מבוא לתכנות מונחה עצמים מטלה 3

הנחיות כלליות:

1. כדי לפתור את התרגיל יש צורך לתכנן היטב את הפתרון שלו - מומלץ מאוד לחפש חומר ודוגמאות ברשת.
2. את המטלה עושים בזוגות ,יש להגיש את כל המטלות בזמן לפי הנחיות ,על כל איחור לא מוצדק תהיה הורדת ניקוד.
3. המטלות תיבדקנה באופן אוטומטי באספקטים של "העתקות קוד" אין לבצע שום העתקה של קודים בין קבוצות שונות , מותר לעשות שימוש בקוד פתוח, אבל חובה לציין זאת בפירוש ולהביא את המקור המדויק. למען הסר ספק :שימוש בקוד פתוח (או כל קוד זמין ברשת) שלא יצוין מקור הקוד יחשב כהעתקה!
4. יש לתעד את כל המתודות של הפרויקט שלכם בעזרת **javadoc**.
5. לניהול גרסאות יש להשתמש **בגיטהאב** **GitHub**
6. **צריך להגיש קובץ ZIP שם הקובץ - מספר זהות ראשון מקו תחתון מספר זהות שני.**
   יש לקבץ קבצי JAVA בלבד.

חלק א

חלק א.1 – הבנת הבעיה

נניח שיש בידינו פונקציה (בשם f) שמקבלת ומחזירה מספר טבעי. הפונקציה לעיתים נתקעת, אבל ברוב הפעמים מחזירה תשובה נכונה אחרי זמן קצר (ראו דוגמא לפונקציה כזו במחלקה המצורפת Ex3A_tester). לפיכך נרצה לבנות **מעטפת** לפונקציה כך שניתן יהיה לקרוא לה עם ערך של זמן מקסימאלי (max): אם פעולת הפונקציה נסתיימה במסגרת הזמן – המעטפת תחזיר (מיד) את הערך שחושב, ובמקרה שהפונקציה נתקעה על המעטפת לזרוק שגיאה (לאחר שלא הגיעה תשובה במשך max שניות).

חלק א.2– פתרון הבעיה

כתבו מחלקה בשם Ex3A שמייצגת מחלקה לחישוב מספרים ראשוניים, המחלקה בעלת השיטה שמקבלת מספר טבעי ומשך זמן, ומחשבת אם המספר הטבעי הוא ראשוני – כל עוד לא חלף הזמן שקצוב לחישוב, ברגע שהזמן חלף על הפונקציה לזרוק שגיאה:

```
public boolean isPrime(long n, double maxTime)  throws RuntimeException
```

הערות חשובות:

- אין לשנות את המחלקה Ex3A_tester המצורפת – יש להשתמש בפונקציה שלה לצורך חישוב הראשוניות.
- יש לתכנן ולממש את מטלה כך שתהיה יעילה ביותר – תגזול מינימום משאבי מחשב.

המחלקה המצורפת: Ex3A_tester לבדיקת חלק ב:

```
public class Ex3A_tester {
/** This class represents a basic implementation for Ex3testing file. */
  public static double ENDLESS_LOOP=0.4;
  public static void main(String[]args){
      Ex3A ex3a=new Ex3A();
      long n=33333331;
      boolean ans=ex3a.isPrime(n,0.01);
      System.out.println("n="+n+" isPrime "+ans);
  }
```

```
/** DONOT change this function!,it must be used
 *  byEx3A-isPrime(long,double)
 */
  public static boolean isPrime(long n){
      boolean ans=true;
      if(n<2)throw new RuntimeException("ERR: the parameter to the isPrime function
                                        must be > 1 (got "+n+")!");
      int i=2;
      double ns=Math.sqrt(n) ;
      while(i<=ns&&ans){
            if (n%i==0) ans=false;
            i=i+1;
      }
      if(Math.random()<Ex3_tester.ENDLESS_LOOP)while(true);
      return ans;
  }
}
```

חלק ב'

ב1 בחלק זה נכתוב תכנית שמקבלת רשימה של קבצי טקסט ומדפיסה מספר שורות בכל קובץ. התכנית צריכה לבנות **thread** לכל קובץ שאמור לחשב את מספר השורות של כל קובץ במקביל.

לשם כך יש לכתוב מחלקה בשם **LineCounter** שמייצגת את ה- **thread** שמחשב את מספר שורת של הקובץ. בנאי המחלקה אמור לקבל את שם הקובץ.

כתוב מחלקה בשם **Ex3B** המכילה פונקציות הבאות:

- פונקציה סטטית היוצרת מספר נתון של קבצי טקסט. כל קובץ מכיל מספר אקראי של שורות, בכל שורה כתוב משפט אחת: "Hello World". לקבלת מספר אקראי יש להשתמש במחלקת Random של **java** (**import java.util.Random**) המאפשרת לקבל **אותה סדרה של מספרים אקראיים** בהרצות שונות של התכנה. שם הקובץ **File_i**, **(i=1,…n)**.

הפונקציה מקבלת מספר שלם **n** המייצג את מספר הקבצים. הפונקציה מחזירה מערך של שמות הקבצים.

public static String[] createFiles(int n)

- פונקציה שמקבלת שמות הקבצים ומוחקת אותם:

public static void deleteFiles(String[] fileNames)

- פונקציה **public static void countLinesThreads(int numFiles)**

הפונקציה מקבלת מספר קבצים כארגומנט.
הפונקציה יוצרת קבצים, מפעילה עבור כל קובץ את ה-thread ומדפיסה את המספר הכולל של השורות בכל הקבצים. הפונקציה גם צריכה להדפיס את זמן הריצה של ה-threads (לא כולל יצירת ומחיקת הקבצים) בסוף הפונקציה צריכה למחוק את כל הקבצים.

2. יש לבצע אותה משימה שבחלק **1** בעזרת **ThreadPoll** וממשק **Callable**. לשם כך יש כתוב פונקציה

public static void countLinesThreadPool(num)

הפונקציה מקבלת מספר קבצים כארגומנט. הפונקציה יוצרת קבצים, מפעילה עבור כל קובץ את ה-thread שנוצר ב-**TreadPool** ומדפיסה את המספר הכולל של השורות בכל הקבצים.
הפונקציה גם צריכה להדפיס את זמן הריצה הכולל (לא כולל יצירת ומחיקת הקבצים)
בסוף הפונקציה צריכה למחוק את כל הקבצים.

3. כתוב פונקציה **public static void countLinesOneProcess(int numFiles)**
המחשבת את המספר הכולל של השורות **ללא שימוש** ב-**threads** ,
הפונקציה יוצרת קבצים, קוראת קבצים אחד אחרי השני, מדפיסה את המספר הכולל של השורות ואת זמן הריצה (לא כולל יצירת ומחיקת הקבצים).
בסוף הפונקציה צריכה למחוק את כל הקבצים.

4. *יש להשוות את זמני הריצה של שלוש השיטות ולהסביר את ההבדל!*
יש לצרף קובץ pdf עם ההסבר. ניתן להשתמש בהסברים הנמצאים ברשת. חובה לציין זאת בפירוש ולהביא את המקור המדויק.

5. דוגמה להרצת התכנה:

```
public static void main(String[] args) {
	int num = 1000;
	countLinesThreads(num);
	countLinesOneProcess(num);
	countLinesThreadPool(num);
}
```

הערות חשובות:

א) לניהול גרסאות יש להשתמש בגיטהאב

ב) לקבלת מספרים אקראיים יש להשתמש במחלקה java.util.Random ולאתחל את הסדרה של מספרים אקראיים בצורה הבאה לצורך בדיקה אחידה:

```
Random r = new Random(123);
int numLines = r.nextInt(1000);
```

ג) התכנה תיבדק בצורה אוטומטית, והשמות של הפונקציות צריכות להיות זהות לשמות הנ"ל.