

כתב וערך
לירון בדיחי

badihil@gmail.com
050-8756869



מבוא

חוברת זו מהווה פריצת דרך בנושא הכנה לבגרות במדעי המחשב, מכמה סיבות:

סיבה ראשונה

דרך מקורית לפתירת תרגילים עם לולאות ומערכים: יש לי תואר למדעי המחשב. לימדתי הרבה שיעורים פרטיים, עבדתי בבית תוכנה וכתבתי הרבה תוכניות. חשבתי לעצמי: "כתיבת לולאות זה דבר מסובך. במיוחד אם נצרכת לולאה בתוך לולאה. גם מערכים חד מימדיים דורשים לולאות ומערכים דו מימדיים דורשים לולאות בתוך לולאות. בתור מורה, האם המשימה שלי היא להראות לתלמיד את הפיתרון לתרגיל מסובך ולהסביר לו למה זה פועל (כמו שלימדו אותי)? חייבת להיות דרך פשוטה להגיע לתשובות המסובכות האלה!"

ישבתי וחשבתי. וה' סייע בידי ופיתחתי דרך חשיבה מקורית לכל לולאה (מורכבת ככל שתהיה). קראתי לה "דרך שלושת השלבים".

דרך זו לפתרון לולאות הלהיבה את כל התלמידים שלימדתי אותם. דרך זו נותנת את האפשרות לפתור לולאה בלי להשקיע יותר מידי מחשבה ולהגיע לתוצאה הנכונה מבלי להתעמק בתוצאה הסופית. לדעתי, אין דרך שמפשטת את התרגיל יותר מהדרך שלי. אם למישהו יש רעיון טוב יותר – אשמח לשמוע.

סיבה שנייה

חוברת קצרה וקולעת: חוברת זו הינה חוברת הכנה לבגרות ליחידות 2 ו-3. חשוב לי לציין, שאת החוברת כתבתי מעצמי ולא עיינתי בהסברים של שום ספר קודם להכנה לבגרות. וכן, הרבה מהתרגילים המצאתי. מה שהנחה אותי: כתיבת חוברת קצרה שמספיקה ללימוד החומר כולו ומספקת הסברים ותרגילים. חוברת שנוכל לגמור מהר ונתחיל לפתור בגרויות. אני מקווה שאתם מבינים שתלמידים לא ינסו אפילו לקרוא את הספרים הארוכים שיש היום (מעל ל-200 עמודים) ! לעומת חוברת זו שיש בה כ-40 עמודים בלבד !

כתבתי המשך לחוברת בנושאים של יחידות מתקדמות (רקורסיה, מחלקות, פונקציה בונה, הרשאות גישה וכו'). אך הפרדתי את ההמשך מחוברת זו, כדי להתמקד ביחידות 2 ו-3 בלבד.

הייתי רוצה להכיר את החוברת לכמה שיותר מורים ולקבל תגובות, הערות, ורעיונות לשיפורים. אני יודע שפרצתי פה דרך חדשה וכדאי למורים, למחברי הספרים, ובעיקר... לתלמידים להכיר אותה !

בתודה:

לירון בדיחי: 050-8756869

אימייל: badihil@gmail.com

תוכן

| | |
|----|---|
| 5 | הטיפוסים הבסיסיים |
| 6 | עץ הטיפוסים הבסיסיים |
| 8 | הדפסה על המסך |
| 8 | קליטת נתונים מהמסך/מהמשתמש |
| 9 | תרגילים – קלט/פלט |
| 10 | שימוש במחלקות Math, Random |
| 10 | Random - מנגנון לייצור מספרים אקראיים ("הגרלת מספרים" בטווח מסויים) |
| 11 | ביטויים חשבוניים |
| 11 | תרגילים – ביטויים חשבוניים |
| 12 | תנאים |
| 13 | דוגמא ראשונה - תנאי רגיל |
| 13 | דוגמא שניה – תנאי מורכב (משני תנאים) |
| 13 | דוגמא שלישית - תנאי מקונן (תנאי בתוך תנאי) |
| 14 | בלוקים |
| 14 | קדימויות בבלוקים |
| 15 | תרשים זרימה / אלגוריתם |
| 16 | תרגילים – תנאים |
| 17 | טבלת מעקב |
| 18 | לולאות |
| 18 | שימוש ראשון: חיסכון בכתיבת פקודות |
| 18 | לולאת while |
| 18 | לולאת for |
| 20 | תרגילים – לולאות (שימוש ראשון) |
| 21 | שימוש שני: שימוש במספרים הרצים שמייצרת הלולאה |
| 21 | דרך כללית לפתרון כל לולאה – שיטת שלושת השלבים |
| 21 | דוגמא ראשונה |
| 21 | דוגמא שניה |
| 22 | דוגמא שלישית |
| 24 | תרגילים – לולאות (שימוש שני) |
| 25 | לולאה בתוך לולאה |
| 25 | דוגמא |
| 27 | דוגמא (מתקדמת) |
| 29 | תרגילים – לולאה בתוך לולאה |
| 30 | פונקציות |
| 30 | דוגמא |
| 31 | המבנה של פונקציה |
| 31 | תיעוד |
| 32 | תרגילים – פונקציות |
| 33 | switch |
| 33 | דוגמא ראשונה |
| 34 | דוגמא שניה |
| 35 | מערך חד מימדי |
| 35 | מהו מערך? |
| 35 | אינסק המערך |
| 36 | איך המערך נשמר בזכרון |
| 37 | 2 אפשרויות להגדיר מערך |
| 37 | גודל המערך |
| 38 | דוגמא |
| 39 | תרגילים – מערך חד מימדי |
| 41 | מחרוזות ותווים – בסיס |

| | |
|----|--|
| 41 |דוגמא |
| 41 |תרגילים – מחרוזות |
| 42 |מערך דו מימדי |
| 42 |3 אפשרויות להגדיר ממעריך דו מימדי |
| 43 |גודל המעריך |
| 44 |דוגמא |

הטיפוסים הבסיסיים

אנחנו רגילים לספור בבסיס דצימלי:

לדוגמא, המספר 126: $10^7 + 10^6 + 10^5 + 10^4 + 10^3 + 10^2 + 10^1 + 10^0$

| | | | | | | | |
|-----|------|-----|----|----|----|----|----|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| ... | 1000 | 100 | 10 | 1 | | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 | 6 |

ספירה בבסיס 16, הנקרא בסיס הקסה:

לדוגמא, המספר 20: $16^7 + 16^6 + 16^5 + 16^4 + 16^3 + 16^2 + 16^1 + 16^0$

| | | | | | | | |
|-----|------|-----|----|----|----|----|----|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 |
| ... | 4096 | 256 | 16 | 1 | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 4 |

הסימון: #14

המחשב סופר בבסיס בינארי (בסיס 2):

מספר אפשרויות

255 =

| | | | | | | | |
|-----|----|----|----|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

byte - 8 ביטים

עוד טיפוסים:

מספר אפשרויות

65535 =

| | | | | | | | |
|----|----|----|----|----|----|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

short - 16 ביטים

מספר אפשרויות

4,294,967,295 =

(10 ספרות)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

int - 32 ביטים

מספר אפשרויות

18,446,766,073,709,551,615 =

(20 ספרות)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

long - 64 ביטים

כל מספר יכול להיות מוצג בכל בסיס.

הסיבה שהמחשב סופר בצורה בינארית, כי הוא בנוי מסיביות. כל סיבית יכולה להיות או דלוקה (1) או כבויה (0) לכן לכל סיבית יש 2 אפשרויות בלבד. הצגנו גם את בסיס "הקסה", כי מתכנתים נפגשים עם מספרים מהסוג הזה כל הזמן.

נפתח את היישום "מחשבון" (התחל ← כל התוכניות ← עזרים ← מחשבון), נעבור לתצוגה מדעית ונבדוק את המעבר בין הבסיסים.

בתוכניות שלנו ובספרים משתמשים רק בטיפוס `int` כדי להסיר מעליכם את השיקול במה להשתמש בכל פעם.

במציאות, כדאי לחשב במה עדיף להשתמש.

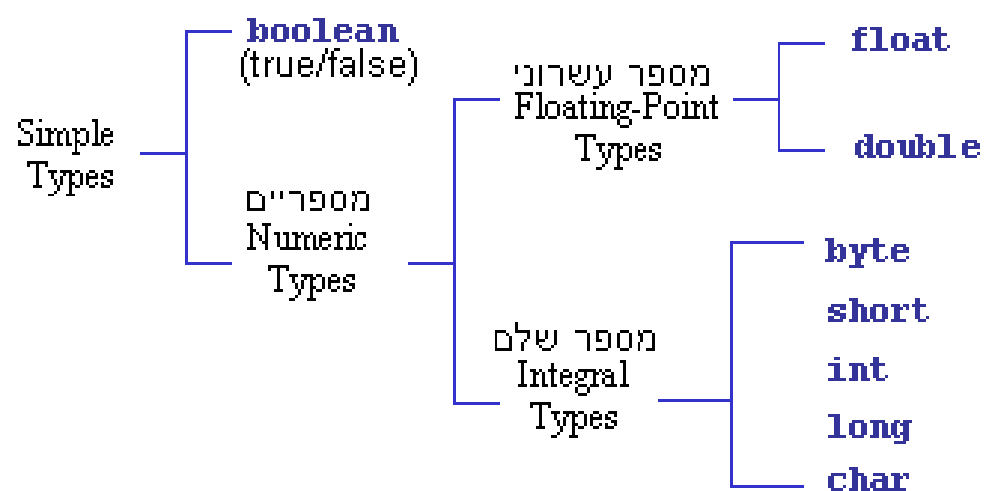
שאלה:

מדוע לא נשתמש בטיפוס הגדול ביותר תמיד ?

תשובה:

כיוון שיש מה שנקרא "ביצועים". ככל שהטיפוס גדול יותר כך הוא תופס מקום יותר גדול בזכרון המחשב ואם לא נתרגל לחשוב בצורה חסכונית, יכול להיות שיגמר המקום במחשב ואז הביצוע של התוכנית יכשל. לכן כדאי לנו תמיד לחשוב בצורה חסכונית.

עץ הטיפוסים הבסיסיים



`char` הינו "תו". מדוע הוא מספר שלם?

כיוון שכל תו מיוצג על ידי מספר שלם, ובכל מחשב יש טבלת תווים: עבור כל מספר – איזה תו הוא מייצג. לדוגמא, המספר 97 עבור 'a' והמספר 65 עבור 'A' והמספר 32 עבור רווח והמספר 48 עבור '0' וכן הלאה...

יש טבלת תווים הנקראת **ASCII** עבור 255 תווים הכוללת רק את האותיות באנגלית ואת שאר התווים במקלדת. מספיק לה `byte` אחד (כי `byte` אחד יש 255 אפשרויות, כפי שראינו בעמוד הקודם). יש טבלת תווים הנקראת **Unicode** עבור כל האותיות של כל השפות בעולם ושאר התווים במקלדת. היא צריכה שני `bytes`. סה"כ – 65,535 אפשרויות.

לאחר שראינו שגודל הטיפוס מוגדר בהתאם למספר הביטים, ננסה להבין את האינפורמציה בטבלה:

| הטיפוס | סוג הערך | מספר הביטים | ברירת המחדל | ערכי מינימום ומקסימום |
|--|--|-------------------|---------------------|---|
| <u>boolean</u> | false או true | 1 | false | true או false |
| <u>char</u> | תו (ב-Unicode) | 16 | #0000 | #0000 עד #FFFF |
| <u>String</u> (אינו טיפוס בסיסי) | מחרוזת תווים | תלוי במספר התווים | "" (מחרוזת ריקה) | |
| byte | מספר שלם חיובי או שלילי או אפס | 8 | 0 | -128 עד 127 |
| short | מספר שלם חיובי או שלילי או אפס | 16 | 0 | -32768 עד 32767 |
| <u>int</u> | מספר שלם חיובי או שלילי או אפס | 32 | 0 | -2147483648 עד 2147483647 |
| long | מספר שלם חיובי או שלילי או אפס | 64 | 0 | ממספר שלם מאוד גדול וחיובי עד מספר שלילי קטן במיוחד |
| float | מספר ממשי (יכול להיות שלם ויכול להיות עם שבר עשרוני) | 32 | 0.0 | דיוק של 7 מקומות עשרוניים |
| <u>double</u> | מספר ממשי (יכול להיות שלם ויכול להיות עם שבר עשרוני) | 64 | 0.0 | דיוק של 15-16 מקומות עשרוניים |
| Object (אינו טיפוס בסיסי) | מחלקת הבסיס של כל הטיפוסים | | | |

הדפסה על המסך

- `System.out.println("...");` - כתיבת שורה למסך, יורד שורה.
- `System.out.print("...");` - כתיבת שורה למסך, לא יורד שורה.

קליטת נתונים מהמסך/מהמשתמש

השלב הראשון לקליטת נתונים מהמשתמש הוא קבוע: הצהרה על משתנה מסוג `Scanner`:
`Scanner input = new Scanner(System.in);`
המחלקה שלנו לא מכירה את המחלקה `Scanner` כי זה לא נמצא באותה חבילה שלה. לכן נצטרך על החבילה ש-`Scanner` נמצאת בה. נכתוב בתחילת התוכנית (לאחר הגדרת ה-package):
`import java.util.*;`

להלן דוגמה לקלט עבור כל אחד מן הטיפוסים הבסיסיים:

```
public static void main(String[] args)
{
    Scanner input = new Scanner(System.in);

    // קלט מספר מסוג שלם:
    System.out.println("Enter an integer number: ");
    int i = input.nextInt();

    // קלט מספר מסוג עשרוני:
    System.out.println("Enter an double number: ");
    double d = input.nextDouble();

    // קלט מספר בולאני:
    System.out.println("Enter an boolean value (true/false): ");
    boolean b = input.nextBoolean();

    // קלט של מחרוזת:
    System.out.println("Enter String: ");
    String str = input.next();

    // קלט של תו:
    System.out.println("Enter char: ");
    char c = input.next().charAt(0);

    // הדפסה:
    System.out.println("Your number is: " + i);
    System.out.println("Your number is: " + d);
    System.out.println("Your value is: " + b);
    System.out.println("Your string is: " + str);
    System.out.println("Your char is: " + c);
}
```


תרגילים – קלט/פלט

- (1) קלוט מחרוזת והדפס אותה.
- (2) קלוט מספר שלם והדפס אותו.
- (3) קלוט מספר עשרוני והדפס אותו.
- (4) קלוט מספר בולאני והדפס אותו.
- (5) קלוט תו והדפס אותו.
- (6) קלוט מהמשתמש מחרוזת והדפס את התו הרביעי שלה (שים לב, הספירה מתחילה מ-0).
- (7) קלוט שם (name), גיל (age), גובה (height), מין (gender - 'F' עבור נקבה ו-'M' עבור זכר), והאם המשתמש אוהב סלט (likeSalad), והדפס את הנתונים בשורות נפרדות.
דוגמא לפלט:

```
name: yoni  
age: 16.5  
height: 1.80  
gender: M  
likeSalad: true
```

- (8) מה יוצג על המסך?

```
int numOfStudents = 23;  
System.out.println(numOfStudents);  
  
int x = 5;  
int y = x + 3;  
System.out.println(y);
```

- (9) קלוט מספר שלם, הוסף לו 10 והדפס אותו.
- (10) קלוט מחרוזת, קלוט תו. הדפס בשורה אחת את התו, המחרוזת ושוב את התו, עם רווחים מפרידים.

שימוש במחלקות Math, Random

Random - מגנון לייצור מספרים אקראיים ("הגרלת מספרים" בטווח מסויים).

```
Random rnd = new Random();  
int num = rnd.nextInt(8);
```

שאלה: איזה מספרים אקראיים המשתנה num יכול לקבל?
תשובה: מספר אקראי בין 0 (כולל) לבין 8 (לא כולל) המספר 8.

```
Random rnd = new Random();  
int num = rnd.nextInt(8)+5;
```

שאלה: איזה מספרים אקראיים המשתנה num יכול לקבל?
תשובה: מספרים אקראיים בין 5 (כולל) לבין 13 (לא כולל) 13.

איך נחשב זאת בעצמנו?

שאלה: הגרל מספר בין 5 ל-12.

תשובה: - שלב ראשון: נחשב כמה מספרים יש בין 5 ל-12 (כולל): $12-5+1$: סה"כ 8 מספרים.

- שלב שני: נכתוב `rnd.nextInt(8)`


- שלב שלישי: נוסיף לתוצאה את המספר 5 (המספר הקטן): $rnd.nextInt(8)+5$

תרגילים

- 1) בנה תוכנית שתגריל מספר בטווח המספרים מ-0 עד 10 ותדפיס אותו.
- 2) כתוב תוכנית שתגריל מספר בטווח המספרים מ-2 עד 3 ותדפיס אותו.
- 3) קלוט מספר שלם מהמשתמש, הגרל מספר בטווח המספרים מ-0 עד המספר של המשתמש והדפס אותו.
- 4) קלוט מספר שלם מהמשתמש, הגרל מספר בטווח המספרים מ-5 עד המספר של המשתמש ועוד 5, והדפס אותו.

Math – מחלקה שמספקת פעולות מתמטיות רבות:

שימוש: נכתוב את המילה Math ולאחריה נקודה, ותפתח לנו רשימת הפונקציות.
בכל פונקציה רואים מה היא מחזירה:

 `abs(int arg0)` `int` Math

הפונקציה הזאת מחזירה טיפוס מסוג `int` (מספר שלם), לכן צריך לקלוט את התוצאה לתוך משתנה מטיפוס שלם:

```
int i = Math.abs(-5);
```

תרגילים (העזר בפונקציות של המחלקה Math):

- 1) הדפס כמה זה פאי (PI).
עבור המספר 10.24:
- 2) הדפס את השורש שלו (`sqrt`).
- 3) הדפס את הריבוע (`pow`) שלו.
- 4) הדפס את העיגול שלו (`round`).
- 5) קלוט שני מספרים והדפס את הקטן מביניהם (`min`).
- 6) קלוט שני מספרים. הדפס את השורש (`sqrt`) של הגדול מביניהם ואת הריבוע (`pow`) של הקטן מביניהם.
- 7) הגרל שני מספרים בין 1 ל-100. הדפס את שני המספרים, ואת הגדול (`max`) מביניהם.

ביטויים חשבוניים

- האופרטורים +, -, * משמשים לחיבור, חיסור, כפל
- האופרטור / מבצע חילוק – חלוקה בין מספרים שלמים תהיה תמיד מספר שלם !
- דוגמא: $10 / 3 = 3$
- האופרטור % משמש לקבל שארית מחלוקה בין מספרים שלמים

דוגמא:

| שארית | תוצאה |
|--------------|-------------|
| $6 \% 2 = 0$ | $6 / 2 = 3$ |
| $6 \% 3 = 0$ | $6 / 3 = 2$ |
| $6 \% 4 = 2$ | $6 / 4 = 1$ |
| $6 \% 5 = 1$ | $6 / 5 = 1$ |

| שארית | תוצאה |
|--------------|-------------|
| $7 \% 2 = 1$ | $7 / 2 = 3$ |
| $7 \% 3 = 1$ | $7 / 3 = 2$ |
| $7 \% 4 = 3$ | $7 / 4 = 1$ |
| $7 \% 5 = 2$ | $7 / 5 = 1$ |
| $7 \% 6 = 1$ | $7 / 6 = 1$ |

- $()$ לסוגריים הקדימות הגבוהה ביותר
- *, /, % קדימות גבוהה
- +, - קדימות נמוכה

דוגמא:

```
int a=5, b=6;
int c,d;
c = (a + 2)*b; // = 42
d = 30 / 6 / 3; // = 1
```

תרגילים – ביטויים חשבוניים

הגדר שני משתנים שלמים: a, b וטען לתוכם את הערכים 6, 4 בהתאמה, והדפס את:

- (1) החיבור שלהם.
- (2) החילוק של הראשון בשני.
- (3) הכפל שלהם.
- (4) הריבוע של הסכום שלהם (מבלי להשתמש במחלקה Math).

הגדר שלושה משתנים שלמים: a, b, c וטען לתוכם את הערכים 6, 3, 2 בהתאמה, והדפס את:

- (5) שארית החילוק של a+b עם c.
- (6) חלק את a עם b והכפל את התוצאה עם c+3.
- (7) זכוב מצליח לעצבן 5 אנשים כל בדקה. כתוב תוכנית הקולטת את מספר הדקות ומדפיסה כמה אנשים הצליח הזכוב לעצבן.
- (8) במאפיה שלושה טבחים. טבח ראשון אופה לחמניה אחת בדקה. טבח שני אופה שתי לחמניות בדקה. טבח שלישי אופה 5 לחמניות בדקה. כתוב תוכנית הקולטת מספר דקות ומדפיסה את מספר הלחמניות המוכנות.
- (9) קלוט מספר עשרוני. הדפס את החלק העשרוני שלו.

תרגילים נוספים: מבט לחלונות חלק א – עמוד 59 תרגילים 6, 7, 11:

תנאים

מבנה הוראה לביצוע מותנה :

```
if ( תנאי )  
{  
    .  
    .  
}  
else  
{  
    .  
    .  
}
```

אם (תנאי) אזי
הוראות לביצוע 1
אחרת
הוראות לביצוע 2

אחרי if לא כותבים ";" אלא בפקודה שבתוכו.

בתוך תנאי יכולים להופיע אופרטורי ההשוואה הבאים:

| המשמעות | דוגמא לשימוש | האופרטור |
|---|--------------|----------|
| האם ערכו של A קטן מערכו של B ? | $A < B$ | < |
| האם ערכו של A גדול מערכו של B ? | $A > B$ | > |
| האם ערכו של A קטן או שווה לערכו של B ? | $A <= B$ | <= |
| האם ערכו של A גדול או שווה לערכו של B ? | $A >= B$ | >= |
| האם ערכו של A שווה לערכו של B ? | $A == B$ | == |
| האם ערכו של A שונה מערכו של B ? | $A != B$ | != |

אפשר לשלב כמה תנאים ביחד בעזרת האופרטורים הבאים:

| המשמעות | דוגמא לשימוש | האופרטור |
|--|--------------------------|----------|
| האם ערכו של התנאי שווה false ? | (תנאי) ! | ! |
| האם ערכו של תנאי א שווה true וגם ערכו של תנאי ב שווה true ? | ((תנאי א) && (תנאי ב)) | && |
| האם ערכו של תנאי א שווה true או ערכו של תנאי ב שווה true ? | ((תנאי א) (תנאי ב)) | |
| האופרטור ! הוא בעל הקדימות הגבוהה יותר, אחריי האופרטור && , והאופרטור הוא בעל הקדימות הנמוכה ביותר. | | |

דוגמא ראשונה - תנאי רגיל

קלוט ציון. אם הציון קטן מ-55 הדפס "לא עבר", אחרת הדפס "עבר".

```
Scanner in = new Scanner(System.in);
System.out.println("Enter grade: ");
int grade = in.nextInt();
```

```
if (grade < 55)
{
    System.out.println("lo avar");
}
else
{
    System.out.println("avar");
}
```

דוגמא שנייה – תנאי מורכב (משני תנאים)

קלוט מספר. בדוק אם הוא מתחלק ב-2 וב-3. הדפס תשובה בהתאם.

```
Scanner in = new Scanner(System.in);
System.out.println("Enter number: ");
int number = in.nextInt();
```

```
if ((number % 2 == 0) && (number % 3 == 0))
{
    System.out.println(number + " divided");
}
else
{
    System.out.println(number + " is not divided");
}
```

דוגמא שלישית - תנאי מקונן (תנאי בתוך תנאי)

כתוב תכנית הקולטת שלשה מספרים, בודקת ומדפיסה את המספר הקטן מבין שלושת המספרים.

```
Scanner in = new Scanner(System.in);
int num1, num2, num3;
```

```
System.out.println("Enter first number:");
num1 = in.nextInt();
```

```
System.out.println("Enter second number:");
num2 = in.nextInt();
```

```
System.out.println("Enter third number:");
num3 = in.nextInt();
```

```
if ((num1 < num2) && (num1 < num3))
{
    System.out.println("The small number is:" + num1);
}
else
{
    if ((num2 < num1) && (num2 < num3))
    {
        System.out.println("The small number is:" + num2);
    }
    else
    {
        if ((num3 < num1) && (num3 < num2))
        {
            System.out.println("The small number is:" + num3);
        }
    }
}
```

בלוקים

למחלקה תמיד פותחים בלוק.

לפונקציה תמיד פותחים בלוק.

ל-if לא חייבים תמיד לפתוח בלוק (עדיף תמיד). מתי חייבים לפתוח בלוק?

כאשר יש יותר מפקודה אחת.

לדוגמא:

אם התנאי לא יתקיים – שתי הפקודות לא יתבצעו -

```
if (תנאי)
{
    System.out.println("hello");
    System.out.println("by");
}
```

אם התנאי לא יתקיים – הפקודה השניה תתבצע, כי היא לא חלק מהתנאי -

```
if (תנאי)
    System.out.println("hello");
System.out.println("by");
```

תנאי if נחשב לפקודה אחת. לכן בדוגמא הבאה, לא חייבים לפתוח בלוק ל-if החיצוני:

```
if (תנאי)
    if(תנאי)
    {
        ...
    }
```

קדימויות בבלוקים

כלל ה"הזחה": כל הקוד בתוך בלוק מסוים – מוזזים קצת פנימה.

אם בזמן הכתיבה לא ביצענו הזחה, נלחץ בתפריט על **Edit** ← **Advances** ← **Format**

Document והמערכת תבצע לנו את ההזחה אוטומטית.

כלל: כל משתנה שהגדרנו בבלוק חיצוני – יוכר גם בבלוק הפנימי.

משתנה שהוגדר בבלוק פנימי – יהיה מוכר אך ורק באותו בלוק ולא יהיה מוכר מחוץ לו!

דוגמא:

```
public class Test
{
    static int x = 20;

    static void main(String[] args)
    {
        int j = 5;
        if (j < 5)
        {
            System.out.println("hello");
            int i = 10;
            j = j + 1;
        }
        // שגיאה: משתנה מקומי שהוגדר בבלוק מסוים, לא מוכר מחוץ לו!
        x = x + 1;
    }

    public static void go()
    {
        x = x + 1;
        // שגיאה: משתנה מקומי שהוגדר בבלוק מסוים, לא מוכר מחוץ לו!
    }
}
```

תרשים זרימה / אלגוריתם

כאשר ניגשים לתוכנית מורכבת, לא מומלץ לגשת לכתיבת הקוד מיד, אלא מומלץ לתכנן אותה תחילה. לתכנון הזה קוראים גם "תרשים זרימה". והכוונה: כתיבה של מה שאנחנו הולכים לעשות בשפה שלנו.

תרשים הזרימה עבור הדוגמא הראשונה מהעמוד הקודם:
קלוט ציון. אם הציון קטן מ-55 הדפס "לא עבר", אחרת הדפס "עבר".

קלוט ציון למשתנה ששמו *grade*

אם $grade < 55$

הדפס "לא עבר"

אחרת

הדפס "עבר"

- באיזה מילים יש להשתמש?
עבור קלט והדפסה, במילים: "קלוט", "הדפס".
עבור תנאי במילים: "אם" ו"אחרת".
- יש להקפיד על כלל ההזחה גם בתרשים הזרימה.

שאלה:

מה תרשים הזרימה הועיל? הרי חזרנו בעצם על המילים של התרגיל !

תשובה:

זה נועד למקרים יותר מורכבים בהם קל יותר (כמעט חובה) לחשוב ולתכנן מה לעשות במילים שלנו, ואח"כ להפוך כל שורה מתרשים הזרימה לקוד.

תרגילים – תנאים

(1) פקיד בנק עובד במילוי טפסים. עבור כל טופס הוא מקבל שכר של 5.3 ₪. כתוב תרשים זרימה הקולט מספר טפסים ומדפיס את השכר של הפקיד.

(2) לפניך קטע תוכנית:

```
int num = 10;
if ((a + b) > 0) num = 7;
else if (a > 0) num = 19;
else num = 50;
```

מה יהיה ערכו של המשתנה num לאחר ביצוע הקטע הנ"ל, עבור הערכים הבאים:

א $a=5; b=-4$; ב $a=3; b=-8$; ג $a=-8; b=3$

(3)

בהנחה שערכי המשתנים A B C D הם : 0 -3 8 5 , מה תהיה תוצאת הביטויים הבוליאניים הבאים: true או false

$(A > B) \ \&\& \ (C == D)$

$(A == 10 / 2)$

$(A >= D) \ \&\& \ (C > B)$

$(C < D) \ || \ (B < A)$

$(A < B) \ \&\& \ (C <= B \% 2) \ || \ (C > 0)$

(4) כתוב את התנאים הבאים המנוסחים במילים לביטויים בוליאניים :

א. ערכו של המשתנה x גדול מ 0 וקטן מ 50.

ב. ערכו של המשתנה a גדול מ 0 או ערכו של המשתנה x קטן מ 10.

ג. ערכו של המשתנה letter אינו התו 'A' ואינו התו 'Z'.

ד. ערכו של המשתנה letter שווה לתו 'A' או שווה לתו 'Z'.

ה. ערכו של המשתנה x שווה לערכו המוחלט וגדול מ 4.

ו. ערכו של המשתנה d גדול מ 0 וערכו של המשתנה x קטן מ 100.

ז. ערכו של המשתנה a גדול מ 0 או ערכו של המשתנה x קטן מ 10.

ח. ערכו של המשתנה num גדול מ 100 או שערכו של המשתנה sum קטן מ 191.

(5) כתוב תוכנית הקולטת שני מספרים ומדפיסה אותם בסדר עולה.

(6) כתוב תוכנית הקולטת שלשה ציונים של תלמיד, מחשבת ומדפיסה את הממוצע, אם הממוצע קטן מ-55 מדפיסה: 'התלמיד נכשל', אחרת מדפיסה: 'התלמיד עבר'.

(7) כתוב תכנית המדפיסה על המסך את התרגיל: " $5+8+3 = ?$ ", התכנית תבקש מהמשתמש שיחשב את התוצאה. התוצאה תשמש בקלט לתכנית, אם המשתמש פתר נכון, תודפס הודעה: 'יפה מאד פתרת!'; אחרת תודפס הודעה: 'טעית, התוצאה היא 16'.

(8) כתוב תכנית, בליווי תרשים זרימה, הקולטת שלשה מספרים, מחשבת ומדפיסה את מכפלתם, אם המכפלה קטנה מהסכום, מדפיסה את המספרים, אחרת מדפיסה את הסכום והמכפלה.

(9) תרגם את האלגוריתם (תרשים הזרימה) הבא למשפטי תוכנית :

קלוט שלוש אותיות מה-ABC למשתנים let1 let2 let3

אם $(let1 < let2)$ וגם $(let2 < let3)$

הצג כפלט את ההודעה "האותיות מסודרות בסדר עולה"

אחרת

הצג כפלט את ההודעה "האותיות אינן מסודרות בסדר עולה"

(10) כתוב תכנית הקולטת מספר בודקת אותו ומדפיסה אחת מההודעות הבאות:

א. המספר בן 4 ספרות ומעלה

ב. המספר בן 3 ספרות

ג. המספר בן 2 ספרות

ד. המספר בן ספרה אחת בלבד

טבלת מעקב

1. נתונה התכנית הבאה:

```
Scanner in = new Scanner(System.in);
double A,B,C,D;

System.out.println("Enter three double numbers");
A = in.nextDouble();
B = in.nextDouble();
C = in.nextDouble();

if (A>B)
{
    D = A;
    A = -B;
    B = D;
}
if (C<A)
{
    D = A;
    A = C;
    C = -D;
}
if (C<B)
{
    D = B;
    B = C;
    C = D;
}
System.out.println("A=" + A);
System.out.println("B=" + B);
System.out.println("C=" + C);
```

(1 מצא בעזרת טבלת מעקב, מה יודפס על המסך עבור הקלטים הבאים:

| | | | |
|------|--------|--------|----|
| A=11 | B=34.6 | C=12.5 | א. |
| A=7 | B= 4 | C=13.3 | ב. |
| A=33 | B=22 | C=22 | ג. |

(2 רשמו, מה מבצעת התכנית הנ"ל.

פתרון עבור סעיף א
נצייר טבלת מעקב.

מספר הטורים של הטבלה הוא כמספר המשתנים + מספר התנאים + עמודת פלט:

| A | B | C | D | A>B | C<A | C<B | פלט |
|----|------|------|------|-------|-------|------|----------|
| 11 | 34.6 | 12.5 | | false | false | true | |
| | 12.5 | 34.6 | 34.6 | | | | A = 11 |
| | | | | | | | B = 12.5 |
| | | | | | | | C = 34.6 |

המשיכו לפתור את סעיף ב ו-ג, ואת שאלה 2.

לולאות

למה צריך לולאות?

שימוש ראשון: חיסכון בכתיבת פקודות



כתוב תוכנית שכותבת 5 פעמים את המילה "hello":

```
System.out.println("hello");
System.out.println("hello");
System.out.println("hello");
System.out.println("hello");
System.out.println("hello");
```

לולאת while

התוכנית הבאה כותבת את המילה "hello" 5 פעמים:

```
int i=0;
while (i<5)
{
    System.out.println("hello");
    i++;
}
```

לולאת for

בלולאת for, האיטחול של i, תנאי הסיום של הלולאה והקידום של i נכתבים באותה השורה:

```
for (int i=0; i<5; i++)
{
    System.out.println("hello");
}
```

שתי הלולאות מתבצעות באותו סדר:

- (1) מאתחלים את i ל-0. פעולה זאת מתבצעת רק פעם אחת.
- (2) בדיקת התנאי: האם i קטן מ-5?
- (3) כיוון ש-i קטן מ-5, נכנס ללולאה ומבצע את הפקודה.
- (4) מקדם את i. ←
- (5) בודק את התנאי $i < 5$
- (6) נכנס ומבצע את הפקודה בתוך הלולאה.
- (7) חוזר לשלב 4. ←

אחרי if, for ו-while לא כותבים ";" אלא בפקודה שבתוכם.

כל מה שניתן לכתוב בלולאת for ניתן לכתוב בלולאת while וכן להיפך, אלא לפעמים נוח להשתמש ב-for ולפעמים נוח להשתמש ב-while.

מתי נוח להשתמש ב-for ומתי נוח להשתמש ב-while?

ב-for נוח להשתמש כאשר ידוע מראש כמה פעמים צריך לבצע את הלולאה.

ב-while נוח להשתמש כאשר מספר הפעמים של ביצוע הלולאה לא ידוע מראש.

בדוגמא הקודמת (כתוב 5 פעמים את המילה "hello") פשוט יותר להשתמש בלולאת for.

דוגמא

כתוב תוכנית שקולטת מספר. על המספר להיות קטן מ-10. כל עוד המספר גדול או שווה ל-10 קלוט מספר אחר.

פתרון

כיוון שלא ידוע כמה פעמים נצטרך לקלוט מספר עד שהמשתמש יכניס מספר חוקי (מספר קטן מ-10) נעדיף להשתמש בלולאת while.

כאשר כותבים לולאה שמספר הפעמים לא ידוע מראש, דרך כלל ישנו קוד שחוזר על עצמו פעמיים:

קלוט מספר

```
while (המספר >= 10)
{
    קלוט מספר
}
```

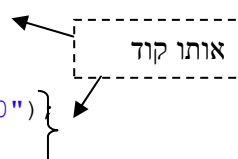
מדוע לא לכתוב את התוכנית הנ"ל בלולאת for? כי זה לא נראה יפה:

קלוט מספר

```
for(;number >= 10;)
{
    קלוט מספר
}
```

הקוד:

```
int number;
Scanner in = new Scanner(System.in);
System.out.println("Enter number less than 10");
number = in.nextInt();
while (number >= 10)
{
    System.out.println("Enter number less than 10");
    number = in.nextInt();
}
System.out.println("your number is: " + number);
```



הרחבות:

(1) אם רוצים להבטיח שהלולאה תתבצע לפחות פעם אחת, משתמשים עם do. התוכנית הבאה תכתוב את המילה "hello" פעם אחת:

```
int i=5;
do
{
    System.out.println("hello");
} while (i<5);
```

(2)

לולאה אין סופית בגירסת לולאת for:

```
for (;)
{
    ...
}
```

לולאה אין סופית בגירסת לולאת while:

```
while(true)
{
    ...
}
```

תרגילים – לולאות (שימוש ראשון)

תרגילים עם לולאת for [מספר הפעמים ידוע מראש]

- (1) הדפס את שמך 12 פעמים.
- (2) תלמיד קיבל עונש לכתוב "לא אדבר בשעת השיעור" 400 פעם. כתוב תוכנית שתעשה עבור התלמיד את העבודה.
- (3) כתוב תוכנית שקולטת משפט ומדפיסה אותו 3 פעמים.
- (4) כתוב תוכנית המדפיסה בשורה 20 כוכביות. הפלט:

(5) כתוב תוכנית שקולטת מספר שלם ומדפיסה כוכביות כמספר הפעמים של המספר השלם.
- (6) כתוב תוכנית המבצעת את התרגיל הקודם שלוש פעמים.
- (7) כתוב תוכנית המדפיסה סוגריים ובתוכם 20 כוכביות. הפלט:
(*****)
(8) רמז: התו '(' אח"כ 20 כוכביות ואח"כ את התו ')'.
כתוב תוכנית המבצעת את התוכנית הקודמת 10 פעמים.

תרגילים עם לולאת while [מספר הפעמים לא ידוע מראש]

- (1) כתוב תוכנית שקולטת אות קטנה (תו בין 'a' ל-'z'). אם המשתמש לא הכניס אות קטנה, תוצג הודעה "הכנס אות קטנה". בסיום, התוכנית תדפיס את האות.
- (2) ממוצע מספרים: כתוב תוכנית שמחשבת ממוצע ל
- (3) משחק "נחש את המספר" - כתוב תוכנית המגרילה מספר. על המשתמש לנחש את המספר. כל עוד המשתמש לא ניחש את המספר, וכל עוד לא עברו שלושה ניחושים תוצג הודעה: "טעות, נסה שוב". אחרי 3 נסיונות, תוצג הודעה: "המשחק הסתיים". אם המשתמש הצליח לגלות את המספר, תוצג הודעה: "ניצחת!"
- (4) כתוב תוכנית שנותנת למשתמש לשחק במשחק "נחש את המספר" כמה פעמים שהוא רוצה. בסיום המשחק הראשון תוצג הודעה: "האם ברצונך לשחק שוב? כ/ל". אם המשתמש לחץ על 'כ' יתחיל משחק חדש. אם המשתמש לחץ על 'ל', תוצג הודעה: "תודה ששיחקת בתוכנית נחש את המספר" והתוכנית תסתיים.



שימוש שני: שימוש במספרים הרצים שמייצרת הלולאה

דרך כללית לפתרון כל לולאה – שיטת שלושת השלבים

שיטת הפיתרון תהיה קבועה. תמיד נצטרך לעבור 3 שלבים:
שלב ראשון: כתיבת כמה פקודות במפורש.

שלב שני: נשאל את עצמנו שתי שאלות:

- (1) מהי התבנית הקבועה?
- (2) מה טווח המספרים הרצים?

שלב שלישי: כתיבת הלולאה:

- התחום של הלולאה יהיה בדיוק הטווח של המספרים הרצים.
- בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה.

דוגמא ראשונה

הדפס את המספרים מ-1 עד 10 בשורות נפרדות.

פתרון:

שלב ראשון: כתיבת כמה פקודות במפורש:

```
System.out.println(1);
System.out.println(2);
...
System.out.println(10);
```

שלב שני: זיהוי מה התבנית הקבועה ומה טווח המספרים הרצים?

התבנית הקבועה: `System.out.println();`

טווח המספרים הרצים: מ-1 עד 10.

שלב שלישי: ניצור לולאה בדיוק בתחום המספרים הרצים (מ-1 עד 10). בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה:

```
for (int i=1; i<=10; i++)
{
    System.out.println(i);
}
```

דוגמא שנייה

כתוב את השם שלך 5 פעמים בצורה הבאה:

```
1
liron
2
liron
3
liron
4
liron
5
liron
```

פתרון:שלב ראשון: כתיבת כמה פקודות במפורש:

```
System.out.println(1);
System.out.println("liron");

System.out.println(2);
System.out.println("liron");

...

System.out.println(5);
System.out.println("liron");
```

שלב שני: זיהוי מה התבנית הקבועה ומה טווח המספרים הרצים?
התבנית הקבועה:

```
System.out.println( );
System.out.println("liron");
```

טווח המספרים הרצים: מ-1 עד 5.

שלב שלישי: ניצור לולאה בדיוק בתחום המספרים הרצים (מ-1 עד 5). בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה:

```
for (int i=1; i<=5; i++)
{
    System.out.println(i);
    System.out.println("liron");
}
```

דוגמא שלישית

חשב את הסכום של המספרים מ-1 עד 10 והצג אותו.

פתרון:שלב ראשון: כתיבת כמה פקודות במפורש:

```
int sum = 0;

sum = sum + 1;
sum = sum + 2;
sum = sum + 3;
...
sum = sum + 10;
```

הערה:

```
sum = 0;
sum = sum + 1;
```

קודם מחושב הצד הימני, ואח"כ התוצאה עוברת לצד השמאלי.

שלב שני: זיהוי מה התבנית הקבועה ומה טווח המספרים הרצים?
התבנית הקבועה:

```
sum = sum + ;
```

טווח המספרים הרצים: מ-1 עד 10.

שלב שלישי: ניצור לולאה בדיוק בתחום המספרים הרצים (מ-1 עד 10). בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה:

```
int sum = 0;

for (int i=1; i<=10; i++)
{
    sum = sum + i;
}
```

הערה:

מי שרואה את הפיתרון מראש, לא חייב לעבור את שלושת השלבים האלה, אלא יכול לגשת ישר לפיתרון. אמנם, כאשר נגיע לדוגמאות מסובכות יותר, שלושת השלבים הנ"ל יהיו הכרחיים כדי להגיע לפתרון בלי לטעות.

תרגילים – לולאות (שימוש שני)

- (1) כתוב תוכנית המדפיסה את המספרים מ-1 עד 100.
 - (2) כתוב תוכנית המדפיסה את המספרים מ-1 עד 100 בסדר הפוך.
 - (3) כתוב תוכנית הקולטת מספר שלם ומדפיסה בשורה אחת את כל המספרים העוקבים עד אותו מספר מופרדים עם פסיקים ביניהם. בסיום, התוכנית תדפיס "גמרתי".
 - (4) אבא הבטיח לבנו דני שאם יתנהג יפה יתן לו כל שבוע 2 ש"ח יותר מאשר בשבוע שעבר. כתוב תוכנית הקולטת את מספר השבועות שדני התנהג יפה ומחשבת כמה כסף יש לדני.
 - (5) הדפס את כל המספרים המתחלקים ב-4 בטווח מ-1 עד 100.
 - (6) במפעל קוקה קולה 26 עובדים. חברת קוקה קולה החליטה לצאת במבצע ולתת לכל עובד מספר אקראי בין 5 ל-20 בקבוקי קולה.
- כתוב תוכנית המגרילה עבור כל עובד מספר בקבוקים אקראי בין 5 ל-20 ומדפיסה לדוגמא:
- עובד 1 קיבל 3 בקבוקים.
עובד 2 קיבל 12 בקבוקים.

...

תרגילים נוספים: מבט לחלונות חלק א – עמוד 113 תרגילים 7,8,9,10,13

תרגילים נוספים: מבט לחלונות חלק א – עמוד 127 תרגילים 37,39

- כמו תרגיל 39, אלא שבמקום להדפיס את מכפלת כל המספרים, יש להדפיס את סכום כל המספרים.

לולאה בתוך לולאה

דוגמא

כתוב תוכנית שמציגה את הפלט הבא:

```
*
**
***
```

שלב ראשון: כתיבת הפקודות באופן מפורש:

```
1) System.out.print("*");
   System.out.println();

2) System.out.print("*");
   System.out.print("*");
   System.out.println();

3) System.out.print("*");
   System.out.print("*");
   System.out.print("*");
   System.out.println();
```

נכניס כל שלב ללולאה:

```
1) for (int i=0; i<1; i++)
    {
        System.out.print("*");
    }
   System.out.println();

2) for (int i=0; i<2; i++)
    {
        System.out.print("*");
    }
   System.out.println();

3) for (int i=0; i<3; i++)
    {
        System.out.print("*");
    }
   System.out.println();
```

שלב שני: נבדוק מהי התבנית הקבועה ומה טווח המספרים הרצים:
התבנית הקבועה:

```
for (int i=0; i< ; i++)
{
    System.out.println("*");
}
System.out.println();
```

טווח המספרים הרצים: מ-1 עד 3.

שלב שלישי: ניצור לולאה בדיוק בתחום המספרים הרצים (מ-1 עד 3). בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה:

```
for (int j=1; j<=3; j++)
{
    for (int i=0; i<j ; i++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

תרגיל דומה:

כתוב תוכנית שתציג את הפלט:

```
***
**
*
```

דוגמא (מתקדמת)

כתוב תוכנית שתציג את הפלט:

```
***
**
*
```

הבנת השאלה:

נדרש לצייר בכל שורה רווחים ואחריהם כוכביות:

```

*   *   *
└   *   *
└   └   *
```

שלב ראשון:

| | |
|--|--|
| ובלולאה: | בשורה הראשונה מציגים רווח 0 פעמים * (עיין הערה) אחריו כוכבית 3 פעמים, ואח"כ יורדים שורה: |
| <pre>for (int j=0; j<0; j++) { System.out.print(" "); } for (int i=0; i<3; i++) { System.out.print("*"); } System.out.println();</pre> | <pre>System.out.print(" "); System.out.print("*"); System.out.print("*"); System.out.print("*"); System.out.println();</pre> |

| | |
|--|--|
| ובלולאה: | בשורה השנייה מציגים רווח 1 פעמים, אחריו כוכבית 2 פעמים, ואח"כ יורדים שורה: |
| <pre>for (int j=0; j<1; j++) { System.out.print(" "); } for (int i=0; i<2; i++) { System.out.print("*"); } System.out.println();</pre> | <pre>System.out.print(" "); System.out.print("*"); System.out.print("*"); System.out.print("*"); System.out.println();</pre> |

| | |
|--|--|
| ובלולאה: | בשורה השלישית מציגים רווח 2 פעמים, אחריו כוכבית 1 פעמים, ואח"כ יורדים שורה: |
| <pre>for (int j=0; j<2; j++) { System.out.print(" "); } for (int i=0; i<1; i++) { System.out.print("*"); } System.out.println();</pre> | <pre>System.out.print(" "); System.out.print(" "); System.out.print("*"); System.out.print("*"); System.out.println();</pre> |

הערה: בשורה הראשונה הצגנו רווח 0 פעמים וכתבנו לשם כך לולאה שלעולם לא תתבצע. מדוע?

כדי שלכל השורות תהיה אותה תבנית. אם לא היינו עושים לולאה עבור הרווחים, אז התבנית של השורה הראשונה היתה שונה מהתבנית של שאר השורות ואז היינו צריכים להשאיר את התבנית של השורה הראשונה בפני עצמה ולא היינו יכולים לקבץ אותה עם התבניות של שאר השורות. אמנם זו גם דרך, זו לא שגיאה! כלומר, ניתן להשאיר את התבנית של השורה הראשונה כמו שהיא, ואח"כ לקבץ את שאר התבניות.

שלב שני:

התבנית הקבועה:

```
for (int j=0; j< ; j++)
{
    System.out.print(" ");
}
for (int i=0; i< ; i++)
{
    System.out.print("*");
}
System.out.println();
```

טווח המספרים הרצים: ב-for העליון: מספר עולה מ-0 עד 2
ב-for התחתון: מספר יורד מ-3 עד 1

שלב שלישי:דרך ראשונה (הדרך הפשוטה):

ניצור לולאה בדיוק בתחום המספרים הרצים (של ה-for העליון ושל ה-for התחתון). בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה: עבור תחום המספרים הרצים של ה-for העליון נגדיר את המשתנה x. עבור תחום המספרים הרצים של ה-for התחתון נגדיר את המשתנה y.

```
int x;
int y;
for(x=0, y=3; x<=2 && y>=1; x++, y--)
{
    for (int j=0; j<x; j++)
    {
        System.out.print(" ");
    }
    for (int i=0; i<y; i++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

דרך שנייה (דרך טריקית):

ניצור לולאה בדיוק בתחום המספרים הרצים (נבחר תחום מספרים אחד: של ה-for התחתון א של ה-for העליון). בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה (נתאים את תחום המספרים הרצים השני בעזרת תחום המספרים הרצים הראשון). נבחר את תחום המספרים הרצים של ה-for העליון. נגדיר עבורו את המשתנה x. עבור תחום המספרים הרצים של ה-for התחתון, נכתוב x-3 כדי שהמספרים ירדו מ-3 עד 1.

```
for(int x=0; x<=2; x++)
{
    for (int j=0; j<x; j++)
    {
        System.out.print(" ");
    }
    for (int i=0; i<3-x; i++)
    {
        System.out.print("*");
    }
    System.out.println();
}
```

הסבר:

| הערכים של ה-for התחתון: 3-x | הערכים של ה-for העליון: x | |
|-----------------------------|---------------------------|-------------------------|
| 3 - 0 = 3 | 0 | בסיבוב הראשון של הלולאה |
| 3 - 1 = 2 | 1 | בסיבוב השני של הלולאה |
| 3 - 2 = 1 | 2 | בסיבוב השלישי של הלולאה |

תרגילים – לולאה בתוך לולאה

(1) כתוב תוכנית המדפיסה את לוח הכפל (עד 10). השתמש בשיטת שלושת השלבים!

פונקציות

רקע: נניח שאתה מנהל ויש לך פועלים ואתה נותן להם פקודות והם מבצעים ומחזירים לך את מה שעשו. הפונקציה הראשית היא המנהל וקוראים לה main. יש רק מנהל אחד לכן יש רק פונקציית main אחת במחלקה (אחרת נקבל שגיאה). כל פעולה חישובית – לא מוטלת על ה-main לעשות, אלא על פונקציה אחרת. כל הפונקציות חוץ מה-main הם הפועלים (פונקציה נקראת גם "פעולה"). כל פעולה חישובית שיש לעשות – מטילים על פונקציה אחרת.

דוגמא

כתוב תוכנית שמקבלת מהמשתמש מספר ושולחת אותו לפונקציה שבודקת אם הוא ראשוני. התוכנית הראשית תדפיס את התשובה.

פתרון:

- 1) הפונקציה הראשית תקלוט מהמשתמש מספר.
- 2) הפונקציה הראשית תשלח את המספר לפונקציה שתבדוק אם הוא ראשוני ומקבלת את התשובה.
- 3) הפונקציה הראשית תחזיר למשתמש את התשובה.

```
public static void main(String[] args)
{
1)   Scanner input = new Scanner(System.in);
      System.out.println("Enter number:");
      int num = input.nextInt();

2)   boolean isRishoni = checkRishoni(num);

3)   if (isRishoni == true)
        System.out.println("Number " + num + " Rishoni !!!");
      else
        System.out.println("Number " + num + " not Rishoni");
}
```

עדיין לא קיימת פונקציה כזו.
עוד מעט ניצור אותה.

הפונקציה שבודקת אם המספר ראשוני:

- 1) מקבלת מהפונקציה הראשית את המספר כפרמטר.
- 2) מפעילה לולאה שבודקת: אם הוא מתחלק במספר אחר חוץ מעצמו – מחזירה לפונקציה הראשית false.
- 3) אם נגמרה הלולאה, סימן שהמספר לא מתחלק בשום מספר – לכן מחזירה לפונקציה הראשית true.

```
1) public static boolean checkRishoni(int number)
   {
2)   for (int i = 2; i < number; i++)
       {
           if (number % i == 0)
               return false;
       }
3)   return true;
   }
```

שים לב!
אין קשר בין הבלוק של הפונקציה הראשית לבלוק של הפונקציה הזאת, לכן יש להשתמש במשתנה החדש!

| תוצאה | שארית |
|-----------|------------|
| $6/2 = 3$ | $6\%2 = 0$ |
| $6/3 = 2$ | $6\%3 = 0$ |
| $6/4 = 1$ | $6\%4 = 2$ |
| $6/5 = 1$ | $6\%5 = 1$ |

| תוצאה | שארית |
|-----------|------------|
| $7/2 = 3$ | $7\%2 = 1$ |
| $7/3 = 2$ | $7\%3 = 1$ |
| $7/4 = 1$ | $7\%4 = 3$ |
| $7/5 = 1$ | $7\%5 = 2$ |
| $7/6 = 1$ | $7\%6 = 1$ |

המבנה של פונקציה

תבנית של פונקציה שעושה פעולה ולא מחזירה ערך:

```
public static void שם הפונקציה ( אם יש )
{
    ...
}
```

שים לב – אין ";" בסוף השורה

הפעלת הפונקציה מהתוכנית:

(פרמטרים לשליחה – אם יש) שם;

תבנית של פונקציה שעושה פעולה ומחזירה ערך int לדוגמא:

```
public static int שם הפונקציה ( אם יש )
{
    ...
    return 5;
}
```

שים לב – אין ";" בסוף השורה

הפעלת הפונקציה מהתוכנית:

int number = שם (אם יש) לשליחה – אם יש;

כפי שרואים, ההבדל בשלושה מקומות:

מקום ראשון:

void – עבור פונקציה שלא מחזירה כלום.

int – עבור פונקציה שמחזירה משתנה מסוג int.

מקום שני:

return – בפונקציה שמחזירה ערך, חייבים להחזיר את הערך, על ידי הפקודה return.

מקום שלישי:

הפעלת פונקציה שלא מחזירה כלום – על ידי כתיבת שמה ואח"כ סוגריים.

הפעלת פונקציה שמחזירה ערך – צריך להעביר למשתנה את הערך שהפונקציה מחזירה.

תיעוד

תיעוד זה הוספת הסברים למי שיקרא את התוכנית שלכם, כדי שיבין את התוכנית מהר.

בתוכניות גדולות התיעוד הוא הכרחי.

כיצד כותבים את התיעוד?

מוסיפים הערה // ואחריה כותבים את התיעוד.

לפני פונקציה, נכתוב תיעוד בו כתוב מה הפונקציה עושה (גם אם זה נראה מיותר).

לפני פעולה חישובית (בפונקציה או בתוכנית), כדאי גם כן לכתוב תיעוד בו כתוב מה הפעולה החישובית עושה.

דוגמא:

```
// this function checking if number is rishoni
public static boolean CheckRishoni(int number)
{
    for (int i = 2; i < number; i++)
    {
        // if number divide in any number - the number is not rishoni
        if (number % i == 0)
        { return false; }
    }
    // if number didn't divide in any number - the number is rishoni
    return true;
}
```

תרגילים – פונקציות [הוסף תיעוד]

- (1) כתוב תוכנית שקולטת שני מספרים שלמים ושולחת אותם לפונקציה שמחזירה את המספר הגדול מביניהם (לך בדיוק לפי השלבים בדוגמא למעלה).
 - (2) כתוב פונקציה שמקבלת מספר ומדפיסה אותו (שים לב, זוהי פונקציה שלא מחזירה ערך ל-Main).
 - (3) כתוב פונקציה הקולטת מספר ומדפיסה אותו.
 - (4) כתוב תוכנית שקולטת מספר ושולחת אותו לפונקציה שמוסיפה לו 10. התוכנית תדפיס את המספר החדש.
 - (5) כתוב תוכנית שקולטת מספר ושולחת אותו לפונקציה שבודקת אם הוא זוגי. אם הוא זוגי, התוכנית הראשית תדפיס "זוגי", אחרת תדפיס "לא זוגי".
 - (6) לפניך תוכנית שקולטת מספר. אם המספר זוגי, היא מדפיסה את כל המספרים הזוגיים בטווח מ-1 עד אותו המספר. אם הוא אי זוגי, היא מדפיסה את כל המספרים האי-זוגיים בטווח מ-1 עד אותו המספר. השלם את הפונקציות החסרות:
- ```

Scanner input = new Scanner(System.in);
System.out.println("Enter number:");
int num = input.nextInt();

if (isZugi(num))
{
 printZugiim(num);
}
else
{
 printEZugiim(num);
}

```
- (7) כתוב תוכנית הקולטת מספר מ-1 עד 10 ושולחת אותו לפונקציה שמדפיסה את כל המספרים שמתחלקים באותו מספר בטווח המספרים מ-1 עד 100.
  - (8) כתוב פונקציה בשם display שתחליף את הפקודה System.out.println(). כלומר, כל פעם שנרצה להדפיס על הלוח משהו, נכתוב: display("hello").
  - (9) כתוב תוכנית שקולטת אורך ורוחב של מלבן ושולחת את הפרמטרים לפונקציות שונות: פונקציה ראשונה – תחשב ותדפיס את שטח המלבן ואת היקפו. פונקציה שנייה – תצייר את המלבן המתקבל באמצעות כוכביות. פונקציה שלישית – תחשב שטח והיקף של מלבן הגדול באורכו פי 2 מהקלט וקטן ברוחבו ב-3 מהקלט. וכן, תצייר אותו. השתמש בפונקציות מסעיפים קודמים.
  - (10) כתוב פונקציה המקבלת כקלט אורך רדיוס r, ומציגה כפלט את היקפו ( $2\pi r$ ) ושטחו ( $\pi r^2$ ) של מעגל שרדיוסו r.



# switch

פתרון שיש בו שימוש מרובה במשפטי תנאי (if), מעיד על תכנון לקוי. כאשר נזהה שימוש מרובה במשפטי תנאי, נעדיף להשתמש במשפט ה-switch.

## מבנה ה-switch

(משתנה) switch

```
{
 case אפשרות ראשונה: ...
 break;
 case אפשרות שניה: ...
 break;
 case אפשרות שלישית: ...
 break;
 default: ...
 break;
}
```

## דוגמה ראשונה

קלוט שני מספרים לתוך משתנים בשם num1 ו-num2.  
קלוט מספר נוסף מהמשתמש לתוך משתנה בשם choice.  
אם choice שווה 1 – הדפס num1 + num2  
אם choice שווה 2 – הדפס num1 - num2  
אם choice שווה 3 – הדפס num1 \* num2  
אם choice שווה 4 – הדפס num1 / num2  
אחרת (אם choice לא שווה לאף אחד מהמספרים) – הדפס "לא בחרת מספר חוקי".

## פתרון

```
Scanner input = new Scanner(System.in);
System.out.println("Enter two numbers:");
int num1 = input.nextInt();
int num2 = input.nextInt();
System.out.println("Enter your choice:");
int choice = input.nextInt();

switch (choice)
{
 case 1:
 System.out.println(num1 + num2);
 break;
 case 2:
 System.out.println(num1 - num2);
 break;
 case 3:
 System.out.println(num1 * num2);
 break;
 case 4:
 System.out.println(num1 / num2);
 break;
 default:
 System.out.println("בחרת מספר לא חוקי");
 break;
}
```

**דוגמא שנייה**

כתוב פונקציה שמקבלת מהתוכנית מספר שלם ומחזירה תו בהתאם למספר:  
 אם num שווה 1, הפונקציה תחזיר את התו 'b'  
 אם num שווה 4, הפונקציה תחזיר את התו 'g'  
 אם num שווה 9, הפונקציה תחזיר את התו 'r'  
 אחרת (במקרה ו-num לא שווה לאף אחד מהמספרים הנ"ל), הפונקציה תחזיר '0'  
פתרון

```
public static char function(int num)
{
 switch (num)
 {
 case 1: return 'b';
 case 4: return 'g';
 case 9: return 'r';
 default: return '0';
 }
}
```

הפקודה return באה במקום הפקודה break, כי שתי הפקודות מבטאות יציאה מה-switch.

**תרגילים**

- (1) קלוט תו.  
 אם התו שווה 'u' הדפס "how are you?".  
 אם התו שווה 't' הדפס "thank you".  
 אם התו שווה 'h' הדפס "hello".  
 אחרת, הדפס "by by".
- (2) כתוב פונקציה שמקבלת מהתוכנית מספר שלם לתוך משתנה בשם num. הפונקציה תקלוט מספר נוסף לתוך משתנה בשם choise.  
 אם choise שווה 1, הפונקציה תחזיר את num  
 אם choise שווה 2, הפונקציה תחזיר את num + num  
 אם choise שווה 3, הפונקציה תחזיר את num + num + num  
 אחרת, תחזיר 0.

## מערך חד מימדי

מהו מערך?

מערך זהו אוסף של משתנים מאותו טיפוס.

לדוגמא, מערך בשם  $a$  שמכיל אוסף של 5 מספרים שלמים:

|   |   |   |
|---|---|---|
| a | 0 | 0 |
|   | 1 | 0 |
|   | 2 | 0 |
|   | 3 | 0 |
|   | 4 | 0 |

אפשר לדמות מערך לארון שיש בו תאים

ובכל תא נכנס מספר אחד.

בהתחלה, הארון ריק. יש בו רק 0 בכל התאים.

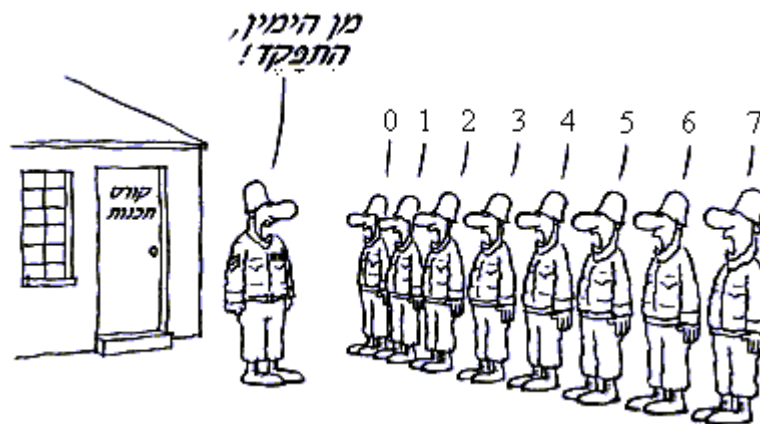
הכנסת מספר לתא:  $a[0] = 5$ ;

$a[3] = 2$ ;

התוצאה:

|   |   |   |
|---|---|---|
| a | 0 | 5 |
|   | 1 | 0 |
|   | 2 | 0 |
|   | 3 | 2 |
|   | 4 | 0 |

אינדקס המערך



אינדקס = מספר התא של המערך.

אינדקס (מספר) התא הראשון במערך תמיד מתחיל מ-0!

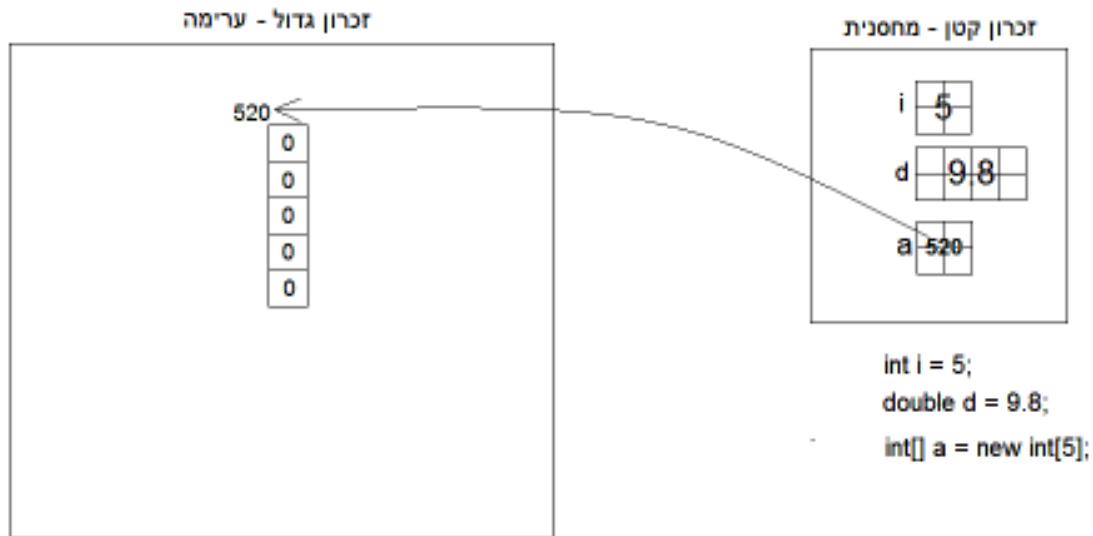
אינדקס התא השני הוא 1.

שאלה:

איזה אינדקס יהיה לתא הרביעי?

תשובה:

3, נכון.

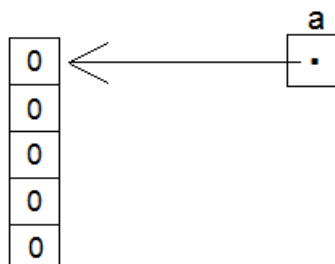
איך המערך נשמר בזכרון

ישנם שני סוגי זכרונות שהתוכנית שלנו משתמשת בהם במהלך תוכנית: זכרון קטן שנקרא מחסנית והוא עובד ישירות עם המעבד. זכרון גדול שנקרא ערימה. הזכרון הקטן מהיר יותר מהזכרון הגדול (כי הוא עובד ישירות עם המעבד).

כאשר מבצעים `int i = 5;` או `double d = 9.8;` הכל מתבצע בזכרון הקטן. מדוע? כי הזכרון הקטן יודע כמה "מחכה" לו: הגודל של `int` הוא 4 בייטים (ולא יכול להיות יותר). הגודל של `double` זה 8 בייטים (ולא יכול להיות יותר). כאשר מגדירים מערך, הגודל לא קבוע. המשתמש יכול להחליט להגדיר מערך עם 5 תאים או עם 100 תאים. לכן המערך נשמר בזכרון הגדול.

ככלל: משתנה שהגודל שלו לא קבוע (אלא המשתמש קובע אותו) נשמר בזכרון הגדול. משתנה שהגודל שלו קבוע נשמר בזכרון הקטן.

מה נשמר בתוך המשתנה `a` עצמו? בתוך המשתנה `a` נשמרת הכתובת של המערך (520). הערה: מה זה כתובת? לכל אדם יש מקום בו הוא גר. לדוגמא, נחל לכיש 8. גם בתוך המחשב יש לכל משתנה כתובת בו הוא נמצא, והכתובת הזאת היא מספר. לסיכום, בתוך המשתנה `a` נשמרת הכתובת של המערך, והמערך עצמו נמצא במקום אחר. לכן נמצא בהרבה ספרים שמשתנה מערך נקרא "משתנה הפניה" (כלומר, מפנה למקום אחר), ומסומן כך:



משתנה הפניה: משתנה שמכיל בתוכו כתובת למקום בזיכרון שמכיל את מערך המספרים.

ההבנה מהו מערך ואיך הוא נשמר בזכרון היתה חשובה לפני שנראה איך משתמשים בו.

## 2 אפשרויות להגדיר מערך

אפשרות ראשונה:



(1) יצירת משתנה מטיפוס מערך

```
int[] a;
```

(2) מגדירים כמה "תאים" יהיו למערך. יצירת מערך בעל 3 "תאים".

```
a = new int[3];
```

שאלת הבנה: מדוע מופיעה המילה new? לא היה מובן בלעדית שהמערך מכיל 3 תאים?  
תשובה: הכוונה במילה new: תקצה לי מקום חדש בזכרון הגדול עבור 3 תאים של המערך.

אפשר לקצר את שני השלבים ביחד (יצירת משתנה מטיפוס מערך והגדרת ה"תאים" שלו):

```
int[] a = new int[3];
```

הכנסת איברים ל"תאים" של המערך:

```
a[0] = 5;
```

```
a[1] = 2;
```

```
a[2] = 9;
```

אפשרות שניה:



כאשר האיברים ידועים מראש, ניתן להגדיר את המערך כך:

```
int[] a = {5,2,9};
```

## כלל

לא ניתן לחרוג מגבולותיו של המערך. לדוגמא, אם ננסה לפנות לאיבר ה-4 במערך בגודל 3, נקבל שגיאה שמציגה (באנגלית) את ההודעה "חריגה מגבולות המערך".

## גודל המערך

נתון המערך:

```
int[] a = new int[3];
```

גודל המערך הוא 3.

ניתן לקבל את גודל המערך כך:

```
int g = a.length;
```

קיבלנו את גודל המערך לתוך המשתנה g. כלומר, g שווה 3.

אם נחליף את כל המקומות בתוכנית בהם כתוב 3 ל-a.Length, מה נרויח?  
נניח שנרצה לשנות את גודל המערך במקום 3 תאים ל-6 תאים. בכל מקום בתוכנית שכתבנו 3 נצטרך לשנות ל-6. אבל אם כתבנו בתוכנית a.Length, לא נצטרך לשנות כלום.

שאלה

כאשר מגדירים מערך, בהתחלה כל התאים ריקים.

אם המערך מסוג מספרים שלמים (int), בכל תא יהיה 0.

אם המערך מסוג מחרוזות (string), מה יהיה בכל תא?

תשובה

בכל תא תהיה מחרוזת ריקה "".

**דוגמא**

הגדר מערך והכנס אליו את המספרים: 5,3,7,8,6,5,9,0,6,3.  
הצג את המערך בשורה אחת.

פתרון:

בניית המערך והכנסת מספרים לתוכו.

```
int[] a = {5,3,7,8,6,5,9,0,6,3};
```

הצגת המערך:

שלב ראשון: כתיבת כמה פקודות במפורש:

```
System.out.print(a[0] + " ");
System.out.print(a[1] + " ");
System.out.print(a[2] + " ");
...
System.out.print(a[9] + " ");
```

שלב שני: נבדוק מהי התבנית הקבועה ומה טווח המספרים הרצים:

התבנית הקבועה: `System.out.println(a[ ])`;

טווח המספרים הרצים: מ-0 עד 9.

שלב שלישי:

ניצור לולאה בדיוק בתחום המספרים הרצים (מ-0 עד 9). בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה:

```
for(int i=0; i<10; i++)
{
 System.out.print(a[i] + " ");
}
```

אם נשנה את הגודל של המערך ל-100, במקום לכתוב `i<10` נצטרך לכתוב `i<100`. אפשר לכתוב `i<a.length` ואז לא נצטרך לשנות כלום.

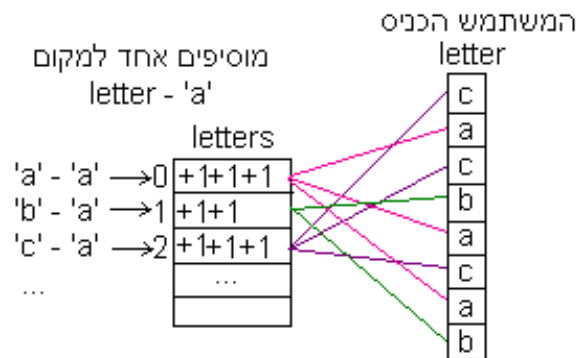
**תרגילים – מערך חד מימדי**

- (1) קלוט 10 מספרים לתוך מערך והצג אותם.
- (2) הגדר מערך בגודל 10. הכנס לתוכו מספרים אקראיים בין 5 ל-12 (כולל 12). הצג את הסכום שלהם.
- (3) הגדר מערך בשם a והכנס אליו את האיברים: 4,7,3,1,5,7,2,9  
צור מערך ריק חדש בשם b בגודל של המערך a.  
הכנס למערך b את האיברים של המערך a בסדר הפוך.  
תוצאת המערך b: 9,2,7,5,1,3,7,4
- (4) הגדר מערך בשם a והכנס אליו את האיברים: 4,7,3,1,5,7,2,9  
צור מערך ריק חדש בשם b.  
לארבעת המקומות השמאליים של המערך b, הכנס בלולאת for את 4 האיברים הימניים של המערך a,  
ולארבעת המקומות הימניים של המערך b, הכנס בלולאת for את 4 האיברים השמאליים של המערך a.  
תוצאת המערך b: 5,7,2,9,4,7,3,1
- (5) הגדר מערך והכנס בו עשרה מספרים: 0,1,2,3,4,5,6,7,8,9.  
הזז את כל האיברים מקום אחד ימינה: האיבר הימני ביותר – יעבור לצד שמאל.  
אין להשתמש במערך עזר!  
הפלט: 9,0,1,2,3,4,5,6,7,8
- (6) קלוט 8 אותיות מהמשתמש (אין צורך לבדוק את תקינות הקלט).  
צריך לספור כמה פעמים כל אות מופיעה ולהציג למשתמש את התוצאה.  
פתרון פשוט: על ידי לולאה בתוך לולאה. לפי הדרך של שלושת השלבים:  
שלב ראשון: כתיבת כמה פקודות במפורש.  
שלב שני: נבדוק מהי התבנית הקבועה ומה טווח המספרים הרצים.  
שלב שלישי: ניצור לולאה בדיוק בתחום המספרים הרצים. בתוך הלולאה נכתוב את התבנית הקבועה, ואיפה שצריך להיות המספר הרץ נכתוב את המשתנה של הלולאה.  
החיסרון בפיתרון הפשוט: אות שמופיעה פעמיים, תוצג בפלט פעמיים.  
עבור התווים: h,e,l,l,o,w,o,r  
הפלט:

```
The letter h show 1 times
The letter e show 1 times
The letter l show 2 times
The letter l show 2 times
The letter o show 2 times
The letter w show 1 times
The letter o show 2 times
The letter r show 1 times
```

פתרון מתקדם (להעשרה בלבד)

הנחה: המשתמש הכניס אותיות קטנות ולא גדולות.

תכנון:

מה מוצג למשתמש כתוצאה

| letters  |               |     |       |   |            |
|----------|---------------|-----|-------|---|------------|
| 0 +1+1+1 | (char)(0+'a') | 'a' | ----> | 3 | letters[0] |
| 1 +1+1   | (char)(1+'a') | 'b' | ----> | 2 | letters[1] |
| 2 +1+1+1 | (char)(2+'a') | 'c' | ----> | 3 | letters[2] |
| ...      |               |     |       |   |            |

הקוד:

```
public static void main(String[] args)
{
 System.out.println((int) ('a'));
 int size = 'z'-'a'+1; // =26
 int[] letters = new int[size];
 Scanner in = new Scanner(System.in);
 for(int i=0; i<10; i++)
 {
 char c = in.next().charAt(0);
 letters[c-'a']++;
 }
 for (int i=0; i<size; i++)
 {
 if (letters[i]>0)
 System.out.println((char) (i+'a')+"-->"+letters[i]);
 }
}
```

היתרון: אות שמופיעה פעמיים, תוצג בפלט רק פעם אחת.

עבור התווים: h,e,l,l,o,w,o,r

הפלט:

```
e-->1
h-->1
l-->2
o-->2
r-->1
w-->1
```



## מחרוזות ותווים – בסיס

String – מחרוזת – סידרה כלשהי של תווים.  
char – תו.

מחרוזות כותבים עם מרכאות (" ")  
תו כותבים עם גרשיים (' ')

דוגמא:

```
String str = "HAPPY BIRTHDAY ";
// מחרוזת בשם str המורכבת מ-2 מילים שאחרי כל אחת
// מהן מופיע רווח.
char c = 'Y' (שגיאה לכתוב: char c = "Y")
```

כיוון שמחרוזת היא סידרה של תווים, אפשר לפנות לכל תו בה באמצעות הפונקציה charAt(index)  
דוגמא:

```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
String str = "HAPPY BIRTHDAY ";
str.charAt(4) → 'Y'
str.charAt(13) → 'Y'
str.charAt(5) → ' '
str.charAt(14) → ' '
```

דוגמא

קלוט מהמשתמש 5 תווים לתוך מחרוזת. הדפס את המחרוזת.

פתרון

נגדיר מחרוזת ריקה:

```
String str = "";
```

כל תו שהמשתמש יכניס, נוסיף למחרוזת:

```
Scanner input = new Scanner(System.in);
for (int i = 0; i < 5; i++)
{
 System.out.println("Enter char");
 char c = input.next().charAt(0);
 str = str + c;
}
```

נדפיס את המחרוזת:

```
System.out.println(str);
```

תרגילים – מחרוזות

- קלוט מחרוזת והדפס את התווים עם רווח בין כל תו לתו.
- קלוט מחרוזת. בדוק אם מופיעה בתוך המחרוזת האות a. אם כן, הדפס "האות a מופיעה".
- נתונה מחרוזת "happy birthday". צור מחרוזת חדשה בשם b והכנס אליה את התווים בסדר הפוך.
- צור מערך מטיפוס char שיכיל את התווים: h,e,l,l,o, ,w,o,r,d, המר את המערך למחרוזת. הדפס את המחרוזת.

## מערך דו מימדי

מהו מערך דו מימדי?

מערך קד מימדי זהו אוסף של משתנים מאותו טיפוס.

מערך דו מימדי זה אוסף של מערכים שכל אחד מהם הוא אוסף של משתנים מאותו טיפוס.

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| a | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |

אפשר לדמות מערך לארון שיש בו תאים  
ובכל תא יש אוסף של תאים.  
בהתחלה, הארון ריק. יש בו רק 0 בכל התאים.

הכנסת מספר לתא:  $a[0][1] = 5;$   
 $a[3][2] = 2;$

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| a | 0 | 5 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 2 | 0 |
| 4 | 0 | 0 | 0 | 0 |

התוצאה:

### 3 אפשרויות להגדיר ממעריך דו מימדי

אפשרות ראשונה: 

(1) יצירת משתנה מטיפוס מערך דו מימדי

```
int[][] a;
```

(2) מגדירים כמה "תאים" יהיו למעריך. יצירת מערך בעל 3 x 4 (3 תאים ובכל תא 4 תאים).

```
a = new int[3][4];
```

אפשר לקצר את שני השלבים ביחד (יצירת משתנה מטיפוס מערך והגדרת ה"תאים" שלו):

```
int[][] a = new int[3][4];
```

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| a | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |

הכנסת איברים ל"תאים" של המעריך:

```
a[0][0] = 3;
a[0][1] = 2;
a[0][2] = 9;
a[0][3] = 8;
```

אפשרות שניה:

כאשר האיברים ידועים מראש:

|     | 0 | 1 | 2 | 3 |
|-----|---|---|---|---|
| a 0 | 1 | 2 | 3 | 4 |
| 1   | 5 | 6 | 7 | 8 |
| 2   | 9 | 0 | 1 | 2 |

ניתן להגדיר את המערך כך:

```
int[][] a = { new int[] { 1, 2, 3, 4 },
 new int[] { 5, 6, 7, 8 },
 new int[] { 9, 0, 1, 2 } };
```

אפשרות שלישית:

מערך עם גודל תאים שונה בין תא לתא.

בארון עם תאים, אין הכרח שבכל תא יהיו אותם מספר תאים. כלומר, לכל תא יכול להיות מספר תאים אחר. יכול להיות ארון שבתא הראשון יש 2 תאים, בתא השני 3 תאים ובתא השלישי 4 תאים:

|     | 0 | 1 | 2 | 3 |
|-----|---|---|---|---|
| a 0 | 0 | 0 |   |   |
| 1   | 0 | 0 | 0 |   |
| 2   | 0 | 0 | 0 | 0 |

הגדרת המערך:

```
int[][] a;
a = new int[3][];

a[0] = new int[2];
a[1] = new int[3];
a[2] = new int[4];
```

גודל המערך

הגודל של המערך הנ"ל:

a.length שווה 3, כי המערך a הוא מערך חד מימדי עם 3 תאים.  
a[0].length שווה 2, כי המערך a[0] הוא מערך חד מימדי עם 2 תאים.  
a[1].length שווה 3, כי המערך a[1] הוא מערך חד מימדי עם 3 תאים.  
a[2].length שווה 4, כי המערך a[2] הוא מערך חד מימדי עם 4 תאים.

**דוגמא**

צור מערך דו מימדי בעל המימדים 4x6.  
קלוט לתוכו מספרים אקראיים בין 10 ל-30.  
הדפס את המערך.  
חשב את ההיקף והדפס את התוצאה.

**פתרון:****תזכורת "שיטת שלושת השלבים"**

שלב ראשון: לכתוב במפורש כמה פקודות.  
שלב שני: לזהות מהי התבנית הקבועה ומה טווח המספרים הרצים.  
שלב שלישי: לבנות את הלולאה בהתאם.

נחלק את הפיתרון לחלקים לפי השאלה.

**(1) בניית מערך דו מימדי בגודל 4x6**

```
int[][] array = new int[4][6];
```

**(2) מילוי המערך במספרים אקראיים**

יש למלא את המערך במספרים אקראיים בין 10 ל-30.  
דרך הפתרון:

- (1) לחשב מהו הביטוי לחישוב המספרים האקראיים.
- (2) דרך הפיתרון היא לפי שיטת שלושת השלבים. שים לב, יש להשתמש בשיטת שלושת השלבים פעמיים. פעם ראשונה כדי ליצור לולאות. פעם שניה כדי ליצור לולאה אחת עבור כל הלולאות.

| a | 0  | 1  | 2  | 3  | 4  | 5  |
|---|----|----|----|----|----|----|
| 0 | 22 | 17 | 28 | 30 | 15 | 12 |
| 1 | 29 | 18 | 13 | 19 | 24 | 16 |
| 2 | 13 | 10 | 11 | 25 | 15 | 17 |
| 3 | 15 | 12 | 14 | 24 | 19 | 27 |

**(3) הדפסת המערך**

דרך הפיתרון היא לפי שיטת שלושת השלבים (גם פה צריך להשתמש בשיטת שלושת השלבים פעמיים).

**(4) חישוב ההיקף****סיכום השורה הראשונה**

שלב ראשון: כתיבת כמה פקודות באופן מפורש (כתבתי את כולן, למרות שלא צריך):

```
int sum=0;
sum = sum + a[0][0];
sum = sum + a[0][1];
sum = sum + a[0][2];
sum = sum + a[0][3];
sum = sum + a[0][4];
sum = sum + a[0][5];
```

**שלב שני**

התבנית הקבועה: `sum = sum + a[0][ ]`

טווח המספרים הרצים: מ-0 עד 5

שלב שלישי: כתיבת הלולאה שמסכמת את השורה הראשונה:

```
int sum=0;
for (int i=0; i<6; i++)
 sum = sum + a[0][i];
```

### סיכום העמודה האחרונה

שלב ראשון: כתיבת כמה פקודות באופן מפורש:

```
sum = sum + a[0][5];
sum = sum + a[1][5];
sum = sum + a[2][5];
sum = sum + a[3][5];
```

שלב שני:

התבנית הקבועה: `sum = sum + a[i][5]` :  
טווח המספרים הרצים: מ-0 עד 3

שלב שלישי: כתיבת הלולאה שמסכמת את העמודה האחרונה:

```
for (int i=0; i<4; i++)
 sum = sum + a[i][5];
```

באותו אופן, ניצור לולאה עבור השורה האחרונה ועבור העמודה הראשונה.

שים לב, אחרי 4 הלולאות שמסכמות את ההיקף, צריך להפחית את 4 הפינות בצורה ידנית, כי חישבנו אותם פעמיים:

```
sum = sum - array[0][0];
sum = sum - array[0][5];
sum = sum - array[3][0];
sum = sum - array[3][5];
```