**מעבדה 2**

**אלי חדאד 207931536**

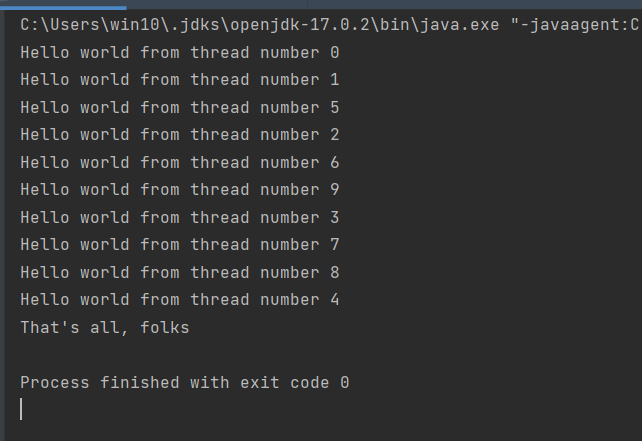
**פייסל סעדיה 208336321**

1)thread.start() יוצרת חוט חדש ואז החוט הזה קורא למתודה run אבל thread.run() לא יוצרת חוטים חדשים, רק מריצה חוט שכבר קיים .

2)

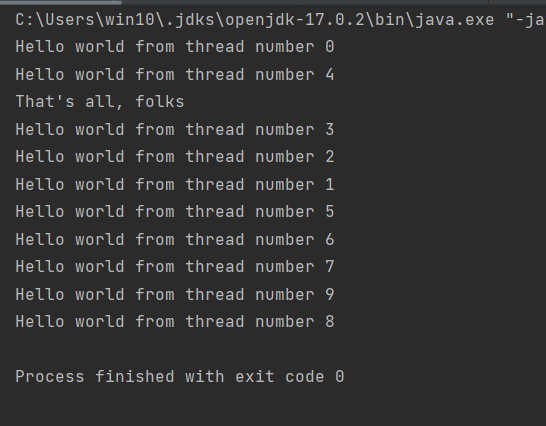
2.1)

התוכנית הדפיסה את מספרי החוטים בסדר שונה כל פעם כי כל החוטים עובדים במקביל, ובסוף בגלל פקודת join התוכנית מחכה עד שכל החוטים יסיימו ואז מדפיסה “that’s all , folks” .



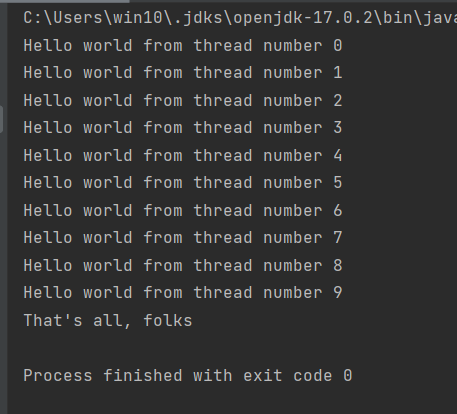
2.2)

הלולאה השלישית המטרה שלה היא לחכות עד שכל החוטים סיימו עבודתם ואז התוכנית ממשיכה לכן אם מוחקים את הלולאה, התוכנית עלולה להדפיס את השורה האחרונה לפני שמדפיסה את כל מספרי החוטים .



2.3) אם מבצעים פקודת join אחרי start באותה לולאה אז התוכנית תמתין עד שהחוט הנוכחי ימות לפני שתייצר חוט חדש

ולכן התוכנית תדפיס את המספרים בסדר הנכון.

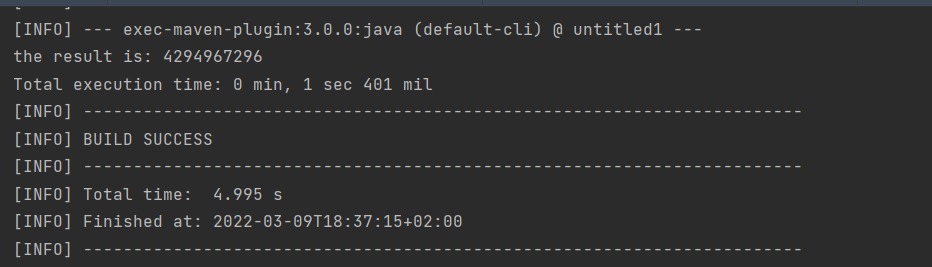


2.4) מצב של deadlock , כי אנחנו מבקשים לחסום את החוט עד ש אותו חוט מסיים עבודתו, שזה לעולם לא יקרה.

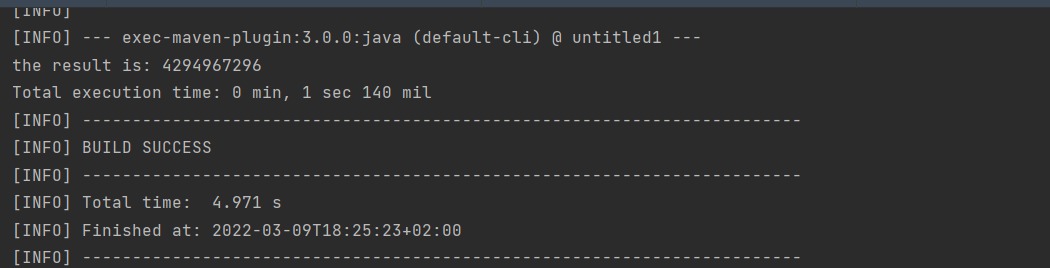
3)

3.1) כן, עם חוטים הקוד יהיה יותר מהיר ויעיל, כי החוטים רצים במקביל וזה חוסך זמן ומאמץ.

Sum :



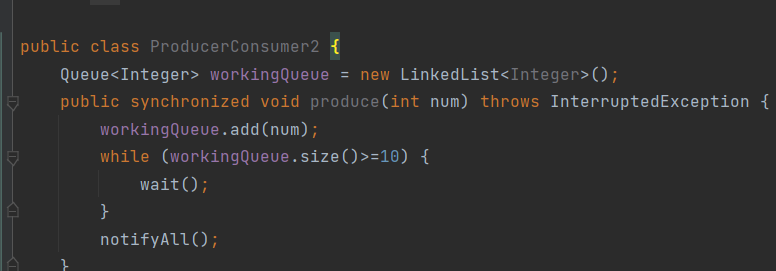
SumThreads :



3.2) לא ,יצא לנו הפרש של כמה milliseconds כל פעם, כי החוטים רצים במקביל ויכול להיות שכל פעם הם רצים בצורה שונה לכן ישפיע קצת על זמן הריצה.

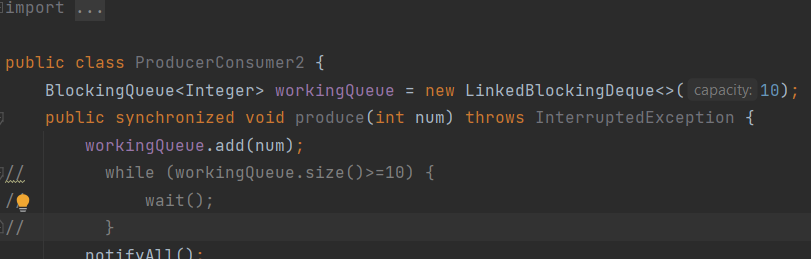
3.3) צירפנו את קובץ SumThreads.java לקובץ ה zip.

4) הוספנו לולאת while שבודקת אם יש 10 איברים בתור אם כן אז קוראים ל wait .



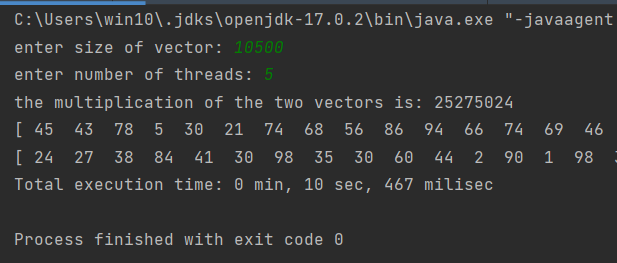
דרך אחרת היא להשתמש ב Concurrent Collections ,נשתמש ב

Blockingqueue במקום queue ולהגדיר את הcapacity כ 10



הוספנו קובץ producerConsumer2.java לקובץ ה zip .

מטלה 2.4)



2.5) כי התוכנית יותר מהירה עם 10 חוטים מאשר 5, אם היו עובדים בצורה סדרתית אז היה יוצא זמן ריצה דומה.