

# תוכנה 1 – אביב תשע"ג

## תרגיל מספר 5

מערכים, מחרוזות, עיבוד טקסט ומבני בקרה

### הנחיות כלליות:

קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

- הגשת התרגיל תיעשה במערכת ה-moodle בלבד (<http://moodle.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש האוניברסיטאי שלכם ומספר התרגיל (לדוגמא, עבור שם המשתמש aviv יקרא הקובץ aviv\_hw5.zip). קובץ ה-zip יכיל:
  - א. קובץ פרטים אישיים בשם details.txt המכיל שם מלא ומספר ת.ז.
  - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש.

**חשוב: יש להקפיד על שמות מחלקות ועל פלטים מדויקים על פי ההוראות והדוגמאות !**

### חלק א' – סכימת מספרים בבסיסי ספירה (20%)

בחלק זה נתאמן על עיבוד מחרוזות מתקדם תוך שימוש בפונקציות ספרייה קיימות של שפת ג'אווה.

מומלץ להשתמש במחלקות ספרייה קיימות של ג'אווה לעיבוד מחרוזות כגון StringTokenizer או Scanner המפרקות מחרוזות לחלקים (Tokens) על פי תווים מפרידים נתונים (Delimiters).

<http://docs.oracle.com/javase/7/docs/api/java/util/StringTokenizer.html>

<http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html>

חשוב: את שתי המתודות של חלק זה יש לממש באותה מחלקה תחת השם Assignment05SecA

1. ממשו מתודה בשם *calcSum* המקבלת מחרוזת בודדת ובה רצף של מספרים המחולקים לקבוצות ע"י פסיקים (רווחים יפרידו בין מספרים בתוך אותה קבוצה, ופסיקים יפרידו בין מספרים השייכים לקבוצות שונות). המתודה תדפיס בשורה את הסכום של כל קבוצת מספרים ע"פ הפורמט הבא:

עבור מחרוזת הקלט:

"1 3 8 , 16 4 , 33 6 7 2 , 18 5"

יודפס הפלט:

"12 , 20 , 48 , 23"

הסבר: במחרוזת הקלט יש 4 קבוצות מספרים (תחומים ע"י 3 פסיקים) ועל כן הודפסו 4 ערכים המייצגים את סכומה של כל קבוצה. החישוב שבוצע הוא:

(1+3+8) , (16+4) , (33+6+7+2) , (18+5)

חתימה המתודה:

```
public static void calcSum (String sequence);
```

2. ממשו מתודה בשם calcSumByRadix המחשבת סכומים של מספרים הנתונים במחרוזת בדומה לסעיף הקודם, אך בסעיף זה מתווסף סיבוך – בסופו של כל מספר במחרוזת הקלט תופיע אות שתייצג את בסיס הספירה של המספר על פי הפירוט הבא:

B	בסיס ספירה 2	(בינארי)
D	בסיס ספירה 10	(דצימאלי)
H	בסיס ספירה 16	(הקסדצימאלי)

דוגמא:

עבור מחרוזת הקלט:

"10B 10D 10H , 111B 1aH 101D , 11B 5D"

יודפס הפלט:

"28 , 134 , 8"

כאשר החישוב שבוצע (לאחר המרה של כל מספר לבסיס 10) הוא:

(2+10+16) , (7+26+101) , (3+5)

רמז: המתודה תסרוק את המחרוזת ותפרק אותה לחלקים על פי התווים המפרידים המתאימים (קודם לקבוצות לפי הפסיקים, ואז למספרים לפי הרווחים), כל מספר יומר על פי הספרה שבסופו לבסיס דצימאלי, ובסופו של דבר יודפס הסכום של כל קבוצה (בבסיס דצימאלי).

חתימה המתודה:

public static void calcSumByRadix (String sequence);

## חלק ב' – ספירת מילים בטקסט (40%)

כתבו תוכנית בשם TextFileAnalyzer המקבלת שם של קובץ טקסט כארגומנט בשורת הפקודה. התוכנית תקרא את קובץ הטקסט, תנתח את המילים המופיעות בו על פי המוגדר בהמשך, ותדפיס למסך את תוצאות הניתוח.

- התוכנית תזהה את 5 המילים בעלות מספר המופעים הגדול ביותר בקובץ הטקסט.
- התוכנית תזהה את 5 האותיות הלועזיות בעלות מספר המופעים הגדול ביותר בראש מילה.
- התוכנית תזהה את 5 המילים הארוכות ביותר בקובץ הטקסט.

הערות:

- מילים מוגדרות כרצף תווים (שאינם Whitespaces) המופרדות ביניהם ע"י Whitespaces. (אוסף התווים שנחשבים כ-Whitespace נקבע לפי המתודה Character.IsWhitespace).
- כמו כן, מילה חייבת להתחיל ע"י אות לועזית. ממילים שאינן מתחילות באות לועזית יש להתעלם.
- יש לבצע את כל החישובים במצב case-insensitive (כלומר המילה 'Hello'-i-'hello' נחשבות זהות ומתחילות באותה אות).

- ניתן לכלול בחישובים רק את 2000 המילים השונות הראשונות אותן נקרא מקובץ הקלט (רמז: כדי שנוכל להגדיר מראש מערך מילים ומערך מונים בגודל זה).
- המילים בפלט (או האותיות בסעיף השני) צריכות להיות ממוינות קודם לפי מספר המופעים, ואז לפי סדר ההופעה בטקסט לראשונה. בכל מקרה יש לכלול רק 5 תוצאות בכל סעיף, גם אם נמצאו מספר מילים (או אותיות בסעיף השני) עם מספר זהה של מופעים.
- בסעיף השלישי בו נדרש לזהות את 5 המילים הארוכות ביותר, יש להדפיס את המילים ממוינות על פי אורכן כשבראש הרשימה המילה הארוכה ביותר. במידה ויש מספר מילים באותו האורך יש להדפיסן על פי סדר ההופעה בטקסט.
- יש להדפיס את המילים עם גודל אותיות כפי שהיה במופע הראשון של כל מילה.
- הקפידו לטפל במקרי קצה בהם הקלט מכיל פחות מ-5 מילים!
- ניתן להניח שקובץ הקלט הוא קובץ טקסט המצוי באותה ספרייה בה התוכנית רצה (ב-Eclipse, כברירת מחדל, זוהי ספריית הפרויקט). על כן ניתן לקבל רק את שם הקובץ.
- לצורך קריאת הטקסט מקובץ הקלט וחלוקתו לחלקים (Tokens) על פי תווים מפרידים נתונים (Delimiters) ניתן להשתמש במחלקה Scanner כפי שהוצגה במצגת תרגול 3.

### דוגמא:

הרצת התוכנית:

TextFileAnalyzer input.txt

כאשר בספרייה המקומית מצוי הקובץ input.txt ומכיל את הטקסט הבא:

```
Everything that can be counted does not necessarily count;
everything that counts cannot necessarily be counted. (Albert
Einstein)
```

תביא להדפסת הפלט הבא למסך:

```
TextFileAnalyzer

Word          Frequency
Everything     2
that          2
be            2
necessarily   2
can           1

Leading letter Frequency
C             6
E             3
N             3
T             2
B             2

Word          Length
necessarily   11
Everything     10
Einstein)     9
counted.      8
counted       7
```

## חלק ג' – זיהוי סדרות מספרים (40%)

בשאלה זו נכתוב תוכנית בשם SequenceAnalyzer הקוראת סדרות של מספרים שלמים מקובץ טקסט נתון, מנסה לזהות את סוג הסדרה (חשבונית, הנדסית, ריבועית או פיבונאצ'י) ומדפיסה את פרטי הסדרה למסך.

סוגי הסדרות אותן תזהה התוכנית הן:

1. סדרה חשבונית (Arithmetic): סדרה בה ההפרש בין איברים סמוכים קבוע.
2. סדרה הנדסית (Geometric): סדרה בה המנה בין איברים סמוכים היא קבועה.
3. סדרת הריבועים (Square): סדרה של ריבועי מספרים עוקבים (ניתן להניח סדרה עולה וחיובית, אך לא חייבת להתחיל ב-1).
4. סדרת פיבונאצ'י (Fibonacci): סדרה בה כל איבר החל מהאיבר השלישי הוא סכום של שני האיברים שלפניו (לא חייבת להתחיל ב-1).

### פרטים נוספים:

- שמו של קובץ הקלט ניתן לתוכנית כארגומנט שורת פקודה.
- כל שורה בקובץ הקלט תכיל סדרה אחת המורכבת ממספרים שלמים, מופרדים ע"י רווחים.
- במידה וזוהתה חוקיות הסדרה, יש להדפיס את סוג הסדרה למסך (בסדרה חשבונית והנדסית יש להדפיס גם את ההפרש/מנה).
- במידה ולא זוהתה חוקיות הסדרה, תודפס הודעה מתאימה למסך.
- במידה וגודל הסדרה קטן מ-3, תודפס הודעת שגיאה מתאימה למסך (ראו דוגמא).
- תיתכן סדרה המקיימת יותר מחוקיות אחת ואז יש להדפיס את כל סוגי החוקיות שזוהו.

### דוגמא:

אם נריץ את התוכנית שלנו משורת הפקודה כך:

```
SequenceAnalyzer sequences1.txt
```

בהנחה שקיים קובץ טקסט בשם sequences1.txt המכיל את השורות הבאות:

```
2 4 6 8 10
2 4 8 16
1 2 3 4 5 6 7 8 9 10
-2 -6 -18
700 600 500 400
100 50 25
2187 729 243 81 27 9 3
1 5
1 2 3 5 8 13
1 4 9 16 25 36 49
16 25 36 49
1 3 6 10 15 21 28 36 45
1 1 1 1 1 1
```

יודפס הפלט הבא למסך (עליכם לממש בדיוק באותו הפורמט עד לרמת הריווח):

```
Sequence 1: [2, 4, 6, 8, 10]
Arithmetic sequence detected: d=2

Sequence 2: [2, 4, 8, 16]
Geometric sequence detected: q=2.000000

Sequence 3: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Arithmetic sequence detected: d=1

Sequence 4: [-2, -6, -18]
Geometric sequence detected: q=3.000000

Sequence 5: [700, 600, 500, 400]
Arithmetic sequence detected: d=-100

Sequence 6: [100, 50, 25]
Geometric sequence detected: q=0.500000

Sequence 7: [2187, 729, 243, 81, 27, 9, 3]
Geometric sequence detected: q=0.333333

Sequence 8: [1, 5]
Error: Sequence length is smaller than 3.

Sequence 9: [1, 2, 3, 5, 8, 13]
Fibonacci sequence detected.

Sequence 10: [1, 4, 9, 16, 25, 36, 49]
Square sequence detected.

Sequence 11: [16, 25, 36, 49]
Square sequence detected.

Sequence 12: [1, 3, 6, 10, 15, 21, 28, 36, 45]
Unidentified sequence type.

Sequence 13: [1, 1, 1, 1, 1, 1]
Arithmetic sequence detected: d=0
Geometric sequence detected: q=1.000000
```

הנחיות נוספות:

✓ הקפידו לבנות את הקוד בצורה מסודרת – למשל, עבור סדרה חשבונית כיתבו מתודת עזר בעלת החתימה:

```
private static String testArithmeticSequence(int[] sequence)
```

אשר תבחן אם מערך המספרים הנתון מחזיק סדרה חשבונית ואם כן, תחזיר (לא תדפיס בעצמה) את המחרוזת אותה יש להדפיס. חישבו מה תחזיר המתודה במידה ולא זוהתה סדרה חשבונית.

✓ מומלץ לעשות שימוש במחלקות ומתודות הבאות הזמינות לכם בג'אווה:

- Scanner – מאפשר קריאת טקסט מקובץ על פי תווים מפרידים כלשהם
- String.split(String delimiter) – מחלק מחרוזת לחלקים על פי התו המפריד המצוין
- String.format(String string) – יוצר מחרוזת המשלבת טקסט עם ערכי משתנים למשל: String.format("Sequence %d",i) יחליף את %d בערכו של המשתנה i
- Arrays.toString(int[] array) – יוצר מחרוזת המייצגת מערך נתון של מספרים שלמים