרנט

# מדריך למשתמש

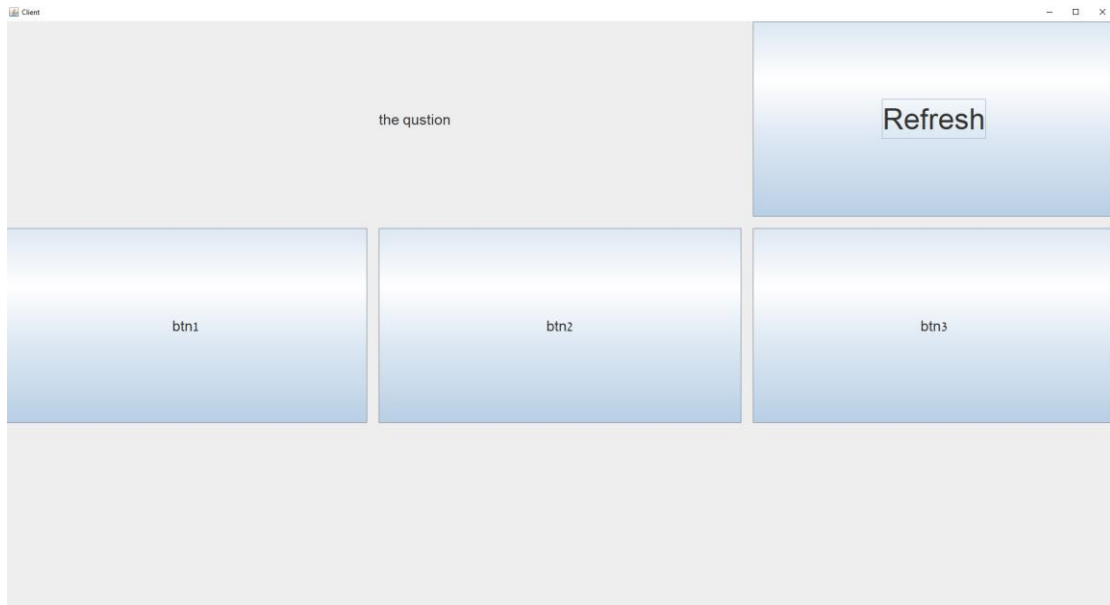
יש להפעיל את השרת(שהוא גם החלון של מנהל הסקר) MultiThreadServerGUI

The screenshot shows a window titled "Host" with a light gray background. At the top left, there is a label "Enter The Question-->" followed by a large empty text input box. Below this, there are three labels: "First Option:", "Second Option:", and "Third Option:". Each label is followed by an empty text input box. At the bottom of the window, there are two blue buttons: "Summary" and "Accept". On the right side, there is a small text box containing the message "MultiThreadServer started at Fri May 22 11:46:20 EDT 2020".

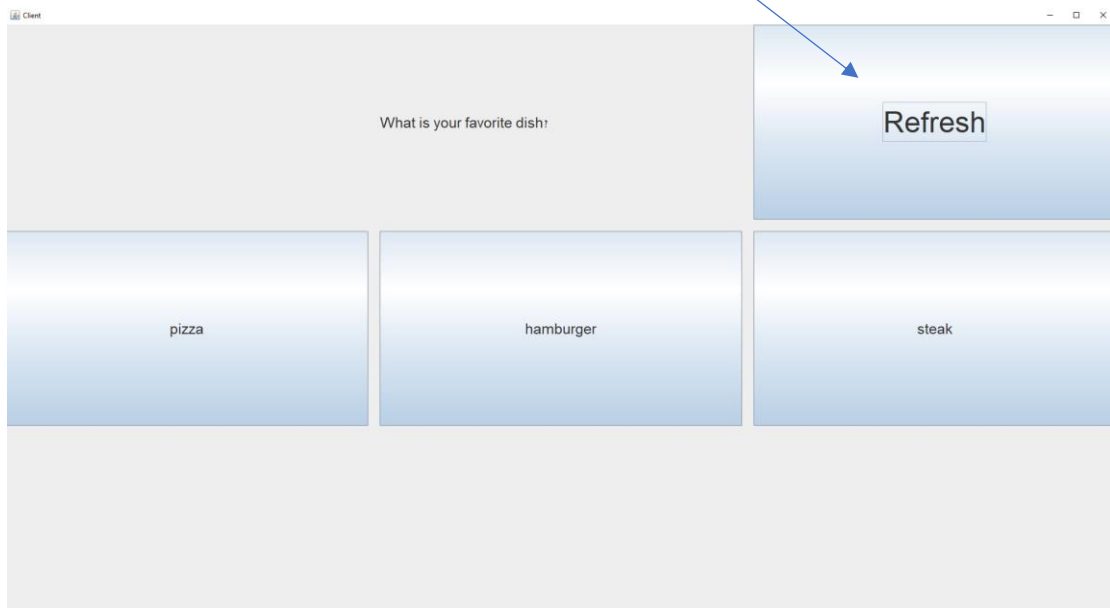
לאחר מכן יש למלא את החלון של המנהל בפרטים של הסקר (השאלה ואפשרויות תשובה לשאלה וללחוץ Accept

The screenshot shows the same window as before, but now it is filled with example data. The "Enter The Question-->" input box contains the text "What is your favorite dish:". The "First Option:" input box contains "בננה", the "Second Option:" input box contains "hamburger", and the "Third Option:" input box contains "steak". A blue arrow points from the "Accept" button to the "Summary" button. The "Accept" button is highlighted with a blue border. The "Summary" button is also highlighted with a blue border. The "MultiThreadServer started at Fri May 22 11:46:20 EDT 2020" message is still visible in the bottom right corner.

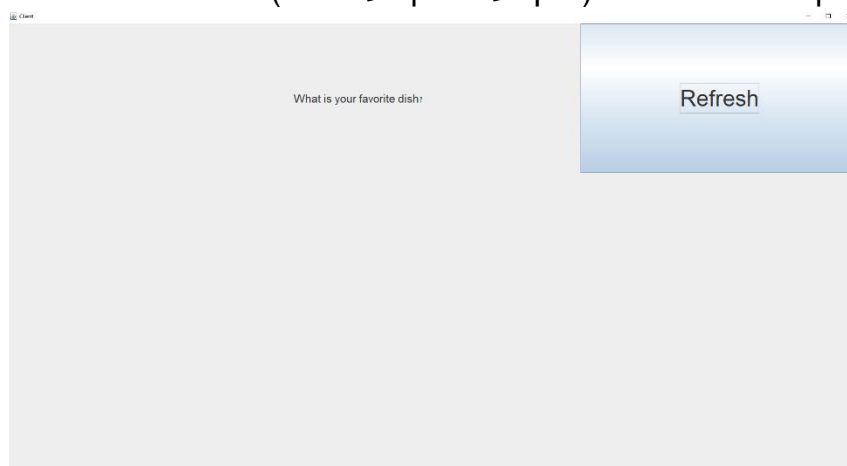
אחרי שמנהל הסקר מילא את השאלה והאפשרויות ולחץ Accept כל המשתתפים בסקר יכולים להתחבר בכך שמפעילים StudentGUI



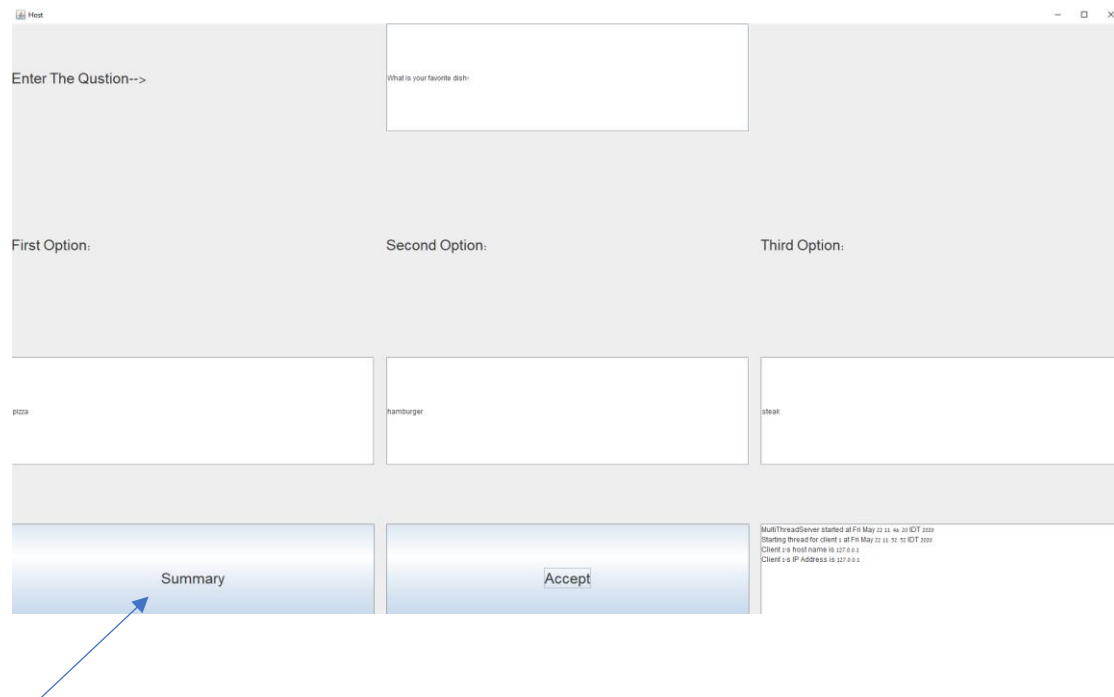
שאר המשתתפים בסקר ילחצו על כפתור Refresh הדף יתענן ויופיע השאלה והאפשרויות של מנהל הסקר



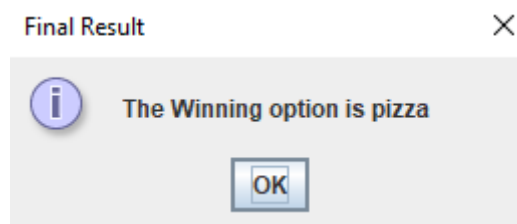
כעת המשתתפים בסקר יכולים לבחור את התשובה לשאלה ותשובתם תישלח בחזרה למנהל הסקר בצורה אונימית (ניתן לענות רק פעם אחת)



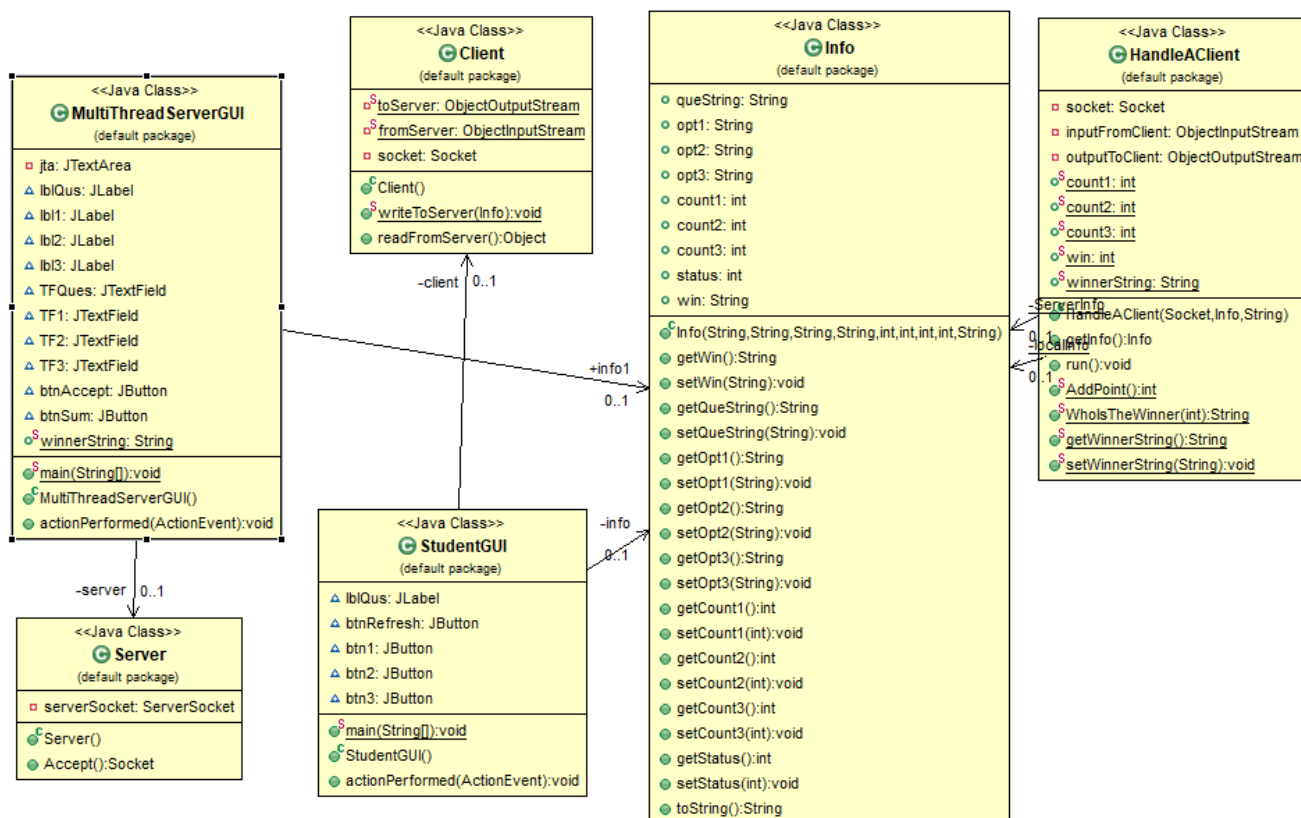
כאשר כל המשתתפים בסקר סיימו לענות על השאלה מנהל הסקר יכול לראות את האפשרות שנבחרה הכי הרבה פעמיים בכך שילחץ על הכפתור Summary ויפיע לו חלון קטן שבו רשום מה התשובה שנבחרה הכי הרבה



התשובה שנבחרה הכי הרבה



# תרשים UML





# פירוט מחלקות

## Client מחלקת

מחלקה המייצגת את פעולות הלקוח

פירוט	שדה
הנתונים ששולחים לשרת	<code>private static ObjectOutputStream toServer;</code>
הנתונים שמתקבלים מהשרת	<code>private static ObjectInputStream fromServer;</code>
תבנית של כתובת אינטרנט	<code>private Socket socket;</code>

פירוט	שיטה
הפעולה מאתחלת את הערוץ הלוגי של התקשורת עם שרת	<code>public Client()</code>
הפעולה שולחת נתונים לשרת	<code>public static void writeToServer(Info info1)</code>
הפעולה מקבלת נתונים מהשרת	<code>public Object readFromServer()</code>

## מחלקת StudentGUI

מחלקה המייצגת את המתאר הגרפי של הלקוח

פירוט	שדה
אובייקט מסוג Client	<code>private Client client = new Client();</code>
כפתור שמרענן את הדף	<code>JButton btnRefresh = new JButton("Refresh");</code>
כפתור מעדכן את ערך 1	<code>JButton btn1 = new JButton("btn1");</code>
כפתור מעדכן את ערך 2	<code>JButton btn2 = new JButton("btn2");</code>
כפתור מעדכן את ערך 3	<code>JButton btn3 = new JButton("btn3");</code>
אובייקט מסוג info	<code>private Info info = new Info(null, "", "", "", 0, 0, 0, 0, "");</code>

פירוט	שיטה
הפעולה מאתחלת את החלון	<code>public StudentGUI()</code>
הפעולה מטפלת באירוע של לחיצה על כפתור	<code>public void actionPerformed(ActionEvent e)</code>

## מחלקת HandleAClient

מחלקה המקשרת בין השרת ללקוח

פירוט	שדה
תבנית של כתובת אינטרנט	private Socket socket;
הנתונים שמתקבלים מהשרת	private ObjectInputStream inputFromClient;
הנתונים ששולחים לשרת	private ObjectOutputStream outputToClient;
אובייקט מקבל מידע מהשרת מסוג info	private static Info localInfo;
אובייקט שולח מידע לשרת מסוג info	private static Info ServerInfo;
ערך סופר מספר 1	public static int count1 = 0;
ערך סופר מספר 2	public static int count2 = 0;
ערך סופר מספר 3	public static int count3 = 0;
ערך סופר מנצח	public static int win = 0;
השם של המנצח	public static String winnerString = "";

פירוט	שיטה
הפעולה מאתחלת את החלון ואת התקשורת של השרת	public HandleAClient(Socket socket, Info info, String winnerString1)
הפעולה מחזירה את המידע	public Info getInfo()
הפעולה שולחת נתונים למשתמש מהשרת ושולחת נתונים לשרת מהמשתמש	public void run()
פעולה שמוסיפה נקודות לערך שנבחר(בצורה מסוכרנת)	public static synchronized int AddPoint()
פעולה שבודקת אם יש תיקו בין ערכים	public static String WholsTheWinner(int win)
פעולה המחזירה את השם של המנצח	public static String getWinnerString()
פעולה המגדירה את השם של המנצח	public static void setWinnerString(String winnerString)

## מחלקת Server

מחלקה המייצגת את השרת

פירוט	שדה
כתובת השרת	<code>private ServerSocket serverSocket;</code>

פירוט	שיטה
הפעולה מאתחלת את כתובת השרת	<code>public Server(){</code>
הפעולה מאשרת את החיבור ללקוח	<code>public Socket Accept()</code>

## מחלקת Info

מחלקה המייצגת את המידע

פירוט	שדה
המחרוזת של השאלה	public String queString;
המחרוזת של אפשרות 1	public String opt1;
המחרוזת של אפשרות 2	public String opt2;
המחרוזת של אפשרות 3	public String opt3;
ערך סופר של הסטטוס	public int status;
ערך סופר מספר 1	public int count1 = 0;
ערך סופר מספר 2	public int count2 = 0;
ערך סופר מספר 3	public int count3 = 0;
השם של המנצח	public String win;

פירוט	שיטה
הפעולה מאתחלת את פרטי info	public Info(String queString,String opt1,String opt2,String opt3, int count1, int count2, int count3,int status,String win)
הפעולה המחזירה את הערך המבוקש	public GET...()
פעולה המגדירה את הערך המבוקש	Public SET...()

## מחלקת MultiThreadServerGUI

מחלקה המייצגת את חלון השרת ואת המנהל של הסקר

פירוט	שדה
חלק המציג טקסט	<code>private JTextArea jta = new JTextArea();</code>
אובייקט מסוג <code>Server</code>	<code>private Server server = new Server();</code>
קופסת טקסט למילוי השאלה	<code>JTextField TFQues = new JTextField();</code>
קופסת טקסט למילוי האפשרות הראשונה	<code>JTextField TF1 = new JTextField();</code>
קופסת טקסט למילוי האפשרות השנייה	<code>JTextField TF2 = new JTextField();</code>
קופסת טקסט למילוי האפשרות השלישית	<code>JTextField TF3 = new JTextField();</code>
כפתור המאשר שליחה של הנושא והאפשרויות	<code>JButton btnAccept = new JButton("Accept");</code>
כפתור המציג את התוצאה של הסקר	<code>JButton btnSum = new JButton("Summary");</code>
מחרוזת שבה יהיה השם של המנצח	<code>public static String winnerString="NOTHING";</code>
אתחול אובייקט מסוג <code>info</code>	<code>public Info info1 = new Info(TFQues.getText(), TF1.getText(), TF2.getText(), TF3.getText(), 0, 0, 0, 0,"default");</code>

פירוט	שיטה
פעולה המריצה את התוכנית	<code>public static void main(String[] args)</code>
הפעולה מאתחלת את חלון השרת, מפעילה את השרת ומשמשת את מנהל הסקר	<code>public MultiThreadServerGUI()</code>
הפעולה מטפלת באירוע של לחיצה על כפתור	<code>public void actionPerformed(ActionEvent e)</code>

## קוד מקור

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

class Client {

    // IO streams
    private static ObjectOutputStream toServer;
    private static ObjectInputStream 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 ObjectOutputStream(socket.getOutputStream());

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

```

    catch (IOException ex) { }
}

public static void writeToServer(Info info1){
    try {
        toServer.writeObject(info1);
        toServer.flush();
        toServer.reset();
    } catch (IOException e) {e.printStackTrace(); }
}

public Object readFromServer(){
    try {
        return fromServer.readObject();
    } catch (IOException e) {e.printStackTrace(); } catch
(ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return 0;
}

}

```



```

import java.awt.print.Printable;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.Vector;

class HandleAClient implements Runnable {
    private Socket socket; // A connected socket

    private ObjectInputStream inputFromClient;
    private ObjectOutputStream outputToClient;

    private static Info localInfo;
    private static Info ServerInfo;
    public static int count1 = 0;
    public static int count2 = 0;
    public static int count3 = 0;
    public static int win = 0;
    public static String winnerString = "";

    public HandleAClient(Socket socket, Info info, String
winnerString1) {
        this.socket = socket;
        this.ServerInfo = info;
        winnerString = winnerString1;
    }
}

```

```

public Info getInfo() {
    return localInfo;
}

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

// Continuously serve the client
        while (true) {
            // Receive info from the client
            localInfo = (Info)
inputFromClient.readObject();

            // load teachers text - send it to client (if
status = 0)

            if (localInfo.getStatus() == 0)
                outputToClient.writeObject(ServerInfo);

            // get result from server(if status = 1)
            else if (localInfo.getStatus() == 1) {

                win = AddPoint(); // function that add
the choose option to the summary and put the highest vote

                // count
into win

                ServerInfo.setWin(WholsTheWinner(win)); // check if it is a TIE

```

```

        localInfo.setWin(ServerInfo.getWin()); //
set the winning option

        System.out.println("count1: " + count1);
        System.out.println("count2: " + count2);
        System.out.println("count3: " + count3);
        System.out.println("win: " +
ServerInfo.getWin());

```

```

    }
}
}
catch (
IOException e) {
    System.err.println(e);
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

```

```

        public static synchronized int AddPoint() { // adding the option
chosen to the summery
            if (localInfo.getCount1() == 1) {
                count1++;
                ServerInfo.setCount1(count1);
            }
            if (localInfo.getCount2() == 1) {
                count2++;
                ServerInfo.setCount1(count2);
            }
        }
    }
}

```

```

    }
    if (localInfo.getCount3() == 1) {
        count3++;
        ServerInfo.setCount1(count3);
    }

    return Math.max(count1, Math.max(count2, count3));
//return who has the most points

}

public static String WholsTheWinner(int win) { //testing if there
is a tie

    if (count1 == win && count2 == win && count3 == win) {
        return "a tie between All The Options ";
    } else if (count1 == win && count2 == win) {
        return "a tie between " + localInfo.getOpt1() + " and
" + localInfo.getOpt2();
    } else if (count1 == win && count3 == win) {
        return "a tie between " + localInfo.getOpt1() + " and
" + localInfo.getOpt3();
    } else if (count3 == win && count2 == win) {
        return "a tie between " + localInfo.getOpt2() + " and
" + localInfo.getOpt3();
    }
    else if (count1 == win) {
        return localInfo.getOpt1();

    } else if (count2 == win) {
        return localInfo.getOpt2();
    }
}

```

```

        } else if (count3 == win) {
            return localInfo.getOpt3();

        }
        return "Error";

    }

    public static String getWinnerString() {
        return winnerString;
    }

    public static void setWinnerString(String winnerString) {
        HandleAClient.winnerString = winnerString;
    }

}

```

```

import java.io.Serializable;

public class Info implements Serializable{

    public String queString;
    public String opt1;
    public String opt2;
    public String opt3;
    public int count1;
    public int count2;
    public int count3;
    public int status;
    public String win;

    public Info(String queString, String opt1, String opt2,
String opt3, int count1, int count2, int count3,int
status, String win) {

        this.queString = queString;
        this.opt1 = opt1;
        this.opt2 = opt2;
        this.opt3 = opt3;
        this.count1 = count1;
        this.count2 = count2;
        this.count3 = count3;
        this.status=status;
        this.win = win;

    }

    public String getWin() {
        return win;
    }

    public void setWin(String win) {
        this.win = win;
    }

    public String getQueString() {
        return queString;
    }
}

```

```

public void setQueString(String queString) {
    this.queString = queString;
}

public String getOpt1() {
    return opt1;
}

public void setOpt1(String opt1) {
    this.opt1 = opt1;
}

public String getOpt2() {
    return opt2;
}

public void setOpt2(String opt2) {
    this.opt2 = opt2;
}

public String getOpt3() {
    return opt3;
}

public void setOpt3(String opt3) {
    this.opt3 = opt3;
}

public int getCount1() {
    return count1;
}

public void setCount1(int count1) {
    this.count1 = count1;
}

public int getCount2() {
    return count2;
}

public void setCount2(int count2) {
    this.count2 = count2;
}

```

```

    }

    public int getCount3() {
        return count3;
    }

    public void setCount3(int count3) {
        this.count3 = count3;
    }
    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }

    @Override
    public String toString() {
        return "Info [queString=" + queString + ",
opt1=" + opt1 + ", opt2=" + opt2 + ", opt3=" + opt3 +
", count1="
                + count1 + ", count2=" + count2 +
", count3=" + count3 + ", status=" + status + "];"
    }

}

```



```

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

import javax.swing.*;

public class MultiThreadServerGUI extends JFrame
implements ActionListener {

    private JTextArea jta = new JTextArea();
    private Server server = new Server();
    JLabel lblQus = new JLabel("Enter The Qustion--
>");
    JLabel lbl1 = new JLabel("First Option:");
    JLabel lbl2 = new JLabel("Second Option:");
    JLabel lbl3 = new JLabel("Third Option:");

    JTextField TFQues = new JTextField();
    JTextField TF1 = new JTextField();
    JTextField TF2 = new JTextField();
    JTextField TF3 = new JTextField();

    JButton btnAccept = new JButton("Accept");
    JButton btnSum = new JButton("Summary");

    public static String winnerString="NOTHING";

    public Info info1 = new Info(TFQues.getText(),
TF1.getText(), TF2.getText(), TF3.getText(), 0, 0, 0,
0,"default");//reset info

    public static void main(String[] args) {
        new MultiThreadServerGUI();
    }

    public MultiThreadServerGUI() //teacher app and
server
    {

```

```

        // Place text area on the frame
        JPanel main = new JPanel();
        main.setLayout(new GridLayout(0, 3, 20,
100));
        main.add(lblQus, BorderLayout.PAGE_START);
        lblQus.setFont(new Font("TimesRoman",
Font.PLAIN, 25));

        main.add(TFQues, BorderLayout.CENTER);

        main.add(new JLabel(""));
        main.add(lbl1);
        main.add(lbl2);
        main.add(lbl3);
        lbl1.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
        lbl2.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
        lbl3.setFont(new Font("TimesRoman",
Font.PLAIN, 25));

        main.add(TF1, BorderLayout.WEST);
        main.add(TF2, BorderLayout.CENTER);
        main.add(TF3, BorderLayout.EAST);

        main.add(btnSum);
        main.add(btnAccept, BorderLayout.EAST);
        main.add(new JScrollPane(jta),
BorderLayout.CENTER);
        btnSum.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
        btnSum.addActionListener(this);
        btnAccept.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
        btnAccept.addActionListener(this);
        add(main);
        setTitle("Host");
        setSize(1920, 1080);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);

```

```

        jta.append("MultiThreadServer started at " +
new Date() + '\n');
        int clientNo = 1;
        while (true) { //server starting
            // 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,info1,winnerString));
            task.start();
            clientNo++;
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnAccept) { // enter
the info in to the object and sent it to the server

            info1.setQueString(TFQues.getText());
            info1.setOpt1(TF1.getText());
            info1.setOpt2(TF2.getText());
            info1.setOpt3(TF3.getText());
        }
    }

```

```
        if (e.getSource() == btnSum)
        {
            JOptionPane.showMessageDialog(null,
"The Winning option is " + info1.getWin(), "Final
Result ", JOptionPane.INFORMATION_MESSAGE);

        }
    }
}
```

```

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;
    }
}

import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class StudentGUI extends JFrame implements
ActionListener {

    private Client client = new Client();
    JLabel lblQus = new JLabel("the qustion");
    JButton btnRefresh = new JButton("Refresh");
    JButton btn1 = new JButton("btn1");
    JButton btn2 = new JButton("btn2");

```

```

        JButton btn3 = new JButton("btn3");

        private Info info = new Info(null, "", "", "", 0,
0, 0, 0, "");

        public static void main(String[] args) {
            new StudentGUI();
        }

        public StudentGUI() {
            JPanel main = new JPanel();
            main.setLayout(new GridLayout(0, 3, 20,
20));
            main.add(new JLabel(""));
            main.add(lblQus, BorderLayout.PAGE_START);
            // add text for the question
            lblQus.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
            main.add(btnRefresh, BorderLayout.EAST); //
add refresh button
            main.add(btn1, BorderLayout.WEST); // add
button1
            main.add(btn2, BorderLayout.CENTER); // add
button2
            main.add(btn3, BorderLayout.EAST); // add
button3
            btnRefresh.setFont(new Font("TimesRoman",
Font.PLAIN, 50));
            btn1.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
            btn2.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
            btn3.setFont(new Font("TimesRoman",
Font.PLAIN, 25));
            btn1.addActionListener(this);
            btn2.addActionListener(this);
            btn3.addActionListener(this);
            btnRefresh.addActionListener(this);

            main.add(new JLabel(""));

            add(main);

```

```

        setTitle("Client");
        setSize(1920, 1080);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnRefresh) // Refresh
the page to see the question and the answers
        {
            info.setStatus(0);
            client.writeToServer(info);
            lblQus.setText("");
            info = (Info) client.readFromServer();
            lblQus.setText(info.getQueString());
            btn1.setText(info.getOpt1());
            btn2.setText(info.getOpt2());
            btn3.setText(info.getOpt3());

        }
        else {
            info.setStatus(1);
            if (e.getSource() == btn1) //if option
1 chosen

            {
                info.setOpt1(btn1.getText());
                info.setOpt2(btn2.getText());
                info.setOpt3(btn3.getText());
                info.setCount1(1);
                info.setCount2(0);
                info.setCount3(0);
                btn1.setVisible(false);
                btn2.setVisible(false);
                btn3.setVisible(false);
                client.writeToServer(info);
            }
        }
    }

```

```

        else if (e.getSource() == btn2) //if
option 2 chosen
        {
            info.setOpt1(btn1.getText());
            info.setOpt2(btn2.getText());
            info.setOpt3(btn3.getText());
            info.setCount1(0);
            info.setCount2(1);
            info.setCount3(0);
            btn1.setVisible(false);
            btn2.setVisible(false);
            btn3.setVisible(false);
            client.writeToServer(info);
        }
        else if (e.getSource() == btn3) //if
option 3 chosen
        {
            info.setOpt1(btn1.getText());
            info.setOpt2(btn2.getText());
            info.setOpt3(btn3.getText());
            info.setCount1(0);
            info.setCount2(0);
            info.setCount3(1);
            btn1.setVisible(false);
            btn2.setVisible(false);
            btn3.setVisible(false);
            client.writeToServer(info);
        }
    }
}
}

```