

## טיפוסי נתונים

### א. מספרים

מספרים שלמים: byte , short , int , long

מספרים עשרוניים: float , double

ששת הטיפוסים הללו נבדלים זה מזה בכמות הזיכרון המוקצת לערכים שלהם. כתוצאה מכך כל טיפוס יכול להכיל ערכים בטווח שונה.

בטיפוסים השלמים: ל byte מוקצת כמות הזיכרון הקטנה ביותר ול long מוקצת כמות הזיכרון הגדולה ביותר.

במספרים עשרוניים ל float מוקצת כמות זיכרון קטנה מזו שמוקצת ל double

### ב. תווים

שם הטיפוס: char. נתון יהיה בין שני סימני גרש.

### ג. טיפוס בוליאני

שם הטיפוס boolean. יכול לקבל רק ערך לוגי: true (אמת) או false (שקר).

טיפוס הנתונים	סוג הצהרה	דוגמה להצהרה	דוגמאות להשמה
מספר שלם	int	int number;	number = 98; count=count+1;
מספר ממשי	double	double average;	average = sum/count;
תו	char	char tav ;	tav = 'a'; tav1=tav2;
טיפוס לוגי	boolean	boolean flag;	flag = false; flag = true;

❖ צריך להצהיר על משתנה לפני הפעם הראשונה שמשתמשים בו.

❖ אסור להצהיר על משתנה יותר מפעם אחת.

❖ אפשר להציב במשתנה מטיפוס מספר ממשי (double), ערך של משתנה מטיפוס מספר שלם (int).

במקרה כזה, יתווסף למספר הממשי אפס אחד לאחר הנקודה העשרונית.

❖ כדי להציב במשתנה מטיפוס מספר שלם (int), ערך של משתנה מטיפוס מספר ממשי (double), צריך

לרשום (int) לפני הערך להצבה. למשל, `num = (int)6.5;` , `numInt = (int)numDouble;`

רישום כזה נקרא: **המרה (Casting)** כי אנו ממירים את המספר הממשי להיות מספר שלם.

כאשר מתבצעת המרה, המספר השלם יקבל את החלק השלם של המספר הממשי.

❖ כאשר מציבים במשתנה מטיפוס מספר שלם (int), ערך של משתנה מטיפוס תו (char), הערך

שנכנס למשתנה השלם הוא הקוד האסקי של התו שנמצא במשתנה התו.

❖ אי אפשר להציב במשתנה מטיפוס תו, ערך של משתנה מטיפוס int, אבל, באמצעות המרה,

למשל: `ch = (char)num;` אפשר להציב במשתנה מטיפוס תו את התו שהקוד האסקי שלו נמצא

במשתנה מטיפוס int.

## הוראות "מקוצרות"

### הוספת / חיבור 1:

הביטוי ++ *פעולה* מוסיף 1 לערך של המשתנה. למשל, הביטוי ++count מוסיף 1 לערך של המשתנה count.

```
count = count+1; ⇔ count++;
```

הביטוי -- *פעולה* מוריד 1 מהערך של המשתנה. למשל, הביטוי --count מוריד 1 מהערך של המשתנה count.

```
count = count-1; ⇔ count--;
```

### איתחול (השמה) בזמן הצהרה:

*צרכי התחלתי* שם של משתנה *טיפוס הנתונים*

```
int count; ⇔ int count = 0;
count = 0;
```

```
char first; ⇔ char first = 'a';
first = 'a';
```

### השמה בו זמנית למספר משתנים:

*צרכי* = שם משתנה n = ..... = שם משתנה 2 = שם משתנה 1

```
num2 = num1; ⇔ num2 = num3 = num1;
sum = 0;
count = 0; ⇔ sum = count = max = 0;
max = 0;
```

השמת ערך למשתנה שמקבל תוצאה של חישוב בו "משתתף" הערך הקודם של אותו משתנה:

; *טיפוי בו משתתף המשתנה עצמו* = שם משתנה

```
sum = sum + grade; ⇔ sum += grade;
```

```
num = num / 10; ⇔ num /= 10;
```

## הוראת קלט

```
import java.util.Scanner;
public class FirstIOProg
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        int num;
        System.out.println ("enter int number");
        num = input.nextInt();
        System.out.println ("num = " + num);
    }
}
```

הפעלה של מחלקה בשם Scanner שמאפשרת לקלוט נתונים מהמשתמש. הוראה זו תופיע תמיד כהוראה ראשונה לפני כותרת המחלקה.

הוראה שאפשרת להשתמש במחלקה Scanner. צריכה להופיע אחרי הפותח { לפתיחת הפעולה הראשית ולפני הוראת הקלט הראשונה בפעולה.

### הוראות קלט:

`input.nextInt()` -- הוראה לקליטת מספר שלם.

`input.nextDouble()` -- הוראה לקליטת מספר ממשי.

`input.nextBoolean()` -- הוראה לקליטת ערך בוליאני.

`input.next()` -- הוראה לקליטת מחרוזת עד לתו הרווח הראשון.

`input.nextLine()` -- הוראה לקליטת מחרוזת כולל רווחים.

לפני כל הוראת קלט חשוב להודיע למשתמש לאיזה מידע מצפים ממנו.

הודעה זו נעשית באמצעות הוראת הפלט `System.out.println`

## פעולות מתמטיות בסיסיות

- ❖ בפעולות המתמטיות  $+$  -  $*$  / , אם לפחות אחד מהערכים הוא מטיפוס מספר ממשי, התוצאה היא מטיפוס מספר ממשי.
- ❖ בפעולות המתמטיות  $+$  -  $*$  / , אם שני הערכים הם מטיפוס מספר שלם, התוצאה היא מטיפוס מספר שלם.
- ❖ כאשר הפעולה / מקבלת שני מספרים שלמים, היא מחזירה את החלק השלם של תוצאת החילוק.
- ❖ הפעולה % מקבלת שני מספרים ומחזירה את השארית של החלוקה. שימושית בעיקר בשלמים.  
דוגמאות:

$$\begin{array}{cccc}
 10 / 5 \longrightarrow 2 & 10 \% 5 \longrightarrow 0 & 12 / 7 \longrightarrow 1 & 12 \% 7 \longrightarrow 5 \\
 (123 / 10) \% 10 \longrightarrow 2 & (123 \% 100) / 10 \longrightarrow 2 & & 
 \end{array}$$

### שימושים נפוצים ל %

- בדיקה אם מספר שלם מתחלק במספר שלם אחר. למשל, כדי לדעת אם `num` הוא זוגי נבדוק אם `num%2==0`
- מציאת ספרת האחדות של מספר שלם `num`: `num%10`

## קבועים

- ❖ הצהרה על קבוע: כמו הצהרה על משתנה בתוספת המילה `final` בתחילת ההצהרה.  
למשל, `final int NUM = 6;` , `final char TAV = 't';`
- ❖ מוסכמה בג'אווה: קבוע כותבים באותיות גדולות.