מסמך דרישות – פרויקט מסכם בקורס תכנות בסביבת האינטרנט 2021

סטודנטים יקרים,

- מסמך זה מכיל את הדרישות לפרויקט המסכם.
- לרשותכם ב-moodle קבצי המקור של הפרויקטים שכתבנו במהלך הסמסטר וחומרי עזר נוספים.
- סטודנטים שצפו בשיעורים ויעשו חזרה על הקוד שכתבנו יחד יכולים לסיים את הפרויקט תוך פרק זמן קצר.
- לרשותכם מעל לחודש להצגת הפרויקט.
- **שימו לב – במהלך הסמסטר פיתחנו מערכות תוכנה עם פונקציונאליות דומה מאוד לזו הנדרשת מכם בפרויקט המסכם ובחלק מהדרישות, עד כדי התאמה של קוד קיים.**

תמצית: שרת אשר יכול לבצע פעולות אלגוריתמיות שונות בהתאם לסוג של ה-task שהתקבל. הפעולות השונות והטיפול בלקוחות בשרת עשויים להתבצע במקביל (concurrently). אתם רשאים להיעזר ב-multithreaded server שמימשנו. עליכם להשתמש בעקרונות OOD (Object Oriented Design) ו-Multithreading שלמדנו במהלך הסמסטר.

- נעסוק במתן מענה לבעיות אלגוריתמיות, בדומה לפרויקטים שכתבנו יחד במהלך הסמסטר.
- כל בעיה היא משימה שהמערכת צריכה לבצע ולהחזיר את תוצאת החישוב ללקוח.
- **אתם רשאים להשתמש בקוד של PriorityTask<V> (כולל הרחבה/שינוי).**
- **אתם רשאים להשתמש במחלקה Node<T> ובכל מנגנוני בהם השתמשנו שכתבנו במהלך הסמסטר**
- המשימות מופיעות בסיפא של המסמך לאחר רשימת ההנחיות המלאה.
- הבעיות הספציפיות יועברו על-גבי Socket הלקוח אל השרת וכל בקשה תטופל ב-thread נפרד באמצעות Handler קונקרטי.
- **במהלך הסמסטר עסקנו במספר פרויקטים ב-networking, Multi-threaded Server Architecture TCP & Sockets כולל שרת להעברת מידע על-גבי רשת האינטרנט (TCPServer)**
  אתם רשאים לעשות שימוש בכל המחלקות שמימשנו (כולל הרחבה/שינוי).
- אני מאפשר לכם לקבוע את ה-API ברכיבים החדשים שתוסיפו (בפרט חתימות מתודות, שמות ממשקים/מחלקות חדשות).

דגשים להרחבת המחלקות TcpServer וה-handlers השונים

1. אתם רשאים להשתמש בקוד להרחיבו ולשנותו.
2. אתם רשאים להרחיב את ה-MatrixIHandler ולשנותו בהתאם ל-API שתיצרו.
3. תזכורת: פעולת ()accept של ה-ServerSocket תחזיר לנו במקרה של הצלחה socket תפעולי.

אנו נעביר את InputStream וה-OutputStream ל-Handler רלוונטי אשר יבצע את הלוגיקה שהוגדרה לו ב-thread נפרד. **הפרויקט שכתבנו בשיעור זה יסייע לכם לממש את הדרישות.**

דוגמאות לשימוש ב-Handlers וב-Socket/ServerSocket תפעולי מופיע בפרויקט השרת שכתבנו וכולל שימוש ב-Decorators, לדוגמה:

```
Runnable handleLogic = () -> {
    try {
        requestConcreteHandler.handle(request.getInputStream(),
                request.getOutputStream());
        // Close all streams
        request.getInputStream().close();
        request.getOutputStream().close();
        request.close();
    } catch (IOException ie) {
    }
};
...
   var objectInputStream = new ObjectInputStream(
           (client.getInputStream()));
   inTransaction transaction = (inTransaction)
   objectInputStream.readObject();
```

4. אינני מגביל אתכם ל-OutputStream/InputStream ספציפי וכן לייצוג המידע שיועבר ע"ג ה-Socketים, כל עוד תאפשרו את הפונקציונאליות של המשימות שהוגדרו להלן.
5. הטיפול בכל בקשה (כלומר קריאה למתודה handle של ה-Handler הקונקרטי) תעשה במסגרת Thread ב-ThreadPoolExecutor.
6. חשבו האם פעולת ()accept של ה-ServerSocket צריכה להתבצע ב-thread הראשי של השרת או לעטוף אותה ב-thread נפרד.
7. וודאו כי יש אפשרות להפסיק את פעולת השרת ושפעולה זו נעשית באופן שהוא Thread-Safe באמצעות המנגנונים השונים שלמדנו במהלך הסמסטר.
8. היררכיית ה-Handlerים והחתימות שלהם הם לשיקולכם (כדאי לחשוב כיצד ניתן לטפל ברמה האבסטרקטית ולהבין כיצד הבדלים בין משימה אחת לאחרת יכול להשפיע על בחירתכם).

<u>הנחיות</u>

1. עליכם לכתוב מימוש למערכת שתספק את השירותים הנדרשים לפתרון המשימות (ברשימת המשימות למימוש) תוך שימוש בעקרונות ה-OOD ו-Multithreading שלמדנו במהלך הסמסטר.
2. במידה ומדובר באלגוריתם שמתבצע באופן מקבילי, קריאה בסגנון algorithm.traverse(graph) צריכה להתבצע ב-Thread נפרד.
3. השתמשו במידת הצורך ב-`ReentrantReadWriteLock` באמצעות אחד או שני המנעולים שמכיל.
4. השתמשו ב-Streams ו-method references בהתאם לצורך. לדוגמה:

```
s.getData()).stream().filter(vertex -> matrix.getValue(vertex) ==1)
    .map(NeighboringVertex ->
     new GraphNode<>( NeighboringVertex,s))
    .collect(Collectors.toList());
....
hashSets.sort((Comparator.comparingInt(HashSet::size)));
```

5. ניתנת חשיבות גם ל-Exceptions עליהם תכריזו ותטפלו. אלו רק חלק מהפרקטיקות שהשתמשנו בהן בפרויקטים במהלך הסמסטר.
6. יש לתעד את הקוד באופן תמציתי כולל פרמטרים
7. בפרויקט המסכם ישנן משימות שדורשות מעבר על המטריצה <u>מכמה מקורות במקביל (concurrently)</u>, עליכם לבצע את החיפוש באופן יעיל ונכון.

להלן טיפים ודגשים אשר עשויים לזכות אתכם במלוא הנקודות עבור סעיף זה (אין חובה לעקוב אחר כל הדגשים של סעיף):

- כל חיפוש מאינדקס מקור עשוי להתבצע ב-Thread משלו. על כן, יש לוודא שאין התנגשות במידע שנשמר ב-Threadים, למשל באמצעות הצהרה על מבני נתונים מסוג ThreadLocal<T>.
- את אתחול הטיפוסים המוצהרים מסוג ThreadLocal<T> יש לבצע באמצעות המתודה:
```
public static <S> ThreadLocal<S> withInitial
(Supplier<? extends S> supplier)
```
- להזכירכם טרם ביצוע מעבר על גרף ממקור נתון יש לגשת גישה למידע מסוג `ThreadLocal` באמצעות המתודה: `public T get()`
- ערך ההחזרה מחיפוש מקומי צריך להיות טיפוס קונקרטי ולא `ThreadLocal`.
- שימו לב שאם תממשו אלגוריתם לחיפוש מקבילי, אך בסופו של דבר תריצו אותו במסגרת אותו Thread (לא יעיל), מבני נתונים שהוגדרו כ-`ThreadLocal` עדיין יכילו מידע שנשמר במסגרת חיפוש קודם - וודאו שאין בהם מידע לפני שמתבצע חיפוש חדש.
- **בניגוד לשימוש במבני נתונים שהם מקומיים לחיפוש**, איחוד של המידע (כלומר הוספתו למבנה נתונים אחד) צריך להתבצע באמצעות מבני נתונים Thread-Safe או באמצעות עטיפה של מבנה נתונים קונקרטי באמצעות אחת מהמתודות הסטטיות של **`Collections.Synchronized`**.

<u>רשימת המשימות</u>

- בסעיף זה נציג את המשימות שהמערכת שלכם תפתור.
- אנו מעוניינים לממש אלגוריתמים באופן בלתי תלוי בבעיות שאותן אנו נפתור. כלומר, נרצה לכתוב קוד שיפתור את המקרה הכללי ובאמצעות התאמות נפתור מגוון use cases.
- בבדיקה של הפרויקט אתן חשיבות לנכונות הקוד כמו גם על Thread-safety ו-design יעיל.

<u>משימה 1- מציאת כל קבוצות ה-1ים (אינדקסים ישיגים כוללים אלכסונים)</u>

קלט: מערך 2D של int או Integer

פלט: רשימה של כל קבוצות ה-1ים ממוינת לפי כמות האינדקסים בכל רכיב וללא כפילויות (רשימה של HashSet<Index>)

כולל אלכסונים.

[1, 0, 0]
[1, 0, 1]
[0, 1, 1]

- הפלט יהיה:

[(2,2), (2,1), (1,2), (1,0), (0,0)]

<u>משימה 2- מציאת מסלולים קצרים ביותר מאינדקס מקור לאינדקס יעד</u>

- קלט: מערך 2D **עד גודל 50 X 50** של int או Integer, אינדקס מקור ואינדקס יעד
- ניתן להסתמך על כך שהמטריצה ריבועית
- פלט: רשימה עם המסלולים הקצרים ביותר מאינדקס המקור לאינדקס היעד (קבוצות ה-1ים שכוללות את המספר הקטן ביותר של אינדקסים בין המקור ליעד.
- כידוע יכולים להיות כמה מסלולים קצרים ביותר בין אינדקס מקור לאינדקס יעד
- לא תמיד כדאי לבצע סריקה של מטריצה מכמה אינדקסים במקביל.למדנו שקיים Overhead למימוש כזה והוא כדאי כאשר הקלט מספיק גדול.
- **במשימה 2 (ובמשימה זו בלבד) עליכם לבצע את החישובים במסגרת thread אחד בלבד.**

<u>משימה 3 - משחק צוללות</u>

קלט: מערך דו-מימדי של int או Integer

פלט: מספר הצוללות התקינות על לוח המשחק

חוקים:

1. צוללת יכולה להיות שני 1ים (לפחות) במאונך
2. צוללת יכולה להיות שני 1ים (לפחות) במאוזן
3. לא יכולים להיות שני 1דים באלכסון אלא אם כן עבור שניהם מתקיימים סעיפים 1 ו-2.
4. המרחק המינימלי בין שתי צוללות (ללא קשר לאוריינטציה) הוא משבצת אחת

דוגמה 1 לקלט לא תקין:

[1, 1, 0, 1, 1]
[1, 0, 0, 1, 1]
[1, 0, 0, 1, 1]

דוגמה 2 לקלט לא תקין

[1, 0, 0, 1, 1]
[1, 0, 0, 1, 1]
[0, 1, 0, 1, 1]

דוגמה 3 לקלט תקין- 2 צוללות

[1, 0, 0, 1, 1]
[1, 0, 0, 1, 1]
[1, 0, 0, 1, 1]

דוגמה 4 לקלט תקין- 3 צוללות

[1, 1, 0, 1, 1]
[0, 0, 0, 1, 1]
[1, 1, 0, 1, 1]

<u>משימה 4 - מציאת מסלולים קלים ביותר (ללא אלכסונים)</u>

- קלט: מערך 2D של int או Integer, אינדקס מקור ואינדקס יעד
- פלט: רשימה שכוללת את המסלולים הקלים ביותר מאינדקס המקור לאינדקס היעד
- לדוגמה בהינתן המערך הבא, אינדקס מקור (1,0) ואינדקס יעד (1,2):
  [100, 100, 100]
- [500, 900, 300]
- הפלט יהיה:
  [(1,0) ,(0,0) ,(0,1) ,(0,2), (1,2)]

<u>הערות</u>

1. ההגנה תיערך באופן מקוון.
2. במעמד ההגנה, כל אחד מהמגישים יציג את הקוד ב-IDE לבחירתו באמצעות שיתוף מסך ב-zoom.
3. אם אין ברשותכם מיקרופון שמחובר למחשב, ניתן יהיה לשוחח איתי בטלפון (במקביל לשיתוף המסך**).** אין צורך במצלמה.
4. בתחילת יולי 2021 יעלה גיליון excel משותף עם חלונות זמן – בחרו את השיבוץ המועדף עליכם.
5. כל אחד מהסטודנטים חייב להכיר את הפרויקט במלואו.
6. **אנא מכם - אל תעתיקו.** לרשותכם נמצאים ב-moodle כל הקבצים שכתבנו במהלך הסמסטר, הקלטות שיעורים וחומרי עזר נוספים.
7. ניתן ליצור איתי קשר במייל nathand@hit.ac.il