

# תוכנה 1 – אביב תשע"ג

## תרגיל מספר 7

### 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\_hw7.zip). קובץ ה-zip יכיל:
  - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
  - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש.
  - ג. קובץ טקסט answers.txt עם התשובות לשאלות (אפשר קובץ טקסט או קובץ Word).

שימו לב,

- לכל אורך התרגיל ניתן להוסיף שירותי עזר בתנאי שאינם פוגעים בנדרש בסעיפים אחרים.
- כדי להקל עליכם, סיפקנו מחלקות הבדקות חלק מהדוגמאות המופיעות בגוף התרגיל. אם הקוד שמימשתם עונה נכונה על הדוגמאות, תתקבל הודעה שהבדיקה עברה בהצלחה. אחרת, תתקבלנה הודעות שגיאה מתאימות.
- חשוב:** הודעת הצלחה אינה מעידה על נכונות הקוד אלא על התאמה לדוגמאות הספציפיות אותן סיפקנו. לעומת זאת, הודעות שגיאה מעידות על כך שהמימוש שגוי.
- חשוב מאוד:** קוד שאינו מתקמפל עם תוכניות הבדיקה לא ייבדק.
- בהצלחה

### חלק א'

בחלק זה נתמודד עם שתי בעיות הקשורות למשחקי מילים.

- 10 (נקודות) נרצה לבנות אפליקציה אשר תעזור לנו בפתרון תשבצים. לשם כך נבנה מחלקה אשר, בהינתן מילון ודוגמא (pattern) מחזירה את כל המילים במילון המתאימות למחרוזת דוגמא. נממש את האפליקציה במחלקה שתקרא: DictionaryHelper  
הדוגמא היא מחרוזת אשר מכילה אותיות או את התו '-', אשר יסמן כי איננו יודעים איזו אות מופיעה במקום זה.
- i. ממשו את הפונקציה findMatches אשר מקבלת מילון (מערך של מילים) ומחרוזת דוגמא ומחזירה מערך המכיל את כל המילים המתאימות לדוגמא שסופקה.  
להלן מספר הרצות לדוגמא:

```
findMatches("m-d-m", dictionary) -> { "madam", "modem" }
findMatches("-n--r-m", dictionary) -> { "anagram", "interim" }
findMatches("cycl--", dictionary) -> { "cycled", "cycles", "cyclic" }
```

```
public static String[] findMatches(String pattern, String[] dictionary) {
}
```

ii. השלימו את המימוש כך שניתן יהיה להפעיל את השירות משורת הפקודה (command line). התוכנה הממומשת תקבל שני ארגומנטים: הראשון הוא שם הקובץ המכיל את המילים במילון והשני הוא מחרוזת הדוגמא. הפלט של התוכנית הוא רשימת המילים המתאימות לדוגמא, כאשר כל מילה מופיעה בשורה נפרדת.

**הערה ראשונה:** ניתן להשתמש במילון dictionary.txt המצוי בספרייה resources<sup>1</sup>.

**הערה שנייה:** על מנת לבדוק את הפונקציה main תוכנית הבדיקה משנה את System.out ולכן הדפסות לזרם זה לא ייראו על המסך וייפורשו על ידי תוכנית הבדיקה כפלט מהתוכנית.

2. (20 נקודות) נרצה לבנות אפליקציה למציאת אנאגרמה של משפט כללי. האפליקציה תקבל משפט (כלומר מחרוזת המכילה מספר מילים) ותדפיס סדרה של מילים חוקיות המורכבות מהאותיות שהופיעו במשפט המקורי. במימוש שלנו נסתפק בפלט של אנגרמה אחת עבור משפט קלט אחד. שימו לב: רווחים וסימני פיסוק אינם נחשבים כאות וניתן להתעלם מהם.

ע"מ לבצע את המטלה ביעילות, נייצג מילה (או משפט) ע"י מערך של מספרים, כאשר במקום ה-i יופיע מספר הפעמים בה האות ה-i בא"ב הלועזי (אנגלי) מופיעה במילה. להלן מספר דוגמאות לייצוג מחרוזות במערך:

abba

2	2	0	0	...	0
---	---	---	---	-----	---

abcccc

1	1	4	0	...	0
---	---	---	---	-----	---

<sup>1</sup> המילון הורד מהאתר: <http://www.mieliestronk.com/wordlist.html>

נשים לב, שבהינתן ייצוגים כנ"ל למשפט  $s$  ומילה  $w$ , המילה עשויה להיכלל באנאגרמה של המשפט רק אם ערכי כל תא בייצוג המילה קטנים או שווים מהערכים המתאימים בייצוג המשפט. כלומר, לכל תא  $i$  מתקיים:  $w[i] \leq s[i]$ .

בנוסף, נבחין כי אם המילה  $w$  היא אכן חלק מאנאגרמה של  $s$  אזי נוכל לקבל בעיה "קטנה" יותר בה יש לחפש אנאגרמה עבור משפט אחר, המורכב מאותיות  $s$  למעט אלו שהופיעו ב- $w$ . משפט כזה ייוצג ע"י מערך בו ערך התא במקום ה- $i$  הוא בדיוק  $s[i]-w[i]$ .

על סמך הבחנות אלו נוכל לממש את האלגוריתם הבא עבור משפט  $s$ :

- i. נבנה מילון של מילים חוקיות
- ii. נייצג את כל מילים במילון עפ"י הייצוג שתואר לעיל
- iii. נייצג את  $s$  עפ"י הייצוג הנ"ל
- iv. נעבור על רשימת המילים החוקיות ונבדוק האם ניתן לשלבן באנאגרמה:  
אם המילה הנוכחית עשויה להיות חלק מאנאגרמה, אזי:
  - (1) נבנה ייצוג חדש  $s'$ , בו  $s'[i]=s[i]-w[i]$ .
  - (2) אם בייצוג  $s'$  ערכי כל התאים הם  $\geq 0$ , אזי מצאנו אנאגרמה חוקית וסיימנו.
  - (3) אחרת, נפעיל שוב את שלב ד' (ברקורסיה) עם  $s'$ .

עליכם לממש את המחלקה `AnagramFinder` המיישמת את האלגוריתם המוצע. להלן דוגמאות הרצה של המחלקה:

```
> java AnagramFinder dictionary.txt "most of it"
moot fist
> java AnagramFinder dictionary.txt "challenge me"
helm glen ace
> java AnagramFinder dictionary.txt "dinosaurs"
runs adios
```

שאלות ותשובות :

- i. מהי רשימת המילים החוקיות?  
**תשובה:** נשתמש במילון כמו בסעיף הקודם.
  - ii. אילו אותיות הן חוקיות?  
**תשובה:** כולן, אולם אנו נקודד רק את האותיות המעניינות אותנו. נשתמש באותיות קטנות ונעזר במתודה isLetter של המחלקה Character.
  - iii. מה לעשות במקרה שיש מספר תשובות חוקיות?  
**תשובה:** מספיק לתת פתרון חוקי אחד.
  - iv. מה לעשות אם לא קיימת מילה המתאימה כחלק מהאנאגרמה באחד משלבי הרקורסיה?  
**תשובה:** במקרה זה לא מדפיסים כלום למסך.
  - v. נראה כי ניתן לייעל את הקוד, לדוגמא, מילה ארוכה לא יכולה להיות אנאגרמה של מילה קצרה יותר.  
**תשובה:** נכון, ניתן להשתמש באורך המחרוזת כדי לייעל את המימוש. ניתן, לדוגמא, להוסיף תא למערך שיכיל את מספר האותיות במילה במיוצגת ולהשתמש בערך זה במהלך החישוב.
- הערה:** על מנת לבדוק את הפונקציה main תוכנית הבדיקה משנה את System.out ולכן הדפסות לזרם זה לא ייראו על המסך וייפורשו על ידי תוכנית הבדיקה כפלט מהתוכנית.

### חלק ב' – המשחק Hangman

בחלק זה נממש את המשחק Hangman.

1. (15 נקודות) השלימו את מימוש המחלקה Hangman. המממשת את לוגיקת המשחק. להלן תיאור הפונקציות הנדרשות:
  - א. הבנאי Hangman מקבל מחרוזת המכילה את המילה אותה יש לנחש.
  - ב. הפונקציה guessLetter() מבצעת ניחוש של אות אחת. ומחזירה מספר המבטא את תוצאת הניחוש: 0 האות אינה מופיעה במילה, 1 האות מופיעה במילה לפחות פעם אחת, ו-2 אם הניחוש אינו חוקי (כלומר איננו אות בא"ב האנגלי או אות שנוסחה בעבר). יש להשתמש בקבועים GUESS\_WRONG, GUESS\_SUCCESSFUL ו-GUESS\_ILLEGAL (בהתאמה) המוגדרים במחלקה.
  - ג. הפונקציה getGuess() מחזירה מחרוזת המכילה את הניחושים הנכונים במקומם ובמקום האותיות החסרות את התו '-.

ד. הפונקציה `getMisses()` מחזירה מחרוזת המכילה את כל הניחושים השגויים (מסודרים לפי א"ב).

ה. הפונקציה `isOver()` מחזירה `true` אם ורק אם המילה נתגלתה במלואה.

דוגמא:

```
Hangman game = new Hangman("Hello");
game.getGuess() -> "-----"
game.getMisses() -> ""
game.isOver() -> false
```

```
game.guessLetter('h')
game.getGuess() -> "h----"
game.getMisses() -> ""
game.isOver() -> false
```

```
game.guessLetter('a')
game.getGuess() -> "h----"
game.getMisses() -> "a"
game.isOver() -> false
```

```
game.guessLetter('r')
game.getGuess() -> "h----"
game.getMisses() -> "ar"
game.isOver() -> false
```

```
game.guessLetter('c')
game.getGuess() -> "h----"
game.getMisses() -> "acr"
game.isOver() -> false
```

```
game.guessLetter('l')
game.getGuess() -> "h-ll-"
game.getMisses() -> "acr"
game.isOver() -> false
```

```
game.guessLetter('p')
game.getGuess() -> "h-ll-"
game.getMisses() -> "acpr"
game.isOver() -> false
```

```
game.guessLetter('e')
game.getGuess() -> "hell-"
game.getMisses() -> "acpr"
game.isOver() -> false
```

```
game.guessLetter('o')
game.getGuess() -> "hello"
game.getMisses() -> "acpr"
game.isOver() -> true
```

2. (30 נקודות) השלימו את המחלקה HangmanTextUI. המחלקה מציגה ממשק טקסט למשחק ומאפשרת שישה נחושים שגויים.  
**חשוב:** תוכנית הבדיקה לא בודקת סעיף זה.

- א. את המחלקה מפעילים משורת הפקודה.
- ב. המחלקה טוענת את המילון מסעיף א' ובוחרת מתוכו מילה באופן אקראי (יש להעזר במחלקה Random של Java).
- ג. המחלקה מאתחלת מופע של המחלקה Hangman ומשתמשת בו כדי לעקוב אחר הניחושים של המשתמש. זרם הקלט נלקח מ-System.in וכל הפלט נכתב ל-System.out (אילו הם אמצעי התקשורת שלנו עם השחקן).
- ד. המחלקה מציגה את "ציור" האיש, מתחתיו את הניחושים השגויים שבוצעו עד כה (אם יש כאלו) ואת הניחוש הנוכחי. לאחר מכן מבקשת התוכנית מהמשתמש לנחש אות נוספת ומחכה לקלט וחוזר חלילה, עד אשר אחד מהאירועים הבאים מתרחש:
  - i. המשתמש גילה את המילה, במקרה זה תוצג הודעה "You Won!".
  - ii. מספר השגיאות המקסימלי הושג. במקרה תוצג ההודעה "You Lost!".
  - iii. זרם הקלט מסתיים לפני תום המשחק. במקרה זה תוצג ההודעה "You Lost!".

שתי דוגמאות הרצה (example1.txt ו-example2.txt) נתונות בספרייה resources.

3. (25 נקודות) נשדרג את המחלקה HangmanTextUI לתמיכה במשחק Hangman של שני שחקנים. הקוד שנבנה יאפשר לשני השחקנים לשחק ממחשבים שונים המחוברים ברשת, אולם, יהיה ניתן גם לעבוד על גבי מחשב יחיד. **למעשה, אנו נכתוב תוכנת שרת (בדומה לשרת web) אשר תשמש את אחד השחקנים ואילו השחקן השני ישתמש בתוכנת תקשורת סטנדרטית כדי ליצור קשר עם השרת.**

**חשוב:** תוכנית הבדיקה לא בודקת סעיף זה.

א. חוקי המשחק לשני שחקנים:

בדומה לסעיף (2) המחלקה טוענת מילון ובוחרת מילה באופן אקראי. מרגע זה המשחק מתנהל בתורות (שחקן 1 תמיד משחק ראשון). כל שחקן מנחש אות, אם האות שניחש אינה מופיעה במילה, התור עובר לשחקן השני. שחקן שמצליח לנחש את המילה מנצח (ובמקרה זה השחקן השני מפסיד). לאחר שישה ניחושים שגויים (הנספרים לשני השחקנים יחדיו) המשחק מסתיים ושני השחקנים מפסידים.

ב. תקשורת עם השחקנים:

a. שחקן 1 יפעיל את התוכנה שנבנה, הקלט ממנו מתקבל ב-System.in והפלט אליו נרשם ל-System.out, בדומה לסעיף הקודם.

b. שחקן 2 יתחבר לתוכנה שלנו באמצעות תוכנת תקשורת סטנדרטית. נשתמש במחלקה (המסופקת) GameServer לתקשורת עם השחקן הזה בצורה הבאה:

i. הבנאי של GameServer מקבל מספר port וממתין לחיבור של שחקן 2.

ii. השירותים `getInputStream()` מחזיר זרם לקבלת קלט מהמשתמש (בדומה ל-`System.in`).

iii. השירותים `getOutputStream()` מחזיר זרם אשר יעביר נתונים אל המשתמש (כמו `System.out`).

ג. הפעלת התוכנה:

כאשר מפעילים את המחלקה משורת הפקודה ללא פרמטרים נוספים, התוכנית פועלת כמו בסעיף (2).

כאשר מפעילים את המחלקה עם הפרמטר הנוסף "--server" המחלקה תעבוד כשרת ותחכה לחיבור של שחקן 2.

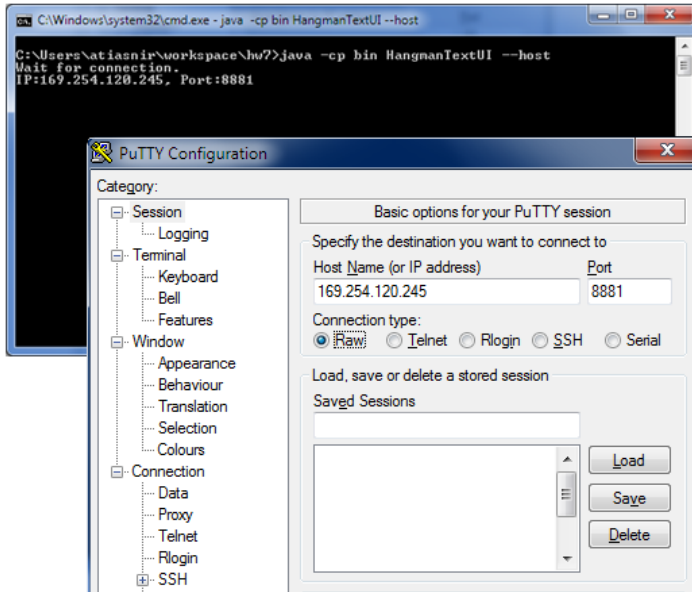
**הערה 1:** כללית, כדי לפתוח ערוץ תקשורת אל אפליקציה אחרת, נדרשת כתובת המורכבת ממזהה למחשב (כתובת IP) ומזהה לאפליקציה הנקרא port. האפליקציה שלנו מאזינה ל-port מספר 8881. המחלקה GameServer מדפיסה ל-System.out את כתובת ה-IP ומספר ה-port אליהם היא מאזינה. ניתן להעתיק אותם אל אפליקציית התקשורת אתה נשתמש.

**הערה 2:** להלן שתי שיטות להתחבר אל התוכנה שכתבנו:

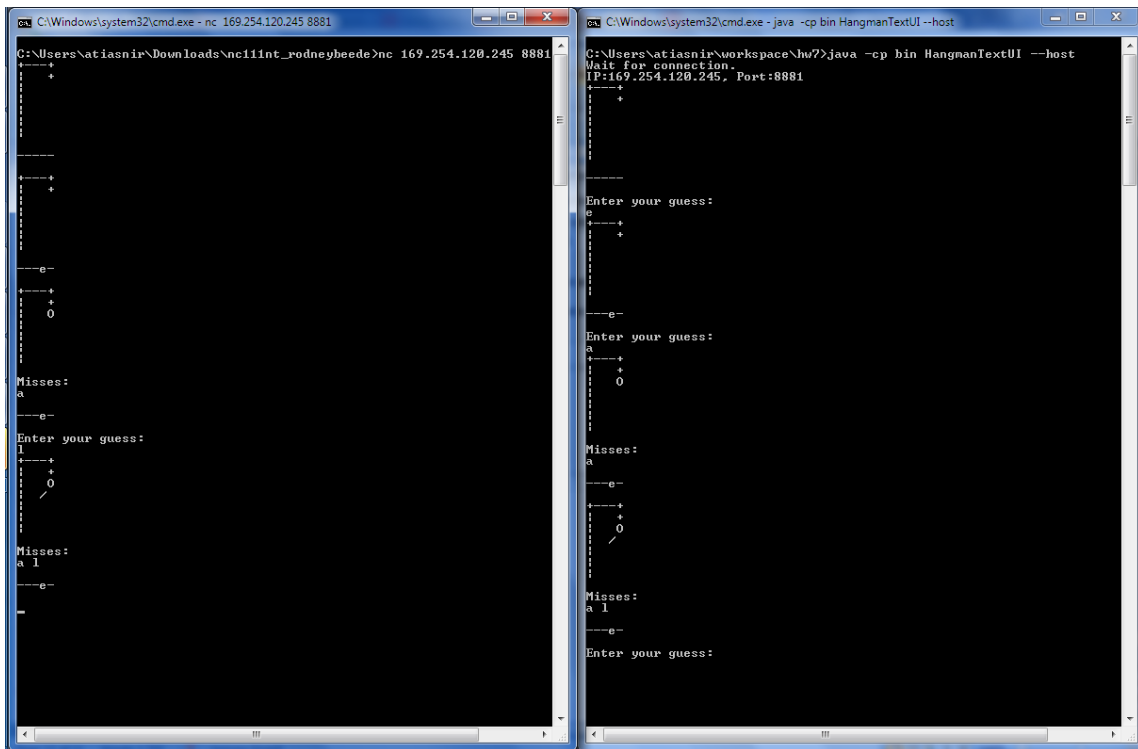
א. במחשבי linux ניתן להשתמש בתוכנה netcat (זמינה גם ב-Windows, וכן התוכנה Ncat בעלת ממשק גרפי). תוכנה זו עובדת משורת הפקודה ומקבלת כארגומנטים את כתובת ה-IP ואת מספר ה-port של האפליקציה. דוגמא:

```
$ nc 169.254.120.245 8881
```

ב. ניתן להשתמש בתוכנה Putty (המשמשת גם להתקשרות לשרתי האוניברסיטה). בשימוש ב-Putty יש להקפיד להזין את מספר ה-port וכן שסוג החיבור (Connection type) יהיה Raw, כפי שמוצג בתמונה הבאה.



דוגמא להרצת המשחק. דוגמא נוספת (סרט ווידאו המדגים את ריצת התוכנית) זמינה באתר הקורס.



ב ה צ ל ח ה