Github : click here!

מטלה 3

ליאורדניאל – 305257347
אילן קיציו – 205358203

חלק א':

בחלק זה התבקשנו להפעיל את הפונקציה `isPrime(long n)` ולבדוק האם תצליח לחשב בזמן מסוים האם מספר כלשהו ראשוני או לא, לשם כך יצרנו מחלקה Timer היורשת מ-Thread. מחלקה זו "הולכת לישון" למיל' שניות וכשמתעוררת זורקת שגיאה. `throw new RuntimeException("TIME OUT");`

עם הפעלת הפונקציה `isPrime(long n, double maxTime)`, נפעיל את הפונקציה `isPrime(long n)` ומיד אחריה את ה-Timer שלנו. במידה והפונקציה תספיק לסיים את החישוב היא מיד תעצור את ה-Timer. במידה ולא, ה-Timer "יתעורר" ויזרוק שגיאה(TIME OUT) .

חלק ב' :

בחלק זה יצרנו שלושה פונקציות היוצרות קבצים עם מס' שורות רנדומלי, מדפיסות את סכום השורות בכל הקבצים ובסיום מוחקת את הקבצים.

- לצורך ההשוואה בין הפונקציות השתמשנו במספר קבצים ושורות זהים עבור כל פונקציה. (מס' קבצים = 100 , מס' שורות = 782000)

פלט :

```
*countLinesThreads*
Total lines : 782000
Time of threads work :312ms

*countLinesOneProcess*
Total lines : 782000
Time of threads work :125ms

*countLinesThreadPool*
Waiting for all results...
Total lines : 782000
Time of threads work :78ms
```

הסבר :

- `countLinesThreads` – הפונקציה הזו מקצה תהליכון עבור כל קובץ הבודק את מס' השורות שלו. במקרה כזה, עלינו לחכות שתהליכון אחד יסיים את הבדיקה על מנת לקבל את הערך בחזרה, רק אז נוכל להמשיך לתהליכון הבא. לכן לוקחת הכי הרבה זמן כי מתעכבת על כל תהליכון.

- `countLinesOneProcess` – הפונקציה מחשבת בתוכנית אחת עבור כל קובץ את מספר השורות תחת לולאת for, ללא עיכוב על יצירת תהליכונים והמתנה שיסיימו.

- `countLinesThreadPool` –עבור הפונקציה הזו יצרנו מחלקת עזר LineCounterCallable הממשמת את ממשק Callable ומייצגת thread, שיחשב את מספר השורות עבור קובץ וידע להחזיר את הערך. בתוך הפונקציה עצמה נעזרנו ב-Future כך ש"יחכה" לערך החזרה מתהליך מסוים. ע"י שימוש ב-Thread Pool הפעלנו LineCounterCallable עבור כל קובץ במקביל, כך ש-Future מחכה לערך החזרה מכל תהליכון שרץ. מכיוון שהם רצים במקביל, הערכים מתקבלים בזמן מהיר וניתן להחזיר את סך כל השורות בזמן המהיר ביותר.