תרגיל בית תכנותי
להגשה עד 27.01.22 בשעה 23:50
בהצלחה!

מתרגל אחראי על התרגיל: שמעון

<u>הוראות:</u>

1. יש להגיש קובץ **zip יחיד** בעל השם EXP_ID1_ID2 כאשר ID1 ו ID2 הם מספרי תעודות הזהות של שני בני הזוג. קובץ ה zip יכיל תיקייה בודדת בשם src ובה כל קבצי ה Java שיצרתם, ללא תיקיות נוספות ותתי תיקיות. אין צורך להגיש קבצים שסופקו ע"י צוות הקורס.
2. ההגשה תתבצע רק ע"י **אחד** מבני הזוג למקום הייעודי באתר הקורס במודל.
3. **עליכם לוודא לפני ההגשה במודל כי הקוד שלכם מתקמפל ורץ בשרת Microsoft Azure שהוקצה לכם** (הוראות מצורפות בקובץ נפרד).
4. זוג שהתרגיל שלו לא יתקמפל בשרת שהוקצה או יעוף בזמן ריצה **ציונו בתרגיל יהיה 0.**
5. יש לכתוב קוד קריא ומסודר עם שמות משמעותיים למשתנים, למתודות ולמחלקות.
6. יש להקפיד למלא את כל דרישות התרגיל. **אי עמידה בדרישות התרגיל תגרור ציון 0.**

הסבר כללי:

בתרגיל בית זה נרצה לממש בשפת java ראוטר הפועל ברשת של ראוטרים. בהתאם לנלמד בכיתה לראוטר מספר תפקידים. ראוטר מתחזק טבלת ניתוב ובהתאם לטבלת הניתוב מבצע ניתוב להודעות המגיעות אליו בהתאם ליעד ההודעה. כמו כן, ראוטר מדי פעם צריך לעדכן את טבלת הניתוב בהתאם לשינויים שהתרחשו במשקלי קשתות הרשת. בתרגיל בית זה ראוטר יעדכן את טבלת הניתוב באמצעות אלגוריתם הניתוב Distance Vector Routing.

הגדרת ראוטר:

לכל ראוטר ברשת יש שם ייחודי שהוא מספר שלם בין 1 לבין מספר הראוטרים ברשת.

בקובץ Router.java עליכם לממש מחלקה **פומבית** בשם Router.

דרישות:

1. מחלקה זו יורשת מהמחלקה Thread וממשת את הפונקציה run שהיא ירשה מהמחלקה Thread.
2. עליכם לממש בונה למחלקה Router עם החתימה הבאה:

public Router(int name, String inputFilePrefix, String tableFilePrefix, String forwardingFilePrefix)

3. בונה המחלקה קורא פרמטרים עבור הראוטר מקובץ קלט בשם:

inputFilePrefix[name].txt

- הקובץ בנוי משורות, כאשר בכל שורה יש פרמטר מסוים. הפרמטרים של הקלט מסודרים בשורות באופן הבא:
  1. פורט להאזין בפרוטוקול UDP
  2. פורט להאזין בפרוטוקול TCP
  3. מספר הראוטרים ברשת
  4. שם שכן
  5. IP של שכן
  6. פורט שבו השכן מאזין בפרוטוקול UDP
  7. פורט שבו השכן מאזין בפרוטוקול TCP
  8. משקל הקשת לשכן

  שורות 4-8 (כולל) חוזרות על עצמן כמספר השכנים של הראוטר (ראו דוגמא בקבצים המצורפים) . הקלט עבור שכניו של ראוטר מסתיים ב * ולאחריו יש חסם עליון על קוטר הרשת (הממושקל).

- לאחר שראוטר קרא את הפרמטרים הוא בונה את טבלת הניתוב ההתחלתית שלו כך שעבור כל ראוטר ברשת (מלבדו) המרחק לראוטר יוגדר להיות החסם על קוטר הרשת ושדה ה next יוגדר להיות השכן הראשון שלו לפי קובץ הקלט. המרחק לעצמו הוא 0.

4. כאשר קוראים למתודה run של ראוטר, הראוטר מתחיל האזנה על הפורטים שקיבל "במקביל".

<u>קבצי פלט:</u>

לראוטר יש שני קבצי פלט:

1. tableFilePrefix[name].txt
2. forwardingFilePrefix[name].txt

כאשר name זהו שמו של הראוטר.

<u>האזנה על פורט בפרוטוקול UDP:</u>

בהאזנה על פורט זה הראוטר יכול לקבל ארבעה סוגי הודעות בלבד.

1. PRINT-ROUTING-TABLE – כאשר הראוטר מקבל הודעה זו הוא כותב לקובץ הפלט tableFilePrefix[name].txt את טבלת הניתוב שלו. טבלת הניתוב תכתב בקובץ הפלט באופן הבא כאשר כל שורה בטבלת הניתוב תכתב בשורה חדשה בקובץ הפלט:

[estimated distance to router i];[neighbour to forward messages for router i]

השורה בטבלת הניתוב המתאימה לראוטר עצמו תכתב באופן הבא:
0;None
ראו דוגמא בקבצים המצורפים.
לאחר שראוטר מסיים לכתוב את טבלת הניתוב לקובץ הפלט הוא שולח את ההודעה FINISH לשולח ההודעה PRINT-ROUTING-TABLE.
2. UPDATE-ROUTING-TABLE – כאשר הראוטר מקבל הודעה זו הוא מעדכן את טבלת הניתוב לפי אלגוריתם הניתוב Distance Vector Routing.
ראו הערות חשובות בהקשר לפעולה זו בהמשך.
3. FORWARD;[destination];[hops];[message];[ip];[port] – כאשר הראוטר מקבל הודעה זו, ראשית הוא כותב אותה כמו שהיא (בשורה חדשה) בקובץ הפלט forwardingFilePrefix[name].txt. בנוסף, הוא מנתב את ההודעה ל destination בהתאם לטבלת הניתוב שלו לאחר שהקטין ב 1 את ה hops. במידה וראוטר מקבל הודעה זו עם hops שווה ל 0 או שהוא ה destination להודעה:
   3.1. הוא לא מנתב את ההודעה (אך כן רושם בקובץ הפלט)
   3.2. שולח את ההודעה message לכתובת ip ופורט port בפרוטוקול UDP.

ניתן להניח כי message לא מכיל את התו ;.

4. SHUT-DOWN – כאשר הראוטר מקבל הודעה זו הוא יפסיק להאזין על **שני** הפורטים, **גם** פורט ה UDP וגם פורט ה TCP.

האזנה על פורט בפרוטוקול TCP:

בהאזנה על פורט זה הראוטר יכול לקבל סוג הודעה אחד בלבד.

- [num] – כאשר הראוטר מקבל הודעה זו הוא שולח לראוטר שממנו הגיעה ההודעה את וקטור המרחקים שלו עבור ההפעלה מספר num של אלגוריתם הניתוב Distance Vector Routing.
  ראו הערות חשובות בקשר לפעולה זו בהמשך.

הרחבה על פעולת עדכון טבלת הניתוב:

רשת הראוטרים המוגדרת בתרגיל היא רשת מכוונת. ברשת מכוונת השכנים של ראוטר הם רק הראוטרים שיש לו קשתות יוצאות אליהם. הניתוב מתבצע **רק** דרך קשתות יוצאות.
על מנת שנוכל להפעיל את אלגוריתם הניתוב Distance Vector Routing ראוטר צריך לקבל משכניו את וקטור המרחקים שלהם. על מנת שראוטר יוכל לקבל את וקטור המרחקים, הוא פונה לשכניו בבקשה לשלוח לו את וקטור המרחקים (בפרוטוקול TCP) המתאים למספר העדכון הנוכחי והשכן משתמש בחיבור שנוצר על מנת לשלוח לראוטר את וקטור המרחקים (שימו לב שכעת במקום שראוטר ימתין עד קבלת וקטורי המרחקים של שכניו כפי שהוגדר באלגוריתם Distance Vector Routing הראוטר פונה בבקשה לשכניו לקבלת הוקטור).

על מנת להימנע מבעיות של סינכרון, לאחר עדכון טבלת הניתוב הראוטר כבר בונה את וקטור המרחקים שיש לשלוח לצמתים שהוא שכן שלהם (שיוצאת מהם קשת הנכנסת אליו) בעדכון הבא. בנייה זו כוללת עדכון של משקלי הקשתות.

<u>הסבר על אופן עדכון משקלי הקשתות:</u>

בקובץ CreateInpute.java שקיבלתם (ראו הסבר מפורט על הקובץ בהמשך) יש מטריצה סטטית בעלת 3 מימדים בשם weightsMatrix כאשר:

CreateInput.weightsMatrix[i][j][k]

זהו הערך של משקל הקשת מראוטר i לראוטר j בסבב העדכון ה k.
כניסות שמייצגות מידע לא רלוונטי (לדוגמא, אם אין קשת מראוטר i לראוטר j או ראוטר עם מספר 0) יש בהן את הערך 0. במידה והערך במטריצה הוא 1- המשמעות היא שהמשקל לא השתנה ונשאר כפי שהוא.

<u>הערות:</u>

1. שימו לב כי את וקטור המרחקים המתאים לסבב העדכון הראשון יש לבנות לפני שהראוטר מתחיל להאזין על הפורטים.
2. כאשר ראוטר מבקש משכן את וקטור המרחקים הוא שולח לו את ההודעה

[num]

כאשר num הוא מספר עדכון טבלת הניתוב הנוכחי (בעדכון הראשון ישלח 1).

3. לאחר סיום עדכון טבלת הניתוב ועדכון וקטור המרחקים של העדכון הבא תוחזר ההודעה FINISH לשולח ההודעה UPDATE-ROUTING-TABLE.

4. במידה ובמהלך עדכון טבלת הניתוב של ראוטר i המרחק לראוטר j הוא זהה דרך שני שכנים שונים, ראוטר i יבחר לנתב הודעות לראוטר j דרך השכן שמספרו קטן יותר.

הרחבה על פעולת שליחת וקטור המרחקים:

כאשר ראוטר מקבל בקשה לשליחת וקטור המרחקים שלו

[num]

הוא ישלח לשכן המבקש את וקטור המרחקים כפי שהוא נבנה בסוף העדכון ה num-1.

דרישות:

1. בתרגיל בית זה ניתן לבצע **רק** את ה imports הבאים:
   - import java.net.SocketException;
   - import java.nio.charset.StandardCharsets;
   - import java.net.UnknownHostException;
   - import java.io.DataOutputStream;
   - import java.net.DatagramSocket;
   - import java.io.DataInputStream;
   - import java.io.FileWriter;
   - import java.io.File;
   - import java.net.DatagramPacket;
   - import java.io.IOException;
   - import java.net.ServerSocket;
   - import java.net.Socket;
   - import java.net.InetAddress;
   - import java.util.*;

2. יש להתייחס לראוטר גם כשרת וגם כלקוח במודל השרת-לקוח (לדוגמא הראוטר מתפקד כשרת כאשר מקבל בקשה לשליחת וקטור המרחקים ומתפקד כלקוח כאשר מבקש את וקטור המרחקים של שכניו). במודל השרת-לקוח על מנת שהשרת יוכל לטפל במספר בקשות "בו-זמנית", לאחר שהשרת מקבל בקשה נפתח thread חדש לטיפול בבקשה והשרת חוזר להאזין על הפורט לבקשות נוספות. שימו לב, ה thread המאזין תפקידו רק לקבל את הבקשה (ואולי לבצע פעולות קצרות מאוד) ואינו רשאי לשלוח ולקבל מידע.
3. ייתכן ושני threads ינסו לגשת "במקביל" למשאבים של ראוטר (טבלת הניתוב, קובץ הפלט,...). מהסיבה הנ"ל יש להשתמש במנגנון נעילה עבור משאבים משותפים. **שימו לב, יש לנעול את המשאב למשך הזמן המינימלי הדרוש.**

הנחות:

1. מספרי הפורט שיינתנו כקלט לראוטר יהיו תקינים. כלומר, הראוטר אכן יכול להאזין על מספרי פורט אלו ושכניו אכן מאזינים על מספרי הפורט שקיבל.
2. הודעות בפרוטוקול UDP לא ילכו לאיבוד ויגיעו ליעדן.
3. גודל הודעה בפרוטוקול UDP יהיה לכל היותר 4096 בתים.
4. ההודעה על עדכון טבלת הניתוב (UPDATE-ROUTING-TABLE) תשלח תמיד לכל הראוטרים ברשת ובערך באותו הזמן.
5. ראוטרים לא יתבקשו לעדכן את טבלת הניתוב לפני שכל הראוטרים ברשת סיימו את העדכון הקודם.
6. ניתוב של הודעה יסתיים בטרם תתקבל בקשה לעדכון טבלאות הניתוב.
7. הודעת SHUT-DOWN תשלח לכל הראוטרים בערך באותו הזמן ורק לאחר שהם שהם סיימו לעדכן את טבלאות הניתוב ואין שום הודעה שמנותבת ברשת.
8. משקלי הקשתות הם מספרים שלמים וחיוביים.

הסבר על הקבצים שצורפו:

1. CreateInput.java – בקובץ הזה מוגדרת המחלקה CreateInput אשר מטרתה:
   1.1. בניית רשת תקשורת מכוונת (אקראית אך קשירה היטב).
   1.2. כתיבת המידע לקבצי הקלט של הראוטרים.
   1.3. בניית מטריצת המשקלים.
2. Test.java – קובץ בדיקה. הבדיקה מתחילה מבניית רשת התקשורת, קובץ הקלט לכל ראוטר ומטריצת המשקלים (ע"י שימוש במחלקה CreateInput).
   בשל אילוצי משאבים אנו מסמלצים את פעולת רשת התקשורת על מחשב אחד ולכן לפני תחילת הבדיקה יפתח thread נפרד לכל שרת ולאחר מכן ישלחו בקשות לראוטרים.
3. text_files – תיקייה ובה קבצי הקלט והפלט של הראוטרים לאחר הרצת פונקציית ה main בקובץ Test.java.

הערות:

1. כל מה שלא הוגדר באופן מפורש נתון לשיקול דעתכם (מבנה הנתונים שישמש לשמירת טבלת הניתוב, אופן שליחת וקטור המרחקים לשכנים וכו')
3. מחלקת CreateInpute שקיבלתם ייתכן ותשתנה בבדיקה, בפרט ייתכן ואופן בניית רשת התקשורת יהיה שונה, מספר הראוטרים ברשת, משקלי הקשתות, ההסתברויות וכו'.
4. הבדיקה תתבצע עם קובץ שונה מ Test.java ובו סדר הפעולות, מספר הפעולות וכו' ייתכן וישתנה.
5. המלצה: לפני שאתם מתחילים לעבוד על התרגיל תנסו ליצור חיבור TCP וחיבור UDP על מנת להבין את נושא התקשורת ב java.
6. שימו לב כי קבצי הקלט והפלט מסתיימים ב txt.
7. כל מחרוזת (String) שתשלח ותתקבל מהראוטרים צריכה להיות בקידוד UTF-8. ניתן להגדיר זאת ע"י שימוש ב StandardCharsets.UTF_8. ראו דוגמא במחלקה Test.java.

הדרכה:

על מנת למנוע גישה של יותר מ thread אחד באותו פרק הזמן לאותו משאב, ניתן להשתמש במילה השמורה synchronized ב java.

הסבר על תהליך הבדיקה האוטומטית:

אנחנו נריץ את הקבצים שלכם עם מחלקת Test ו CreateInpute שונה מזאת שקיבלתם עם פרסום התרגיל. ב Test ו CreateInpute שישמשו לבדיקה ייתכן ויתווספו אובייקטים, הפעולות וסדר הפעולות ישתנה, גודל הקלט ישתנה וכו'

במהלך הבדיקה יקומפלו **כל** הקבצים שהגשתם בתוספת Test ו CreateInpute (שונים) בשרת Microsoft Azure. **חשוב מאוד שתגישו את כל קבצי ה java שיצרתם ותוודאו שהקוד מתקמפל בשרת.**
בהנחה והקוד מתקפל, הקוד יורץ והפלט שיתקבל יושווה לפלט חוקי באופן הבא:

1. קבצי הפלט של הראוטרים שבהם הודפסו טבלאות הניתוב (tableFilePrefix[name].txt) צריכים להיות זהים לקבצי הפלט החוקיים.
2. קבצי הפלט של הראוטרים שבהם הודפסו הודעות הניתוב (forwardingFilePrefix[name].txt) לא חייבים להיות זהים לקבצי הפלט החוקיים (תיתכן שונות בשל תיזמון של ה threads) אך יבדק כי כל השורות הנמצאות בקבצי הפלט החוקיים נמצאות בקבצי הפלט שלכם וההפך.

המלצה:

אל תשאירו את הבדיקה בשרת לרגע האחרון. ייתכן והקוד לא יתקמפל בשרת ותצטרכו לתקנו לפני ההגשה.

בהצלחה