דפי סיכום

מבנה חוקי של מחלקה בשפת ג'אווה:

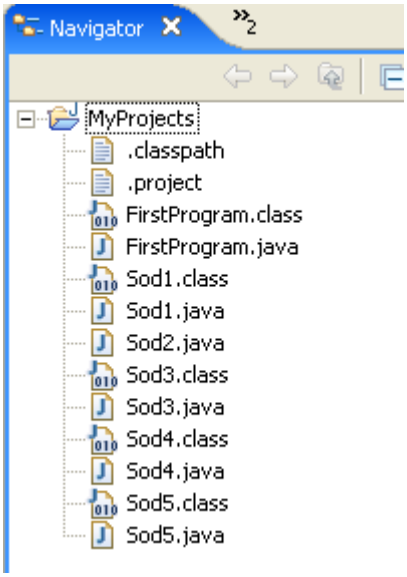
```
כותרת המחלקה -- השורה הפותחת את המחלקה        שם המחלקה public class
{                                                סוגריים לפתיחת המחלקה
    public static void main(String[] args)       חתימה של הפעולה הראשית לביצוע
    {                                            סוגריים לפתיחת הפעולה הראשית לביצוע
            הוראות לביצוע
    }                                            סוגריים לסגירת הפעולה הראשית לביצוע
}                                                סוגריים לסגירת המחלקה
```

- הפעולה הראשית main מופעלת אוטומטית עם הרצת המחלקה.
- לאחר כל הוראה לביצוע מופיע התו **;**
- יש הבחנה בין אותיות גדולות לבין אותיות קטנות.
- שם של מחלקה חייב להתחיל באות גדולה, ויכולים להכלל בו אותיות גדולות, אותיות קטנות, ספרות ומקף תחתון _.
- הוספת הערות לתכנית:

אפשרות א: הערה תיכתב בין סימני /* למשל: /* ...... */

אפשרות ב: הערה תיכתב החל משני סימני // ועד לסוף השורה //.....

פרויקט ומחלקה



- בחלון השמאלי של תכנת ג'אווה (Navigator) מופיעה רשימת הקבצים עימם אנו עובדים.
- הקבצים מאוגדים בפרויקטים. כל **פרויקט** מכיל מספר מחלקות. למשל, MyProjects הוא **פרויקט** שמכיל 6 **מחלקות:** FirstProgram , Sod1 , Sod2 , Sod3 , Sod4 , Sod5

הערות "טכניות"

- לאחר שמשנים תוכן של מחלקה, יש לבצע שמירה שלה. אם מריצים לפני ששומרים, מופיעה שאלה האם לשמור את המחלקה, ולאחר מכן התכנית מבצעת את הנדרש, אבל הפלט נעלם לפני שמספיקים לראות אותו!
- לאחר שמריצים מחלקה פעם אחת, אפשר להריץ אותה גם באמצעות לחיצה על כפתור (כפתור חמישי משמאל בשורת הכפתורים בחלק העליון של המסך).

הוראות הדפסה

S גדולה!

<u>אפשרות ראשונה:</u> **System.out.println (ערך להדפסה);**

הערך להדפסה יכול להיות מחרוזת ויכול להיות מטיפוס מוכר בשפה: מספר שלם, מספר ממשי, תו, לוגי.

ההוראה להדפסה	הערך שיודפס
System.out.println (" we did it! ");	we did it!
System.out.println (88);	88
System.out.println ('t');	t
System.out.println (false)	false

- אפשר להשתמש ב- **print** במקום ב- println ואז ההדפסה הבאה תהיה באותה שורה.
- אם המספר להדפסה הוא מספר שאינו שלם, המספר מודפס בדיוק כפי שהוקלד.
- אם המספר שיש להדפיס הוא כתוצאה מחישוב של מספרים ממשיים, אז מודפסות 14 ספרות אחרי הנקודה.

שילוב של ערכים מטיפוסים שונים בהדפסת מחרוזת

- המחרוזת להדפסה נשארת בין סימני " ", מוסיפים את הערכים הרצויים באמצעות הסימן +.
- הסימן + לא יודפס. תפקידו להפוך טיפוסים שונים (מספר, תו, ערך בוליאני) למחרוזת, וליצור מחרוזת אחת מכל הערכים להדפסה.

הסבר	**הערך שיודפס**	**ההוראה להדפסה**
הדפסת תוצאה חשבונית	18	System.out.print (9+9);
שרשור של מחרוזת ריקה ושני מספרים.	99	System.out.print (""+9+9)
שרשור של 3 מחרוזות	there are 10 boys	System.out.print (" there are " + 10 + " boys");
חילוק קודם לחיבור. שרשור של שתי מחרוזות	the average is 87	System.out.print (" the average is " + 875/10);
חילוק קודם לחיבור. שרשור של 4 מחרוזות	there are 10 students. the average is 87	System.out.print (" there are " + 10 + " students. the average is " + 875/10);
שרשור של 3 מחרוזות	1+398	System.out.print ("1+3"+9+8);
חיבור של שני מספרים ולאחריו שרשור של שתי מחרוזות	171+3	System.out.print (9+8+"1+3");
חיבור של שני מספרים ולאחריו שרשור של 4 מחרוזות	17 hi 24	System.out.print (9+8+" hi "+2+4);
חיבור של שני מספרים ולאחריו שרשור של 4 מחרוזות	171+324	System.out.print (9+8+"1+3"+2+4);
חיסור של שני מספרים ולאחריו שרשור של 4 מחרוזות	8 hi 99	System.out.print (10-2+" hi "+9+9);
כפל של שני מספרים ולאחריו שרשור של 4 מחרוזות	20 hi 99	System.out.print (10*2+" hi "+9+9);
כפל קודם לחיבור. פעמיים כפל של שני מספרים ולאחר מכן שרשור של 3 מחרוזות	20 hi 81	System.out.print (10*2+" hi "+9*9);
חיסור בסוגריים. אחריו כפל. אחריו שרשור של 3 מחרוזות	20 hi 0	System.out.print (10*2+" hi "+(9-9));

ההוראה System.out.print (10-2+" hi "+9-9) היא שגויה כיוון ש- "hi"היא מחרוזת. למחרוזת אפשר רק לחבר (לשרשר) ערכים.

תווי בקרה שולטים על עיצוב ההדפסה.

מבנה תו בקרה: הסימן **** ולאחריו אות.

תווי בקרה שימושיים:

תו הבקרה	תפקיד
\n	העברת הראש המדפיס לתחילת שורת ההדפסה הבאה.
\t	העברת הראש המדפיס לתחילת ה tab הבא.

ניתן לשלב תווי בקרה בכל מקום בתוך המחרוזת להדפסה. דוגמאות:

הערך שיודפס	ההוראה להדפסה
we learn	System.out.println (" \n we \t learn \n ");
how to print	System.out.println (" \t how to \nprint ");
7=3	System.out.println (" \n 7=3");
there are 10 students. the average is 87	System.out.println (" there are " + count + " students.\nthe average is " + sum/count);

- **תווי בקרה אפשר לשלב רק בתוך הדפסה של מחרוזת**. למשל: **System.out.println ("\n"+9)** לכן, ההוראה System.out.println (\n+9+9) איננה תקינה.
  בהוראה **System.out.println ("\n"+9+9)** מופיעה מחרוזת, לכן הערכים המשורשרים אחריה הופכים למחרוזות, ולכן מודפסת המחרוזת 99.
- גם בהוראה System.out.println ("\n"+(9+9)) מופיעה מחרוזת, וגם בה הערכים המשורשרים אחרי המחרוזת הופכים למחרוזות, אבל הפעם הודפס 18 כי קודם חושב ערך הביטוי שבסוגריים (9+9) ורק לאחר מכן הוא הפך למחרוזת (כמו במקרה בו במקום תו הבקרה היתה מחרוזת ריקה).
- ההוראה System.out.println (9+9+"\n"+9+9) גורמת להדפסת שתי שורות: בשורה הראשונה הודפס 18 ובשורה השניה הודפס 99 כי רק הערכים שלאחר המחרוזת "\n" הפכו למחרוזות.

<u>אפשרות שניה</u>: הדפסת באמצעות תווי המרה: System.out.format (מחרוזת להדפסה, ערכים);

- בתוך המחרוזת להדפסה, במקום בו רוצים שיופיע התו או המספר, רושמים תו המרה.
- לאחר המרכאות " רושמים פסיק , ולאחריו את התו או המספר (או הביטוי החשבוני) הרצוי.
- אפשר לשלב בהוראת הדפסה מספר תווים ומספרים. במקרה כזה, הערך הראשון יכנס במקום תו ההמרה הראשון, הערך השני יכנס במקום תו ההמרה השני וכן הלאה. תווי המרה שימושיים:

הטיפוס המודפס	תו ההמרה
מספר שלם	%d
מספר ממשי	%f
תו	%c

דוגמאות:

ההוראה	הערך שיודפס
System.out.format ("\nthe temperature is %d degree" , 37);	the temperature is 37 degree
System.out.format (" \n %d + %d = %d ", 2 , 3 , 2+3);	2 + 3 = 5
System.out.format (" \n the average is \t %f ", 89.93);	the average is 89.93
System.out.format (" \n %c comes before %c", 'a' , 'b');	a comes before b

❖ אם משתמשים בתו ההמרה %c בהדפסת מספר שלם, יודפס התו שזהו הקוד האסקי שלו.

הדפסה מעוצבת

❖ הוספת מספר בתוך תו ההמרה קובעת את גודל שדה ההדפסה.

❖ במספרים ממשיים:

- מספר התווים כולל: נקודה עשרונית וסימן מינוס.
- ברירת המחדל: 6 ספרות אחרי הנקודה.
- אפשר להוסיף שני מספרים. המספר הראשון קובע את גודל שדה ההדפסה והמספר השני קובע את מספר הספרות אחרי הנקודה העשרונית (במידת הצורך, מתבצע עיגול של השבר).
- אם מספר הספרות המבוקש אחרי הנקודה, גדול ממספר הספרות אחרי הנקודה, יודפסו אפסים להשלמת מספר הספרות המבוקש.

❖ אם מספר התווים (לאחר העיגול) גדול מגודל שדה ההדפסה המבוקש, לא תהייה התחשבות בגודל שדה ההדפסה המבוקש.

❖ אם מספר התווים קטן מגודל שדה ההדפסה המבוקש, יודפסו רווחים לפני הערך המודפס כדי להגיע לגודל שדה ההדפסה המבוקש.

דוגמאות:

ההוראה	הערך שיודפס
System.out.format ("\n %5d is integer", 37);	37 is integer
System.out.format ("\n %10d is integer", 37);	37 is integer
System.out.format ("\n %4f is read number", 3.65);	3.650000 is read number
System.out.format ("\n %4.2f is read number", 3.65);	3.65 is read number
System.out.format ("\n %9.3f is read number", 3.65);	3.650 is read number
System.out.format ("\n %9c is char", 'x');	x is char

אותיות גדולות וקטנות

1. **מילים שמורות** של השפה נכתבות באותיות קטנות, למשל int , double
2. מוסכמה 1: **שמות של משתנים** נכתבים באותיות קטנות. למעט תחילתה של כל מילה פנימית חדשה שתכתב באותיות גדולות. למשל, numStudent
3. מוסכמה 2: **שמות של מחלק**ות (class) נפתח באותיות גדולות (כמו Sod1). בהמשך, יופיעו רק ספרות ואותיות קטנות למעט תחילתה של כל מילה פנימית חדשה שתכתב באותיות גדולות (כמו FirstProgram).

קוד אסקי

(ASCII = American Standard Information Interchange)

המחשב "יודע" לקרוא רק ספרות בינאריות (0 ו- 1). אולם שפת תכנות יודעת לטפל בתווים שונים (כולל ספרות שונות) – איך הדבר מתבצע?

כל תו "מיוצג" במחשב באמצעות רצף של ספרות בינאריות. למשל התו 'A' מיוצג באמצעות הרצף 01000001 ואילו התו 'B' מיוצג באמצעות הרצף 01000010. רצף הספרות שמייצג כל תו נקרא: קוד אסקי (ASCII) של התו המדובר.

לנוחיותנו, מתורגמים המספרים הבינאריים למספרים עשרוניים. כך למשל, הקוד 01000001 (המייצג את התו A) מתורגם למספר 65. לכן, נאמר כי הקוד האסקי של התו A הוא המספר 65. ובאותו אופן הקוד האסקי של התו B הוא המספר 66.

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 מוקצית כמות הזיכרון הקטנה ביותר ול long מוקצת כמות הזיכרון הגדולה ביותר.

במספרים עשרוניים ל float מוקצית 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 התוי.
- 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 / חיסור 1:

הביטוי ++ שם של משתנה מוסיף 1 לערך של המשתנה. למשל, הביטוי count++ מוסיף 1 לערך של המשתנה count.

```
count = count+1;  ⇔  count++;
```

הביטוי -- שם של משתנה מוריד 1 מהערך של המשתנה. למשל, הביטוי count-- מוריד 1 מהערך של המשתנה count.

```
count = count-1;  ⇔  count--;
```

איתחול (השמה) בזמן הצהרה:

;ערך התחלתי שם של משתנה טיפוס הנתונים

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

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

השמה בו זמנית למספר משתנים:

;ערך = שם משתנה n = ... = שם משתנה2 = שם משתנה1

```
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:

; ביטוי 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 שמאפשרת להשתמש במחלקה Scanner (יצירת עצם מטיפוס המחלקה 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 אחת מהוראות ההדפסה System.out.println או System.out.print

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:

10 / 5 ⟶ 2 10 % 5 ⟶ 0 12 / 7 ⟶ 1 12 % 7 ⟶ 5

(123 / 10) %10 ⟶ 2 (123 % 100) / 10 ⟶ 2

- **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.

הוראות תנאי

דוגמה	משמעות	הוראה
if (grade >95) **System.out.println** ("very good");	אם התנאי מתקיים בצע הוראה	**if** (תנאי) הוראה ; *(או בלוק של הוראות)*

if (grade >60)
{ System.out.println ("pass");
System.out.print ("good grade");
}
else
System.out.println ("not pass");
	אם התנאי מתקיים
	בצע הוראה1
	אחרת
	בצע הוראה 2
		if (תנאי)
		הוראה1 ; *(או בלוק)*
		else
		הוראה2 ; *(או בלוק)*

תנאים מורכבים (לוגיים)

סימן לוגי	משמעות	דוגמה	הערך של תנאי יהיה true
&&	וגם	(a>0 && b>0)	רק אם הערך של כל הביטויים הוא true
||	או	(a>0 || b>0)	אם הערך של לפחות אחד מהביטויים הוא true
!	לא	(! a>0)	אם הערך של הביטוי הוא false

סימנים בביטויי תנאי

התנאי	סימן	משמעות
x < y	<	קטן
x <= y	<=	קטן או שווה
x > y	>	גדול
x >= y	>=	גדול או שווה
x != y	!=	לא שווה
x == y	==	שווה

הוראת בחירה switch

דוגמה	משמעות	הוראה
switch (a) { case 1 : System.out.print ("one"); break; case 2 : System.out.print ("two"); break; case 3 : System.out.print ("three"); break; default : System.out.print ("no one"); a--; }	אם ערך המשתנה = ערך1, בצע הוראות 11 12 אם ערך המשתנה = ערך2, בצע הוראות 21 22 ..... אם ערך המשתנה = ערךn, בצע הוראות n1 n2 אם ערך המשתנה איננו ערך1 ..... ערךn, בצע הוראות 1 2	**switch** (שם של משתנה) **{ case** ערך1 : הוראה11; הוראה12; ..... break; **case** ערך2 : הוראה21; הוראה22; ..... break; ..... **case** ערךn : הוראהn1; הוראהn2; ..... break; **default** : הוראה1; הוראה2; ..... **}**

- הוראת switch היא המקרה היחיד שמאפשר ביצוע של מספר הוראות, לא בתוך בלוק!!
- הביצוע של ההוראה switch (num), מתחיל מהשורה case x שבה הערך של x זהה לערך של num, ונפסק בהוראת break הראשונה.

  לכן, אם רוצים לבצע משהו אחר עבור כל ערך של num (כמו בדוגמה לעיל), צריך לרשום break בסיום סדרת ההוראות עבור כל ערך, ואם רוצים להמשיך את הביצוע, אפשר להשמיט את הוראת break .
- אם רוצים לבצע את אותה הוראה (או בלוק של הוראות) עבור מספר ערכים, רושמים את הערכים ללא break ביניהם, ורושמים את סדרת ההוראות לאחר הערך האחרון.
- כאשר ידועים כל הערכים שהמשתנה עשוי לקבל, אפשר לוותר על שורת ה- default.
- ההוראה break משמשת ליציאה מהבלוק של הוראת ה- switch לכן, אין צורך לכתוב break לאחר סדרת ההוראות של default (או לאחר סדרת ההוראות של ה case האחרון).

הוראות חזרה (לולאות)

חזרה מותנית

הוראה	משמעות	דוגמה

while (תנאי)
גוף הלולאה
	כל זמן שהתנאי מתקיים
	בצע את גוף הלולאה
		while (grade > 100)
		{
		System.out.print ("enter grade");
		grade = input.nextInt();
		}

do גוף הלולאה **while** (תנאי);	בצע גוף הלולאה **כל זמן ש**התנאי מתקיים	do { System.out.print ("enter grade"); grade = input.nextInt(); } while (grade>100);

גוף הלולאה יכול להכיל הוראה אחת או בלוק של הוראות

מסננת קלט לולאה שחוזרת על עצמה עד שיוקלד קלט רצוי.

מכיוון שבהוראת החזרה do-while **הלולאה מתבצעת לפחות פעם אחת,** היא מתאימה למקרים בהם רוצים לבצע את הלולאה לפחות פעם אחת כמו במקרה של מסננת קלט!

חזרה קבועה: לולאת מונה

מבנה ההוראה:

for (הוראות השמה שכוללות ביטוי לשינוי המונה ;תנאי לבדיקה ;ערך התחלתי = מונה הלולאה)
גוף הלולאה

אופן ביצוע ההוראה:

1. הערך ההתחלתי מוצב במונה הלולאה.
2. נבדק התנאי לבדיקה.

אם התנאי אינו מתקיים, מסתיים ביצוע הלולאה.

אם התנאי מתקיים: 2.1. מתבצע גוף הלולאה.

2.2. הערך של מונה הלולאה משתנה בהתאם לביטוי לשינוי המונה.

2.3. חוזרים לבצע את סעיף 2.

דוגמאות לביטויים לשינוי המונה:

דוגמה	משמעות	הביטוי
for (count = 8; count<100; count++)	הוספת 1 למונה הלולאה	++מונה הלולאה
for (i = 10; i>=-10; i--)	הפחתת 1 ממונה הלולאה	--מונה הלולאה
for (i = 0; i <100; i=i+3)	הוספת 3 למונה הלולאה	3 + מונה הלולאה= מונה הלולאה

- כדי שהלולאה תסתיים, הוראות ההשמה צריכות לגרום לכך שלאחר מספר סופי של פעמים בהם מתבצעת הלולאה, התנאי לבדיקה לא יתקיים!
- כאשר יש מספר הוראות השמה, מפרידים ביניהן באמצעות פסיק.
- אפשר להצהיר על מונה הלולאה בתוך הלולאה. למשל, for (int count = 8; count<100; count++)

המחלקה Math

המחלקה Math מאפשרת לבצע פעולות מתמטיות. לדוגמה:

טיפוס הערך המוחזר	טיפוס הערך המתקבל	הערך המוחזר	הפעולה
מספר ממשי	מספר ממשי	הערך המוחלט של הפרמטר	**abs**
מספר שלם	מספר שלם		
מספר ממשי	מספר ממשי	הערך השלם התחתון של הפרמטר	**floor**
מספר ממשי	שני מספרים ממשיים	הערך של הפרמטר הגדול יותר	**max**
מספר שלם	שני מספרים שלמים		
מספר ממשי	שני מספרים ממשיים	הערך של הפרמטר הקטן יותר	**min**
מספר שלם	שני מספרים שלמים		
מספר ממשי	שני מספרים ממשיים	הפרמטר הראשון בחזקת הפרמטר השני	**pow**
מספר ממשי	מספר ממשי	ערך מעוגל של הפרמטר	**round**
מספר ממשי	מספר ממשי	השורש של הפרמטר	**sqrt**
מספר ממשי	----------	מספר אקראי בתחום 0-1 (כולל 0, לא כולל 1).	**random**

- הפעלה של פעולה שמקבלת פרמטר אחד (פועלת על ערך יחיד): **Math.*שם הפעולה*(*פרמטר*);** למשל ההוראה, **Math.abs (num);** היא הפעלה של הפעולה abs . הפרמטר של הפעולה הוא הערך של המשתנה num.
- הפעלה של פעולה שמקבלת שני פרמטרים (פועלת על שני ערכים): **Math.*שם הפעולה*(*פרמטר1, פרמטר2*);** למשל ההוראה, **Math.min (n1 , n2);** היא הפעלה של הפעולה min. הפרמטרים של הפעולה הם הערכים של המשתנים n1 , n2.

- הפעלה של הפעולה random (שלא מקבלת אף פרמטר): **Math.random();**
- הביטוי: ***(1 + המספר הקטן – המספר הגדול)*** מחזיר את מספר המספרים בתחום שבין המספר הקטן לבין המספר (כולל הקצוות)!
- הביטוי ;***המספר הקטן*** + ((1 + ***המספר הקטן – המספר הגדול***) * (Math.random() (int) מחזיר מספר אקראי בתחום שבין המספר הקטן לבין המספר הגדול (כולל הקצוות)!

- מכיוון שכל הפעולות מחזירות ערך, צריך לעשות משהו עם הערך המוחזר. למשל, להציב אותו במשתנה אחר: ;p = Math.pow (2,3)

פעולות (שיטות)

הגדרה של פעולה:

private static **טיפוס הערך המוחזר** שם-הפעולה (**רשימת פרמטרים וטיפוסיהם**)

```
{
  הצהרות של משתנים מקומיים
  הוראות גוף הפעולה
}
```

```
private static double max (double a , double b)
{
  double maximum;
  if (a>b)  maximum = a;
  else  maximum = b;
  return maximum;
}
```

שורה ראשונה == החתימה של הפעולה

הערך שהפעולה תחזיר

הפעולה max מקבלת שני ערכים מטיפוס double (יוצבו בפרמטרים a ו- b).

הפעולה מחזירה ערך מטיפוס double

- כאשר רוצים שפעולה תקבל מספר פרמטרים רשמים ביניהם , למשל כך:

private static double max (double a , double b)

- **הערך שהפעולה מחזירה** יכול להיות מכל אחד מהטיפוסים המוכרים בשפה: int , double , char , boolean

זימון לפעולה (קריאה לפעולה)

בעת הקריאה לפעולה, צריך לומר מה לעשות עם הערך המוחזר על-ידי הפעולה. אפשר לעשות עם הערך המוחזר על-ידי פעולה, כל מה שניתן לבצע עם משתנה מטיפוס המוחזר על-ידי הפעולה.

דוגמאות:

```
System.out.println ("the maximum is "+ max (num1, num2) );
if (max (n1 , n2) < 0) System.out.println ( …….);
max_three = max (a , (max (b , c) ) ;
```

- פעולה יכולה לזמן פעולה אחרת שמוגדרת באותה מחלקה.

פרמטר- משתנה שנמצא בחתימה של הפעולה. מקבל את ערכו בזמן הקריאה (הזימון) לפעולה (בדוגמא: a , b).

משתנה מקומי- משתנה שמקבל את ערכו במהלך הביצוע של הפעולה (בדוגמא: maximum).

פעולות (כולל הפעולה main) לא מכירות את הפרמטרים ואת המשתנים המקומיים של פעולות אחרות !!!!

- **פעולה לא חייבת לקבל פרמטרים**. כאשר פעולה לא מקבלת פרמטרים:

א. בחתימה של הפעולה, במקום רשימת הפרמטרים, לא רושמים כולם. למשל,

private static double message ( )

ב. בעת הזימון לפעולה, צריך להוסיף ( ) לאחר שם הפעולה. למשל, big = message ( );

- **פעולה לא חייבת להחזיר ערך**. כאשר פעולה לא מחזירה ערך:

א. בחתימה של הפעולה, במקום הערך המוחזר, רושמים: void.

למשל, private static void max (int a , int b)

ב. בגוף הפעולה אין הוראה להחזרת ערך.

ג. בעת הזימון של הפעולה, לא צריך לעשות כלום עם ערך מוחזר. למשל, average (a ,b); פעולה כזו שלא מחזירה ערך נקראת פרוצדורה.

- גם **main היא פעולה!!!!** public static void main(String[] args)
- main היא **פעולה ראשית** המופעלת "באופן אוטומטי" עם הפעלת המחלקה.
- main היא פעולה שלא מחזירה שום ערך.
- הערך שהפעולה main מקבלת הוא args מטיפוס String[]

- **פרמטר פורמלי** – הפרמטר כפי שהוא מופיע בחתימה ובגוף של הפעולה.
- **פרמטר אקטואלי** – הערך שניתן לפרמטר בעת הזימון של הפעולה.

משתנים גלובליים – משתנים שמוגדרים בתחילת המחלקה, כל הפעולות מכירות אותן, יכולות להשתמש בהם ויכולות לשנות אותם. למשל,

```
public class MathEx
{
static int n1 , n2;
…….
```