**פ ר ו 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 הפרויקט: מימוש "שרת - לקוח" ("Client - server")**

**שם הסטודנט:**

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

**שנה"ל תש"ף 2020**

ה צ ה ר ת ס ט ו ד נ ט

אני הסטודנט מס' ת.ז החתום מטה, מצהיר בזאת שכל עבודת הגמר/ הפרויקט המוגש/ת בחוברת זו הינו/ה פרי עבודתי בלבד .

על בסיס הנחייתו של המנחה ותוך הסתמכות על מקורות הידע והמידע האחרים המצויים בביבליוגרפיה 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

[הקדמה 4](#_Toc40953007)

[הפעלת התוכנה: 5](#_Toc40953008)

[תרשים UML: 7](#_Toc40953009)

[תיאור מחלקות: 8](#_Toc40953010)

[מחלקת – Client 8](#_Toc40953011)

[מחלקת – ClientGUI 8](#_Toc40953012)

[מחלקת – Server 9](#_Toc40953013)

[מחלקת ThreadServerConsole 9](#_Toc40953014)

[מחלקת ClientHandler 9](#_Toc40953015)

[מחלקת CHANGE 10](#_Toc40953016)

[קטעי קוד: 11](#_Toc40953017)

[Class Server: 11](#_Toc40953018)

[Class ThreadServerConsole: 12](#_Toc40953019)

[Class Client: 13](#_Toc40953020)

[Class ClientHandler: 15](#_Toc40953021)

[Class ClientGUI: 17](#_Toc40953022)

[Class CHANGE: 19](#_Toc40953023)

[ביבליוגרפיה: 21](#_Toc40953024)

# הקדמה

בפרויקט זה פיתחתי תוכנה המממשת את מנגנון "שרת – לקוח" (Client - Server).

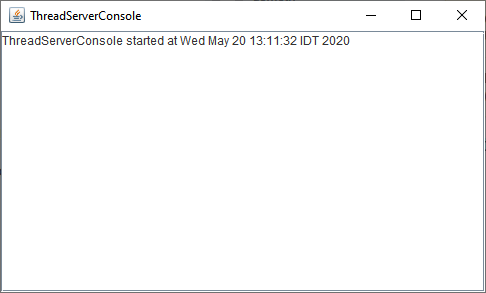
בשרת מוחזק ווקטור שמכיל את יחסי המרת המטבעות משקל אל המטבע הזר. כאשר מתחבר לקוח אל השרת, כאשר הוא ירצה לבצע המרה ויבחר במטבע הרצוי ובסכום, השרת מקבל את המידע וממיר לו את המטבע לפי שערי המטבעות ששמורים לו בווקטור.

לאחר שהשרת בעצם ביצע את תהליך ההמרה הוא יחזיר למסך את הערך המומר והלקוח יקבל את סכום הכסף אותו ביקש במטבע הזר שאליו רצה להמיר את כספו.

בפרויקט נעשה שימוש בפעולה שהינה synchronized, זאת בכדי שכאשר לקוח אחד משתמש בשירותי השרת על הלקוח השני יהיה להמתין עד שהלקוח הראשון יסיים, על מנת שלא ייוצר מצב ששני לקוחות ניגשים אל פעולת ההמרה ובעצם משנים את הנתונים, מצב שבמקרה קיצון יכול לגרום לכך שהלקוח לא יקבל את סכום ההמרה הנכון עבורו אלא קיבל סכום המרה של לקוח אחר.

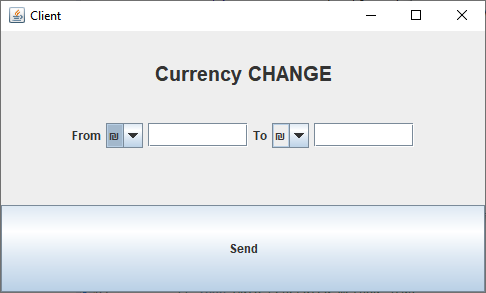
# הפעלת התוכנה:

1. ראשית הפעל את השרת

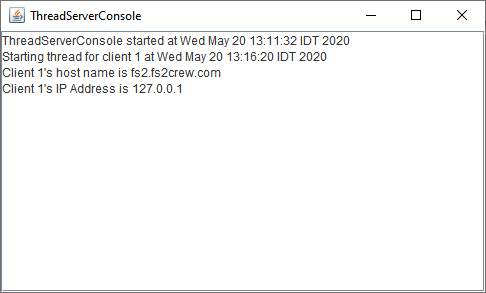
* הרץ את המחלקה "ThreadServerConsole" ובעקבות זאת יופיע החלון הבא:

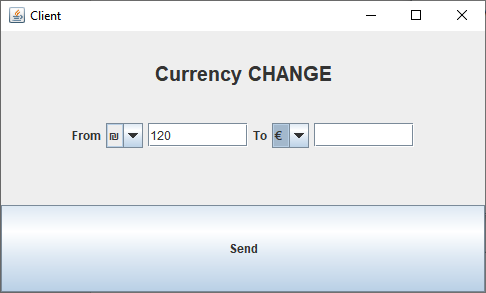
חלון השרת, בו אני יכול לראות גם מידע על מי מחובר לשרת

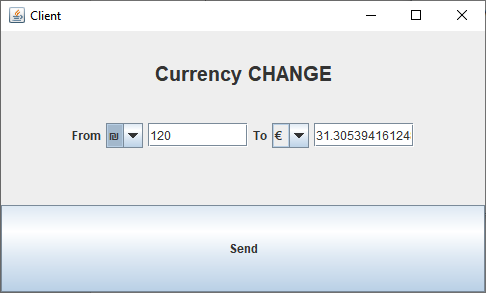
1. הרץ את הצד לקוח

- הרץ את המחלקה "ClientGUI" ובעקבות זאת יופיע החלון הבא:

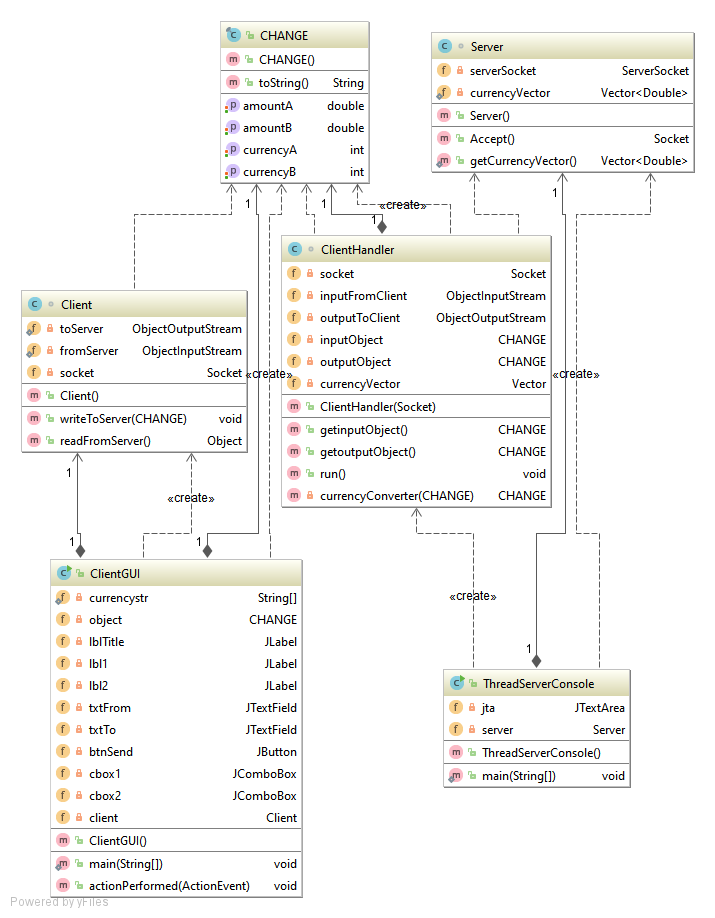
חלון הלקוח - ביצוע המרת המטבעות

כאשר לקוח מתחבר על הסרבר הוא מופיע בו:

1. על מנת לבצע המרת מטבע, נבחר בצד שמאל של המסך את סוג במטבע המקורי, ולידו נכניס את הסכום אותו נרצה להמיר. לאחר מכן בצד ימין נבחר בערך המטבע החדש שאליו נרצה להמיר את הכסף שלנו, ואז נלחץ למטה על "Send" או "שלח" בעברית.

1. לאחר לחיצה על כפתור ה – "Send" יוצג לנו בצד ימין סכום הכסף החדש שהמרנו במטבע החדש:

# תרשים UML:



# תיאור מחלקות:

## מחלקת – Client

|  |  |
| --- | --- |
| **פירוט** | **שדה** |
| הנתונים ששולחים לשרת | **private** **static** ObjectOutputStream *toServer* |
| הנתונים שמתקבלים מהשרת | **private** **static** ObjectInputStream *fromServer* |
| תבנית של כתובת אינטרנט | **private** Socket socket |

|  |  |
| --- | --- |
| **פירוט** | **פעולה** |
| הפעולה מאתחלת את הערוץ הלוגי של התקשורת עם השרת | **public** Client() |
| הפעולה שולחת נתונים אל השרת | **public** **void** writeToServer(CHANGE object) |
| הפעולה מקבלת נתונים מהשרת | **public** Object readFromServer() |

## מחלקת – ClientGUI

|  |  |
| --- | --- |
| **פירוט** | **שדה** |
| אובייקט מסוג CHANGE עם נתונים שישלחו לשרת | **private** CHANGE object |
| שדה טקסט לכתיבת סכום הכסף אותו נרצה להמיר | **private** JTextField txtFrom |
| שדה טקסט בו יופי לאחר ההמרה סכום הכסף המומר | **private** JTextField txtTo |
| לחצן לשליחת הנתונים אל השרת | **private** JButton btnSend |
| תיבת בחירה לבחירת סוג המטבע | **private** JComboBox cbox1, cbox2 |
| משתנה מסוג Client (לקוח) | **private** Client client |

|  |  |
| --- | --- |
| **פירוט** | **פעולה** |
| פעולה שמתבצעת כאשר נלחץ על המקש הנבחר (במקרה שלנו זה כאשר נבחר את סוג המטבע ואז נעדכן אותו ולאחר מכן בתקשורת עם השרת – שליחת וקבלת מידע) | **public** **void** actionPerformed |

## מחלקת – Server

|  |  |
| --- | --- |
| **פירוט** | **שדה** |
| שקע (Socket) לחיבורי משתמשים חדשים | **private** ServerSocket serverSocket |
| Vector שמכיל את ערכי המרת המטבע | **private** **static** Vector<Double> *currencyVector* |

|  |  |
| --- | --- |
| **פירוט** | **פעולה** |
| פעולה בונה | **public** Server() |
| הפעולה ממתינה לחיבור שיבוצע לשקע ומקבלת אותו | **public** Socket Accept() |
| פעולה סטטית המחזירה את ה – Vector | **public** **static** Vector<Double> getCurrencyVector() |

## מחלקת ThreadServerConsole

|  |  |
| --- | --- |
| **פירוט** | **שדה** |
| שדה טקסט | **private** JTextArea jta |
| משתנה מסוג Server (שרת) | **private** Server server |

|  |  |
| --- | --- |
| **פירוט** | **פעולה** |
| פעולה בונה היוצרת תהליכים לפי כמות השקעים (Sockets) ללקוחות | **public** ThreadServerConsole() |

## מחלקת ClientHandler

|  |  |
| --- | --- |
| **פירוט** | **שדה** |
| שקע (Socket) לחיבורים | **private** Socket socket; |
| זרם קלט הנתונים (אובייקט) | **private** ObjectInputStream inputFromClient |
| זרם פלט הנתונים (אובייקט) | **private** ObjectOutputStream outputToClient |
| אובייקט מסוג CHANGE שמכיל את המידע ששלח הלקוח אל השרת | **private** CHANGE inputObject |
| אובייקט מסוג CHANGE שמכיל את כל המידע לאחר ההמרה שביצע השרת | **private** CHANGE outputObject |
| משתנה מסוג Vector שמכיל את שערי ההמרה של המטבעות לפי מטבע ה"שקל" | **private** Vector currencyVector |

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

## מחלקת CHANGE

|  |  |
| --- | --- |
| **פירוט** | **שדה** |
| משתנה מסוג int שמייצג את סוג מטבע המקור והמטבע החדש | **private** **int** currencyA, currencyB |
| משתנה מסוג double שמייצג את סכום הכסף של מטבע המקור וסכום הכסף לפי ערך המטבע החדש | **private** **double** amountA, amountB |

|  |  |
| --- | --- |
| **פירוט** | **פעולה** |
| פעולה בונה | **public** CHANGE() |
| פעולה המחזירה את הערך המבוקש | **public** Object get() |
| פעולה המגדירה את הערך המבוקש | **public** Object set() |

# קטעי קוד:

## Class Server:

**import** java.io.IOException;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.util.Vector;

**class** Server {

**private** ServerSocket serverSocket;

**private** **static** Vector<Double> *currencyVector* = **new** Vector<Double>();

**public** Server() {

*currencyVector*.add(1.0); // Shekel

*currencyVector*.add(3.83320521); // euro

*currencyVector*.add(3.46291564); // dollar

*currencyVector*.add(4.48631109); // British Pounds

**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> getCurrencyVector() {

**return** *currencyVector*;

}

}

## Class ThreadServerConsole:

**import** java.io.\*;

**import** java.net.\*;

**import** java.util.\*;

**import** java.awt.\*;

**import** javax.swing.\*;

**public** **class** ThreadServerConsole **extends** JFrame {

**private** JTextArea jta = **new** JTextArea();

**private** Server server = **new** Server();

**public** **static** **void** main(String[] args) {

**new** ThreadServerConsole();

}

**public** ThreadServerConsole() {

// Place text area on the frame

setLayout(**new** BorderLayout());

add(**new** JScrollPane(jta), BorderLayout.***CENTER***);

setTitle("ThreadServerConsole");

setSize(500, 300);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setVisible(**true**); // It is necessary to show the frame here!

jta.append("ThreadServerConsole 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** ClientHandler(socket));

task.start();

clientNo++;

}

}

}

## Class Client:

**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** **void** writeToServer(CHANGE 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 ClientHandler:

**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 {

**private** Socket socket; // A connected socket

**private** ObjectInputStream inputFromClient;

**private** ObjectOutputStream outputToClient;

**private** CHANGE inputObject = **new** CHANGE();

**private** CHANGE outputObject = **new** CHANGE();

**private** Vector currencyVector = Server.*getCurrencyVector*();

**public** ClientHandler(Socket socket) {

**this**.socket = socket;

/\*

\* currencyVector.add(1.0); // Shekel currencyVector.add(3.83320521); // euro

\* currencyVector.add(3.46291564); // dollar currencyVector.add(4.48631109); //

\* British Pounds

\*/

}

**public** CHANGE getinputObject() {

**return** inputObject;

}

**public** CHANGE 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 = (CHANGE) inputFromClient.readObject();

System.***out***.println(inputObject + "in server");

} **catch** (ClassNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

// Compute currency change

outputObject = currencyConverter(inputObject);

// System.out.print(amountB+"\n");

// Send converted money back to the client

outputToClient.writeObject(inputObject);

}

} **catch** (IOException e) {

System.***err***.println(e);

}

}

**synchronized** **private** CHANGE currencyConverter(CHANGE inputObject) {

**int** currencyA = inputObject.getCurrencyA(), currencyB = inputObject.getCurrencyB();

**double** amountA = inputObject.getAmountA(), amountB = 0;

System.***out***.print("Amount A " + amountA + "\n");

System.***out***.print(" Amount B" + amountB + "\n");

System.***out***.print(" Currency A " + currencyA + "\n");

System.***out***.print(" cureency B " + currencyB + "\n");

**if** (currencyA == 0) // original currency is SHEKEL

amountB = (amountA / ((Double) (currencyVector.elementAt(currencyB))).doubleValue());

**else** // convert original currency to SHEKEL than convert it to the requested new

// currency

amountB = (amountA \* (Double) currencyVector.elementAt(currencyA)

/ ((Double) (currencyVector.elementAt(currencyB))).doubleValue());

inputObject.setAmountB(amountB);

**return** inputObject;

}

}

## Class ClientGUI:

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** java.io.\*;

**import** java.net.\*;

**import** javax.swing.\*;

**public** **class** ClientGUI **extends** JFrame **implements** ActionListener {

**private** **static** String[] *currencystr* = { "₪", "€", "$", "₤" };

**private** CHANGE object = **new** CHANGE(); // object to be sent for server handling

**private** JLabel lblTitle = **new** JLabel("Currency CHANGE");

**private** JLabel lbl1 = **new** JLabel("From");

**private** JLabel lbl2 = **new** JLabel("To");

**private** JTextField txtFrom = **new** JTextField();

**private** JTextField txtTo = **new** JTextField();

**private** JButton btnSend = **new** JButton("Send");

**private** JComboBox cbox1, cbox2;

**private** Client client = **new** Client();

**public** **static** **void** main(String[] args) {

**new** ClientGUI();

}

**public** ClientGUI() {

JPanel pMain = **new** JPanel(), p1 = **new** JPanel(), p2 = **new** JPanel(), p3 = **new** JPanel();

pMain.setLayout(**new** GridLayout(0, 1));

p1.setLayout(**new** GridLayout(1, 1));

lblTitle.setFont(**new** Font("Arial Bold", 0, 20)); // resize text

lblTitle.setHorizontalAlignment(SwingConstants.***CENTER***);

p1.add(lblTitle, BorderLayout.***NORTH***);

p2.setLayout(**new** FlowLayout());

p2.add(lbl1);

cbox1 = **new** JComboBox(*currencystr*);

cbox1.addActionListener(**this**);

p2.add(cbox1);

p2.add(txtFrom);

p2.add(lbl2);

cbox2 = **new** JComboBox(*currencystr*);

cbox2.addActionListener(**this**);

p2.add(cbox2);

p2.add(txtTo);

txtFrom.setHorizontalAlignment(JTextField.***LEFT***);

txtFrom.setPreferredSize(**new** Dimension(100, 24));

txtTo.setHorizontalAlignment(JTextField.***LEFT***);

txtTo.setPreferredSize(**new** Dimension(100, 24));

p3.setLayout(**new** GridLayout(1, 1));

btnSend.setPreferredSize(**new** Dimension(200, 50));

p3.add(btnSend);

pMain.add(p1);

pMain.add(p2);

pMain.add(p3);

setContentPane(pMain);

btnSend.addActionListener(**new** ButtonListener());

setTitle("Client");

setSize(500, 300);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setVisible(**true**);

}

**private** **class** ButtonListener **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

// Send the CHANGE object to the server

object = **new** CHANGE();

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

// txtFrom.

object.setCurrencyA(cbox1.getSelectedIndex()); // getting currency inx.

object.setCurrencyB(cbox2.getSelectedIndex());

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

client.writeToServer(object);

// Get outputString from the server

CHANGE outputObject = (CHANGE) 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() == cbox1) {

// object.setAmountA(Integer.valueOf((txtFrom.getText().trim())));

object.setCurrencyA(cbox1.getSelectedIndex()); // getting currency inx.

}

**if** (e.getSource() == cbox2)

object.setCurrencyB(cbox2.getSelectedIndex());

}

}

## Class CHANGE:

**import** java.io.Serializable;

/\*\*

\* **@author** amits This class represents an instance used to send data across the

\* client/server network.

\*/

**public** **final** **class** CHANGE **implements** Serializable {

**private** **int** currencyA, currencyB;

**private** **double** amountA, amountB;

/\*\*

\* Empty constructor for the class parameters.

\*/

**public** CHANGE() {

**super**();

**this**.currencyA = 0; // index in currencyVector and the same in currrencystr

**this**.amountA = 0;

**this**.currencyB = 0; // index in currencyVector and the same in currrencystr

**this**.amountB = 0;

}

**public** **int** getCurrencyA() {

**return** currencyA;

}

**public** **void** setCurrencyA(**int** currencyA) {

**this**.currencyA = currencyA;

}

**public** **int** getCurrencyB() {

**return** currencyB;

}

**public** **void** setCurrencyB(**int** currencyB) {

**this**.currencyB = currencyB;

}

**public** **double** getAmountA() {

**return** amountA;

}

**public** **void** setAmountA(**double** amountA) {

**this**.amountA = amountA;

}

**public** **double** getAmountB() {

**return** amountB;

}

**public** **void** setAmountB(**double** amountB) {

**this**.amountB = amountB;

}

@Override

**public** String toString() {

**return** "CHANGE [currencyA=" + currencyA + ", currencyB=" + currencyB + ", amountA=" + amountA + ", amountB="

+ amountB + "]";

}

}

# ביבליוגרפיה:

[/https://stackoverflow.com](https://stackoverflow.com/) - עזרה עם הקוד ותשובות לבאגים ובעיות.