פרויקט גמר

מקבילי מבוזר | משרד החינוך

מדעי המחשב

2021

פרויקט תיכנות מקבילי מבוזר ומימוש שרת-לקוח

התלמידה המבצעות: רות זילבר 212332514 .

העבודה בוצעה בהנחיית: גב' חנה סירוקה.

תוכן העניינים

מבוא

קילו-פאונד-מארק-אגורה-מיליגרם-קראט-דג-קילו טון-אבן-אונקיה-מאסה טון.

כולם מיצגים משקל. אז מה ביניהם למעשה?

לכל מדינה אמות משקל משלה בנוסף למשקלים הקבועים. יש חומרים הנמדדים בקראט כמו זכיכות זהב, לעומתם בטונים – רכבים לדוגמא, ואחרים הנמדדים בגרמים: כסף... מינון התרופה נמדד במיליגרם, ואחרת בציטגרם. מגוון היצוג של המשקלים רחב מאד, ופיתחתי פרויקט בשפת ג'אווה הממיר את הערכים באופן מדויק ונעים לעין.

לאחר הרצת השרת ברשת המחשבים של המכללה בה אני לומדת, כמה לקוחות מכמה מחשבים ניסו להמיר משקולות וקיבלו ערכים מעוותים. התברר שבמהלך שליחת הנתונים לשרת, ערכי ההמרה התחלפו בין הלקוחות. לשם כך, טיפלתי בנושא באופן של חישוב מקבילי מבוזר.

במקביל למימוש "שרת לקוח" החזקתי בשרת ווקטור (עבור אפשרות לנעול באופן מקבילי) המכיל את יחסי המרת המשקולות, כאשר קילו הוא המשקולת בעלת המשקל 1. הפעולה ConvertWeight שהינה synchronized מאפשרת ל2' לקוחות להמיר בו זמנית בלי לפגוע בנכונות ההמרה. כך הנתונים נשמרים בבטחה, וכך כל לקוח יקבל את סכום ההמרה עבור עצמו. כך הפרויקט רץ באופן מבוזר ומחזיר תוצאות מהימנות.

על תרחישי השרת:

בתחילה האפליקציה היתה stand-alone, כלומר, לא תקשרו עם אף-אחד וכעת רציתי לכתוב תוכנית שיודעת "לדבר" עם תוכנית אחרת. לדוגמא:

במערכת בנק: יש ישות מרכזית המחזיקה את הנתונים של כל הלקוחות, ויש ישויות המבקשות מידע מישות מרכזית זו. הישות המרכזית המספקת מידע נקראת "שרת" (Server).

ישות הקצה המבקשת מידע נקראת "לקוח" (Client).

שרת יכול לספק מידע לכמה לקוחות.

העבודה מול הסרבר במימוש זה הינה סימטרית ועל כל הודעה של הקליינט יש לחכות להודעה מהסרבר, כלומר:

לא ניתן לשלוח לסרבר 2 הודעות ברצף ללא מענה, מאחר והפקודה readLine הינה פקודת blocking .

לכן לא ניתן לעבור לפקודה הבאה עד שהיא תסתיים, משמע עד שהסרבר ישלח מידע.

אפליקציית ה- server מופעלת ומחכה שאפליקציות client יתחברו עליה.

עבור כל אפליקציית client המתחברת ל- server נוצר ערוץ תקשורת ייחודי (בעזרת thread) כלומר, client לא יכול לראות מהי התקשורת המועברת ל- server מ- client אחר.

לאחר יצירת ערוץ התקשורת ה- client וה- server יכולים "לדבר" וכאשר ה- client מתנתק ה- server סוגר את ערוץ התקשורת.

ה- server הוא אפליקציה שאמורה לרוץ תמיד, ולכן לא תיזום ניתוק של client'ים המחוברים אליה

המערכת שלנו מורכבת מ- 2 תוכניות נפרדות, אחת ל- **client** ואחת ל- **server**.

הרעיון הוא להפעיל את אפליקציית ה- **server** פעם אחת בלבד, ואז להפעיל את אפליקציית ה- **client** מספר כלשהו של פעמים, אחת עבור כל **client**.

כל אפליקציית client תתחבר לאפליקציית ה- server וכך התקשורת בין השרת והלקוח היא סינכרונית, כלומר, על כל הודעה שהלקוח שולח לשרת הוא מצפה לקבל תשובה.

יתכן והלקוח ירצה לשלוח הודעה בכל רגע נתון, או שתי הודעות ברצף ויתכן והסרבר ירצה ליזום הודעה לקליינט (כמו בדוגמא עם ה- broadcast). כלומר, יש לבצע שינוי במימוש של הקליינט כך שיהיה thread נפרד לשליחת הודעות לסרבר ו- thread נפרד המטפל בהודעות מהסרבר.

המימוש בקונסול הוא בעייתי מאחר והתצוגה תשתבש (יציג "אנא הכנס קלט" ופתאום תוצג הודעה מהסרבר).

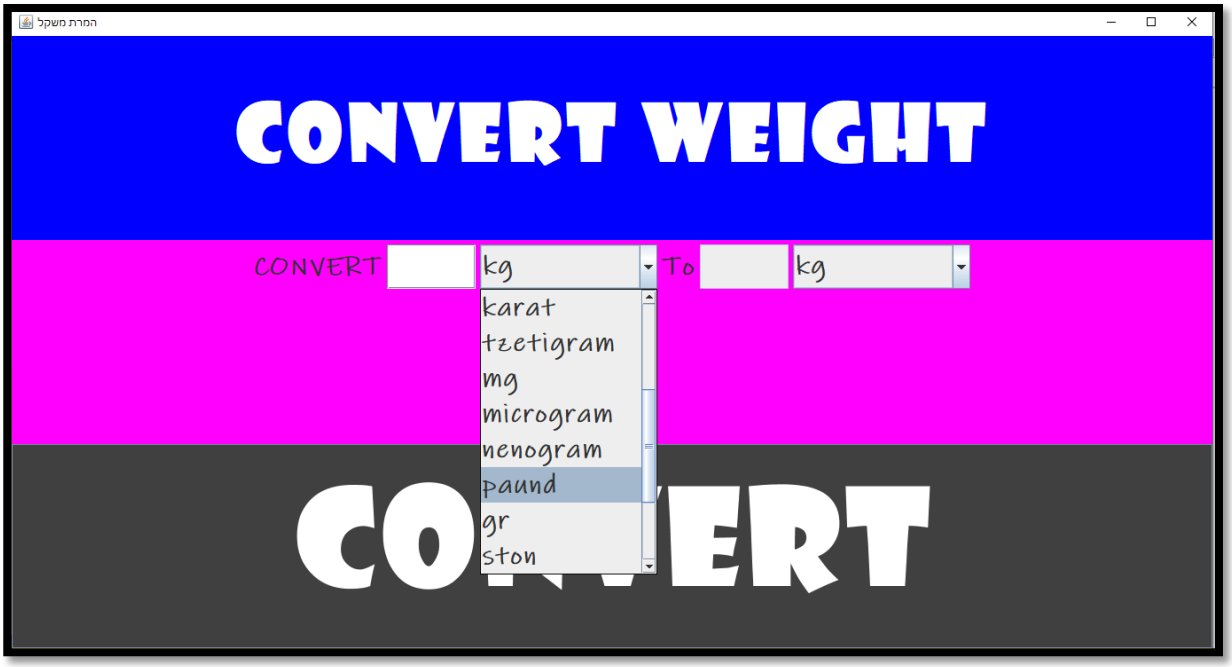
לכן מימשתי את הקוד בGUI.

הרצת השרת וכניסת לקוחות

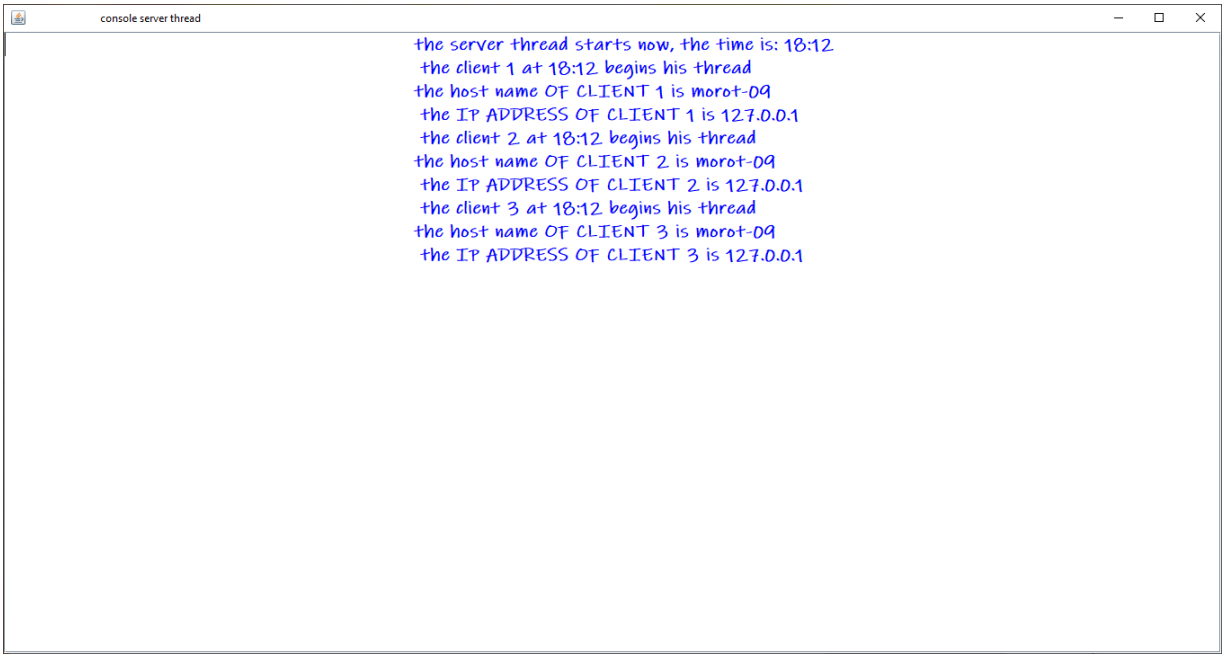
- ❖ יש להריץ את השרת:
- ❖ הרצת המחלקה "ThreadServerConsole"

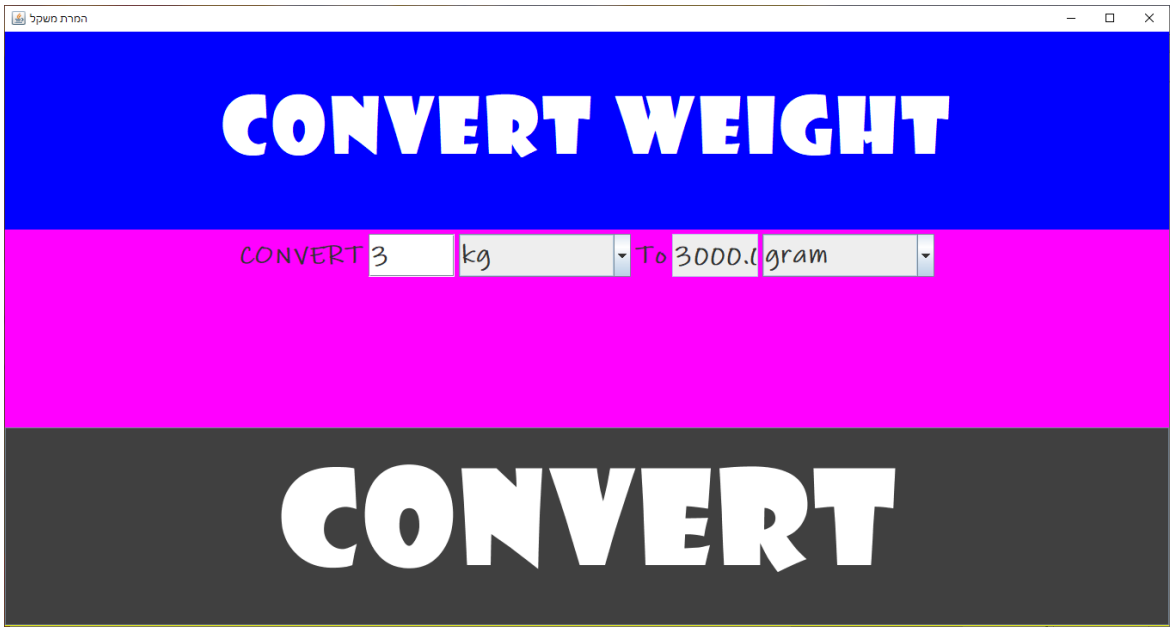
תניב את החלון הבא, בו יופיע בהמשך מידע לגבי כניסת לקוחות נוספים לשרת:

- יש להריץ את הלקוח:
- הרצת המחלקה " ClientGUI " תניב את החלון
הבא בו מופיעה תיבת טקסט לבחירת כמות,
ותיבות רב-ברירה לבחירת מקור ויעד להמרה.



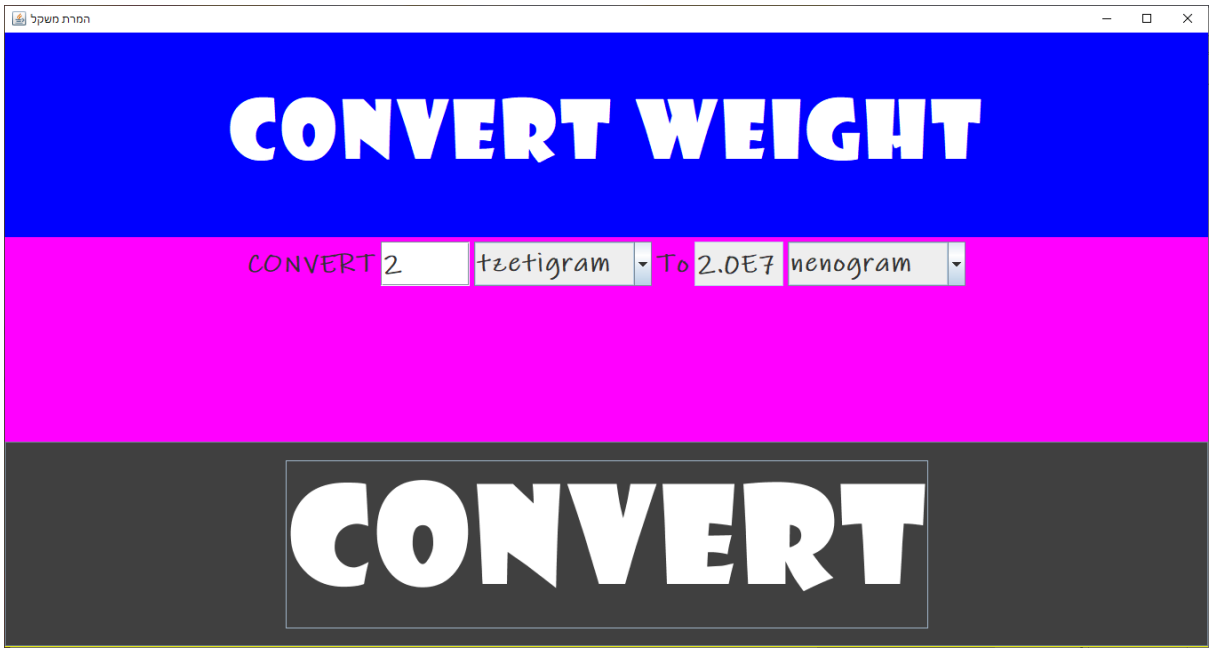
כל לקוח המתחבר לשרת, מוסיף שורה בחלונית השרת:

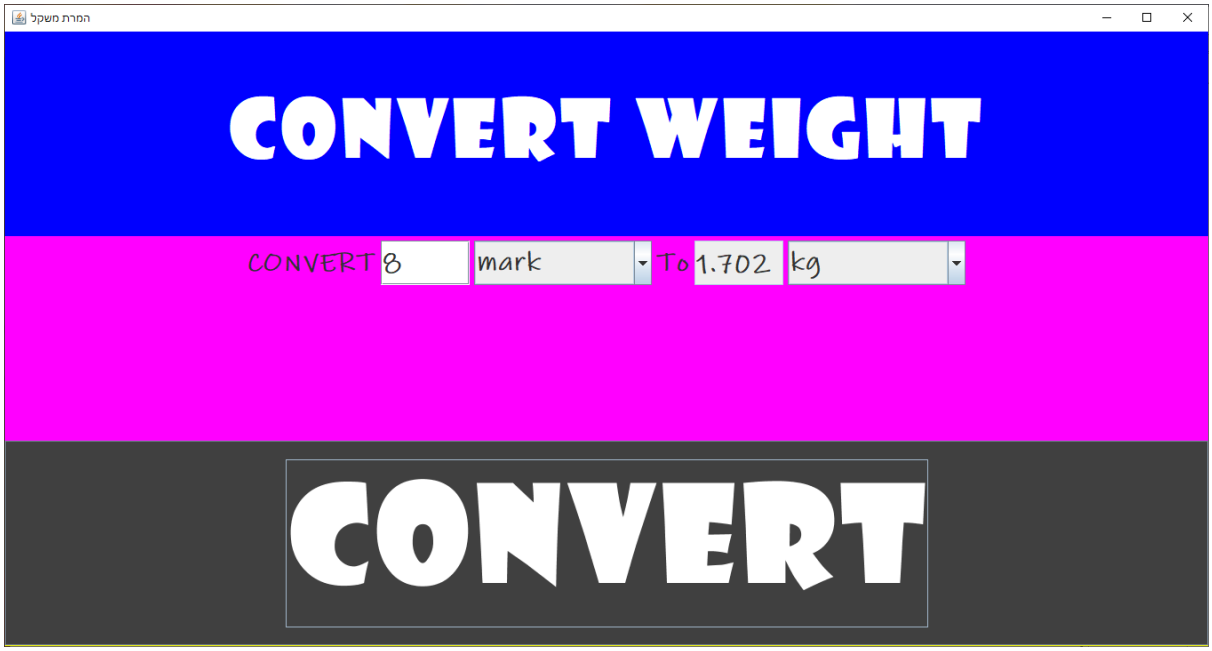




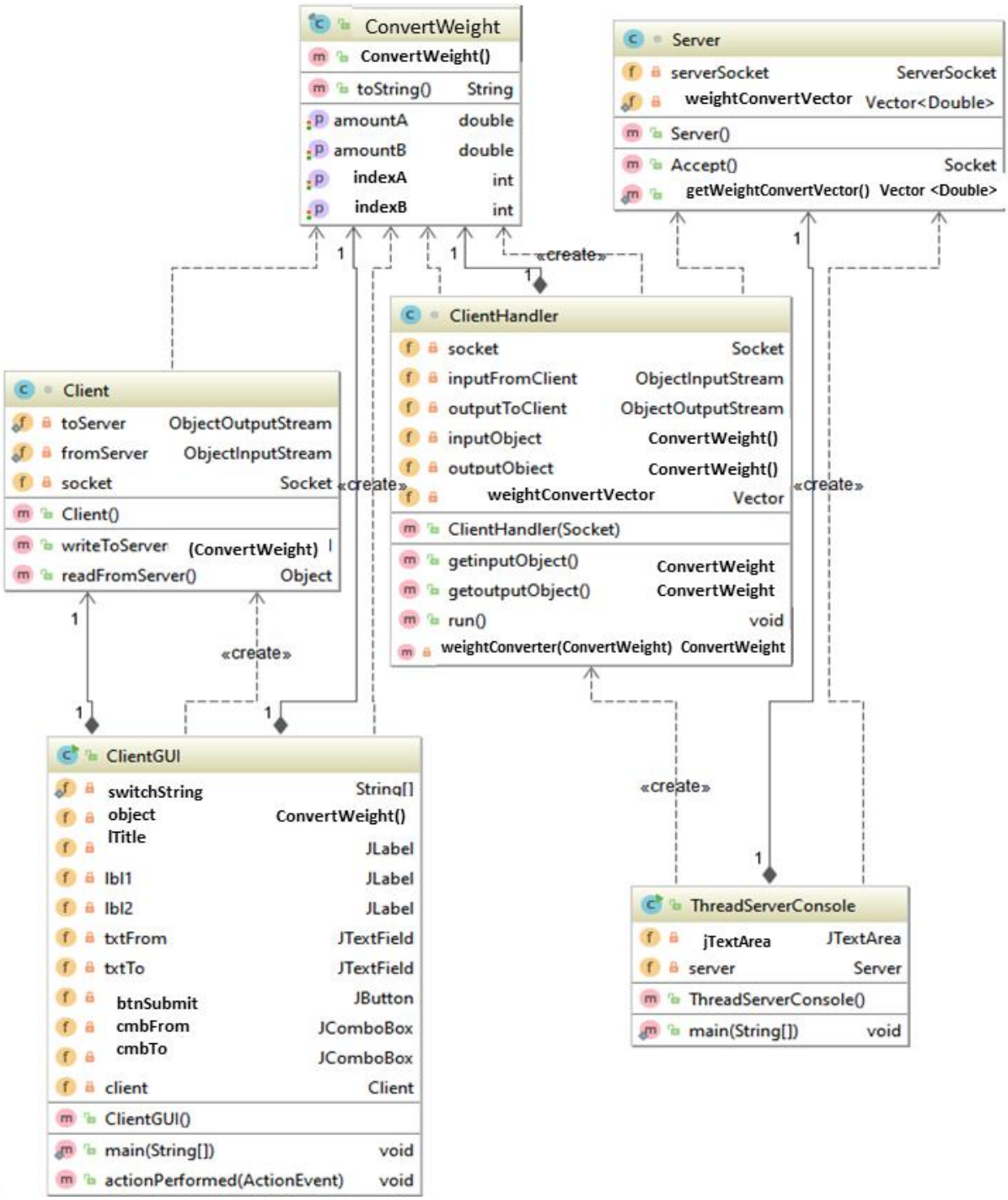
על מנת לבצע המרת משקולת, יש לבחור את הסכום להמרה, את סוג המשקל ממנו רוצים להמיר ואת הערך שרוצים לקבל. כדי לשלח את הנתונים לשרת יש לבחור בכפתור "CONVET".

לאחר לחיצה על כפתור ההמרה, יופיע הייצוג של המשקל בערך הדרוש. הוא נכון גם כאשר כמה לקוחות ניגשים יחד לשרת.


המרת משקל
CONVERT WEIGHT
CONVERT 2
tzetigram
To 2.0E7
nenogram
CONVERT


המרת משקל
CONVERT WEIGHT
CONVERT 8
mark
To 1.702
kg
CONVERT

תרשים UML:


ConvertWeight
ConvertWeight()
toString() String
amountA double
amountB double
indexA int
indexB int
Server
serverSocket ServerSocket
weightConvertVector Vector<Double>
Server()
Accept() Socket
getWeightConvertVector() Vector <Double>
«create»
ClientHandler
socket Socket
inputFromClient ObjectInputStream
outputToClient ObjectOutputStream
inputObject ConvertWeight()
outputObject ConvertWeight()
weightConvertVector Vector
ClientHandler(Socket)
getinputObject() ConvertWeight
getoutputObject() ConvertWeight
run() void
weightConverter(ConvertWeight) ConvertWeight
Client
toServer ObjectOutputStream
fromServer ObjectInputStream
socket Socket
Client()
writeToServer (ConvertWeight)
readFromServer() Object
«create»
ClientGUI
switchString String[]
object ConvertWeight()
lTitle JLabel
lbl1 JLabel
lbl2 JLabel
txtFrom JTextField
txtTo JTextField
btnSubmit JButton
cmbFrom JComboBox
cmbTo JComboBox
client Client
ClientGUI()
main(String[]) void
actionPerformed(ActionEvent) void
ThreadServerConsole
jTextArea JTextArea
server Server
ThreadServerConsole()
main(String[]) void

תיאור מחלקות:

מחלקת Server –

שדה	תפקיד
private ServerSocket serverSocket	שקע (Socket) לחיבורי משתמשים חדשים
private static Vector<Double> weightConvertVector	Vector שמכיל את ערכי המרת המשקולת

פעולה	תפקיד
public Server()	פעולה בונה
public Socket Accept()	הפעולה ממתינה לחיבור שיבוצע לשקע ומקבלת אותו
public static Vector<Double> getWeightConvertVector ()	פעולה סטטית המחזירה את ה – Vector

מחלקת ThreadServerConsole

שדה	תפקיד
private JTextArea jTextArea	שדה טקסט
private Server server	משתנה מסוג Server (שרת)

פעולה	תפקיד
public ThreadServerConsole()	פעולה בונה היוצרת תהליכים לפי כמות השקעים (Sockets) ללקוחות

מחלקת ClientHandler

שדה	תפקיד
private Socket socket;	שקע (Socket) לחיבורים
private ObjectInputStream inputFromClient	זרם קלט הנתונים (אובייקט)
private ObjectOutputStream outputToClient	זרם פלט הנתונים (אובייקט)
private ConvertWeight inputObject	אובייקט מסוג ConvertWeight המכיל את המידע ששלח הלקוח אל השרת

private ConvertWeight outputObject	אובייקט מסוג ConvertWeight המכיל את כל המידע לאחר ההמרה שביצע השרת
private Vector weightConvertVector	משתנה מסוג Vector שמכיל את ערכי המשקולות לפי קילוגרם=1 יחידה.

פעולה	תפקיד
public ClientHandler(Socket socket)	פעולה בונה
public CHANGE getinputObject()	הפעולה מחזירה את האובייקט מסוג ConvertWeight המכיל את המידע ששלח הלקוח את השרת
public CHANGE getoutputObject()	הפעולה מחזירה את האובייקט מסוג ConvertWeight המכיל את המידע לאחר ביצוע ההמרה בשרת
public void run()	הפעולה מבצעת תהליך להמרת ערך המשקל שהתקבל מהלקוח דרך השקע (Socket) אל השרת ולאחר מכן שולחת את כל המידע המומר חזרה אל הלקוח.
synchronized private ConvertWeight weightConverter (ConvertWeight inputObject)	הפעולה מקבלת אובייקט מסוג ConvertWeight המכיל את המידע אודות המטבע המקורי מסכום הכסף אותו צריך להמיר למטבע החדש, מבצעת את ההמרה ולאחר מכן מחזירה אובייקט מסוג ConvertWeight חדש עם הערך המומר.

מחלקת – Client

שדה	תפקיד
private static ObjectOutputStream *toServer*	הנתונים הנשלחים לשרת
private static ObjectInputStream *fromServer*	הנתונים המתקבלים מהשרת
private Socket socket	תבנית של כתובת אינטרנט

פעולה	תפקיד
public Client()	הפעולה מאתחלת את הערוץ הלוגי של התקשורת עם השרת
public void writeToServer(ConvertWeight object)	הפעולה שולחת נתונים אל השרת
public Object readFromServer()	הפעולה מקבלת נתונים מהשרת

מחלקת – ClientGUI

שדה	תפקיד
private ConvertWeight object	אובייקט מסוג ConvertWeight המכיל נתונים שישלחו לשרת
private JTextField txtFrom	שדה טקסט לכתיבת ערך המשקל אותו רוצים להמיר
private JTextField txtTo	שדה טקסט בו יופיע סכום הערך לאחר המרתו למשקל הדרוש
private JButton btnSubmit	כפתור לשליחת הנתונים אל השרת
private JComboBox cmbFrom, cmbTo	תיבת בחירה לבחירת ערך המשקל
private Client client	משתנה מסוג Client (לקוח)

פעולה	תפקיד
public void actionPerformed	פעולה המתבצעת כאשר נלחץ על המקש הנבחר (במקרה זה כאשר נבחר ערך המשקל, ומהלך שליחת וקבלת המידע מהשרת)

מחלקת ConvertWeight

שדה	תפקיד
private int indexA, indexB	משתנים המייצגים את מיקום המשקל המקורי והמשקל הדרוש להמרה בווקטור שבשרת.
private double amountA, amountB	משתנים המייצגים את הערך לפני ההמרה ולאחריה

פעולה	תפקיד
Public ConvertWeight ()	פעולה בונה
public Object get()	פעולה המחזירה את הערך המבוקש
public Object set()	פעולה המגדירה את הערך המבוקש

קטעי קוד:

צד שרת

Class Server:

```
package changeserver;

import java.lang.Math;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Vector;

class Server {
//server socket and vector with weight values, the server socket enters to port 8000.
//it can return the vector. (for sychronize...)

    private ServerSocket serverSocket;
    private static Vector<Double> weightConvertVector = new Vector<Double>();

    public Server() throws IOException {
        double toTon = Math.pow(10, -3);
        double kiloTon = Math.pow(10, -3) * 9.8;
        double nenoGram = Math.pow(10, 12);
        double masaAtom = Math.pow(10, 26) * 6;
        weightConvertVector.add(1.0); // kg
        weightConvertVector.add(toTon); // ton
        weightConvertVector.add(kiloTon); // kiloTon
        weightConvertVector.add(10.0); // hectogram
        weightConvertVector.add(100.0); // dag
```

```
    weightConvertVector.add(1000.0); // gram
    weightConvertVector.add(5000.0); // karat
    weightConvertVector.add(100000.0); // tzetigram
    weightConvertVector.add(1000000.0); // mg
    weightConvertVector.add(1000000000.0); // microgram
    weightConvertVector.add(nenoGram); // nenogram
    weightConvertVector.add(2.2); // paund
    weightConvertVector.add(15432.4); // gr
    weightConvertVector.add(0.2); // ston
    weightConvertVector.add(35.8); // onkia
    weightConvertVector.add(4.7); // mark
    weightConvertVector.add(643.0); // agora
    weightConvertVector.add(masaAtom); // masaAtom

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

}

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

```
    public static Vector<Double> getWeightConvertVector() {
        return weightConvertVector;
    }

}
```

Class ThreadServerConsole:

```
package changeserver;

import changeserver.Server;
import changeserver.ClientHandler;
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalTime;
import javax.swing.*;


public class ThreadServerConsole extends JFrame {

    private JTextArea jTextArea = new JTextArea();
    private Server server;

    public static void main(String[] args) throws IOException {
        new ThreadServerConsole();
    }

    public ThreadServerConsole() throws IOException {
```

```
Date dt = new Date();  // current time
int hours = dt.getHours();
int minutes = dt.getMinutes();
Font font = new Font("Ink Free", Font.BOLD, 20);//Showcard GothicCooper Black
jTextArea.setFont(font);
jTextArea.setForeground(Color.BLUE);
server = new Server();
jTextArea.setSize(100,499);
// Place text area on the frame
setLayout(new BorderLayout());
add(new JScrollPane(jTextArea), BorderLayout.CENTER);
setTitle("                    console server thread                                    ");
setSize(1370, 730);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true); // It is necessary to show the frame here!Segoe Script

jTextArea.append("                                    the server thread starts now, the time is: " + hours +
":" + minutes + '\n');

int clientNo = 1;

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

    // Display the client number
    jTextArea.append("                                    the client " + clientNo + " at " + hours + ":" +
minutes + " begins his thread" + '\n');

    // Find the client's host name, and IP address
    InetAddress inetAddress = socket.getInetAddress();
```

```
        jTextArea.append("                                    the host name OF CLIENT " + clientNo + " is " +
inetAddress.getHostName() + "\n");
        jTextArea.append("                                    the IP ADDRESS OF CLIENT " + clientNo + " is " +
inetAddress.getHostAddress() + "\n");

        // Create a new task for the connection
        Thread task = new Thread(new ClientHandler(socket));
        task.start();
        clientNo++;
      }
    }
}
```

Class ClientHandler:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package changeserver;

import changeclient.ConvertWeight;
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.ArrayList;
import java.util.StringTokenizer;
import java.util.Vector;
```

```
class ClientHandler implements Runnable {

//get in and output stream from client, and the class cw from client, and the vector.
//it calculate the weight value

    private Socket socket; // A connected socket

    private ObjectInputStream inputFromClient;
    private ObjectOutputStream outputToClient;

    private ConvertWeight inputObject = new ConvertWeight();
    private ConvertWeight outputObject = new ConvertWeight();

    private Vector weightConvertVector = Server.getWeightConvertVector();

    public ClientHandler(Socket socket) {
        this.socket = socket;
    }

    public ConvertWeight getinputObject() {
        return inputObject;
    }

    public ConvertWeight getoutputObject() {
        return outputObject;
    }

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

```
        inputFromClient = new ObjectInputStream(socket.getInputStream());

        // Continuously serve the client
        while (true) {
            // Receive inputString from the client
            try {
                inputObject = (ConvertWeight) inputFromClient.readObject();
                System.out.println(inputObject + "-----found entering to server");
            } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            // Compute currency change
            outputObject = weightConverter(inputObject);

            // Send converted money back to the client
            outputToClient.writeObject(inputObject);

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

public ClientHandler() {
}

synchronized private ConvertWeight weightConverter(ConvertWeight inputObject) {
    int indexA = inputObject.getIndexA(), indexB = inputObject.getIndexB();
    double amountA = inputObject.getAmountA(), amountB = 0;
```

```
        System.out.print("amount A " + amountA + "\n");//got   1000 sm
        System.out.print(" amount B" + amountB + "\n");////the result 1 km
        System.out.print(" index A " + indexA + "\n");//which : kilo or... sm
        System.out.print(" index B " + indexB + "\n");//to what? km
        if (indexA == 0) // original index means to kilogram (kilo)
        {
            amountB = (amountA * ((Double) (weightConvertVector.elementAt(indexB))).doubleValue());
            inputObject.setAmountB(amountB);
            return inputObject;
        } else {
            if (indexB == 0) {
                amountB = (amountA * (1.0 / (Double) weightConvertVector.elementAt(indexA)));
                inputObject.setAmountB(amountB);
                return inputObject;

            } else {
                amountA = (amountA * (1.0 / (Double) weightConvertVector.elementAt(indexA)));
                indexA = 0;
                amountB = (amountA * ((Double) (weightConvertVector.elementAt(indexB))).doubleValue());
                inputObject.setAmountB(amountB);
                return inputObject;
            }

        }

    }

}
```

Class Client:

```
package changeclient;

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 {
   //read and write to server
   // 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 void writeToServer(ConvertWeight object) {
    try {
        toServer.writeObject(object);
        toServer.flush();
    } 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 null;
```

```
    }
```

Class ClientGUI:

```
package changeclient;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JTextField;

public class ClientGUI extends JFrame implements ActionListener {
    //gui: with switchString
    private static String[] switchString = {"kg", "ton", "kiloTon", "hectogram", "dag", "gram", "karat", "tzetigram",
        "mg", "microgram", "nenogram", "paund", "gr",
        "ston", "onkia", "mark", "agora", "masaAtom"};

    private ConvertWeight object = new ConvertWeight(); // object to be sent for server handling
    private JLabel lTiltle = new JLabel("CONVERT WEIGHT");
    private JLabel lbl1 = new JLabel("CONVERT");
    private JLabel lbl2 = new JLabel("To");
    private JTextField txtFrom = new JTextField();
    private JTextField txtTo = new JTextField();
    private JButton btnSubmit = new JButton("CONVERT");
    private JComboBox cmbFrom, cmbTo;
    private Client client = new Client();

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

    public ClientGUI() {

        Font font1 = new Font("Showcard Gothic", Font.BOLD, 150);//Showcard GothicCooper Black
        Font font2 = new Font("Ink Free", Font.BOLD, 30);//Showcard GothicCooper Black
        Font font3 = new Font("Showcard Gothic", Font.BOLD, 90);//Showcard GothicCooper Black

        //Showcard Gothic Cooper Black
        lTiltle.setFont(font3);
        btnSubmit.setFont(font1);
        txtTo.setEditable(false);
        JPanel pMain = new JPanel(), pHeader = new JPanel(), middlePannel = new JPanel(), pButtonArea = new
JPanel();
        pMain.setLayout(new GridLayout(0, 1));
        pHeader.setLayout(new GridLayout(1, 1));
        lTiltle.setHorizontalAlignment(SwingConstants.CENTER);
        pHeader.add(lTiltle, BorderLayout.NORTH);
```

```
    middlePannel.setLayout(new FlowLayout());
    middlePannel.add(lbl1);
    pHeader.setBackground(Color.BLUE);

    cmbFrom = new JComboBox(switchString);
    cmbFrom.addActionListener(this);
    cmbFrom.setPreferredSize(new Dimension(200, 50));
    middlePannel.add(txtFrom);
    middlePannel.add(cmbFrom);
    middlePannel.add(lbl2);
    cmbTo = new JComboBox(switchString);
    cmbTo.addActionListener(this);
    cmbTo.setPreferredSize(new Dimension(200, 50));
    middlePannel.add(txtTo);
    middlePannel.add(cmbTo);
    txtFrom.setHorizontalAlignment(JTextField.LEFT);
    txtFrom.setPreferredSize(new Dimension(100, 50));
    txtTo.setHorizontalAlignment(JTextField.LEFT);
    txtTo.setPreferredSize(new Dimension(100, 50));
    pButtonArea.setLayout(new GridLayout(1, 1));
    btnSubmit.setPreferredSize(new Dimension(200, 50));
    pButtonArea.add(btnSubmit);
    pMain.add(pHeader);
    pMain.add(middlePannel);
    pMain.add(pButtonArea);
    setContentPane(pMain);
    pMain.setBackground(Color.yellow);
    btnSubmit.addActionListener(new ButtonListener());
    setTitle("המרת משקל");
    setSize(1370, 730);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    txtFrom.setFont(font2);
    txtTo.setFont(font2);
    cmbFrom.setFont(font2);
    cmbTo.setFont(font2);
    lbl1.setFont(font2);
    lbl2.setFont(font2);
    lTiltle.setForeground(Color.white);
    middlePannel.setBackground(Color.magenta);
    btnSubmit.setForeground(Color.white);
    btnSubmit.setBackground(Color.darkGray);

}

private class ButtonListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        // Send the ConvertWeight object to the server
        object = new ConvertWeight();

        object.setAmountA(Double.valueOf(txtFrom.getText().trim()) + 0);// +0 is used if no value is written in
        // txtFrom.
```

```
        object.setIndexA(cmbFrom.getSelectedIndex()); // getting index inx.
        object.setIndexB(cmbTo.getSelectedIndex());

        System.out.println(object + "before sending"); // check data prior sending - debug
        client.writeToServer(object);

        // Get outputString from the server
        ConvertWeight outputObject = (ConvertWeight) client.readFromServer();
        System.out.println(outputObject + "after sending");// debug

        // Display to the text the changed amount of money
        txtTo.setText(outputObject.getAmountB() + "");
    }
  }

  @Override
  public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub

    if (e.getSource() == cmbFrom) {
        // object.setAmountA(Integer.valueOf((txtFrom.getText().trim())));
        object.setIndexA(cmbFrom.getSelectedIndex()); // getting index inx.
    }
    if (e.getSource() == cmbTo) {
        object.setIndexB(cmbTo.getSelectedIndex());
    }
  }
}
```

Class ConvertWeight:

```
package changeclient;

import java.io.Serializable;
import java.text.DecimalFormat;

public final class ConvertWeight implements Serializable {

   //value of indexes in the vector and amounts.
   private int indexA, indexB;
   private double amountA, amountB;

   public ConvertWeight() {
      super();
      this.indexA = 0; // index in WeightConvertVector and the same in convertList
      this.amountA = 0;//the amount of A
      this.indexB = 0; // index in WeightConvertVector and the same in convertList
      this.amountB = 0;//the amount of B
   }

   public int getIndexA() {
      return indexA;
   }

   public int getIndexB() {
      return indexB;
   }

   public void setIndexA(int indexA) {
      this.indexA = indexA;
   }

   public void setIndexB(int indexB) {
      this.indexB = indexB;
   }

   @Override
   public String toString() {
      return "ConvertWeight{" + "indexA=" + indexA + ", indexB=" + indexB + ", amountA=" + amountA + ",
amountB=" + amountB + '}';
   }



   public double getAmountA() {
      amountA = Double.parseDouble(new DecimalFormat("##.###").format(amountA));
      return amountA;
   }

   public void setAmountA(double amountA) {
      this.amountA = amountA;
   }
```

```
    public double getAmountB() {

        amountB = Double.parseDouble(new DecimalFormat("##.###").format(amountB));
        return amountB;
    }

    public void setAmountB(double amountB) {
        this.amountB = amountB;
    }



}
```

ביבליוגרפיה:

המר משקל(convertworld.com)

300 Core Java Interview Questions (2021) - javatpoint

UML Class Diagram Tutorial (visual-paradigm.com)

https://stackoverflow.com/

Java Programming: Solving Problems with Software | Coursera

Java programming Exercises, Practice, Solution - w3resource

https://www.ccs.neu.edu/home/vkp/Papers/PSF-ccscne2003.pdf

50+ Java Coding and Programming Problems for Interviews | Java67

Solve a real-world problem using Java | Opensource.com

Solve Java | HackerRank