

תרגיל בית 2

MultiThreading in Networking

מועד הגשה: 29.12.2025

דרישות התרגיל –

התרגיל עוסק בנושא תקשורת שרת – לקוח בשפת JAVA תוך שימוש בפרוטוקול תקשורת מוגדר מראש + הרחבה לפרוטוקול נוסף + תמיכה בריבוי לקוחות + שמירה לקובץ.

אפיון מקוצר לתרגיל "Knock Knock" שהקוד שלו נתון עבורכם (קבצי הקוד מצורפים לתיבת הגשה תחת הקובץ exe3KnockKnock.zip או בתחתית הקובץ לנוחיותכם):

- צד הלקוח - מחלקת KnockKnockClient
 - הלקוח יוצר חיבור לשרת, שולח קלט מהמשתמש ומדפיס את התשובה שמתקבלת מהשרת
- צד השרת – מחלקת KnockKnockServer
 - השרת מאזין לפורט 4444, מקבל חיבור של לקוח אחד ומנהל עמו שיחה
 - השרת עצמו לא מכיל לוגיקה עסקית של תקשורת עם הלקוח.
 - מקבל קלט מהלקוח
 - מעביר את הקלט למתודה ProcessInput שקיימת במחלקת KnockKnockProtocol.
 - את התשובה שמתקבלת הוא מחזיר ללקוח
- פרוטוקול התקשורת המוגדר – KnockKnockProtocol
 - הפרוטוקול מנהל **שיחה מובנת** בין השרת ללקוח. מקבל קלט ועונה לפי State Machine מצב פנימי מוגדר במחלקה.
- יש לזכור קודם כל יש להריץ את תוכנית השרת ורק אחר כך את הלקוח. נסו להבין את הקוד הנתון לעומק טרם השלמת המשימות
- **אם יש פורטים שנכתבו בתרגיל והם נעולים ע"י מערכת ההפעלה. ניתן לשנות פורטים רק ציינו זאת בהגשה.**

המשימות –

1. יש לשנות את הקוד הנתון כך שיתמוך **בריבוי לקוחות במקביל**. השרת ימשיך להאזין לפורט 4444. יטפל בכל לקוח שמתחבר בthread נפרד. אין לאפשר מצב שבו לקוח אחד חוסם לקוחות אחרים. יש לממש מחלקה ייעודית לטיפול בלקוח (למשל – clientHandler המממשת ממשק Runnable או Thread. יורשת את מחלקת Thread.

2. על בסיס המחלקות הנתונות + השרת החדש שתומך בריבוי לקוחות במקביל : יש לכתוב תוכנית שמוסיפה עוד פרוטוקול תקשורת עם שרת בשם RuppinRegistrationProtocol הפרוטוקול החדש יעבוד מול שרת המאזין לפורט 4445. בחירת הפרוטוקול תתבצע בקוד השרת עצמו להחלטתכם איך השרת יקבע את הפרוטוקול תקשורת KnockKnockProtocol, או RuppinRegistrationProtocol **(הסבירו בקובץ PDF שאתם מגישים את החלטתכם)** בפרוטוקול החדש תקשורת נראית כך – תרחיש תקשורת משתמש לא קיים (קלטים תקינים):

```
Server: Do you want to register? (yes/no)
Client: yes
Server: Enter a username:
Client: Rina
Server: Checking name...
Server: OK. Enter a strong password:
Client: 1234aBB
Server: Password accepted.
Server: What is your academic status? (student/teacher/other)
Client: student
Server: How many years have you been at Ruppin?
Client: 2
Server: Registration complete.
```

לאחר שליחת ההודעה האחרונה, השרת מנתק את החיבור עם הלקוח. **(חשבו איך לעשות את זה. יש להסביר בקובץ PDF שאתם מגישים)**

תרחיש תקשורת – משתמש קיים (קלטים תקינים):

```
Server: Do you want to register? (yes/no)
Client: no
Server: Username:
Client: Rina
Server: Password:
Client: 1234aBB
Server: Welcome back, Rina.
Server: Last time you defined yourself as student for 2 years.
Server: Do you want to update your information? (yes/no)
```

- במקרה והלקוח עונה no השרת שולח הודעת סיום ומתנתק
- במקרה והלקוח עונה yes יש לקלוט מחדש את הנתונים, לעדכן את מצב המשתמש ולסיים.

```
Server: Do you want to change your username? (yes/no)
Client: yes
Server: Enter new username:
Client: RinaZ
Server: Username updated successfully.
Server: Do you want to change your password? (yes/no)
Client: yes
Server: Enter new password:
Client: NewPass9a
Server: Password updated successfully.
Server: Do you want to update your years of study? (yes/no)
Client: yes
Server: Enter number of years:
Client: 3
Server: Years of study updated successfully.
Server: Thanks. Your information has been updated.
```

בכל אחת מן השאלות אם המשתמש עונה no יש להמשיך לשאול אם ירצה לעדכן את שאר הפרטים.

עליכם להוסיף:

- יש לוודא ששמות הלקוחות ייחודיים (לא חוזרים על עצמם).
דוגמה לתגובה של השרת במקרה של כפילות:
Server: Name not OK. Username exists. Choose a different name:
- לגבי סיסמא יש לבדוק שמכילה:
 - לפחות אות גדולה אחת
 - לפחות אות קטנה
 - לפחות ספרה אחת
 - אורך הסיסמא לפחות 9 תווים.← במקרה שסיסמא לא תקינה השרת ישלח תגובה מתאימה לבחירתכם.
- יש ליצור מחלקה Client שמייצגת לקוח יש להסביר בקובץ PDF שאתם מגישים, מה יהיה (בה (שדות, בנאים, מתודות) יש לממש מתודה equals שמתודה contains תעבוד.
- כל פעם שלקוח פונה לשרת יש לשמור את מצבו – מכיון ששרת לא מתנתק אף פעם יש להשתמש ברשימת הלקוחות בשם clientState. למשל:
List<Client> clientState
 - הרשימה תשמש את השרת לצורך אימות משתמשים קיימים, הוספת משתמשים חדשים ומניעת כפילות במשתמשים, עדכון פרטי משתמשים. לשם כך הגדירו מתודות רצויות (יש לציין הסבר למתודות בקובץ PDF שאתם מגישים)
- ניתן להוסיף מחלקות, מתודות כרצונכם לפי הצורך.
- סעיף יצירתיות (שימוש בכלי AI) כל פעם שמתווסף לקוח חדש ומספר הלקוחות ברשימה מתחלק ב 3 (3,6,9 וכו') יש לשמור נתוני הלקוחות לקובץ גיבוי (קובץ מסוג CSV שם הקובץ backup_DATE.csv כאשר DATE יוחלף ביום ושעה של יצירת הקובץ
 - קובץ csv זהו קובץ שנתוניו נשמרים כטקסט. כל פריט מידע בקובץ מופרד בתו פסיק ';'
 - להגשה של הPDF יש לצרף צילומי מסך של תיקיה בה יהיו לפחות 2 קבצים (ז"א יצרתם לפחות 6 לקוחות).
 - להגשה של הPDF לצרף צילום תוכן של אחד הקבצים.

הנחיות הגשה:

יש להגיש הכל דרך מערכת ה"מודול" באופן הבא:

- קובץ St1ID_st2ID_hw2.pdf שבתחילתו יש לרשום שמות המגישים, מספר ת.ז. לינק לgithub שמכיל את תיקיית הפרויקט.
 - יש להוסיף פירוט מפורט של המחלקות, שדות, מתודות שהוספתם במשימה 2. הוסיפו לקובץ זה גם את ההסברים שדרשנו באדום להוסיף.
 - יש לצרף העתקים של הקוד מקבצי JAVA של כל אחת מן המחלקות.
 - יש לצרף צילומי מסך של השיחה גם בצד השרת וגם בצד הלקוח. של לפחות שני לקוחות
 - יש לארוז כל הקבצים בקובץ זיפ הנקרא St1ID_st2ID_hw2.zip ולשלוח במודול.
 - הקובץ זיפ St1ID_st2ID_hw2.zip יראה באופן הבא:
 - קבצי JAVA של כל המחלקות שיצרתם
 - קבצי csv שיצרתם במטלה
- יש להגיש בזוגות. רק אחד מבני הזוג יצרף את הקובץ ל"מודל".
-

```
import java.io.*;
import java.net.*;

public class KnockKnockClient {
    public static void main(String[] args) throws IOException {

        Socket kkSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try {
            kkSocket = new Socket("127.0.0.1", 4444);
            out = new PrintWriter(kkSocket.getOutputStream(), true);
            in = new BufferedReader(new
InputStreamReader(kkSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: your host.");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection
to: your host.");
            System.exit(1);
        }

        BufferedReader stdIn = new BufferedReader(new
InputStreamReader(System.in));
        String fromServer;
        String fromUser;

        while ((fromServer = in.readLine()) != null) {
            System.out.println("Server: " + fromServer);
            if (fromServer.equals("Bye."))
                break;

            fromUser = stdIn.readLine();
            if (fromUser != null) {
                System.out.println("Client: " + fromUser);
                out.println(fromUser);
            }
        }

        out.close();
        in.close();
        stdIn.close();
        kkSocket.close();
    }
}
```

```
import java.net.*;
import java.io.*;

public class KnockKnockServer {
    public static void main(String[] args) throws IOException {

        ServerSocket serverSocket = null;
        try
        {
            serverSocket = new ServerSocket(4444);
        }
        catch (IOException e)
        {
            System.err.println("Could not listen on port: 4444.");
            System.exit(1);
        }

        while(true)
        {
            Socket clientSocket = null;
            try {
                clientSocket = serverSocket.accept();
            } catch (IOException e) {
                System.err.println("Accept failed.");
                System.exit(1);
            }

            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(),
true);
            BufferedReader in = new BufferedReader(
                new InputStreamReader(
                    clientSocket.getInputStream()));
            String inputLine, outputLine;
            KnockKnockProtocol kkp = new KnockKnockProtocol();

            outputLine = kkp.processInput(null);
            out.println(outputLine);

            while ((inputLine = in.readLine()) != null) {
                if (inputLine.equals("q")) break;
                outputLine = kkp.processInput(inputLine);
                out.println(outputLine);
            }
            out.close();
            in.close();
            clientSocket.close();
        }
    }
}
```

נושאים מתקדמים בתכנות מונחה עצמים
מרצה: ד"ר רינה צביאל-גירשין
מתרגלת: מעיין זנו

```
import java.net.*;
import java.io.*;

public class KnockKnockProtocol {
    private static final int WAITING = 0;
    private static final int SENTKNOCKKNOCK = 1;
    private static final int SENTCLUE = 2;
    private static final int ANOTHER = 3;

    private static final int NUMJOKES = 5;

    private int state = WAITING;
    private int currentJoke = 0;

    private String[] clues = { "Turnip", "Little Old Lady", "Atch", "Who",
"Who" };
    private String[] answers = { "Turnip the heat, it's cold in here!", "I
didn't know you could yodel!", "Bless you!",
    "Is there an owl in here?", "Is there an echo in here?" };

    public String processInput(String theInput) {
        String theOutput = null;

        if (state == WAITING) {
            theOutput = "Knock! Knock!";
            state = SENTKNOCKKNOCK;
        }
        else if (state == SENTKNOCKKNOCK) {
            if (theInput.equalsIgnoreCase("Who's there?")) {
                theOutput = clues[currentJoke];
                state = SENTCLUE;
            } else {
                theOutput = "You're supposed to say \"Who's
there?\"! " + "Try again. Knock! Knock!";
            }
        }
        else if (state == SENTCLUE) {
            if (theInput.equalsIgnoreCase(clues[currentJoke] + "
who?")) {
                theOutput = answers[currentJoke] + " Want another?
(y/n)";
                state = ANOTHER;
            } else {
                theOutput = "You're supposed to say \"\" +
clues[currentJoke] + " who?\"\" + "! Try again. Knock! Knock!";
                state = SENTKNOCKKNOCK;
            }
        }
        else if (state == ANOTHER) {
            if (theInput.equalsIgnoreCase("y")) {
                theOutput = "Knock! Knock!";
                if (currentJoke == (NUMJOKES - 1))
                    currentJoke = 0;
            }
        }
    }
}
```


נושאים מתקדמים בתכנות מונחה עצמים

מרצה: ד"ר רינה צביאל-גירשין

מתרגלת: מעיין זנו

```
        else
            currentJoke++;
            state = SENTKNOCKKNOCK;
    } else {
        theOutput = "Bye.";
        state = WAITING;
    }
}
return theOutput;
}
}
```