



**אורט בראודה**  
ORT Braude College

# שיטות הנדסיות לפיתוח מערכות תוכנה

Object Client Server Framework (OCSF)

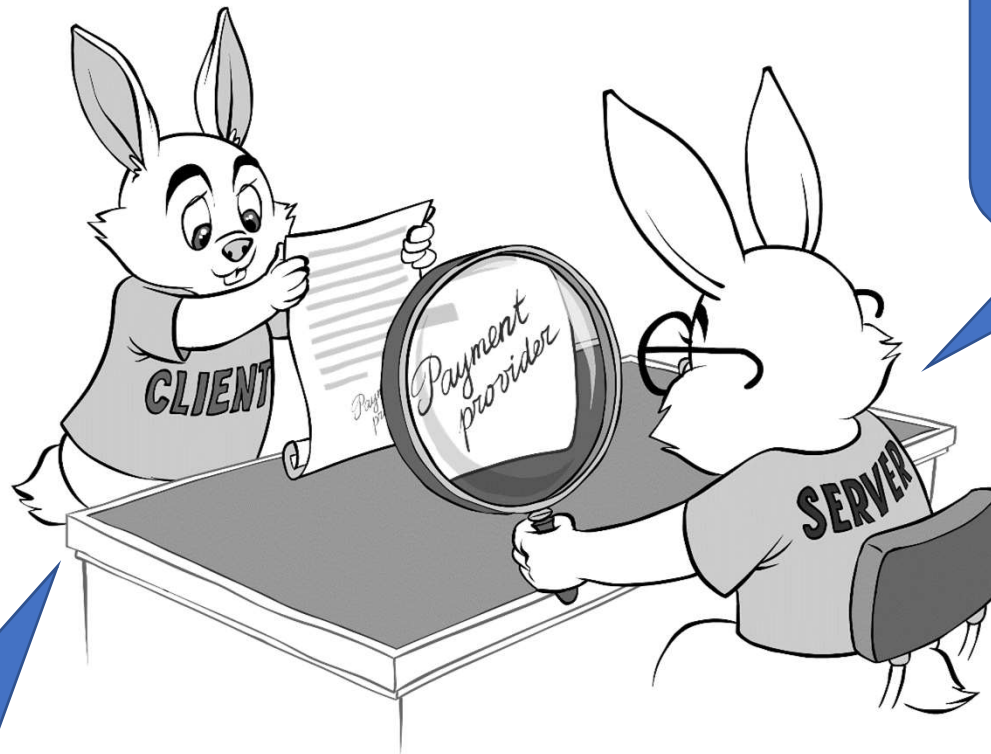
1. מתחילים להסתכל מקרוב על Design Pattern (תבניות תכן)

2. השיעור נתמקד בארכיטקטורת Client-Server בכלל וב-  
**Object Client Server Framework (OCSF)** בפרט

3. כמה דוגמאות:

- The World Wide Web
- Email
- Transaction Processing System
- Remote Display System
- Database System

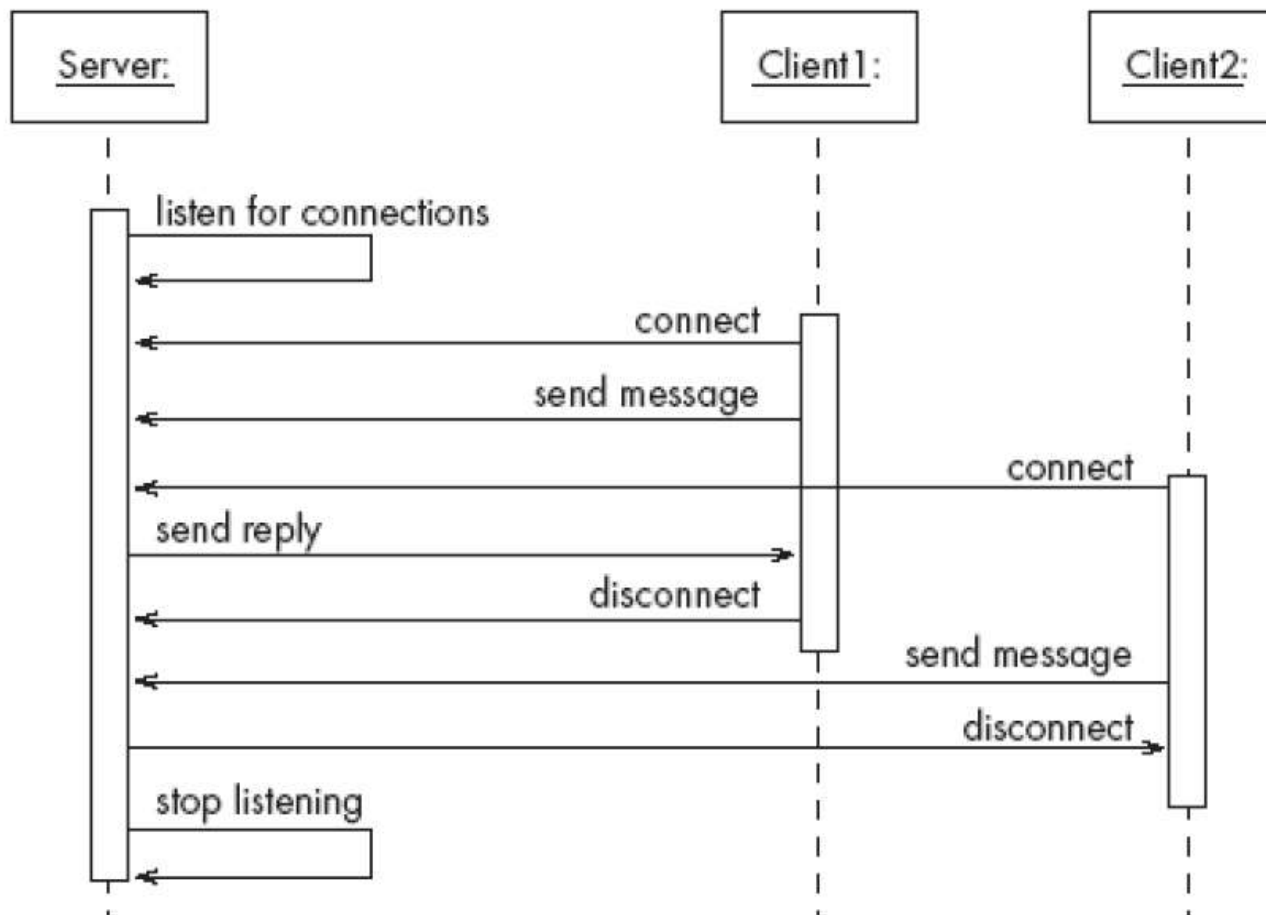




תוכנית המספקת  
שירות עבור תוכניות  
אחרות המתחברות  
אליה באמצעות  
ערוץ תקשורת

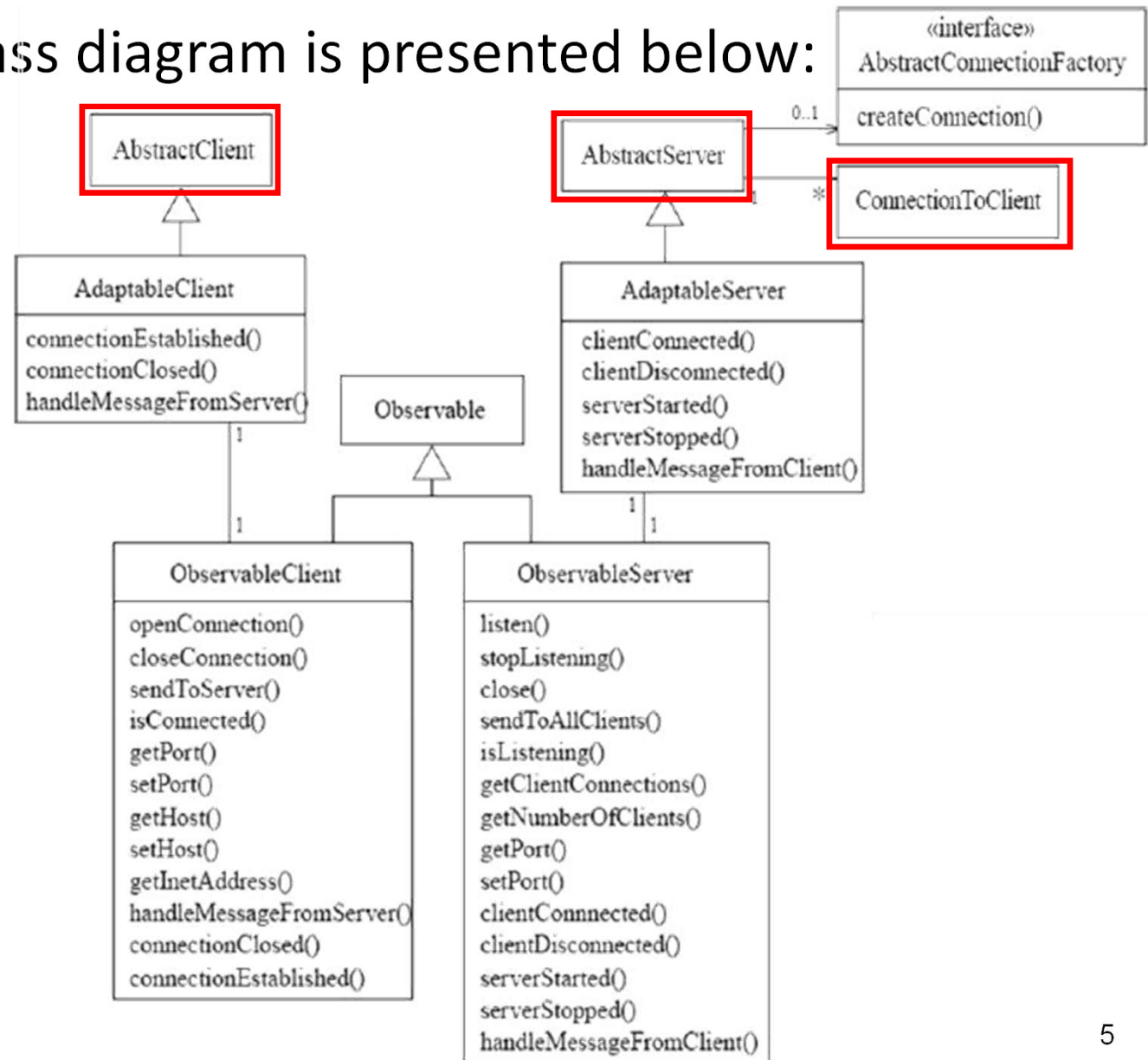
תוכנית שניגשת  
לשרת (או מספר  
שרתים) כדי לקבל  
שירותים

דוגמא לתקשורת בין שרת לשני לקוחות:



The general class diagram is presented below:

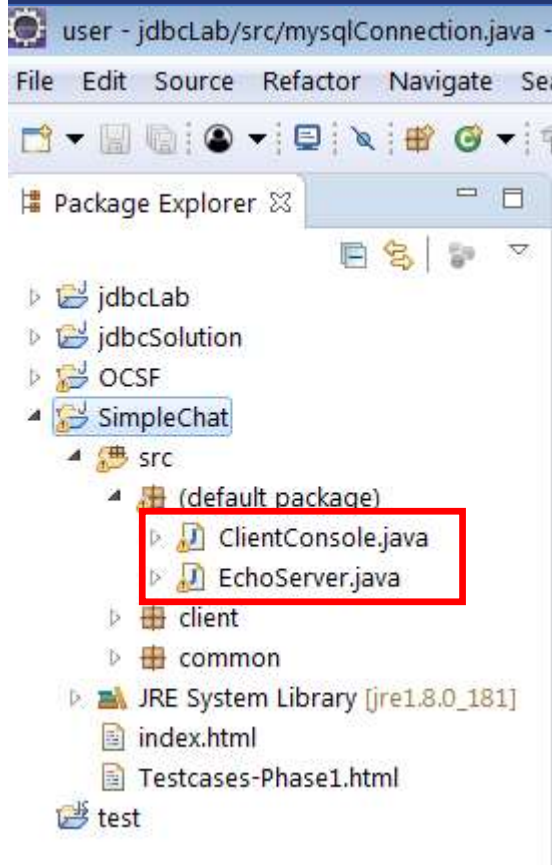
Never modify  
these three  
classes!!!



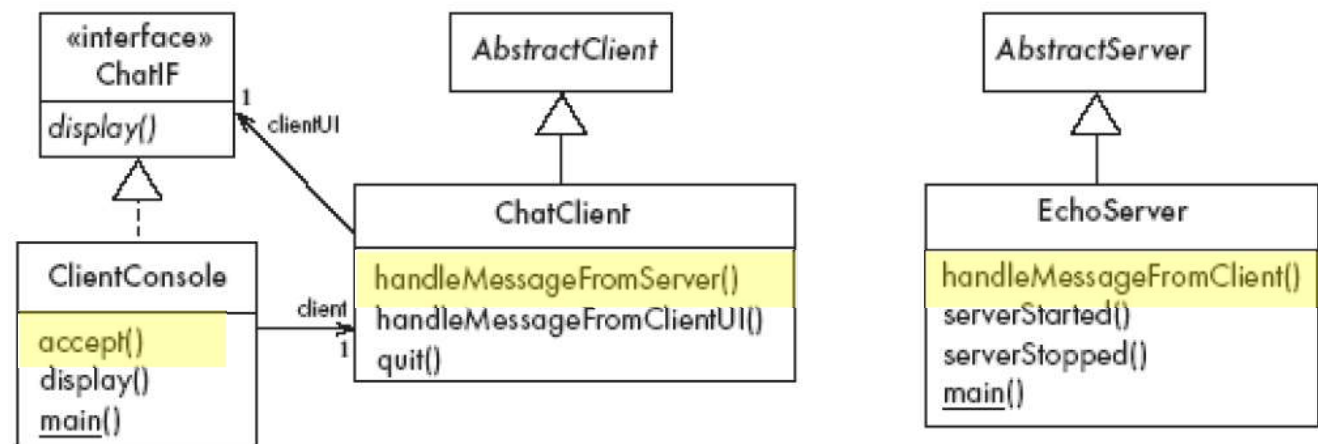


יש קלאס אחד:  
AbstractClient

יש שני קלאסים:  
AbstractServer  
ConnectionToClient

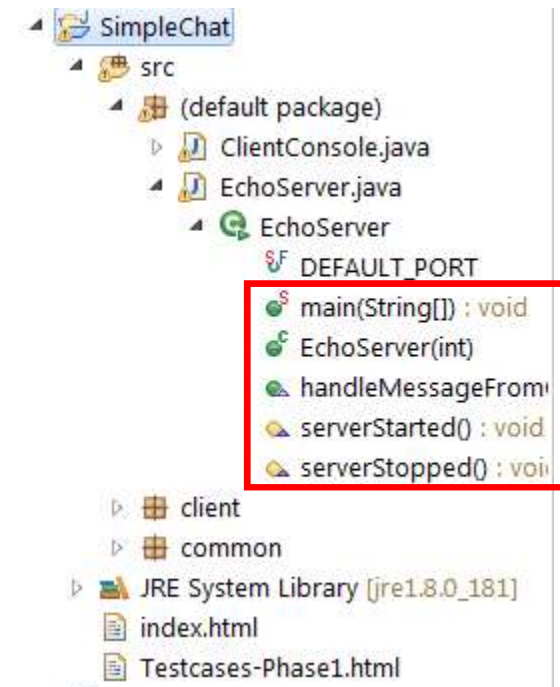
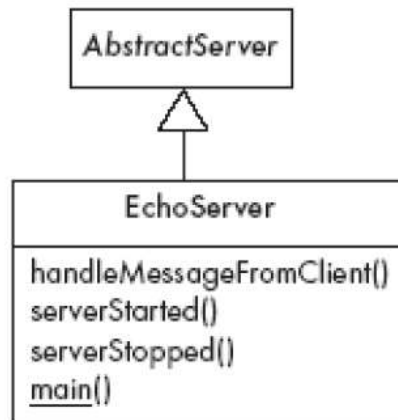


## דוגמא לאפליקציית OCSF:



ClientConsole can eventually be replaced by ClientGUI

:EchoServer

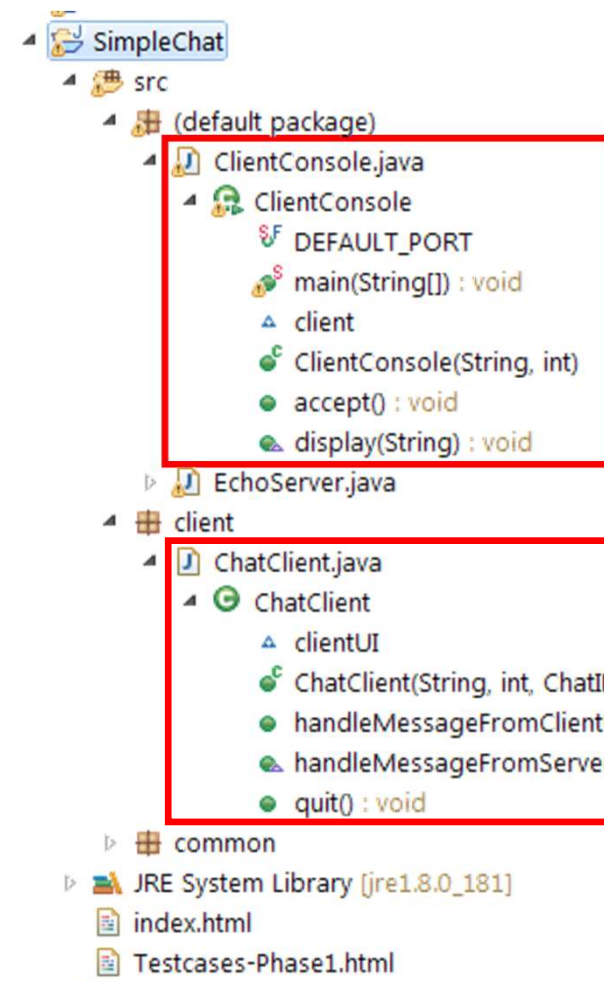
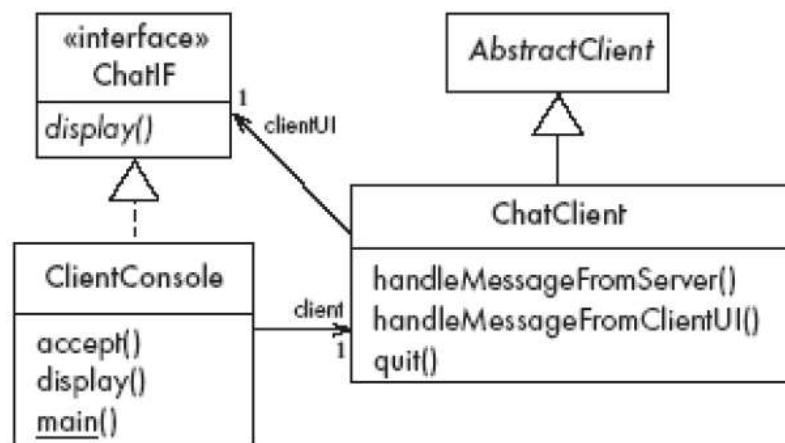




## קטע קוד מרכזי ב-EchoServer:

```
/**
 * This method handles any messages received from the client.
 *
 * @param msg The message received from the client.
 * @param client The connection from which the message originated.
 */
public void handleMessageFromClient
(Object msg, ConnectionToClient client)
{
    System.out.println("Message received: " + msg + " from " + client);
    this.sendToAllClients(msg);
}
```

:ChatClient



## קטע קוד מרכזי ב-ChatClient:

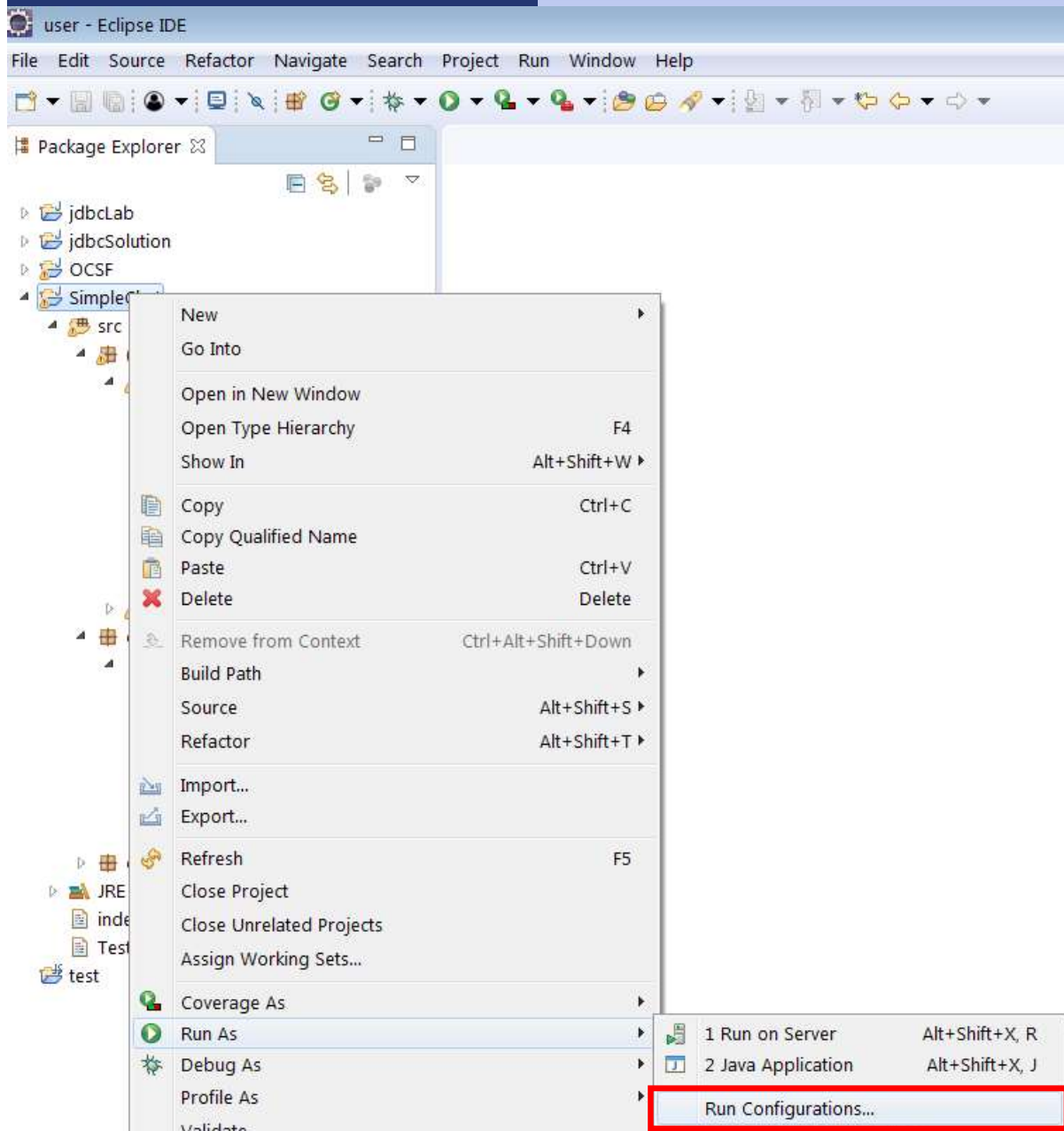
```
/**
 * This method handles all data that comes in from the server.
 *
 * @param msg The message from the server.
 */
public void handleMessageFromServer(Object msg)
{
    clientUI.display(msg.toString());
}

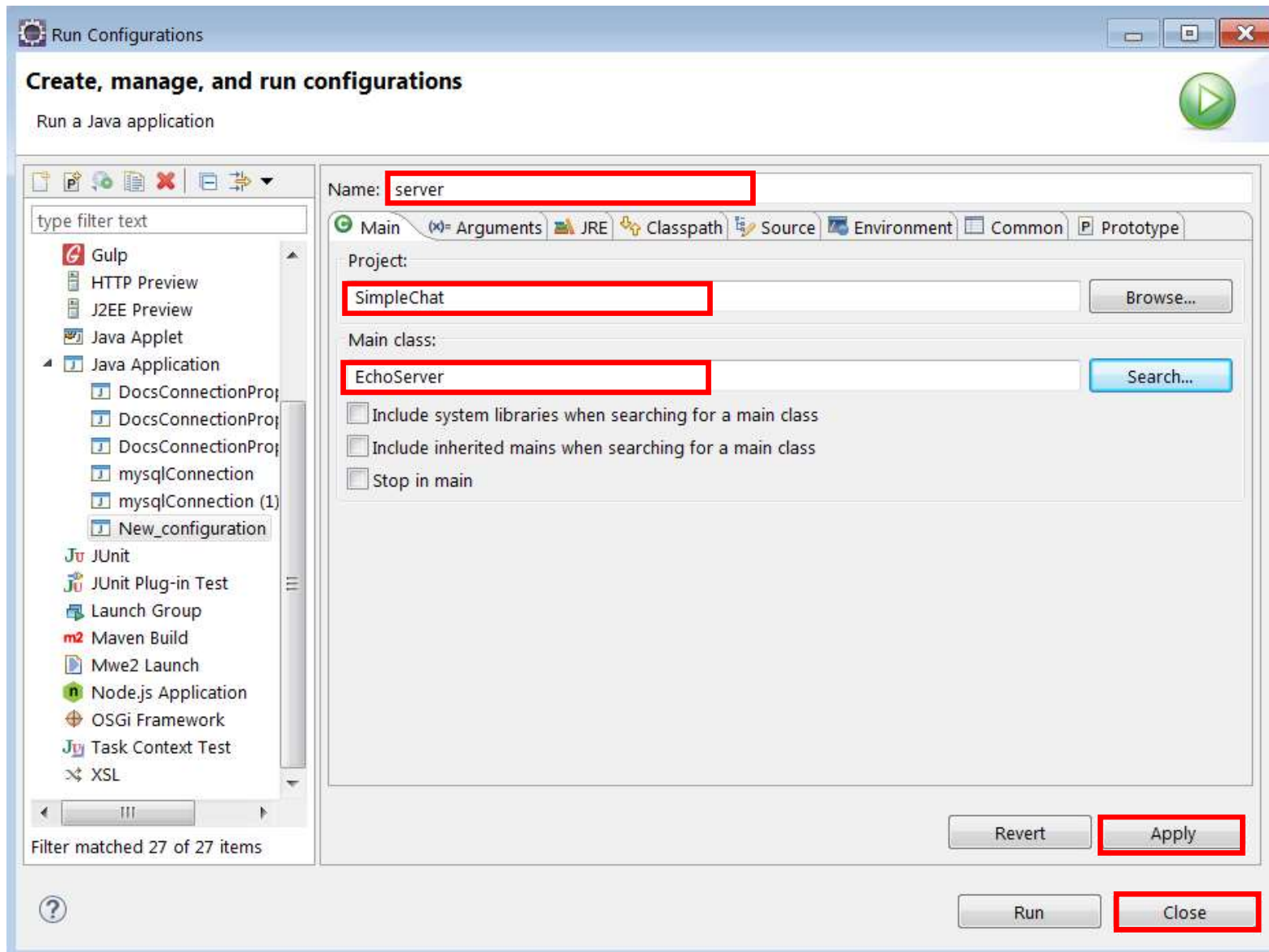
/**
 * This method handles all data coming from the UI
 *
 * @param message The message from the UI.
 */
public void handleMessageFromClientUI(String message)
{
    try
    {
        sendToServer(message);
    }
    catch(IOException e)
    {
        clientUI.display
            ("Could not send message to server. Terminating client.");
        quit();
    }
}
```

## קטע קוד מרכזי ב-ChatConsole:

```
ChatClient.java EchoServer.java *ClientConsole.java ✖
63 //Instance methods *****
64
65 /**
66  * This method waits for input from the console. Once it is
67  * received, it sends it to the client's message handler.
68  */
69 public void accept()
70 {
71     try
72     {
73         BufferedReader fromConsole =
74             new BufferedReader(new InputStreamReader(System.in));
75         String message;
76
77         while (true)
78         {
79             message = fromConsole.readLine();
80             client.handleMessageFromClientUI(message);
81         }
82     }
83     catch (Exception ex)
84     {
85         System.out.println
86             ("Unexpected error while reading from console!");
87     }
88 }
```

נגדיר את השרת  
והלקוח:

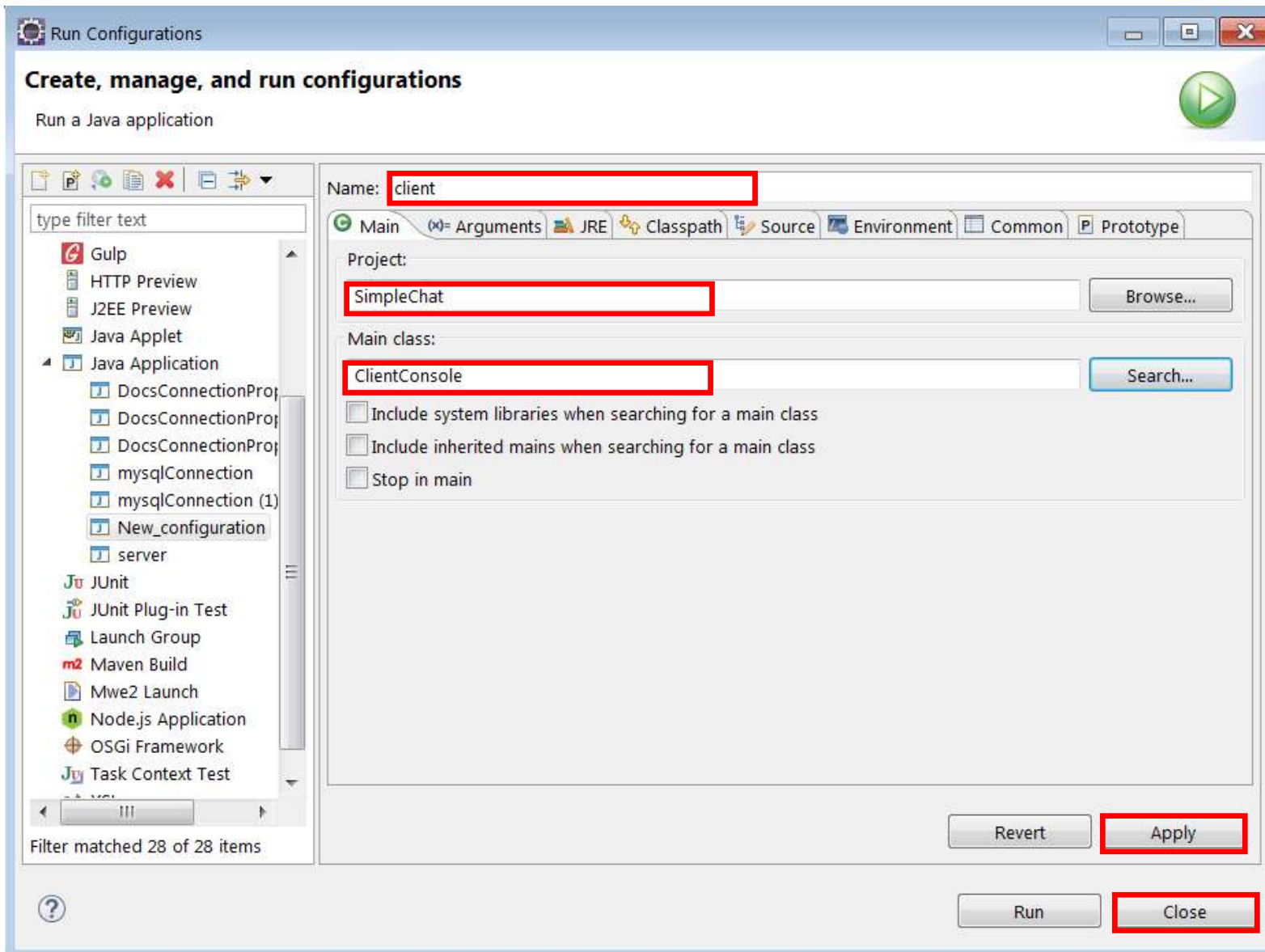




השרת:



הלקוח:



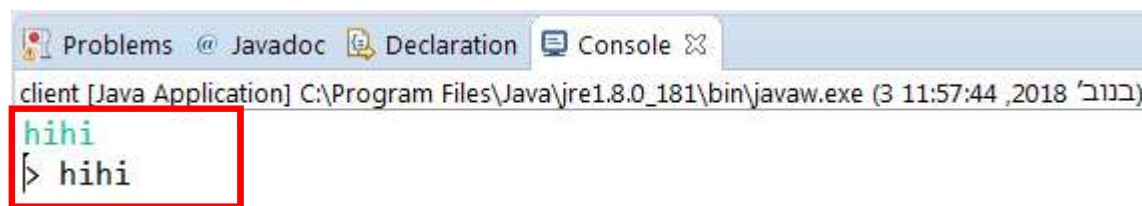
נריץ את השרת:

The screenshot illustrates the steps to run a Java application in Eclipse IDE:

- Package Explorer:** Shows the project structure with 'SimpleChat' selected.
- Context Menu:** Right-clicked on 'SimpleChat', showing options like 'New', 'Go Into', 'Copy', 'Paste', 'Delete', 'Run As', etc.
- Run As Submenu:** The 'Run As' option is expanded, showing '1 Run on Server' and '2 Java Application' (highlighted with a red box).
- Select Java Application Dialog:** A dialog box titled 'Select Java Application' is open, showing 'EchoServer - (default package)' selected under 'Matching items'.
- Console Output:** The console shows the output of the Java application: 'server [Java Application] C:\Program Files\Java\jre1.8.0\_181\bin\javaw.exe (3 11:54:51, 2018 ב'נוב' )' and 'Server listening for connections on port 5555' (highlighted with a red box).
- SimpleChat\bin Dialog:** A dialog box titled 'SimpleChat\bin - (default package)' is open, showing the path to the application's bin directory.



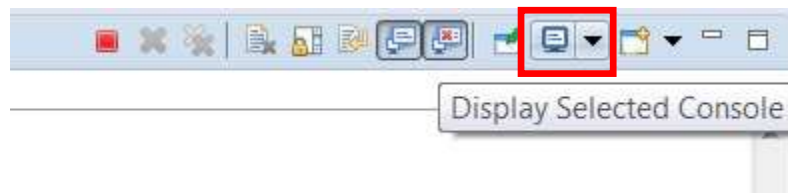
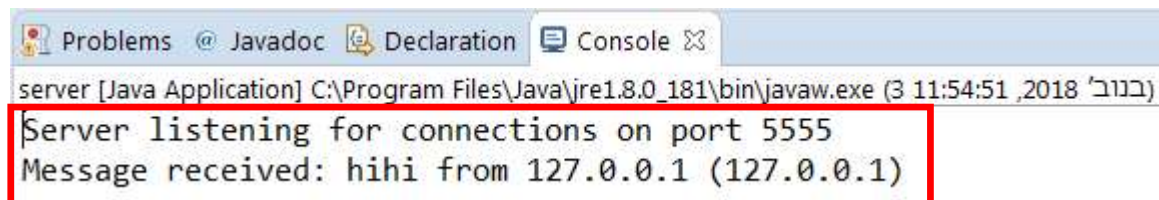
- נריץ את הלקוח באופן
- ה console של הלקוח מופיע תחילה ריק (לא מקשיב)  
✓ אפשר לכתוב hihi ולראות שהשרת יענה



client [Java Application] C:\Program Files\Java\jre1.8.0\_181\bin\javaw.exe (3 11:57:44 ,2018 'בנוב')

```
hihi
[> hihi
```

- אפשר לראות מה קורה בצד הלקוח ע"י פתיחת console נוסף ע"י לחיצה על:
- ובחירה ברלוונטי

server [Java Application] C:\Program Files\Java\jre1.8.0\_181\bin\javaw.exe (3 11:54:51 ,2018 'בנוב')

```
Server listening for connections on port 5555
Message received: hihi from 127.0.0.1 (127.0.0.1)
```

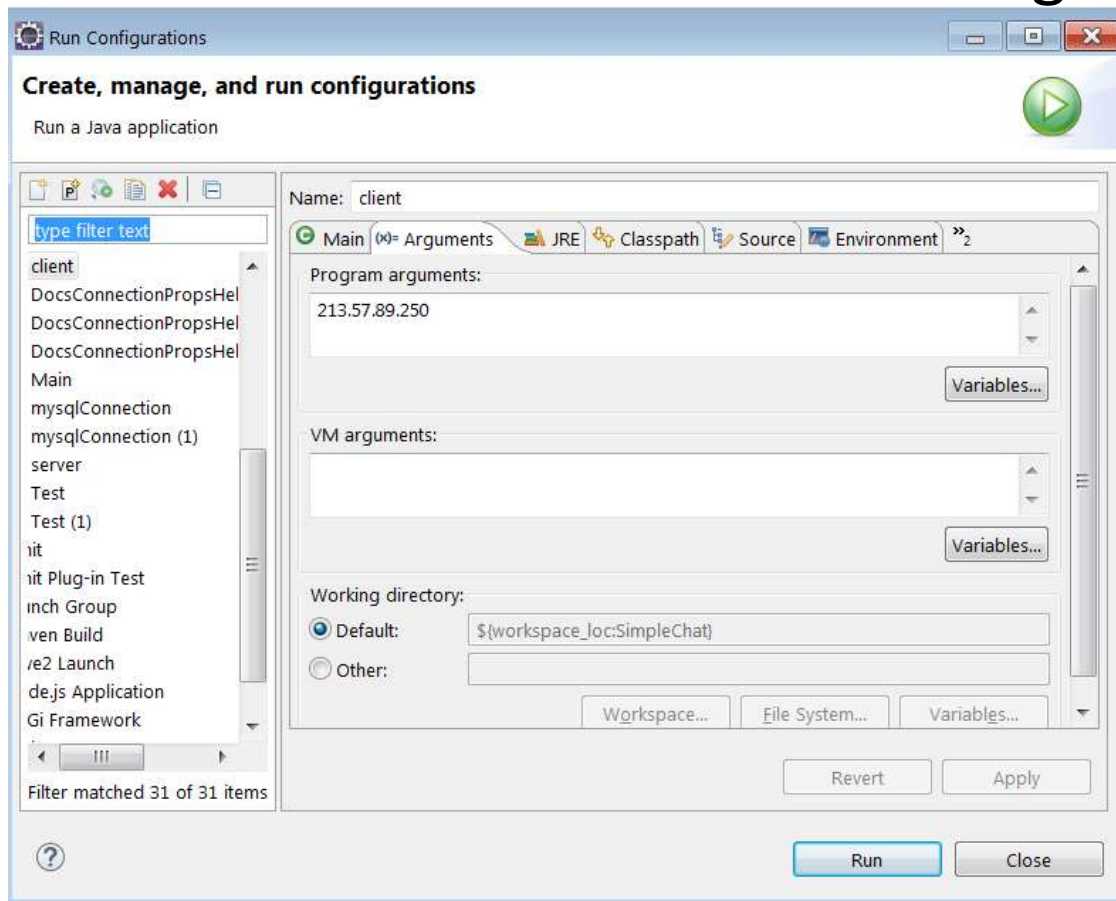
שימו לב:

- לאחר ביצוע שינויים בקוד של השרת, חשוב "להרוג" את השרת הישן (לפני הרצת החדש) ע"י לחיצה על:



כדי לתקשר עם שרת שנמצא במחשב אחר:

- יש להזין בצד הclient את כתובת IP של המחשב השני (server) בטאב Arguments:



1. Import attached projects (OCSF & SimpleChat – the file **Exercise\_06-Client-Server.zip**) to Eclipse.
2. Look first at the structure of the projects and understand the role of the various components.

**Note:** For help look at attached document  
*“Client-Server architecture.PDF”*.

3. Run appropriated Java programs (Client, Server) on one computer and be sure that simple message (string) is passed between Client – Server.
4. Run the client on another computer on the network and perform sending and inserting the data into DB.
5. Implement sending simple object ArrayList of strings from Client to Server.

For example: UserName (Bob), ID (123456),  
Department (VIF project), Tel. (7654321).

**Note:** The ArrayList has to be sent **after typing the key word “send”**.

6. On the Server part, implement inserting the data (ArrayList) to DB. An implementation has to be performed as several methods like:  
*parsingTheData(), connectToDB(), saveUserToDB().*

**Note:** For help see solution of previous exercise (JDBC) => [Add class that connects to MySQL](#)

7. Present the solution to a lecturer.

## Working with GUI: JavaFX

