

## תרגיל בית 2

### MultiThreading in Networking

מועד הגשה: 29.12.2025

#### דרישות התרגיל –

התרגיל עוסק בנושא תקשורת שרת – ל��וח בשפת JAVA תוך שימוש בפרוטוקול תקשורת מוגדר מראש + הרחבת פרוטוקול נוסף + תמייח בריבוי ל��וחות + שמירה לקובץ.

אפיון מקוצר לתרגיל "Knock Knock" שהקוד שלו ניתן עבורהם (קבצי הקוד מצורפים לתיבת הגשה תחת הקובץ exe3KnockKnock.zip) :

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

### המשימות –

1. יש לשנות את הקוד הנוכחי כך שתתאפשר **בריבוי ל��וחות במקביל**. השירות ימשיך להאזין לפורט 4444. יטפל בכל לקוח שמתחבר בthread נפרד. אין לאפשר מצב שבו לקוח אחד חוסם לקוחות אחרים. יש למשתמש מחלקה **יעודית לטיפול בלקוח** (למשל – clientHandler) הממסת משק **Runnable** או **Thread**.  
יורשת את **מחלקת Thread**.

2. על בסיס המחלקות הנתונות + השירות החדש שתומך בריבוי ל��וחות במקביל :  
יש לכתוב תוכנית שמוסיפה עוד פרוטוקול תקשורת עם שירות בשם RuppinRegistrationProtocol  
הפרוטוקול החדש יעבד מול שירות המאזין לפורט 4445.  
בחירת הפרוטוקול תבצע בקוד השירות עצמו להחלטתכם איך השירות יקבע את הפרוטוקול  
תקשורת RuppinRegistrationProtocol, KnockKnockProtocol, או (הסבירו בקובץ PDF שאתם  
**מגישים את החלטתכם**)

בפרוטוקול החדש תקשורת נראה כך – תרשיש תקשורת משתמש לא קיים (קלטיים תקין) :

```
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 הודעה האחורונה, השירות 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)

- במקורה והלkoוח עונה no השרת שולח הודעה סיום ומונתק.
- במקורה והלkoוח עונה yes יש לקלוט מחדש את הנתונים, לעדכן את מצב המשתמש ולסיטים.

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 so יש להמשיך לשאול אם ירצה לעדכן את שאר הפרטיהם.

עליכם להוסיף :

- יש לוודא ששמות הלוקוחות ייחודיים (לא חוזרים על עצמם).  
דוגמא לתגובה של השירות במקרה של כפילות:

Server: Name not OK. Username exists. Choose a different name:

- לגבי סיסמה יש לבדוק שמכילה:
  - לפחות אחת גזרה אחת
  - לפחות אחת קטנה
  - לפחות שורה אחת
  - לפחות 9 תווים.

← במקרה שסיסמא לא תקין השירות ישלח תגובה מתאימה לבחירתכם.

- יש לזכור מחלוקת Client שמייצגת לךו יש להסביר בקובץ PDF שאתם מגישים, מה היה  
בها (שדות, בנאים, מתחדשות) יש למשם מתחודה equals שמתודעה contains תעבוד.
- כל פעם שלוקוח פונה לשרת יש לשמור את מצבו – מכון שהשרת לא מتنתק אף פעם יש  
להשתמש ברשימה הלוקוחות בשם clientState. למשל :

List<Client> clientState

- הרשימה תנשש את השירות לצורך אימוט משתמשים קיימים, הוספה משתמשים  
חדים ומינית כפילות משתמשים, עדכון פרטי משתמשים. לשם כך הגדרו  
מתחדשות רצויות יש לציין הסבר למתחדשות בקובץ PDF שאתם מגישים)
- ניתן להוסיף מחלקות, מתחדשות כרצונכם לפי הצורך.
- **סעיף יצירתיות (שימוש במכשיר AI)** כל פעם שמתווסף לךו חדש ומספר הלוקוחות ברשימה  
מתחלק ב 3 (3,6,9 וכו') יש לשמור נתוני הלוקוחות לקובץ גיבוי (קובץ מסוג CSV שם הקובץ  
backup\_DATE.csv כאשר DATE יוחלף ביום ושעה של יצירת הקובץ)
- קובץ csv זהו קובץ שנותני נשמרים כתקסט. כל פריט מידע בקובץ מופרד בטו פסיק  
,
- להגשה של הPDF יש לצרף צילומי מסך של תינוקה בה יהיה לפחות 2 קבצים (זיה  
יצרתם לפחות 6 לוקוחות).
- להגשה של הPDF לצרף צילום תוכן של אחד הקבצים.

### הנחיות הגשה:

יש להגיש הכל דרך מערכת "מודול" באופן הבא :

- קובץ St1ID\_st2ID\_hw2.pdf שבתחליתו יש לרשום שמות המציגים, מס' טז. לינק/github שמכליל את תיקייתuproject.
  - יש להוסיף פירוט מפורט של המחלקות, שדות, מתודות שהוספות במשימה 2.  
הויסיפו לקובץ זה גם את ההסברים שדרשו באודום להוסיף.
  - יש לצרף העתקים של הקוד מקבצי JAVA של כל אחת מן המחלקות.
  - יש לצרף צילומי מסך של השיחה גם **בצד הלכנו**. של לפחות שני  
לקוחות
  - יש לאפשר 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 שייצרתם
    - קבצי csv שייצרתם במתלה
- יש להגיש בזוג. רק אחד מבני הזוג יצרף את הקובץ ל"מודול".
-

```

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();
        }
    }
}

```

```

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: מעין זנו

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