

7	פרק ראשון - אלגוריתם
8	אלגוריתם – הגדרה
8	תיאור תהליך
8	קבוצת הוראות
8	חד משמעית
8	סדר הביצוע
9	אלגוריתם ופתרון בעיות במחשב
9	בעיה אלגוריתמית – הגדרה
9	נקודת המוצא
9	יעד מטרת האלגוריתם
12	מבנה תנאי ומבנה לולאה באלגוריתמים
12	הוראת בדיקה
12	הוראה לביצוע חוזר
13	ניסוח סופי של מבנה הפקודות באלגוריתמים - סיכום
13	מבנה סדרתי
13	מבנה תנאי
13	מבנה תנאי מורחב
13	מבנה חוזר – לולאה
14	מבנה חוזר לא מותנה – for
14	מבנה חוזר מותנה – while
14	שפת תכנות ואלגוריתם
14	פקודות בשפה החדשה שלנו:
16	מספור והזחה בכתיבת אלגוריתמים
16	מספור
17	הזחה (Indent)
18	קומפילרים ומפרשים - COMPILER & INTERPRETER
19	פרק שני – מבוא לשפת ג'אווה
20	הקדמה
20	מחלקה בג'אווה
21	תכונות והתנהגות של מחלקה
21	משתני האובייקט - object variables
21	משתנה מחלקה – class variable
22	ערך קבוע למשתנה מחלקה
22	תכונות מחלקה
22	שיטות במחלקה – METHODS
23	Instance methods - שיטות מופע
23	חתימת שיטה המחזירה ערך
23	שיטה שאינה מחזירה ערך
24	Class methods – שיטות מחלקה
24	יצירת מחלקה
24	השיטה main
24	השיטה הבונה constructor
25	השוואה בין ג'אווה לפסקל – שתי תוכניות פשוטות
25	השיטות print ו- println
27	קבלת קלטים מהמשתמש: המחלקה IO
27	קבלת קלטים מהמשתמש: המחלקה Scanner
28	השיטה hasNext()
29	שם משתנה בג'אווה - כללים
30	פעולות מתמטיות בסיסיות
30	סוגי משתנים בסיסיים בג'אווה
31	טבלת טיפוסים נתונים בסיסיים
31	הגדרה של טיפוסים נתונים בסיסיים - primitive types

31	סוגי טיפוסים מסוג שלם:
31	טיפוסים מסוג ממשי:
31	טיפוס תו:
31	טיפוס לוגי:
32	שמות המשתנים - סיכום
33	משפט השמה בג'אווה
34	המרה בין טיפוסים משתנים באמצעות Casting –
35	מחלקה Basicop - ביצוע פעולות אריתמטיות פשוטות
35	כתיבת ביטויים מתמטיים בג'אווה
35	טבלת משפטי השמה
36	טבלת אופרטורים
36	תחום ההכרה (מרחב "המחיה" של משתנים) - Scope
39	פרק שלישי – מתחילים לתכנת בג'אווה
41	הכרזה על אובייקטים:
41	מה מבצע האופרטור new?
41	יצירה של מופע מחלקה:
41	הקצאת ערכים לאובייקטים.
42	סיכום - מבוא לאובייקטים
42	מהו "אובייקט" וממה הוא מורכב?
42	שיטות:
43	סיכום
44	אופרטורים, משפטי השמה, קלט פלט של תוכניות
44	מחלקה ג'אווה MoreBasicop – מחשבון פשוט
45	מחלקה Math - סכום וממוצע
45	מחלקה Gader
47	מחלקה IncDec - מוסיפים ומחסירים 1.
48	מחלקה Monit - כמה מוניות?
49	המחלקה ArithmeticDemo מציגה סיכום פעולות
51	מחלקה sidra
53	חלוקה שלמה - תזכורת
54	מחלקה DigitNum – מפרידה מספר לספרותיו
54	סיכום כללים בכתיבת תוכנית בג'אווה
55	האופרטור נקודה – Dot notation
57	פרק רביעי – לשאול שאלות
58	אופרטורי יחס
60	משפט תנאי בג'אווה - if
61	מחלקה ifClass
61	טבלת דוגמא - שימוש באופרטורי יחס עם המשתנים הוגדרו
63	מחלקה MyMath
65	אופרטורים לוגים בג'אווה
66	קינון משפטי תנאי
67	אופרטורים לוגים
68	האופרטור and
68	האופרטור or
69	האופרטור not
70	שילוב בין האופרטורים הלוגיים
81	סיכום – שלבים בבניית מחלקה בג'אווה
82	מחלקה STRING
84	שיטות לטיפול במחרוזות
87	פרק חמישי – לולאות

94.....	חישוב ממוצע – אלגוריתם
98.....	המחלקה SumEven – סכום המספרים הזוגיים
99.....	משפט DO WHILE
99.....	משימה –סכום המספרים $n-1$
100.....	המחלקה myDiv – ביצוע חלוקה שלמה ללא פעולת חילוק
101.....	משימה –סכום ספרות של מספר שלם
101.....	המחלקה SumDigit
102.....	המחלקה MyTest
104.....	המחלקה LetDigit
105.....	המחלקה FindMaxPlace
107.....	משחק "נחש את המספר"
109.....	המחלקה MonteCarlo
111.....	לולאות מקוננות
112.....	המחלקה lokefel - יצירת לוח כפל
113.....	המחלקה stars
114.....	המחלקה printStar – הדפסת מלבן כוכביות
115.....	המחלקה ZipCode
116.....	המחלקה pascalTriangle – הדפסת משולש פסקל
117.....	לולאות וצבים
117.....	המחלקה turtle1
119.....	המחלקה turtle2
120.....	המחלקה turtle3
122.....	המחלקה turtle4
125.....	המחלקה fibonachiGraph
127.....	תבניות
128.....	תבנית הסכום
129.....	תבנית צבירת מכפלה
130.....	תבנית מניה
131.....	תבנית ממוצע
132.....	תבנית ממוצע לולאת while
133.....	תבנית מניה לולאת while
137.....	תבניות מציאת מינימום ומקסימום
138.....	התבנית מציאת המקסימום מינימום כולל מיקום המספר בסדרת המספרים
140.....	משימת סיכום שילוב תבניות מציאת מקסימום, מקסימום, ממוצע
141.....	פרק שישי – מערכים
142.....	מערך – הגדרה כללית
142.....	הגדרת מערך בג'אווה
142.....	סוגים שונים (TYPE) של מערך
150.....	מערכים דו ממדיים
153.....	פרק שביעי – תכנות מונחה עצמים
154.....	על אובייקטים תכונות ושיטות
155.....	האופרטור new
158.....	דוגמאות לעצמים
161.....	חתימת השיטה main
162.....	סיכום מושגים – אובייקטים
163.....	מחלקה KLAf
166.....	מחלקה Point
166.....	המחלקה Point
167.....	המחלקה testPoint
169.....	שיטה בונה מעתיקה – Copy Constructor
171.....	מערך של אובייקטים
171.....	המחלקה testPointTurtle
173.....	המחלקה dirot

177.....	Dia המחלקה
178.....	DiscGame המחלקה
179.....	Forms המחלקה
181.....	DrawForm המחלקה
182.....	משחק "מלחמה"
182.....	Card המחלקה
183.....	Player המחלקה
185.....	game המחלקה
187.....	RaceTurtle המחלקה
190.....	Objectsus המחלקה
196.....	LifeGame המחלקה
201.....	פרק שמיני – הורשה, כימוס ועצמים מורכבים
202.....	Encapsulation עקרון הכימוס
204.....	על נקודות ומלבנים
205.....	Point המחלקה
206.....	Malben המחלקה
207.....	malbenProject המחלקה
210.....	Class methods – שיטות מחלקה - דוגמאות מימוש
210.....	Sons המחלקה
212.....	טבלת סיכום והשוואה בין תכונת מופע ותכונת מחלקה
213.....	INHERITANCE - הורשה
214.....	דוגמא: ארץ האגדות FAIRY TALE
215.....	inheritance – הורשה
218.....	המחלקה גמד
218.....	המחלקה הגמד המקפץ
220.....	דוגמא: קווי נסיעה
220.....	Line המחלקה
222.....	BusLine המחלקה
223.....	MeasefBus המחלקה
225.....	polymorphism - פולימורפיזם
225.....	האם זה יתכן?
226.....	Form – מחלקת האב
227.....	הגדרה מחדש של שיטות (Method's Overriding)
228.....	Circle מהמחלקה Form
229.....	Square היורשת מהמחלקה Form
230.....	Trapez היורשת מהמחלקה Form
231.....	Worker המחלקה
231.....	המחלקות היורשות של Worker
232.....	Manger היורשת מהמחלקה Worker
236.....	WorkerApp - המחלקה
237.....	הגדרה מחדש של שיטות overriding
237.....	גישה לשיטה מוסתרת באמצעות super
238.....	המורות – דוגמא מפורטת
238.....	caramic המחלקה
239.....	wallcarmic המחלקה
241.....	item המחלקה
242.....	instanceof האופרטור
243.....	carmicApp המחלקה
245.....	Overloading – העמסת שיטות
246.....	תרגילים
246.....	תרגילים – פרק 3
248.....	תרגילים – פרק 4
252.....	תרגילים – פרק 5
255.....	תרגילים – פרק 6

257.....	נספח א –ספריית מחלקות
259.....	נספח ב- שיטות הספרייה MATH
263.....	נספח ג – המחלקה STRING - שיטות המחלקה
269.....	נספח ד- מילים שמורות בג'אווה
271.....	נספח ה- שיטות המחלקה IO
272.....	נספח ו – עבודה עם ג'אווה סביבת ECLIPSE
277.....	נספח ז' – עבודה בסביבת JELIOT
283.....	נספח ח' – מדריך מקוצר לפעולות ב-ECLIPSE
285.....	נספח ט' – מילון מונחים
289.....	נספח י' – פתרון תרגילים
289.....	תרגילי פרק 3
289.....	תרגיל 1
289.....	תרגיל 2
290.....	תרגיל 3
291.....	תרגיל 4
291.....	תרגיל 5
292.....	תרגיל 6
292.....	תרגיל 7
293.....	תרגיל 8
293.....	תרגיל 9
293.....	תרגיל 10
294.....	תרגיל 11
294.....	תרגיל 12
294.....	תרגיל 13
295.....	תרגיל 14
296.....	תרגילי פרק 4
296.....	תרגיל 1
297.....	תרגיל 2
297.....	תרגיל 3
297.....	תרגיל 4
297.....	תרגיל 5
298.....	תרגיל 6
298.....	תרגיל 7
299.....	תרגיל 8
300.....	תרגיל 9
300.....	תרגיל 10
300.....	תרגיל 11
301.....	תרגיל 12
301.....	תרגיל 13
302.....	תרגיל 14
302.....	תרגיל 15
303.....	תרגיל 16
303.....	תרגיל 17
304.....	תרגיל 18
304.....	תרגיל 19
304.....	תרגיל 20
305.....	תרגיל 21
307.....	תרגילי פרק 5
307.....	תרגיל 1
307.....	תרגיל 2
308.....	תרגיל 3
308.....	תרגיל 4
309.....	תרגיל 5

309.....	תרגיל 6
309.....	תרגיל 7
310.....	תרגיל 8
311.....	תרגיל 9
312.....	תרגיל 10
312.....	תרגיל 12
313.....	תרגיל 13
314.....	תרגיל 14
316.....	תרגילי פרק 6
316.....	תרגיל 1
318.....	תרגיל 2
319.....	תרגיל 3
321.....	נספח יא' – רקורסיה
322.....	מחלקה <code>sum1_To_K</code>
323.....	מחלקה <code>power</code>
324.....	מחלקה <code>digit</code>
325.....	מחלקה <code>fact</code>
326.....	מחלקה <code>fibonachi</code>
327.....	מחלקה <code>multy</code>
328.....	מחלקה <code>moneDigit</code>
329.....	מחלקה <code>sum_digit_zugi</code>
330.....	מחלקה <code>sortNumber</code>
331.....	מחלקה <code>divider</code>
332.....	מחלקה <code>sumarray</code>
333.....	מחלקה <code>bigAarray</code>



פרק ראשון - אלגוריתם

פרק זה אסביר מהו אלגוריתם, האלגוריתם אינו תלוי בשפת התכנות בה הוא ממומש. אנו מנסחים אלגוריתמים כדי לפתור בעיות אלגוריתמיות. הפרק מתאר ומסביר מהו אלגוריתם, מציג אלגוריתמים הפותרים בעיות שונות, וגם מסביר כיצד נכתוב אלגוריתם בשפה שאינה שפת תכנות, אלא "פסאודו קוד". בתום קריאת הפרק תוכל לכתוב אלגוריתמים לפתרון בעיות שונות, תדע להשתמש במבני תכנות הכוללים:

מבנה סדרתי

מבנה תנאי

מבנה לולאה

פרק זה יכין אותך לכתיבת התוכניות הראשונות שלך בשפת ג'אווה.



פרק ראשון – אלגוריתם

אלגוריתם – הגדרה

אלגוריתם הוא "תיאור תהליך" לביצוע משימה. האלגוריתם בנוי מאוסף הוראות חד משמעיות לביצוע המשימה וסדר ביצוען מוגדר היטב.

תיאור תהליך

השלבים בביצוע המשימה, המתכון, כלומר כיצד לבצע את המשימה.

קבוצת הוראות

קבוצת ההוראות חייבת להיות מובנת "למכונה" המבצעת את ההוראות, לדוגמא, במתכון לאפיית עוגה כל ההוראות חייבות להיות ברורות לאופה, ואכן קבוצת ההוראות לאפיית עוגה כוללת הוראות מהסוג:

- הדלק תנור
- קבע חום ל- 180 מעלות
- בחש את העיסה
- הוסף כוס מים.

כל ההוראות הללו ידועות למבצעים שהם האופים. אני מניח שאם הייתה ההוראה :
אם התנור לא דולק, אזי החלף גוף חימום של התנור ! מרבית האופים לא היו מסוגלים לבצעו.
לכן קבוצת ההוראות באלגוריתם חייבת להיות ברורה וידועה למבצע.

חד משמעית

האם האופה יכול לבצע את ההוראה: **חמם את התנור לטמפרטורה הקרובה ל- 180 מעלות ?**
התשובה היא לא. במקרה הזה כל אופה יחמם את התנור לטמפרטורה אחרת. ההוראה לאופה חייבת להיות חד משמעית, אחרת ישנו סיכוי רב שלא נטעם מהעוגה.

סדר הביצוע

להבנת חשיבות סדר הביצוע של ההוראות באלגוריתם, הנה מתכון לאפיית עוגה שכתבתי:

1. הכנס את העוגה לתנור
2. לוש את הבצק
3. הגש את העוגה לשולחן
4. הוצא מהתנור את העוגה
5. הוסף שמרים לקמח
6. פזר סוכריות צבעוניות על העוגה המוכנה

ההוראה האחרונה "פזר סוכריות צבעוניות על העוגה המוכנה", היא אכן השלב האחרון, אולם האם ניתן להכניס את העוגה לתנור, ולאחר מכן ללוש את הבצק? האם ניתן להגיש את העוגה לשולחן, ולאחר מכן להוציא את העוגה מהתנור?

ברור אם כן שסדר הביצוע של האלגוריתם (מתכון) הוא חשוב.

אלגוריתם ופתרון בעיות במחשב

במדעי המחשב עוסקים באלגוריתמים הפותרים בעיות אלגוריתמיות.

בעיה אלגוריתמית – הגדרה

בעיה אלגוריתמית היא בעיה שבה מתוארת נקודת המוצא, והיעד שהאלגוריתם נדרש להשיג.

נקודת המוצא

באלגוריתם חייבים לתאר את המצב ההתחלתי, ואת תנאי הבעיה שהתקבלו. הגדרה מדויקת מהווה מרכיב חשוב בפתרון הבעיה באמצעות האלגוריתם.

יעד \ מטרת האלגוריתם

אלגוריתם אינו יכול להיות ללא יעד מוגדר, לעיתים יתכן שהיעד אינו ניתן להשגה (כן גם במחשבים יש בעיות בלתי פתירות).

דוגמא:

לרשותך שני מכלי מים, מיכל א' ומיכל ב'. מיכל א' יכול להכיל 5 ליטר מים ומיכל ב' יכול להכיל 3 ליטר מים.

הערה: אין סימון כמות על מיכלים.

הפעולות שהינך יכול לבצע הן:

מלא מיכל מים א'

מלא מיכל מים ב'

העבר מים ממכל א' למיכל ב'

העבר מים ממכל ב' למיכל א'

שפוך מיכל מים א'

שפוך מיכל מים ב'

מטרה: מלא במיכל א' 4 ליטר מים

אלגוריתם הפתרון:

1. מלא מיכל מים ב'
2. העבר מיכל מים ב' למכל א'
3. מלא מיכל מים ב'
4. העבר מיכל מים ב' למיכל מים א' עד שיתמלא מיכל מים א'
5. שפוך מיכל מים א'
6. העבר מיכל מים ב' למיכל מים א'
7. מלא מיכל מים ב'
8. העבר מיכל מים ב' למיכל מים א'

תוצאת סיום: במיכל מים א' יש בדיוק 4 ליטר מים, בדוק!!

שים לב !

- כל ההוראות, בהן השתמשנו באלגוריתם, שייכות לקבוצת הפעולות החוקיות.
- המצב ההתחלתי של הבעיה האלגוריתמית מוגדר היטב.
- היעד מוגדר היטב.
- סדר הביצוע חשוב (שנה את סדר האלגוריתם רק בהוראה אחת וצפה בתוצאה...)

שאלה

האם האלגוריתם שכתבנו הוא היחיד הפותר את הבעיה?

תשובה

יתכן, אולם, על פי רוב, התשובה לשאלה היא לא. לבעיה יתכנו אלגוריתמים שונים שכולם פותרים את הבעיה האלגוריתמית שהוצגה, כלומר, מגיעים באמצעות אלגוריתמים שונים למטרה

שאלה

האם האלגוריתם שהוצע הוא המהיר ביותר?

תשובה

יתכן שזה האלגוריתם המהיר ביותר, אולם יתכן שישנו, או ישנם, אלגוריתמים שיפתרו את הבעיה בפחות הוראות, וביצועם ימשך פחות זמן. השאלה איזה אלגוריתם מהיר יותר, יעיל יותר היא שאלה חשובה במדעי המחשב, ומדעני המחשב עסקו ועוסקים בה.

שאלה

האם האלגוריתם שכתבנו נכון?

תשובה

זו שאלה, לא פחות, מורכבת וחשובה מהשאלה על מהירות ויעילות אלגוריתמים. נכונות ואי נכונות אלגוריתם הוא נושא מתקדם, בשלב זה ניתן להגדיר הגדרה מצומצמת ופשוטה: אלגוריתם נכון אם הפלט יהיה נכון עבור כל קלט שיקלט. לא פשוט לבחון אלגוריתם ולהפעילו עבור כל קלט אפשרי, ולכן אנשי מדעי המחשב פתחו דרכים ושיטות לבחינת אלגוריתמים. אנו נסתפק בביקורת מצומצמות ונגיד שאלגוריתם והתוכנית המממשת אותו נכונים לאחר בדיקה של מספר קלטים (לא ברמת הוכחה מתמטית מלאה..)

שאלה

האם אלגוריתם " יודע לשאול שאלות" ?

תשובה

נניח שיש לנו מיכל מים של 5 ליטר (מיכל א') ומיכל מים של 1 ליטר (מיכל ב') . קבלנו בעיה למלא את מיכל המים של 5 הליטר, אולם עם מגבלה: אסור למלא את מיכל המים של ה-5 ליטר באופן ישיר אלא באמצעות מיכל המים של 1 ליטר. כיצד יראה האלגוריתם לפתרון הבעיה?

תזכורת: אלו הפקודות שהאלגוריתם "יודע" לבצע.

מלא מיכל מים א'

מלא מיכל מים ב'

העבר מים ממיכל א' למיכל ב'

העבר מים ממיכל ב' למיכל א'

שפוך מיכל מים א'

שפוך מיכל מים ב'

פתרון א.

1. מלא מיכל מים ב'
2. העבר מים ממיכל ב' למיכל א'
3. מלא מיכל מים ב'
4. העבר מים ממיכל ב' למיכל א'
5. מלא מיכל מים ב'
6. העבר מים ממיכל ב' למיכל א'
7. מלא מיכל מים ב'
8. העבר מים ממיכל ב' למיכל א'
9. מלא מיכל מים ב'
10. העבר מים ממיכל ב' למיכל א'

בתום 10 השלבים האלגוריתם הגיע למטרה, מילוי מיכל מים א' ב-5 ליטר באמצעות מיכל המים של 1 ליטר.

שאלה

כיצד יראה הפתרון לבעיה האלגוריתמית: מלא מיכל מים א' (שמכיל הפעם 100 ליטר) באמצעות מיכל מים ב' (שמכיל 1 ליטר).

תשובה

אנו נכתוב אלגוריתם מאוד ארוך עם 200 פקודות של מילוי מיכל מים ב' והעברתו למיכל מים א'.

מבנה תנאי ומבנה לולאה באלגוריתמים

ברור שלא ניתן לפתור בעיות אלגוריתמיות מורכבות באמצעות כתיבה של אותם הוראות עשרות, מאות ולעיתים אלפי פעמים. הפתרון לבעיה, להוסיף לאלגוריתמים יכולת "לבדוק" מצבים שונים, "ולפעול" בהתאם לתוצאות הבדיקה ולחזור על קבוצת הוראות מספר פעמים לפי קביעה מראש, או כאשר יתרחש "אירוע".

הוראת בדיקה

נניח, שאנו מוסיפים לקבוצת הפקודות בבעיות המיכלים את הפקודה הבאה:

אם תנאי מתקיים אז בצע פקודה

דוגמאות:

אם מיכל א' = מלא אז

סיים

אם תכולת מיכל א' < תכולת מיכל ב' אז

שפוך מיכל ב'

אם תכולת מיכל א' = תכולת מיכל ב' אז

סיים

הוראה לביצוע חוזר

נניח שאנו מוסיפים לקבוצת הפקודות בבעיות המיכלים את הפקודה הבאה:

בצע N פעמים את קבוצת ההוראות הבאה

(N מייצג מספר כללי של פעמים).

שאלה

כיצד נכתוב עתה את האלגוריתם של מילוי מיכל א' ב-5 ליטר באמצעות מיכל ב' של 1 ליטר.

תשובה :

בצע 5 פעמים את קבוצת ההוראות הבאה:

1. מלא מיכל ב'

2. העבר למיכל א'

בדוק: אלגוריתם זה פותר את הבעיה שפתרנו ב-10 הוראות.

ניסוח סופי של מבנה הפקודות באלגוריתמים - סיכום

אנו מבחינים ב-3 אופנים לאגד קבוצות פקודות באלגוריתם:

מבנה סדרתי

קבוצת פקודות אשר מתבצעת הוראה אחר הוראה לפי סדר כתיבתם

דוגמא:

1. מלא מיכל מים א'
2. שפוך מיכל מים ב'
3. העבר מיכל מים א' למיכל מים ב'

מבנה תנאי

קבוצת פקודות המתבצעת אם תנאי מתקיים

דוגמא:

1. אם תכולת מיכל מים א' = תכולת מים ב' אז
 - 1.1 שפוך מיכל מים א'
 - 1.2 שפוך מיכל מים ב'
- ההוראות 1.1 ו-1.2 מתבצעות רק אם התנאי נכון (שני המיכלים מכילים את אותו כמות מים)

מבנה תנאי מורחב

1. אם תכולת מיכל מים א' > תכולת מיכל מים ב' אז
 - שפוך מיכל מים א'
 - העבר מיכל ב' למיכל מים א'
2. אחרת
 - שפוך מיכל מים ב'
 - העבר מיכל א' למיכל מים ב'

מבנה חוזר – לולאה

בצע "מספר פעמים"

קבוצת הוראות

דוגמא

1. בצע 10 פעמים
 - 1.1 מלא מיכל מים א'
 - 1.2 העבר מיכל מים א' למיכל מים ב'
 - 1.3 רוקן מיכל מים ב'

במבנה חוזר – לולאה יש שתי גירסאות, מבנה חוזר לא "מותנה" ומבנה חוזר "מותנה". נהוג לרשום את המבנים באופן הבא:

מבנה חוזר לא מותנה – for

עבור (ערך התחלתי, ערך סופי) בצע
קבוצת הוראות

מבנה חוזר מותנה – while

כל עוד (תנאי) בצע
קבוצת הוראות

לסיכום

כדי לפתור בעיות כלליות ומורכבות אנו חייבים לכלול באלגוריתמים שאנו כותבים את שלושת המבנים בהם ניתן לארגן את ההוראות: סדרתי, תנאי, לולאה .

צורת הכתיבה בה אנו כותבים אלגוריתמים נקראת קוד מדומה (פסאודו קוד) של אלגוריתם הוא כתיבה כללית של האלגוריתם ללא תלות בשפת תכנות כלשהי. הכתיבה בפסאודו קוד מאפשרת לנו לבנות ולנסח פתרונות לבעיה ללא תלות בשלבי ישום הפתרון בשפת תכנות.

שפת תכנות ואלגוריתם

אם מוצגת לנו בעיה, ואנו מנסחים וכותבים אלגוריתם קוד מדומה (Pseudo Code) בשלב הראשון, ובשלב שני אנו כותבים בשפת תכנות (לדוגמא Java) אנו מבצעים "מימוש" בשפת התכנות של האלגוריתם. הערה: בספרים רבים כתיבת אלגוריתם בקוד מדומה – נקראת **אלגוריתם מילולי** והתהליך נקרא **"פיתוח אלגוריתם"**

בעיות אלגוריתמיות עם מספרים טבעיים

בבנה עתה שפת אלגוריתמים חדשה הכוללת מספר "עצמים" ופקודות.

עצמים בשפה החדשה

בשפה שלנו יש "משתנים" משתנים אלו מכילים ערכים שהם מספרים שלמים. יש לתת שמות למשתנים הללו שיכילו ערכים שונים

פקודות בשפה החדשה שלנו:

פקודת השמה: השם ערך במשתנה - הפקודה מכניסה ערך לתוך משתנה
דוגמא:

$$X = 35$$

ניתן גם לתת שמות בעברית:
דוגמא:

תשלום = 34

פקודת קלט : קלוט (משתנה) - הפקודה מאפשרת הכנסת ערך מקלט חיצוני למשתנה
דוגמא: קלוט(מספר_הזמנות)

פקודת פלט: הצג(ביטוי) – הפקודה מציגה את הערך שמכיל הביטוי כפלט
דוגמאות:

הצג(מספר_הזמנות)

הצג(מספר_הזמנות + מספר_ממתינים)

פקודת תנאי: אם תנאי אז

בצע – קבוצת פקודות

אחרת

בצע – קבוצת פקודות

דוגמא:

1. קלוט(מספר_הזמנות)

2. אם מספר_הזמנות < 0 אז

1.1 הצג(מספר_הזמנות)

1.2 תשלום = מספר_הזמנות * מחיר

1.3 הצג תשלום

3. אחרת

הצג(מספר הזמנות לא חוקי")

פקודת חזור for : עבור (ערך התחלתי ; תנאי סיום; עדכון ערך התחלתי) בצע
קבוצת פקודות

דוגמא :

1. סכום = 0

2. עבור(מספר_פעמים=1; מספר_פעמים=10; הוסף 1)

2.1 קלוט(מספר_הזמנות)

2.2 סכום = סכום + מספר_הזמנות

3. הצג(סכום)

פקודת חזור while :

כל עוד (תנאי) בצע

קבוצת פקודות

דוגמא:

1. סכום = 0
 2. קלוט(מספר_הזמנות)
 3. כל עוד(מספר_הזמנות < 0) בצע
 - 3.1 סכום = סכום + מספר_הזמנות
 - 3.2 קלוט(מספר_הזמנות)
 4. הצג(סכום)
- משימה: כתוב אלגוריתם הקולט מספר (שלם וחיובי) ומדפיס את סדרת המספרים 1 עד המספר שנקלט. השתמש בשפה החדשה שהגדרנו ובמבנה הפקודות שהגדרנו(סדרתי, תנאי, לולאה)

פתרון

1. קלוט(מספר)
 2. מספר_פעמים = 1
 3. כל עוד(מספר < מספר_פעמים) בצע
 - הצג(מספר_פעמים)
 - מספר_פעמים = מספר_פעמים + 1
- משימה: כתוב אלגוריתם הקולט שני מספרים ומציג את המספר הגדול (אם הם שווים יוצג אחד)
1. קלוט(מספר_א , מספר_ב)
 2. אם מספר_א < מספר_ב אז
 - 2.1 הצג(מספר_א)
 3. אחרת
 - 3.1 הצג(מספר_ב)

מספור והזחה בכתיבת אלגוריתמים

מספור

בכל הפתרונות מספרנו את ההוראות, המספור מאפשר מעקב אחר סדר ביצוע התוכנית (גם העורך של ג'אוה נותן מספור אוטומטי של הפקודות). המספור מסייע לנו לקבץ קבוצת פקודות ולהציג שמדובר אכן בקבוצת פקודות. זאת אנו משיגים באמצעות מספור משני של קבוצות פקודות.

לקבוצת פקודות המתבצעת בלולאה או כתוצאה של תנאי אנו קוראים "בלוק – Block"

בלוק הפקודות המאגד קבוצת פקודות ללולאה, משפט תנאי מרובה פקודות או מחלקה ושיטה בג'אוה מיוצגים באופן שונה בשפות תכנות שונות.

בפסקל בלוק של פקודות מתוחם באמצעות המלים :

BEGIN

קבוצת הפקודות השייכת לבלוק

END

בג'אוה בלוק פקודות "חסום" באמצעות סוגריים מסולסלים {}

{

קבוצת הפקודות השייכת לבלוק

}

הזחה (Indent)

כאשר אנו כותבים אלגוריתמים מילוליים, ומיישמים את האלגוריתמים בשפת תכנות, אנו משתדלים ליצור כתיבה שתהייה מובנת למעיינים בתוכנית. אחת הדרכים ליצור הבנה של מבנה התוכנית היא לכתוב כך שכל פקודה בקבוצת פקודות מתחילה באותה עמודה, וכן אם קבוצת פקודות מהווה חלק משני מקבוצת פקודות אחרת, אנו מבצעים "הזחה" פנימה של קבוצת הפקודות כך שהמעין יכול להבחין מצורת הכתיבה שאכן מדובר בקבוצה השייכת לקבוצת פקודות אחרת.

דוגמא:

1. עבור (מספר = 1 ; מספר = 10 ; הוסף 1 למספר)

1.1 הצג (מספר)

1.2 סכום = סכום + מספר

1.3 עבור (מספר_פעמים = סכום ; מספר_פעמים = 100 ; הוסף 1

למספר_פעמים)

1.3.1 הצג(מספר_פעמים)

1.3.2 הצג(סכום)

שים לב!

קבוצת ההוראות 1.1 – 1.3 שייכת ללולאה המרכזית

קבוצת ההוראות 1.3.1 – 1.3.2 אף היא שייכת ללולאה המרכזית ובעצם מהווה הוראה אחת בתוך לולאה זו, כתיבה באמצעות מיספור ושימוש בעקרון ההזחה, גם במיספור וגם במיקום הפקודה, מאפשר להבין טוב יותר את האלגוריתם.

קומפילרים ומפרשים - *Compiler & interpreter*

כתיבת תוכנית מחשב בסביבת העבודה יוצרת קובץ, קובץ זה אינו מובן למחשב, ואינו ניתן להרצה. המתכנת יוצר בעת כתיבת התוכנית קובץ במעבד תמלילים בסיסי – Editor. כאשר אנו פותרים בעיה אלגוריתמית, מנסחים את הפתרון בפסאודו קוד, כפי שלמדנו וכותבים תוכנית בשפת התוכנית לא נשלמה מלאכתנו. מהתוכנית שכתבנו יש ליצור תוכנית שתובן ע"י המחשב, והמחשב יוכל "להריץ" את התוכנית, כלומר, להפעילה. התוכנית המקורית שכתבנו (ניתן לומר גם "הקוד" שכתבנו) נקראת תוכנית המקור – Source code, קוד המקור מתורגם באמצעות תוכנית המתרגמת אותו לתוכנית לה אנו קוראים Target code, התוכנית המבצעת את תהליך התרגום נקראת Compiler (מהדר) או Interpreter (מפרש). ההבדל בין **compiler** ו- **interpreter** הוא, שמהדר מבצע תהליך את יצירת קוד המטרה ולאחר מכן מתבצעת הרצת התוכנית, לעומת זאת המתרגם משלב בין תהליכי התרגום והריצה, עובדה המאטה את ביצוע התוכנית. בחלק משפות התכנות המהדר מבצע תהליך שבסופו מתקבלת תוכנית "בשפת מכונה" (שפת 0, 1 המובנת למחשב) ובחלק לא כך הדבר. בג'אווה התהליך אינו תהליך של שלב אחד, אלא מספר שלבים;

בשלב הראשון התוכנית עוברת הידור באמצעות המהדר, ויוצרת תוכנית הנקראת Byte code בשלב השני המתרגם של ג'אווה מתרגם ומבצע את קובץ Byte code. שלבים אלו שהתהליך מבצע חורגים מנושאי הספר, אציין שהסיבה החשובה לתהליך התרגום, היותר מורכב, בשפת ג'אווה, נעוצה בעובדה ששפת ג'אווה אינה תלויה בסוג המחשב ומערכת ההפעלה עליו היא פועלת, ואותו תוכנית ניתנת להרצה על "פלטפורמות" שונות. היכולת הזו של שפת ג'אווה מהווה את אחד היתרונות המרכזיים של השפה שהפכו את השפה וסביבת העבודה שלה לפופולארית.

בתהליך ההידור (בשפת המתכנתים תהליך "הקומפילציה") תוכנית ג'אווה נבדקת ולכותב התוכנית מוצגות כל בעיות "התחביר" אם היו. אילו היו בעיות לא יהיה המשך לתהליך ולא ייווצר קובץ Byte code. על כותב התוכנית לתקן את כל השגיאות עד לסיום מוצלח של תהליך ההידור.

בתהליך כתיבת התוכנית עלולות להיות טעויות משלושה סוגים:

שגיאה תחבירית – לא כתבת נכון לדוגמא כתבת את המילה System כך System
שגיאת ריצה - שגיאה המתרחשת בעת הרצת התוכנית, לדוגמא, נסיון לחלק באפס
שגיאה לוגית - שגיאה שהמחשב אינו מודיע עליה, כי התוכנית רצה ללא שגיאות (לפחות מבחינת המחשב), אולם אינה מפיקה את התוצאות המקוות (המתוכננות)



```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello JAVA World!");  
    }  
}
```



```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello JAVA World!");  
    }  
}
```

פרק שני – מבוא לשפת ג'אווה

הפרק השני בספר לוקח אותנו לסיור מהיר בשפת ג'אווה, מהי מחלקה בג'אווה, מהו אובייקט, מהי שיטה וכיצד מחברים הכול לכלל תוכנית שפותרת בעיה אלגוריתמית שהוצגה לנו. המשך הפרק מתאר משתנים בשפה, אופרטורים לביצוע פעולות על ערכים שונים. בפרק מספר תוכניות פשוטות הכוללות, הכרזה על משתנים, ושימוש בפעולות השמה.

מומלץ להריץ בפרק זה את כל התוכניות.

.



```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello JAVA World!");  
    }  
}
```



```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello JAVA World!");  
    }  
}
```

פרק שני – מבוא לשפת ג'אווה

הקדמה

ג'אווה היא שפה מונחת עצמים, המאפשרת תכנות מונחה עצמים. מהו בדיוק תכנות מונחה עצמים, ובמה הוא שונה מתכנות רגיל? התשובות לשאלות אינן פשוטות לא למתכנתים מתחילים וגם לא לדוברי שפות פרוצדורליות (פסקל, C וכיו"ב..), בספר זה נסביר בפרוטרוט דוגמאות רבות, מהו תכנות מונחה עצמים, וכיצד כותבים תוכניות מחשב בג'אווה?

הערה: בג'אווה אנו כותבים מחלקות, הכוללות משתנים בסיסים, שיטות בונות ושיטות לעיתים נשתמש במילה "השגרתית" – תוכנית או יישום כאשר נתייחס למחלקה בג'אווה.

מחלקה בג'אווה

מחלקה בג'אווה מהווה את התבנית ליצירת אובייקטים בעלי אותן תכונות. האובייקטים מהווים חלק מהמודל שאנו יוצרים, מהמציאות של "עולם הבעיה". כאשר כותבים תוכנית בשפה מונחת עצמים, אין אנו יוצרים אובייקט מסוים, אלא, יוצרים מחלקה של אובייקטים. לדוגמא, ייצוג של ספר במחלקה אשר מתארת את כל התכונות של ספר כולל:

- שם הספר
- שם הסופר
- מס' עמודים
- סוג הספר

מחלקה יכולה ליצור ספרים שונים :

- ספרי ילדים
- ספרים עם כריכה קשה
- ספרים מאוירים

כאשר כותבים תוכנית בג'אווה מעצבים ויוצרים קבוצה של מחלקות. כאשר התוכנית בג'אווה "רצה" נוצרים אובייקטים של המחלקות בהתאם למחלקות שהגדרת, והאובייקטים מזמנים שיטות המוגדרות על האובייקטים הללו, זאת כדי, לבצע את המשימה שנועדה לתוכנית.

תכונות והתנהגות של מחלקה

כל מחלקה בג'אווה מורכבת משני מרכיבים : תכונות והתנהגות (attributes , behavior)

תכונות מחלקה

תכונות מחלקה מבדילות בין מחלקות. קובעים את ייצוג המחלקה, לדוגמא, במחלקה ספרים התכונות יהיו:

- תמונת הכריכה על הספר
- צבע הכריכה
- הוצאת הספרים

תכונות המחלקה מייצגים את מצב האובייקט, לדוגמא עבור האובייקט חיה בגן החיות, תכונות יכולות לציין את:

- האם בעל החיים בהריון
- האם בעל החיים חולה
- האם בעל החיים רעב

במחלקה אנו נשתמש במשתנים כדי להגדיר את תכונות המחלקה, כל אובייקט שנוצר על פי תבנית המחלקה יכול לקבל ערכים שונים עבור אותם משתנים.

משתני האובייקט - object variables

משתני האובייקט שומרים את המידע על אובייקט מסוים, המחלקה מגדירה את סוג נתונים שיהיו, ולכל אובייקט יש את המשתנים הללו המגדירות את תכונותיו, משתני האובייקט נקראים גם : instance variable .

לדוגמא, עבור אובייקט ספר מסוים myBook, נקבע את הערך, מס' דפים בספר זה:

myBook.pages = 146;

הסבר: myBook הוא הספר (האובייקט המסוים) ו- pages מייצג את תכונת מס' הדפים בספר, עבור הספר המסוים מס' הדפים הוא 146.

משתנה מחלקה – class variable

משתנה מחלקה הוא פריט מידע שאינו מאפיין אובייקט מסוים אלא את כל המחלקה. לכל האובייקטים של המחלקה יש להם אותו ערך של המחלקה. לדוגמא, עבור המחלקה של בעלי החיים בגן החיות, שם גן החיות יכול להיות תכונה של כל האובייקטים של המחלקה

londonZoo

ערך קבוע למשתנה מחלקה

לעיתים, נחפוץ לתת ערך למשתנה מחלקה אשר יהיה הערך של כל המופעים של המחלקה. אופציה זו ניתנת למימוש באופן הבא:

```
class Animal {  
    String name;  
    int age;  
    static String typeZoo "London" }
```

יצרנו מחלקה של בעלי חיים, עם התכונות: שם החיה, גיל החיה ושם גן החיות, בשם גן החיות הקדמנו את המילה static וקבענו ערך. ערך זה יהיה קבוע ללא שינוי בכל אחד מהמופעים של המחלקה Animal

תכונות מחלקה

1. התכונות המאפיינות את המחלקה באופן כללי ולא רק מופע של המחלקה (אובייקט)
2. בזיכרון יש רק עותק יחיד של התכונה, ללא קשר למספר המופעים שנוצרו מהמחלקה

תכונת המחלקה קיימת לפני שנוצר המופע הראשון של המחלקה. אם מופע מסוים שינה את הערך של התכונה, מאחר ויש רק עותק יחיד של התכונה, הרי שהערך ישתנה עבור כל המופעים של המחלקה. כדי לאפיין תכונה כתכונה של מחלקה, אנו מקדימים את המילה static, מילה משמעותה אינה מתייחסת לפירוש המילה בשפה האנגלית, אלא, פירושה שהתכונה היא תכונת מחלקה.

שיטות במחלקה – Methods

שיטות הן חלק מהמחלקה, שורות קוד המגדירות פעולות שניתן לבצע על אובייקטים או שיטות מחלקה, כדי לבצע משימות דרושות. לדוגמא:

- שיטה לביצוע האכלה של בעל חיים – שיטה של אובייקט
- שיטה לבדיקה האם בעל החיים קבל חיסון- שיטה על אובייקט
- שיטה לבדיקה כמה בעלי חיים יש לנו בגן החיות. – שיטה של מחלקה

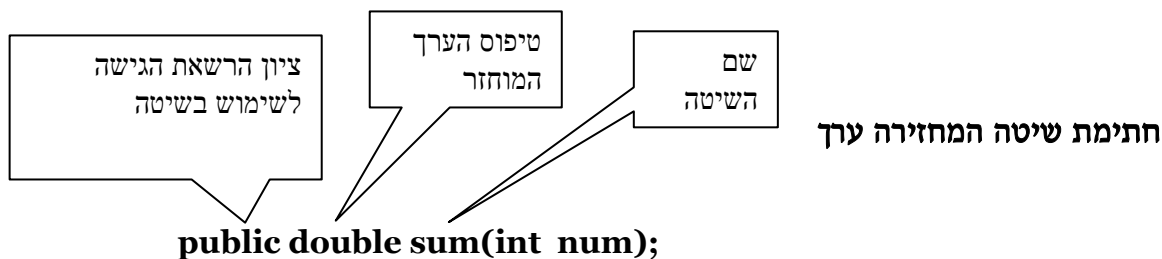
המחלקה או אובייקט של מחלקה, יכולים לזמן שיטה השייכת למחלקה או למחלקה אחרת כדי לבצע משימה לדוגמא:

- להעביר מידע על שינוי לאובייקט אחר (לדוגמא העברת מידע לאובייקט "עובד בגן" שהאובייקט ג'רפה כבר אכל ארוחת צהרים)
- לבקש מאובייקט אחר לבצע משימה (לבקש מהאובייקט "עובד בגן" לצאת לחופשה)

ישנם 2 סוגי שיטות

Instance methods - שיטות מופע

שיטות הפועלות על האובייקטים שיצרנו ונקראים בקיצור methods



שיטות המחזירות ערך מבצעות זאת באמצעות המשפט return, השיטה יכולה לקבל פרמטרים (בדוגמא מופיע פרמטר יחיד) או לא לקבל פרמטר כלל ויופיעו רק סוגריים (חייבים לכתוב את הסוגריים גם אם אין פרמטרים). כותרת השיטה נקראת "חתימת השיטה".

דוגמא

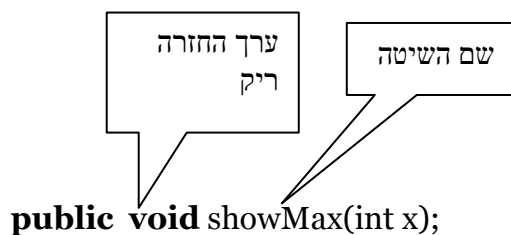
חתימת שיטה ללא פרמטרים

Public int creatRandomNumber()

השיטה מחזירה מספר מסוג int אולם אינה מקבלת פרמטרים לביצוע המשימה (לדוגמא שיטה היוצרת מספר אקראי)

שיטה שאינה מחזירה ערך

השיטה מבצעת חישובים ופעולות שונות אולם, אינה מחזירה ערך לשיטה שזימנה אותה. שיטה זו מקבילה למה שמתכנתים בשפות פרוצדורליות קוראים "פרוצדורה" או "שגרה"



Class methods – שיטות מחלקה

שיטות מחלקה הן פעולות שהמחלקה עצמה מבצעת ולא אובייקט של המחלקה (מופע). שיטות אלו ניתן לזמן גם בלי לבנות מופעים של המחלקה.

יצירת מחלקה

בתום ההסבר על מושגים בסיסיים על מחלקה נוכל ליצור מחלקה של ספרים

```
class Book {  
    }
```

יצרנו מחלקה, המילה class, שם המחלקה ושני סוגריים "מסולסלים" לתיחום המחלקה. אך כפי שציינו, המחלקה מגדירה את תכונות המחלקה, לכן, נרחיב את ההגדרה ונוסיף תכונות

```
class Book {  
    String name;  
    int numPages;  
    double price;  
    }
```

בהגדרת מחלקה בג'אווה אנו מגדירים שתי שיטות נוספות מיוחדות

השיטה main

כל תוכנית ג'אווה כוללת את השיטה הראשית main(...). שממנה מתחיל ביצוע התוכנית. חתימת השיטה main

```
public static void main(String[] args )
```

השיטה הבונה constructor

בשלב הזה נסתפק בהסבר בסיסי מהי השיטה הבונה, כדי ליצור מופעים של המחלקה אשר מקבלים ערכים למשתני האובייקט אנו מגדירים שיטות בונות במחלקה. השיטה הבונה יוצרת מופעים לפי תבנית המחלקה ומאתחלת את תכונות העצם. אם המתכנת לא הגדיר שיטה בונה, ג'אווה "מספקת" שיטה בונה ללא ארגומנטים הנקראת *default constructor*, שיטה בונה זו אינה מבצעת דבר, אינה מבצעת השמה של ערכים אבל נוצרת באופן אוטומטי כי לכל מחלקה חייבת להיות לפחות שיטה בונה אחת.

סיכום – מבנה מחלקה

המחלקה בעלי חיים
תכונות: גיל שם סוג מצב חיסון
שיטות בעלחיים.חסן() בעלחיים.מין?()

השוואה בין ג'אווה לפסקל – שתי תוכניות פשוטות

תוכנית פשוטה בפסקל 1 - התוכנית מדפיסה טקסט

```
PROGRAM PRINT_LINE;  
  BEGIN  
    WRITELN(' HELLO WORLD ');  
  END.
```

מחלקה **PrintLine** - המדפיסה טקסט

```
class PrintLine {  
  public static void main(String []args ) {  
    System.out.println(" hello world "); }  
}
```

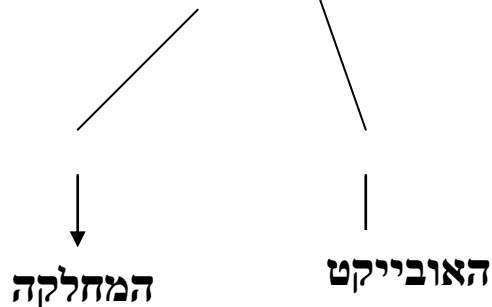
השיטות **println** ו- **print**

אנו נשתמש בשיטות להדפסה בכל הספר וחשוב להסביר מה בעצם משמעות השורות:

System.out.println
System.out.print

ג'אוה מתייחסת לאמצעי הפלט כאובייקט. **out** הוא אמצעי הפלט של התוכנית במידה ואין אנו מציינים זאת אחרת זה **מסך**, המסך הוא אובייקט של המחלקה `System`, ולכן, כאשר אנו רוצים להתייחס לאובייקט של מחלקה נכתוב לפי מה שלמדנו:

System.out



`Print` ו-`println` הן שיטות המוגדרות על האובייקט `out` ולכן תחביר משפט פלט הוא:

System.out.println() או System.out.print()

הסוגריים מציינים שאנו מפעילים שיטה, בין הסוגריים יופיעו הפרמטרים שאנו מעבירים לשיטה. ההבדל בין `print` ו-`println` הוא, שכאשר משתמשים ב-`println` בתום הדפסת הפרמטרים הפלט "עובר לשורה חדשה". אם נשתמש ב-`print` הסמן לא יעבור לשורה חדשה.

דוגמא:

```
System.out.print("Willam ");
System.out.print(" farjun ");
System.out.print(" The writer ");
```

הפלט של ביצוע שתי הפקודות יהיה:

William farjun The writer

דוגמא:

```
System.out.println("Willam ");
System.out.println(" Farjun ");
System.out.println(" The writer ");
```

הפלט יהיה:

William

Farjun

The writer

שאלה

מה משמעות ההוראה: `System.out.println()`;

תשובה:

אנו מזמנים את השיטה `println`, איננו מעבירים פרמטרים להדפסה, ואכן השיטה אינה מציגה פלט על המסך, אבל! , הפלט "עובר" לשורה חדשה, כלומר, המערכת בעצם מדפיסה "מעבר לשורה חדשה"

קבלת קלטים מהמשתמש: המחלקה IO

על מנת לקבל קלטים למשתנים מסוגים שונים עלינו להשתמש במחלקה IO במחלקה זו אנו פונים ל- 4 שיטות שונות לקליטת קלטים בהתאם לסוג – type המשתנה קליטת מספר שלם

```
int num=IO. readInt (" .....");
```

קליטת מספר ממשי מסוג double

```
double avg = IO. readDouble(" .....");
```

קליטת מספר ממשי מסוג float

```
double avg = IO.readFloat (" .....");
```

קליטת תו

```
String name IO.readChar (" .....");
```

שים לב: בסביבות עבודה שונות של Java יש שיטות שונות לביצוע פעולות קלט/פלט. גם בסביבת Jeliot המוצגת בנספחים שיטות הקלט והפלט שונות. חשוב להתאים ולשנות את שיטות IO (קלט/פלט) בהתאם לסביבה בה עובדים או למחלקה בה מחליטים להשתמש. פירוט כל השיטות של המחלקה IO מופיעים בנספחים. המחלקה פותחה ע"י פרופ' מוטי בן ארי – מכון ויצמן

קבלת קלטים מהמשתמש: המחלקה Scanner

המחלקה Scanner מגדירה פעולות קלט שונות, המחלקה כלולה במארז java.util ולכן אם רוצים להשתמש בפעולות המחלקה יש להכניס הצהרה לפני כותרת המחלקה:

```
import java.util.Scanner ;
```

בשונה מהמחלקה IO השימוש במחלקה Scanner "נאמן" לרעיון תכנות מונחה עצמים ולכן עלינו להגדיר עצם של המחלקה ורק אז נוכל להפעיל את אחת מפעולות הקלט המוגדרות על עצם זה. יצירת עצם מסוג Scanner נעשית כך:

```
Scanner in = new Scanner(System.in);
```

הסבר:

המילה Scanner בתחילת המשפט מעידה שאנו מגדירים עצם ממחלקה זו המילה השמורה new יוצרת עצם בדומה לכל יצירת עצם בג'אווה הצירוף System.in משמעותו שהקלט מתבצע מלוח המקשים

קליטת ערך שלם

```
int num = in.nextInt();
```

קליטת ערך ממשי מסוג double

```
int num = in.nextDouble();
```

קליטת ערך ממשי מסוג float

```
in.nextFloat();
```

קליטת תו

```
int num = in.nextInt().charAt(0);
```

בשלב הזה אציג שיטה נוספת ללא הסבר ונחוצה מאוד לכתיבת מחלקות עם מבנים מורכבים:

השיטה hasNext()

השיטה הזו בודקת האם יש עוד קלט, בהמשך אציג שימוש בשיטה ואסביר.

בדוגמאות והתרגילים בספר נשתמש בעיקר במחלקה Scanner זאת בהתאם להנחיות ועדת המקצוע, אולם גם נשתמש במחלקה IO זאת מפאת העובדה שימוש במחלקה זו נפוץ בכמה ספרים וראוי שנלמד ליישם פתרונות ושימוש בקלט גם באמצעות שיטות מחלקה זו.

שם משתנה בג'אווה - כללים

- התו הראשון בשם של משתנה חייב להיות אות "קטנה" (lowercase)
- אם שם המשתנה מורכב מכמה מילים, כל מילה חדשה תתחיל באות גדולה (capital letter)
- כל שאר האותיות בשם המשתנה הן אותיות קטנות או גדולות, וניתן גם להשתמש בספרות ו- הסימנים \$ _
- שם מחלקה מתחיל באות גדולה, רצוי שכל חלק מהשם המהווה מילה בעלת משמעות יתחיל באות גדולה.

דוגמאות לשמות משתנים חוקיים:

sumOfChildren numberOfclass trainNum

דוגמאות לשמות לא חוקיים:

4number sum*5gr

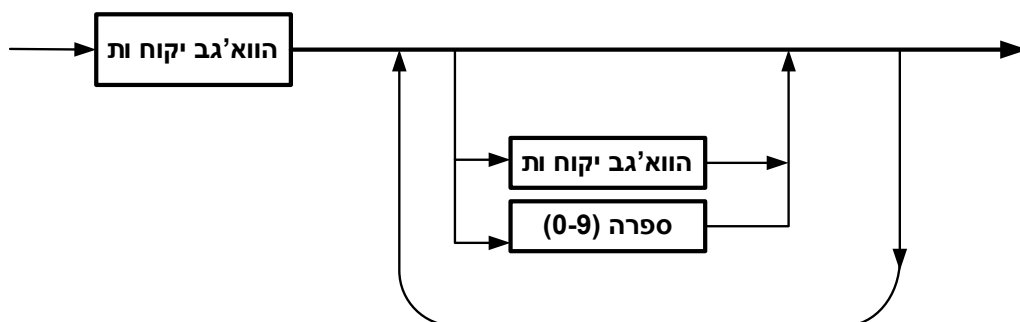
חשוב

ג'אווה "רגישה" לאות גדולה או אות קטנה, המשתנה num שונה מהמשתנה Num . שני המשתנים הללו בתוכנית אחת יהיו שונים.
כל הכרזה של משתנה חייבת להגדיר את סוג המשתנה, סוג המשתנה. יהיה אחד מהסוגים הבאים:

- אחד מסוגי המשתנים הבסיסיים (משתנה מספרי או לוגי או תו) .
- שם של מחלקה או ממשק (לדוגמא class Point) .
- מערך .

ישנם משתנים עם ייצוג שונה המגדיר את אופן האחסון של סוג המשתנה ואת גודל הזיכרון המוקצה למשתנה, בהמשך הספר נרחיב ונסביר את הסוגים השונים.

מבנה תחבירי של שם משתנה



הערות בגוף התוכנית

תוכנית מחשב נכתבת ע"י אדם או יותר, לעיתים המתכנת שכתב את התוכנית אינו זוכר בדיוק מה הוא כתב לאחר מספר ימים, לעיתים תוכניתן אחר ממשיך את עבודתו. כל שפות התכנות מאפשרות להוסיף הערות בתוך התוכנית:

הערה בשורה אחת ע"י הסימנים //

דוגמא

// Set the sum to zero

ניתן לכתוב קטע שלם של הערות זאת ע"י הסימן /* שיפתח את הקטע וסגירת קטע ההערות
*/

דוגמא:

/* This program draw a line

and compute the sum of the numbers */

פעולות מתמטיות בסיסיות

דוגמא	משמעות	הפעולה
5 + 3	חיבור	+
2 - 6	חיסור	-
9 * 4	כפל	*
2 / 5	חילוק	/
30 % 15	שארית חלוקה של שני מספרים שלמים	%

סוגי משתנים בסיסיים בג'אווה

לג'אווה יש קבוצה של סוגי משתנים בסיסיים – **primitive type**, משתנים אלו משמשים אותנו לתכנות שוטף, חישוב ערכים ואחסוןם באופן זמני, את המופעים של משתנים אין צורך ליצור ע"י פקודת new, בדומה לאובייקטים, אלא להגדירם כמו בשפות תכנות רבות. טיפוסים הנתונים הבסיסיים נבדלים בסוג הערכים שהם מאחסנים, גודל הזיכרון שנדרש לאחסונם וטווח הערכים.

דוגמאות:

double sum, number;

int num;

int tax = 15;

char op ;

char let = 'p';

char sifar = '6';

Boolean ok;

הגדרת קבוע בג'אווה

בתוכנת הנהלת חשבונות מחשבים מע"מ, המס המתווסף לכל רכישה, מס זה קבוע וערכו 16.5%. ערך זה מופיע בודאי בחלקים רבים של תוכנת הנהלת חשבונות ואסור לשנותו. למצב זה ולמצבים רבים אחרים בהם אנו מעוניינים שערך של משתנה יהיה קבוע קיימת בג'אווה ההכרזה **final**

public final double msMam = 16.5;

טבלת טיפוסים נתונים בסיסיים

מקסימום	ערך מינימום	גודל זכרון נדרש	סוג הנתונים
		8 סיביות	Boolean
Unicode $2^{16}-1$	Unicode 0	16 סיביות	char
+127	-128	8 סיביות	byte
$+2^{15}+1$	-2^{15}	16 סיביות	Short
$+2^{31}-1$	-2^{31}	32 סיביות	Int
$+2^{63}-1$	-2^{63}	64 סיביות	Long
		32 סיביות	float
		64 סיביות	double

הגדרה של טיפוסים נתונים בסיסיים - primitive types

סוגי טיפוסים מסוג שלם:

```
byte x;
short num;
int sum = 100;
long totalTax;
```

טיפוסים מסוג ממשי:

```
float avg;
double fxValue;
```

טיפוס תו:

```
char tempChar = '*';
```

טיפוס לוגי:

```
boolean aBoolean = false;
```

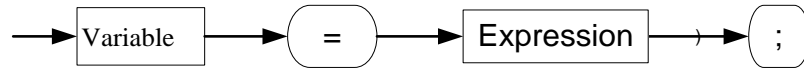
שמות המשתנים - סיכום

תוכנית בג'אווה מתייחסת לערכי המשתנים באמצעות שם המשתנה, לדוגמא, כאשר התוכנית מתייחסת ל- $sum = sum + num$, התוכנית מתייחסת לשני ערכים באמצעות שם המשתנה המכיל ערכים אלו sum ו- num . ההתייחסות למשתנה בג'אווה היא באמצעות "שם פשוט" – simple name וזה בניגוד להתייחסות יותר מורכבת של ג'אווה לאובייקטים ותכונות של אובייקטים. שם פשוט בג'אווה חייב לקיים את הכללים הבאים:

- לא ניתן לתת למשתנה שם שהוא מילת מפתח בשפה, לדוגמא לא ניתן לקרוא למשתנה בשם `for`, `while`, `int` וכו'..
- חייב להתחיל באות.
- השם יכול רק תווים חוקיים, ההמלצה להשתמש באותיות ובמספרים ושילוב של תווים אלו (והסימנים `_` `$`).
- לא ניתן לתת אותו שם לשני משתנים באותו תחום של הכרה (scope)

משפט השמה בג'אווה

מבנה תחבירי של משפט השמה



במשפט השמה יש שני צדדים. בצד הימני של סימן השוויון יש ביטוי בג'אווה, בצד השמאלי יש משתנה שהוכרז. ביצוע משפט ההשמה "אומר" למחשב לחשב את ערך הביטוי המופיע בצד ימין ואת תוצאת החישוב להכניס למשתנה המופיע בצד שמאל.

דוגמא:

num = 4 * x;

הסבר:

הביטוי הימני $4 * x$ מחושב, המשתנה num "מקבל" את התוצאה וזה ערכו בסיום ביצוע ההוראה. סוג ערך הביטוי המחושב חייב להיות סוג של המשתנה בו הוא מושם. שפת ג'אווה אינה "מגלה פשרות" ואינה מאפשרת (כמעט תמיד) השמת ערך מסוג מסוים במשתנה מסוג אחר.

דוגמא:

double x;

int y;

y = x + 5;

פעולת ההשמה אינה חוקית. הוראת ההשמה מבצעת השמת ערך ממשי במשתנה מסוג שלם. השגיאה תתגלה עוד בשלב תרגום התוכנית ותתקבל הודעה מהמפרש, אולם, ישנם מצבים בהם המהדר של ג'אווה מאפשר השמות של סוגים שונים, ההשמה מתבצעת עם תהליך של "שינוי" בסוג המשתנה.

דוגמא:

נתונות ההגדרות הבאות:

int num = 7;

double x;

המשפט הבא הוא חוקי

x = num;

הערך 7 השלם מושם לתוך המשתנה x הממשי, ובתוך התהליך משתנה הערך 7 מייצוג של מספר שלם לערך 7 בייצוג של מספר ממשי שהיה 7.0, שינוי סוג ערך בתהליך של ביצוע משפט השמה נקרא : assignment conversion, שינוי זה יכול להתרחש רק כאשר ההשמה מתבצעת מסוג משתנה שתחום ערכיו מצומצם לסוג משתנה שתחום ערכיו גדול יותר ומכיל את הסוג המצומצם, כמו, השמת ערך שלם לתוך ערך ממשי (במקרה זה ההיפך לא ניתן לביצוע)

ישנם בג'אוה צורות רבות של שינוי ערכי משתנים, פקודת השמה מורכבת כמו הפקודה הבאה:
`x=y=num = 4;`

העיקרון של זהות בסוג המשתנה וסוג הערך נשמר

המרה בין טיפוסים משתנים באמצעות – Casting

תרגום ביטוי מערך ממשי לטיפוס שלם באמצעות (int)

`int price = (int)(2.8);`

ביצוע casting בג'אוה מתבצע באמצעות אופרטור אשר מצוין בתוך הסוגריים ובדוגמא הזו הוא: int.

דוגמא:

`double avg;`

`int sum, n,temp;`

קבלנו משימה לחשב ממוצע של n מספרים שלמים. הסכום של המספרים חושב והושם בתוך משתנה מסוג שלם sum.
 ביצוע המשימה בדרך א:

`temp = sum / n ;`

התוצאה שתתקבל במשתנה temp, תהיה ערך שלם, החישוב לא ייתן את הערך המדויק אלא את המספר השלם הקרוב ביותר, כלומר, המנה השלמה. (עבור סכום מספרים 17 ומספר המספרים 4 תתקבל תוצאת הממוצע 4)
 כיצד נקבל את הערך המדויק של הממוצע?, נוכל לבצע זאת באמצעות casting. באופן הבא:

`avg = (double) sum / n;`

שים לב!

לאופרטור השינוי שאנו מפעילים יש עדיפות בביצוע על פני פעולת החילוק לכן סדר הביצוע הוא :

מתבצע שינוי הטיפוס של הערך הנמצא במשתנה sum (הערך בתוך sum לא משתנה)
ערך זה עתה בייצוג double מחולק ב-n והתוצאה מושמת בתוך המשתנה avg

מחלקה Basicop - ביצוע פעולות אריתמטיות פשוטות

```
class Basicop {
    public static void main(String args []) {
        int x = 5;
        int num1=24,num2=6;
        System.out.println(" The sum of the 3 numbers is " + (num1 + num2 + x));
        System.out.println(" the mul of the three numbers is " + num1*num2*x );
        System.out.println(" The division between num1 and num2 is " +
num1/num2);
    }
}
```

כתיבת ביטויים מתמטיים בג'אווה

ג'אווה מאפשרת תחביר גמיש בכתיבת ביטויים מתמטיים ומשפטי השמה

דוגמא

x=y=n=5;

הסבר: שלושת המשתנים מקבלים את הערך 5

בכל משפט השמה החלק הימני של המשפט מחושב וערכו מושם במשתנה המופיע בצד שמאל.
בשפת פסקל ואחרות תחביר פקודת ההשמה "קשיח" ואינו מאפשר צורות שונות, ג'אווה מאפשרת משפטי השמה שונים

טבלת משפטי השמה

משמעות	ביטוי
$x = x + y$	$x += y$
$x = x - y$	$x -= y$
$x = x * y$	$x *= y$
$x = x / y$	$x /= y$
$x = x \% y$	$x \% = y$

בכתיבת תוכניות תוכל לכתוב את הביטויים הן בצורה במורחבת והן בצורה המקוצרת
ג'אוה מאפשרת פעולת הוספת או הורדת 1 ממשתנה ע"י כתיבת הביטויים הבאים:

$y = x++$;

הסבר: הוסף 1 ל x והשם את הערך ב- y

$y = x--$;

הסבר: הורד 1 מ- x והשם את הערך ב- y

ההבדל בין $x++$ ל $++x$

אם מופיע הביטוי $y = x++$ ערכו של y מקבל את הערך x ולאחר מכן ערכו של x גדל ב-1

אם מופיע הביטוי $y = ++x$ הרי שבתחילה ערכו של x גדל ב- 1 ולאחר מכן הערך מושם ב- y

טבלת אופרטורים

אופרטור	הסבר	דוגמאות
[]	משמש להגדרת מערך	int [] mark; int [] data = {1,2,3,4,5};
.	נקודה מהווה אופרטור מרכזי בג'אוה ומאפשרת גישה לתכונות ושיטות של אובייקטים	card.num=8; num.bigInteger(x,y);
(סוג)	תרגום – שינוי טיפוס ערכים - casting	(int)(Math.sin(num))
new	אופרטור המשמש ליצירה של אובייקטים ומערכים	chair stool = new chair();

תחום ההכרה (מרחב "המחיה" של משתנים) - Scope

תחום ההכרה של משתנה הוא האזור בתוכנית בו ניתן לגשת אליו. אזור ההכרה נקבע ויכול להשתנות בשתי דרכים:

(1) אופן הגדרת הגישה למשתנה (public , private וכו..).

(2) מיקום הגדרת המשתנה קובע את אזור "המחיה" של המשתנה. ישנן 4 אפשרויות לאזורי הכרה של משתנים.

אזור ההגדרה של משתנים- חברים של מחלקה שהוגדרה

Class myClass{

{ הגדרת משתנים (חברים) של המחלקה }

אזור הכרה של הפרמטרים בין הסוגריים של חתימת השיטה.

Public void myFirstMethod(method parameters)

המשתנים יוכרו בכל תחום השיטה

אזור משתנים מקומיים של השיטה

```
public int sum( int x){
// local variable declaration
    int sum; { המשתנים מוכרים בבלוק המגדיר אזור הביצוע של השיטה
code of the method ....} .
```

אזור ההכרה של משתנים בתוך Block

```
public int sum( int x){
// local variable declaration
    int sum;
// code of the method {
    if (sum > 0){
        המשתנים מוכרים בתחום הבלוק המסוים בו הם הוגדרו;
        int x;
        x=4; }
    }
```

בתוך בלוק בשיטה ניתן לפי הצורך להגדיר משתנים חדשים משתנים אלו מוכרים רק בתחום הבלוק.

כל הגדרה של משתנה בשיטה חייבת להיות מוגדרת לפני השימוש במשתנה. שפת ג'אווה מאפשרת הכרזה על משתנים בכל מקום בתוכנית, אולם, ראוי לזכור, לא ניתן להשתמש במשתנים אלא רק אחרי שהוגדרו.

אילו הוגדרו משתנים מקומיים בתוך בלוק של קוד (block) . הם יוכרו עד לסיום שורות הקוד של אותו הבלוק. פרמטרים המועברים לשיטות (רגילות ובונות) המשמשים להעברת ערכים אל ומאת השיטה. מרחב ההכרה של פרמטרים אלו הוא בכל השיטה בה הם הוגדרו בחתימת השיטה. הקוד הבא מתאר משתנה שמוגדר בתוך Block, והפניה למשתנה מחוץ לבלוק תיצור שגיאה כי המשתנה אינו מוכר מחוץ ל-Block

```
if (...) {
    int i = 17;
}

System.out.println("The value of i = " + i); //error
```

תחום ההכרה של משתנים:

```

Class MyClass {
הכרזה על תכונות המחלקה
.....
.....
Member variable declarations
.....
.....
Public void aMethod(method parameters){
תחום ההכרה של הפרמטרים שהועברו לשיטה
.....
.....
Local variable declarations
.....
תחום ההכרה של משתנים המקומיים שהוכרזו בשיטה
.....
.....
for(int x=0;.....){
תחום ההכרה של משתנים שהוגדרו בתוך ה-
    block
    .....
    .....
}
}
.....
}
    
```



פרק שלישי – מתחילים לתכנת בג'אווה

בפרק זה אנו מתחילים לכתוב "ברצינות" מחלקות בג'אווה, הפרק כולל הרחבה של אופרטורים (פעולות) יצירת מופעים של מחלקות באמצעות הפקודה new. הפרק מתאר את הדרך לקלוט קלט לתוכנית ולהציג פלט, וכן שימוש בפעולות מתמטיות שונות כולל חישוב שורש ופונקציות (שיטות) נוספות.



פרק שלישי – מתחילים לתכנת בג'אווה

מבנה הגדרת מחלקה class בג'אווה

```
class NameOfTheClass {  
    // class body  
}
```

דוגמא: מחלקת תלמידים

```
class Talmid {  
    char name;  
    int age;  
    boolean pay;  
}
```

דוגמא : מחלקת כלי תחבורה

```
class Trans {  
    private int yearOfProduction;  
    private int numOfWheel;  
    char color ;  
}
```

דוגמא: מחלקת כסאות

```
class Chair {  
    private int legs;  
    private char color;  
}
```

מבנה מפורט של מחלקה

```
public class nameOfClass {    // הכרזה על המחלקה  
    private int age;          // תכונות של המחלקה  
    public nameOfClass(int x) { // השיטה הבונה של המחלקה  
        this.age=x;  
    }  
    public int addOne(int x){  // פירוט שיטות של המחלקה  
        x++;  
    }  
}
```


הכרזה על אובייקטים:

```
talmid T1;  
trans ford;  
chair stool;
```

בניגוד להגדרת משתנים בסיסיים (boolean, int וכיו"ב) המקצה זיכרון למשתנים ומאפשרת גישה אליהן, מאותו הרגע, ההכרזה על אובייקט איננה מקצה זיכרון אלא רק מגדירה למפרש שם שיכיל בהמשך את הפנייה לעצם מסוג מסויים ללא יצירת העצם. כדי לאפשר גישה לעצם (חברים, שיטות וכו') עלינו להקצות זיכרון לאותו האובייקט ולעדכן את ההפניה לעצם שנוצר והמקום שקיבל בזכרון – פעולה המתבצעת ע"י האופרטור **new**.

מה מבצע האופרטור new?

כאשר משתמשים באופרטור new מתרחשים מספר דברים, מופע חדש של המחלקה שמופיעה במשפט נוצר, זיכרון מוקצה למופע של המחלקה לפי גודל הזיכרון הנדרש, והשם שנתנו לאובייקט מכיל את ההפניה לאובייקט.

יצירה של מופע מחלקה :

```
chair stool = new chair();
```

אל תוותר על כתיבת הסוגריים, הסוגריים מציינים שמדובר בשיטה (השיטה הבונה).

הקצאת ערכים לאובייקטים.

```
stool.legs = 1;  
stool.color = 'g';
```

האובייקט stool של המחלקה chair קבל ערכים:

מס' רגליים = 1

צבע = g

מספר הרגליים וצבע הם תכונות המחלקה ונקראים גם member כלומר נתונים חברים במחלקה. הקישור ביו אובייקט לבין נתונים, החברים במחלקה של האובייקט, הוא באמצעות סימון נקודה או "חיבור באמצעות נקודה" או "יחוס נקודה"

סיכום - מבוא לאובייקטים

שרפרף בר – אובייקט של המחלקה

chair

תכונות צבע- g מס רגליים - 1

מהו "אובייקט" וממה הוא מורכב?

אובייקט בג'אוה ובכל שפה מונחית עצמים מורכב:

תכונות – attributes

שיטות – methods

התכונות והשיטות נקראים "חברים" של האובייקט – members

שיטות:

- מתארות התנהגות של העצם.
- יכולות לשנות את התכונות של העצם.
- יכולות לאחזר את התכונות של העצם.
- יכולות לתקשר בין עצמים.

תכונות ושיטות מתארות לנו מהו האובייקט ומה האובייקט "יודע לעשות".

שיטות עבור האובייקט כסא

1. אחזר_צבע()
2. אחזר_מס_רגליים()

שיטות עבור האובייקט תלמיד

1. שלם()
2. אחזר_שם()
3. הגדל_גיל()

האובייקטים הם מופעים של המחלקה

דוגמאות למחלקות ומופעים

מחלקה: יונקים

מופעים: פיל, אריה, עכבר וכו..

מחלקה: חשבון בנק

מופעים: חשבון הלוואה, חשבון חסכון, חשבון קופת גמל, חשבון עו"ש.

מחלקה: עובד

מופעים: עובד שיווק, עובד אחזקה, עובד מינהלה

מחלקה: כוכבי לכת

מופעים: מאדים, צדק, נוגה, ארץ

סיכום

בגישת תכנות מונחה עצמים המתכנת "בונה עולמות", כל תוכנית מחשב מטרתה ליצור מציאות הקרובה ככל האפשר לעולם האמיתי, עולם המחלקות והעצמים משחרר את המתכנת ממגבלות הטיפוסיים הבסיסיים ומאפשרים להגדיר טיפוסי נתונים חדשים, טיפוסי נתונים אשר ייצגו את עולם הבעיה שלו באמצעות מושגים הלקוחים מעולם הבעיה, יצירת הטיפוסיים – מחלקות מהווה את השלב הראשון בבניית המודל. השלב השני הגדרת תכונות האובייקטים של המחלקה והשיטות המאפשרות קבלת מידע מהאובייקט, שינוי מצב האובייקט ויוצרות קשר בין אובייקטים. בהמשך ארחיב על אובייקטים, בניית אובייקטים מורכבים, הורשה של תכונות בין מחלקות.

אופרטורים, משפטי השמה, קלט פלט של תוכניות

מחלקה ג'אוה MoreBasicop – מחשבון פשוט

```
class MoreBasicop {
    public static void main(String args[]) {
        int x = 12;
        int num1=34,num2=32;
        System.out.println(" The sum of the numbers is " + (num1 + num2 + x));
        System.out.println(" The division num1 and num2 is " + num1/num2);
        System.out.println(" num1 modulo x is " + num1 % x);
        System.out.println(" The multiplication of the three numbers is " +
num1*num2*x);
    } // end of main
} // end of MoreBasicop class
```

המחלקה Basicop מיישמת חלק מהמאפיינים שהצגתי כולל:

- למחלקה יש את השיטה main
- הגדרת משתנים בתחביר הכולל הכרזה על המשתנה והשמת ערך באותה פקודה
- שימוש באופרטורים מתמטיים +, /, *, %
- רעיון חדש המוצג בתוכנית הוא שימוש בשרשור מרכיבי הפלט שברצוננו להדפיס זאת על ידי הפעולה +, במקרה זה ה + אינו משמש כאופרטור מתמטי אלא כמשרשר של שני ארגומנטים, ביטוי מתמטי וקבוע מחרוזת (עוד על האופרטור + ואופן פעולתו על מחרוזות וערכים בהמשך)

קדימות של אופרטורים

אילו קבלת מהמורה למתמטיקה את התרגיל הבא: $4+3*5$

מהי תוצאת התרגיל?

האם התוצאה היא 35 או 19?

התשובה היא 19, כי לכפל יש קדימות בסדר הפעולות, ורק לאחר מכן מבצעים חיבור. גם בג'אוה יש כללי קדימות לאופרטורים.

- פעולות הוספה והחסרה של 1 (Increment and Decrement)

- כפל וחילוק, %

- חיסור וחיבור

כאשר אנו מכניסים ביטוי לסוגריים אנו נותנים לו את העדיפות הגבוהה ביותר.

מחלקה Math - סכום וממוצע

```
import java.util.Scanner;
class Math {
    public static void main (String args [] ) {
        int num1,num2,num3;
        double x;
        Scanner input = new Scanner(System.in);
        System.out.println("Please enter 3 numbers:");
        num1=input.nextInt();
        num2= input.nextInt();
        num3= input.nextInt();
        System.out.println(" The sum is " + (num1 + num2 + num3) );
        x = (num1 + num2 + num3)/3 ;
        System.out.println( "The average for the 3 numbers is: " + x);
    } // end of main
} // end of Math class
```

הגדרתי בשיטה main שלושה משתנים מסוג שלם, משתנה מסוג ממשי.
בתוכנית פקודות לקליטת 3 מספרים שלמים ע"י שימוש בשיטה של האובייקט input
השייכת לספריית המחלקה Scanner.
num1=input.nextInt();

בסוף השיטה main מודפס הסכום של 3 מספרים והממוצע של 3 מספרים.

מחלקה Gader

חקלאי נדרש לגדר את חלקתו, המחלקה Gader כוללת את השיטה main הקולטת את אורך
ורוחב החלקה, ומחיר מטר גדר ומחשבת ומדפיסה את עלות הקמת הגדר.
במחלקה Gader אני מציג את השימוש במשתנה מסוג float, משתנה זה הוא "ממשפחת"
המשתנים המממשים וטווח הערכים שלו יותר מצומצם מ-double על ההבדלים בין סוגי
המשתנים ראה בטבלה המופיעה בפרק שני.
לקליטת מספרים מסוג float השתמשתי בשיטת הקלט:

```
width= input.nextFloat();
length= input.nextFloat();
price= input.nextFloat();
```

```
import java.util.Scanner;
class Gader {
    public static void main (String args [] ){
        Scanner input = new Scanner(System.in);
        float width,length,area,price;
        System.out.println("Please enter Width and Length :");
        width= input.nextFloat();
        length= input.nextFloat();
        System.out.println("Please enter Width and price :");
        System.out.println();
        price= input.nextFloat();
        area = width * length;
        System.out.println(" The area is " + area );
        System.out.println(" The payment is " + area * price + " shekel ");
    } // end of main
} // end of Gader class
```

מחלקה IncDec - מוסיפים ומחסירים 1.

```
class IncDec {
    public static void main( String args[]) {
        int x,y;
        x=5;
        System.out.println(" x= " + x);
        x++;
        System.out.println(" x= " + x);
        y=x++;
        System.out.println(" x= " + x + " y= " + y);
        y= ++x;
        System.out.println(" x= " + x);
        x--;
        System.out.println(" x= " + x);
        y=x--;
        System.out.println(" x= " + x + " y= " + y);
        y= --x;
        System.out.println(" x= " + x);
    } // end of main
} // end of class
```

הסבר:

התוכנית ממחישה את פעולה ++ ו --, את ההבדל בין הופעת הסימנים מימין למשתנה למצב בו הם מופיעים לשמאל למשתנה. הפלט של התוכנית הוא:

```
x=5
y=6
x=7 y=6
x=8
x=7
y=6 y=7
x=5
```

הרץ את התוכנית ובדוק את הפלט, שנה את מיקום הסימנים ++ ו --

מחלקה Monit - כמה מוניות?

קבוצת אנשים ממתינה למוניות, מס' הנוסעים במונית הוא 5, כתוב שיטה המקבלת את מספר האנשים שממתין למוניות ומחשבת ומציגה כמה מוניות ידרשו לקבוצת הנוסעים הממתינה.

פתרון

התוכנית הבאה, קולטת את מס' האנשים שממתין למונית, ומחשבת תוך שימוש באופרטור השארית - % את מס' המוניות הנדרש. התוכנית מדפיסה את מס' המוניות הנדרש.

```
import java.util.Scanner;
class Monit {
    public static void main( String args[]){
        Scanner input = new Scanner (System.in);
        int men;
        System.out.println("Enter num of men");
        men= input.nextInt();
        System.out.println("You need " + (men - men % 5) / 5 + " moniut" );
    } // end of main
} // end of class Monit
```

שים לב!

בתוכנית אנו מפעילים את השיטה: nextInt() לקריאת מס' שלם, שהיא חלק מחבילת Scanner. השיטות לביצוע פעולות קלט של המחלקה Scanner.

המחלקה ArithmeticDemo מציגה סיכום פעולות

```
class ArithmeticDemo {
    public static void main(String[] args) {
        //a few numbers
        int i = 12;    int j = 3;
        double x = 5.4;    double y = 3.25;
        System.out.println("The value of variables");
        System.out.println("  i = " + i);
        System.out.println("  j = " + j);
        System.out.println("  x = " + x);
        System.out.println("  y = " + y);
        // חיבור המספרים
        System.out.println("Adding...");
        System.out.println("  i + j = " + (i + j));
        System.out.println("  x + y = " + (x + y));
        // החסרת המספרים
        System.out.println("Subtracting...");
        System.out.println("  i - j = " + (i - j));
        System.out.println("  x - y = " + (x - y));
        // הכפלת הערכים
        System.out.println("Multiplying...");
        System.out.println("  i * j = " + (i * j));
        System.out.println("  x * y = " + (x * y));
        // חלוקת הערכים
        System.out.println("Dividing...");
        System.out.println("  i / j = " + (i / j));
        System.out.println("  x / y = " + (x / y));
        // חישוב תוצאת השארית מחלוקת שני מספרים
        System.out.println("Computing the remainder...");
        System.out.println("  i % j = " + (i % j));
        System.out.println("  x % y = " + (x % y));
    }
}
```

לפני הרצת התוכנית, רשום מה הפלט של התוכנית, הרץ והשווה. מאוד מומלץ להריץ את התוכנית בסביבת Jeliot שממחישה בצורה ויזואלית את השינוי בערכי המשתנים.

המחלקה ArithmeticDemo באמצעות השיטה main מציגה סיכום של שימוש באופרטורים מתמטיים כולל: חיבור, חיסור, כפל, חילוק ותוצאת השארית מחלוקה של שני מספרים שלמים.

פעולת חיבור

```
System.out.println(" i + j = " + (i + j));
```

פעולת חיסור

```
System.out.println(" i - j = " + (i - j));
```

פעולת כפל

```
System.out.println(" x * y = " + (x * y));
```

פעולת חילוק

```
System.out.println(" i / j = " + (i / j));
```

פעולת שארית

```
System.out.println(" i % j = " + (i % j));
```

שאלה

כאשר יש ביטוי חשבוני הכולל טיפוסים מסוגים שונים מהו טיפוס התוצאה?

תשובה

- אם הביטוי מכיל ערך אחד ממשי – תוצאת הביטוי ממשי
- אם כל הערכים הם שלמים – תוצאת הביטוי שלם

מחלקה **SqFunction** – חישוב פתרונות למשוואה ריבועית

המחלקה **SqFunction** משתמשת בשיטה `sqrt` לחישוב שורש ריבועי. `sqrt` הינה שיטה במחלקה `Math` המרכזת מגוון רב של שיטות מתמטיות. `Math` כוללת גם השיטה `random` בה השתמשתי בדוגמאות קודמות ליצירת מספרים אקראיים, כל השיטות במחלקה `Math` הן סטטיות.

ההוראה לחישוב הדסקרימיננטה $b^2 - 4 * a * c$:

```
temp = Math.sqrt(b*b - 4*a*c); //חישוב הדסקרימיננטה
```

חישוב והדפסת שורשי המשוואה:

```
System.out.println("The first root is " + (-b + temp)/(2*a));
```

```
System.out.println("The second root is " + (-b - temp)/(2*a));
```

המחלקה **SqFunction**

```
import java.util.Scanner;

class SqFunction {
    public static void main( String args[]) {
        double a,b,c,temp;
        Scanner input = new Scanner(System.in);
        a=input.nextDouble();
        b= input.nextDouble();
        c= input.nextDouble();
        temp = Math.sqrt(b*b - 4*a*c);
        System.out.println( " the first root is " + (-b + temp)/(2*a));
        System.out.println( " the second root is " + (-b - temp)/(2*a));
    } // end of main
} // end of SqFunction
```

מחלקה **sidra**

במחלקה **sidra** הגדרתי 5 שיטות סטטיות, שיטות אלו שהן שיטות של המחלקה ולא פועלות על אובייקטים. זו לא דרך התכנות המומלצת בג'אווה כי במידה רבה זה תכנות פרוצדורלי בסביבת תכנות שהיא בבסיסה סביבה מונחת עצמים, הדוגמא נועדה להציג כיצד אנו קוראים לשיטות, ומעבירים פרמטרים ואת האופן בו השיטה מחזירה ערך (אם לא מוגדרת כ- void). המחלקה **sidra** כוללת 5 שיטות.

המחלקה **sidra** באמצעות השיטות הכלולות בה מבצעת חישובי איברי סדרות. כל שיטה מקבלת את נתוני הסדרה ומחשבת את האיבר הבא בסדרה

המחלקה sidra היא מחלקת שירות בדומה למחלקה Math. כל השיטות של sidra הן סטטיות.

השיטה heshbonit

מקבלת כארגומנטים את האיבר הנוכחי בסדרה, את הפרש הסדרה, ומחזירה את האיבר הבא בסדרה

השיטה handasit

מקבלת כארגומנטים את האיבר הנוכחי בסדרה ואת מכפיל הסדרה (בסדרה הנדסית הוא נקרא q), ומחזירה את האיבר הבא.

השיטה binarit

מקבלת כארגומנט את האיבר הנוכחי בסדרה ומחשבת את האיבר הבא, שהוא האיבר הקודם מוכפל ב- 2

השיטה fibonacci

מקבלת כארגומנט את שני איברי הסדרה ומחשבת את האיבר הבא שהוא סכום שני קודמיו

השיטה main

קולטת מהקלט את 3 הערכים $a1, a2, dif$ ומפעילה את 4 השיטות.

שים לב!

4 השיטות מלבד השיטה main מחזירות ערך ולכן אנו משתמשים בהוראה **return** המחזירה ערך לשיטה שקראה (במקרה שלנו הערך מוחזר לשיטה main שבצעה את הזימון)

המחלקה Sidra

```

import java.util.Scanner;
class Sidra {
    public static double heshbonit(double a1, double dif){
        return a1+dif; }
    public static double handasit(double a1, double dif){
        return a1*dif; }
    public static double binarit(double a1){
        return a1*2; }
    public static double fibonacci(double a1,double a2){
        return a1+a2;}
    public static void main(String args[]){ // starting main
        Scanner input = new Scanner (System.in);
        double a1,a2,dif;
        a1= input.nextDouble();
        a2= input.nextDouble();
        dif= input.nextDouble();
        System.out.println("the next number in sidra heshbonit is " +
heshbonit(a1,dif));
        System.out.println("the next number in sidra handasit is " +
handasit(a1,dif));
        System.out.println("the next number in sidra binarit is " + binarit(a1));
        System.out.println("the next number in sidra fibonacci is " +
fibonacci(a1,dif));
    } // end of main
} // end of sidra class

```

חלוקה שלמה - תזכורת

ישנן שפות (כדוגמת פסקל) בהן מוגדר אופרטור נפרד המבצע חלוקה בין מספרים שלמים. בג'אווה אין צורך בשימוש באופרטור מיוחד, שכן כאשר משימים תוצאת חלוקה של מספרים שלמים למשתנה מסוג שלם **int** הערך שיחושב באופן אוטומטי הוא מנת המספרים השלמה ללא שארית. דוגמא:

```

int x, y, a;
x= 17; y=3; a= x/y;

```

המשתנה **a** מכיל את הערך 5, שהוא תוצאת החלוקה השלמה בין 17 ל-3.

מחלקה DigitNum –מפרידה מספר לספרותיו

המחלקה DigitNum באמצעות השיטה main קולטת מספר תלת ספרתי, "מפרקת" את המספר לספרותיו, ומדפיסה את ספרותיו וסכום ספרותיו.

```
import java.util.Scanner;

class DigitNum {
    public static void main (String[] args) {
        Scanner input = new Scanner(System.in);
        int num;
        int n100,n10,n1;
        num = input.nextInt();
        n100 = num / 100;
        n10 = (num % 100) / 10;
        n1= num % 10;

        System.out.println("First digit: " + n100 + " Second: " + n10 + " Third " +
n1);
        System.out.println("The sum of the three digits is: " + (n100+n10+n1));
    } // end main
} // end of class digitNum
```

סיכום כללים בכתיבת תוכנית בג'אווה

ג'אווה כמו שפות תכנות אחרות מחייבת כללי כתיבה קשיחים, המהדר של השפה לא מסיים בהצלחה את תהליך התרגום של התוכנית, אם לא מקפידים על תחביר הפקודות, כמו כן בג'אווה ישנם כללי כתיבה, שעבור חלק מהם אי שמירה על הכללים אינה פוגעת בתהליך הידור התוכנית, אולם, הכללים אומצו ע"י המתכנתים בכל העולם ואימוצם מקל על הבנת התוכנית.

1. נקודה-פסיק בסוף כל פקודה. –חובה
2. הקפדה על אינדנטציה – ההזחה של בלוקים באופן שהמעין יודע לאיזה קטע או בלוק שייכת כל הוראה - מומלץ.
3. שמות ברורים ובעלי משמעות. - מומלץ
4. שם מחלקה - מתחיל באות כל מילה חדשה בשם באות גדולה – מומלץ
5. שם משתנה - מתחיל באות קטנה וכל מילה נוספת בשם מתחילה באות גדולה – מומלץ
6. מילים שמורות נכתבות באותיות קטנות - חובה.

האופרטור נקודה – Dot notation

בתוכנית אנו יוצרים אובייקטים, לאובייקטים יש תכונות ושיטות אליהן אנו מעוניינים "לגשת". שפת ג'אווה מאפשרת גישה לתכונות המחלקה ושיטות המחלקה באמצעות האופרטור "נקודה" אנו כותבים את שם האובייקט, את הסימן נקודה ולאחר מכן את שם התכונה של האובייקט או השיטה שהאובייקט מפעיל.

נניח שנתונה ההגדרה של המחלקה הבאה:

```
Class car {  
    String color;  
    Double price;  
    String typeCar;  
}
```

נניח שיצרנו שני אובייקטים של המחלקה מכונת: myCar, myWifeCar
ועתה ברצוננו לתת ערך לצבע המכונת:

myWifeCar.color = "green";

myCar.price= 123400;

אילו הוגדרה שיטה במחלקה שמשנה את מחיר המכונת ע"י שיטה הנקראת discount המחשבת מחדש את מחיר המכונת, הזימון של השיטה באמצעות האופרטור "נקודה"

myWifeCar.discount();

הסוגריים מעידים שאנו מזמנים שיטה המופעלת ע"י האובייקט.
במידה ויש לשיטה "ארגומנטים" (ערכים) שעלינו להעביר כדי שתפעל (לדוגמא אחוז ההנחה)
אנו נרשום אותם בין הסוגריים.
על יחוס נקודה – dot notation ארחיב בהמשך פרקי הספר המתקדמים.

פרק רביעי – לשאול שאלות

פרק רביעי מציג את מבנה התנאי ומימושו בשפת ג'אוה. מבנה תנאי פשוט ומורחב. טבלת אופרטורי היחס (קטן, גדול, גדול שווה, קטן שווה, שווה, שונה) מבנה תנאי ואופרטורי היחס מאפשרים לנו ליצור תוכניות אשר פועלות שונה בהתאם לתוצאת התנאי שנבדק. בהמשך הפרק אני אציג את המורכבות של שילוב משפטי תנאי, כולל, ביצוע משפטי תנאי מקוננים.

עקוב אחר האלגוריתמים המילוליים ומימושם בשפת ג'אוה. חשוב להריץ את התוכניות המופיעות בפרק ואף לתרגל תוכניות נוספות הכוללות משפטי תנאי מקוננים. מומלץ להשתמש בסביבה של Jeliot להרצת התוכניות.

החלק האחרון של הפרק מציג שני נושאים חשובים: אופרטורים לוגיים המאפשרים לנו "לשאול" שאלות מורכבות, אופרטורים אלו (או, וגם, שלילה) מאפשרים ליצור ביטויים לוגיים מורכבים. בחלק האחרון של הפרק מוצגת המחלקה String מחלקה זו והשיטות הכלולות בה מאפשרים לנו לבצע עיבוד של טקסטים.

פרק רביעי – לשאול שאלות בג'אווה

אופרטורי יחס

אופרטור	משמעות	דוגמא
==	שוויון	<code>X == 3</code>
!=	אי שוויון	<code>X != 3</code>
<	קטן יותר	<code>X < y</code>
>	גדול	<code>x > 0</code>
<=	קטן או שווה	<code>Num <= 7</code>
>=	גדול או שווה	<code>Sum >=</code>

אופרטורי היחס מאפשרים לבנות ביטויים לוגיים שערכם הוא אמת או שקר – **true** או **false**

הערכים הלוגיים ניתנים להשמה במשתנה לוגי

```
boolean test;
```

```
test = mark > 55;
```

במשתנה `test` יהיה ערך `true` או `false` אם הערך ב- `mark` גדול מ- 55 ערכו של `test` יהיה אמת, ושקר אחרת.

התוכנית הבאה מסכמת את נושא אופרטורי יחס, הרץ את התוכנית ועקוב אחר הפלט. רצוי לרשום לפני הרצת התוכנית מה לדעתך יהיה הפלט.

```
class RelationalDemo {
    public static void main(String[] args) {
        //a few numbers
        int i = 23;
        int j = 12;
        int k = 65;
        System.out.println("ערכי המשתנים");
        System.out.println(" i = " + i);
        System.out.println(" j = " + j);
        System.out.println(" k = " + k);

        //greater than
```

```

System.out.println("- גדול מ");
System.out.println(" i > j = " + (i > j)); //false
System.out.println(" j > i = " + (j > i)); //true
System.out.println(" k > j = " + (k > j)); //false
// (they are equal)
// greater than or equal to
System.out.println("- גדול או שווה");
System.out.println(" i >= j = " + (i >= j)); //false
System.out.println(" j >= i = " + (j >= i)); //true
System.out.println(" k >= j = " + (k >= j)); //true
// less than
System.out.println("- קטן מ");
System.out.println(" i < j = " + (i < j)); //true
System.out.println(" j < i = " + (j < i)); //false
System.out.println(" k < j = " + (k < j)); //false
// less than or equal to
System.out.println("- קטן או שווה");
System.out.println(" i <= j = " + (i <= j)); //true
System.out.println(" j <= i = " + (j <= i)); //false
System.out.println(" k <= j = " + (k <= j)); //true
// equal to
System.out.println(" שווה ל-");
System.out.println(" i == j = " + (i == j)); //false
System.out.println(" k == j = " + (k == j)); //true
// not equal to
System.out.println(" שונה");
System.out.println(" i != j = " + (i != j)); //true
System.out.println(" k != j = " + (k != j)); //false
}
}

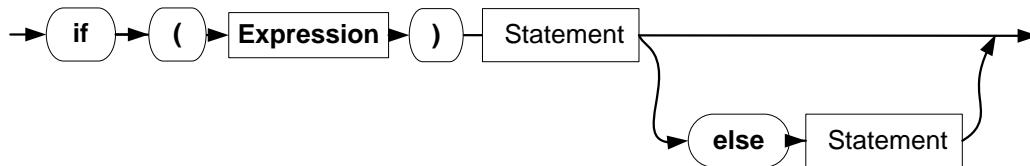
```

משפט תנאי בג'אווה - if

בפרק ראשון – אלגוריתם, הצגתי את מבנה תנאי, מבנה זה מאפשר לבצע פקודה או קבוצת פקודות רק אם מתקיים תנאי כלשהו, שפת ג'אווה כמו שפות אחרות מממשת את מבנה

התנאי באמצעות פקודה תנאי **if**

מבנה תחבירי משפט if



If (תנאי)

```

{
    בלוק פקודות המתבצע אם התנאי מתקיים ( ערכו אמת )
}
else
{
    בלוק פקודות המתבצע אם התנאי אינו מתקיים ( ערכו שקר )
}
  
```

דוגמא:

המחלקה IfClass

השיטה main קולטת שני מספרים ומדפיסה את הגדול מבין שניהם.

הסבר:

if מילה שמורה שמופיעה בתחילת משפט התנאי

(num1>num2) התנאי שמשפט ה-**if** בודק האם ערך הביטוי הוא אמת או שקר.

השיטה main כוללת שני בלוקים של הוראות, בלוק שמתבצע אם התנאי אמת (true)

והבלוק השני שאחרי המילה השמורה **else** (אחרת) מתבצע כאשר ערכו של התנאי שקר)

(false

מחלקה ifClass

```
import java.util.Scanner;
class ifClass {

    public static void main (String[] args){
        Scanner input = new Scanner (System.in);
        float num1=input.nextFloat();
        float num2=input.nextFloat();
        if (num1 > num2) {
            System.out.println("The bigger number is " + num1 );
        }
        else {
            System.out.println("The bigger number is " + num2 );
        }
        } // end of main
    } // end of ifClass
```

ניתן לרשום תנאים שונים, לג'אווה קבוצת של אופרטורי יחס רבים (ראה טבלה בתחילת הפרק)
דוגמאות לביטויים לוגיים:

נניח שיש לנו את ההגדרה הבאה:

int num1=5; int num2 = 9;

טבלת דוגמא - שימוש באופרטורי יחס עם המשתנים הוגדרו

הביטוי	הסבר	ערך הביטוי לפי ערכי num1 ו- num2
num1 > num2	num1 גדול מ- num2	שקר
num1 > num2	num2 גדול מ- num1	אמת
num1 == num2	num1 שווה ל- num2	שקר
num1 != num2	num1 שונה מ- num2	אמת
num1 >= num2	num1 גדול או שווה ל- num2	שקר
num2 <= num1	num2 גדול או שווה ל- num1	אמת

במחלקה SqFunction שכתבנו לחישוב שורשי משוואה ריבועית, ישנה בעיה "קטנה", מה קורה במקרה שלמשוואה אין שורשים ממשיים כלומר, הביטוי $(b*b - 4*a*c)$ הוא שלילי ולא ניתן לחשב את השורש הריבועי של ביטוי שערכו קטן מ-0? עתה, לאחר שלמדנו לכתוב משפטי תנאי נוכל לבדוק אם הערך שלילי ומקרה זה לא נחשב את השורשים של המשוואה אלא נציג הודעה מתאימה.

הבלוק בשיטה main לחישוב השורשים לפני השינוי וללא שימוש ב- if

```
a=input.nextDouble();
b= input.nextDouble();
c= input.nextDouble();
temp = Math.sqrt(b*b - 4*a*c);
System.out.println( " the first root is " + (-b + temp)/(2*a));
System.out.println( " the second root is " + (-b - temp)/(2*a));
```

השיטה main לחישוב השורשים עם השימוש במשפט if

```
import java.util.Scanner;
class SqFunction {
    public static void main( String args[]) {
        Scanner input = new Scanner(System.in);
        double a,b,c,temp;
        a=input.nextDouble();
        b= input.nextDouble();
        c= input.nextDouble();
        if ((b*b- 4*a*c) < 0 ) {
            System.out.println(" no sulution "); }
        else
        { temp = Math.sqrt(b*b - 4*a*c);
            System.out.println( " the first root is " + (-b + temp)/(2*a));
            System.out.println( " the second root is " + (-b - temp)/(2*a));}
        } // end of main
    } // end of SqFunction
```

מחלקה MyMath

המחלקה MyMath, במחלקה שירות זו נאגד כמה שיטות מתמטיות המבצעות חישובים שונים (מחלקה זו בדומה למחלקה Math המוכרת לנו כל השיטות הם סטטיות).
 השיטה myAbs - מקבלת מספר ומחזירה את ערכו המוחלט
 השיטה theBigNumber - מקבלת 3 מספרים ומחזירה את הגדול ביותר
 השיטה myPred - מקבלת מספר שלם וחיובי בטווח 1 - 100 ומחזירה את הקודם לו, אם ערך המספר 0 השיטה מחזירה את הערך 100

```
import java.util.Scanner;

class MyMath {
    public static int myAbs(int x) {
        if(x>0) {
            return x; }
        else
        {
            return -x;
        }
    }

    public static int theBigNumber(int x, int y, int z) {
        int t1,t2;
        if( x>y) {
            t1= x; }
        else
        {t1= y;
        }
        if (y>z) {
            t2= y;
        }
        else {
            t2=z;
        }
        if(t1>t2) {
            return t1; }
        else
```

```

        { return t2;}
    }
    public static int myPred(int x) {
        if (x<100) {
            return x-1 ;
        }
        else
        {
            return 100;
        }
    }

    public static void main( String args[]) {
        Scanner input = new Scanner(System.in);
        int x,y,z;
        x=input.nextInt();
        y= input.nextInt();
        z= input.nextInt();
        System.out.println(" the abs number of " + (x) + "is " + myAbs(x));
        System.out.println(" the abs number of " + (y) + "is " + myAbs(y));
        System.out.println(" the abs number of " + (z) + "is " + myAbs(z));
        System.out.println(" the pred number of " + (x) + "is " + myPred(x));
        System.out.println(" the pred number of " + (y) + "is " + myPred(y));
        System.out.println(" the pred number of " + (z) + "is " + myPred(z));
        System.out.println(" the biggest number of is " + theBigNumber(x,y,z));
    }
} // end of myMath

```


אופרטורים לוגיים בג'אווה

ברצוננו לכתוב שיטה המקבלת 3 מספרים ומחזירה את המספר הגדול מבין השלושה.

נפתור את הבעיה באמצעות אלגוריתם הכי_גדול(מס,1,מס,2,מס,3)

הערה: מס,1,מס,2,מס,3,זמני,1,זמני,2 – הם מטיפוס double

הכי_גדול(מס,1,מס,2,מס,3)

1. אם מס1 < מס2 אז

זמני1 = מס1 אחרת

זמני2 = מס2

2. אם מס2 < מס3 אז

זמני2 = מס2 אחרת

זמני3 = מס3

3. אם זמני1 < זמני2 אזי

החזר זמני1 אחרת

החזר זמני2

הערות לאופן כתיבת שיטות באמצעות אלגוריתם מילולי:

יש לפתוח את האלגוריתם בכותרת האלגוריתם עם פירוט הפרמטרים שהשיטה מקבלת, אין

צורך להכריז על סוג הנתונים, זאת, ניתן לבצע באמצעות הוספת הערה.

את ההוראה **return** המופיעה בשיטה המחזירה ערך אנו מציינים באלגוריתם המילולי

באמצעות המילה החזר.

האלגוריתם המילולי לחישוב המספר הגדול בין 3 מספרים, כלול במחלקה class myMath

שמופיעה בעמוד קודם.

שאלה: האם האלגוריתם הזה הוא הדרך היחידה לביצוע המשימה

תשובה: לא, ניתן לכתוב את האלגוריתם בדרכים נוספות, לדוגמא ע"י קינון משפטי תנאי:

הכי_גדול(מס,1,מס,2,מס,3)

1. אם מס1 < מס2 אז

אם מס1 < מס3 אזי החזר מס1 אחרת

החזר מס2

2. אחרת

אם מס2 < מס3 אזי החזר מס2 אחרת

קיגון משפטי תנאי

נממש בג'אוה את האלגוריתם: הכי גדול (מס,1,מס,2,מס,3)

המחלקה NestedIf

```
class NestedIf {  
    public static double bigNumber(double x, double y, double z) {  
        if (x>y) {  
            if (x>z) {  
                return x; }  
            else  
                { return z; }  
        }  
        else  
            if (y>z) {  
                return y; }  
            else  
                { return z; }  
    }  
    public static void main(String args[]) {  
        System.out.println("The big number is " + bigNumber(12,6,23) );  
    } // end of main  
} // end of class
```

שים לב!

כל **else** שייך ל-**if** הקרוב אליו. זאת משמעות הקיגון של משפטי תנאי.

אופרטורים לוגיים

הביטויים:

`malben.x == malben.y`

`stool.legs > 4`

`num1 == num2`

`ok = false`

הם ביטויים לוגיים, ערכם אמת או שקר.

אנו משתמשים במשתנים לוגיים, וביטויים לוגיים בשיטות אותם אנו כותבים זאת כדי "להחליט החלטות":

האם חשבון הלקוח נמצא ביתרה שלילית?

האם המשתמש לחץ על כפתור ימני בעכבר?

האם המלבן הוא גם ריבוע?

האם המשתמש רוצה לשחק שוב?

האם $x > y$?

לעיתים צריך לבדוק מצבים יותר מורכבים כגון:

האם $x > y$ וגם $x > z$?

האם $x = y$ וגם $y = z$?

האם יתרת חשבון לקוח < 0 וגם בקשה-להלוואה = אמת?

האם טמפרטורה < 10 או לחות < 50

בכל שפות התכנות וגם בג'אווה יש אופרטורים לוגיים, האופרטורים מאפשרים חיבור בין ביטויים לוגיים פשוטים ובניית ביטויים יותר מורכבים.

האופרטור and

עבור שני ביטויים לוגיים המחוברים באמצעות האופרטור and הטבלה הבאה מציגה את תוצאת חיבור הביטויים:

טבלת אמת – and

ביטוי א	ביטוי ב	ביטוי א and ביטוי ב
שקר	שקר	שקר
אמת	שקר	שקר
שקר	אמת	שקר
אמת	אמת	אמת

האופרטור and בג'אווה הוא מסומן ע"י &&

דוגמא: בדיקה האם הנקודה pointA והנקודה pointB הם אותה נקודה, ערכי x שווים וגם ערכי y שווים.

```
if((pointA.x==pointB.x) && (pointA.y == pointB.y))
    { System.out.println("its the same point "); } else
    { System.out.println("its the not same point " );}
```

האופרטור or

עבור שני ביטויים לוגיים המחוברים באמצעות האופרטור or הטבלה הבאה מציגה את תוצאת חיבור הביטויים:

טבלת אמת – or

ביטוי א	ביטוי ב	ביטוי א or ביטוי ב
שקר	שקר	שקר
אמת	שקר	אמת
שקר	אמת	אמת
אמת	אמת	אמת

האופרטור or בג'אווה הוא מסומן ע"י ||

דוגמא: בדיקה האם הנקודה pointA והנקודה pointB יש להם ערכי x או ערכי y שווים.

```
if((pointA.x==pointB.x) && (pointA.y == pointB.y))
    { System.out.println("its the same point "); } else
    { System.out.println("its the not same point " );}
```

שם לב! הגישה לתכונת הערך של האובייקט נקודה מתבצעת באופן ישיר, לא אפשרי אם התכונות הוגדרו כפרטיות ופוגע בעקרון הכימוס ומוצג כך כדי להתמקד בנושא המסויים של תנאים המופיע כאן.

נכתוב שוב את השיטה למציאת המספר הגדול ביותר מבין 3 מספרים ע"י שימוש באופרטור &&

```
class ifAnd {
    public static double bigNumber(double x, double y, double z) {
        if ((x>=y) && (x>=z)) {
            return x; } else
        if
        ((y>=x) && (y>=z)) {
            return y; } else
        if ((z>=x) && (z>=y)) {
            return z; }
        }
    public static void main(String args[]) {
        System.out.println("The big number is " + bigNumber(19,17,11) );
    } // end of main
} // end of ifAnd class
```

האופרטור not

האופרטור not הוא אופרטור אונרי, כלומר פועל על ביטוי יחיד. האופרטור not הופך את הערך של ביטוי לוגי.

ביטוי	not(ביטוי)
שקר	אמת
אמת	שקר

האופרטור not בג'אווה הוא מסומן ע"י !

הבהרה:

ישנו האופרטור וגם (&) וישנו האופרטור &&

ישנו האופרטור או (|) וישנו האופרטור ||

מהו ההבדל ומתי משתמשים ב- & ומתי ב- &&, מתי משתמשים ב- | ומתי משתמשים ב- ||

דוגמא:

נניח שנתונות ההגדרות הבאות:

```
int [] arr = new int[5];  
arr[1] = 1;
```

ושורת הקוד הבאה:

```
If(arr.length > 0 & arr[1] != 1)
```

המשמעות שביצוע השורה יגרום לבדיקת שני התנאים המופיע בתנאי המורכב, ואם המערך arr לא היה מוגדר הייתה שגיאה בזמן ריצה. ולכן רצוי להשתמש ב- &&, השימוש ב- && גורם לביצוע שונה של תנאי הבדיקה, אם $arr.length = 0$ והמערך לא מוגדר, לא יבדק התנאי השני ולא תתרחש שגיאת ריצה.

שילוב בין האופרטורים הלוגיים

ניתן לבנות ביטויים לוגיים מורכבים הכוללים ביטויים לוגיים בסיסיים עם אופרטורים לוגיים, השאלה היא כיצד מחושב ערך הסופי של הביטוי?

דוגמא:

אם (מספר_א < מספר_ב) וגם (מספר_ב = מספר_ג) או (מספר_א < 0)
ביטוי א ביטוי ב ביטוי ג

נניח: ביטוי א = שקר, ביטוי ב = אמת, ביטוי ג = שקר
תוצאת הביטוי: שקר

הסבר: המחשב מחשב בתחילה את האופרטור **and**, לאופרטור **and** קדימות בחישוב לפני האופרטור **or**,

האופרטור **not** קודם בחישוב לפני **and** ו- **or**
סיכום: סדר עדיפות החישוב של ביטויים לוגיים

1. האופרטור **not**

2. האופרטור **and**

3. האופרטור **or**

סוגריים משנים את סדר העדיפות.

משימה: כתוב אלגוריתם מילולי הקולט 3 מספרים ממשיים, על האלגוריתם לבדוק האם המספרים מייצגים אורכי צלעות משולש.

הסבר "גיאומטרי":

3 מספרים יכולים לייצג את אורכי 3 צלעות משולש, אם ורק אם אף צלע אינה גדולה או שווה לסכום שתי הצלעות האחרות, ובשפה מתמטית אנו נרשום זאת כך:

אם a, b, c מייצגים את אורכי הצלעות במשולש הרי שחייב להתקיים:

$$a < b + c \text{ וגם } b < a + c \text{ וגם } c < a + b$$

האלגוריתם

משולש.אם_משולש()

אם משולש.א. $>$ משולש.ב. + משולש.ג. וגם

משולש.ב. $>$ משולש.א. + משולש.ג.

משולש.ג. $>$ משולש.א. + משולש.ב.

החזר אמת אחרת

החזר שקר

המחלקה TriAngle

```
public class TriAngle {
    private double a,b,c;
    public TriAngle(double a, double b,double c){
        this.a=a;
        this.b=b;
        this.c=c;
    }
    public double getA(){
        return this.a;
    }
    public double getB(){
        return this.b;
    }
    public double getC(){
        return this.c;
    }
    public boolean isMesh(){
        double x,y,z;
        x=this.getA();
        y=this.getB();
        z=this.getC();
        if ((x>=y+z) || (y>=x+z) || (z>=x+y))
            return false;
        else
            return true;
    }
}
```

הסבר על היישום בג'אווה

בחרתי להגדיר מחלקה משולש הכוללת 3 תכונות שהם צלעות המשולש (הערה: מידע זה קובע גם את זוויות המשולש)


```
public class TriAngle {  
private double a,b,c;
```

השיטה הבונה של המחלקה TriAngle

```
public TriAngle(double a, double b,double c){  
    this.a=a;  
    this.b=b;  
    this.c=c;  
}
```

המחלקה TestTriAngle

```
public class TestTriAngle {  
    public static void main(String[] args) {  
        TriAngle m1 = new TriAngle(3,4,5);  
        TriAngle m2 = new TriAngle(3,12,5);  
        TriAngle m3 = new TriAngle(3,2,1);  
        if (m1.isMesh())  
            System.out.println("no triangle");  
        else  
            System.out.println(" triangle");  
        if (m2.isMesh())  
            System.out.println("no triangle");  
        else  
            System.out.println(" triangle");  
        if (m3.isMesh())  
            System.out.println("no triangle");  
        else  
            System.out.println(" triangle");  
    }  
}
```

בשיטה main אנו יוצרים 3 משולשים ובודקים האם אכן יצרנו משולשים חוקיים.
יצירת משולשים:

```
TriAngle m1 = new TriAngle(3,4,5);  
TriAngle m2 = new TriAngle(3,12,5);  
TriAngle m3 = new TriAngle(3,2,1);
```

בדיקה האם המשולש שנוצר, אכן משולש חוקי:

```
if (m1.isMesh())  
    System.out.println("no triangle");  
else  
    System.out.println(" triangle");
```

Vector המחלקה

המחלקה וקטור מגדירה וקטורים. לוקטור שתי תכונות: גודל הוקטור וזווית הוקטור.
המחלקה כוללת:

שיטה בונה

שיטה לקביעת ערך גודל הוקטור

שיטה לקביעת זווית הוקטור

שיטה להחזרת גודל הוקטור

שיטה להחזרת זווית הוקטור

שיטה להחזרת מחרוזת פלט עם נתוני הוקטור

משימה:

כתוב מחלקה testVector שתכלול את השיטות הבאות:

שיטה להכפלת גודל הוקטור בגודל ממשי

שיטה המקבלת וקטור ומחזירה את הוקטור השקול של שני הוקטורים, הוקטור שזמן את

השיטה והוקטור שהועבר כארגומנט.

קוד המחלקה **vector**

```
import java.util.Scanner;

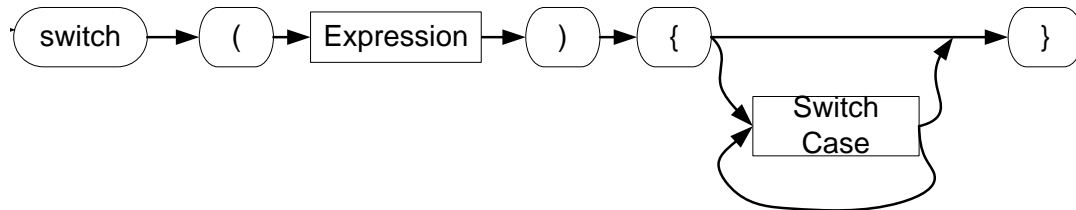
public class Vector {
    private double size;
    private double angle;
    public Vector(double s, double a) {
        this.size=s;
        this.angle=a;
    }
    public void setSize(double s){
        this.size=s;
    }
    public void setAngle(double a){
        this.angle=a;
    }
    public double getSize(){
        return this.size;
    }
    public double getAngle(){
        return this.angle;
    }
    public String toString(){
        String str;
        str="size=" + this.size+ "<angle=>" + this.angle;
        return str;
    }
    public static void main(String[] args) {
        Scanner input = new Scanner (System.in);
        double s,a;
        System.out.println(" Enter the size of vector");
        s= input.nextDouble();
        System.out.println(" Enter the angle of the vector");
        a=input.nextDouble();
    }
}
```

```
Vector v1= new Vector(s,a);
System.out.println(" Enter the size of vector");
s= input.nextDouble();
System.out.println(" Enter the angle of the vector");
a=input.nextDouble();
Vector v2= new Vector(s,a);
System.out.println(v1.toString());
System.out.println(v2.toString())
}
}
```

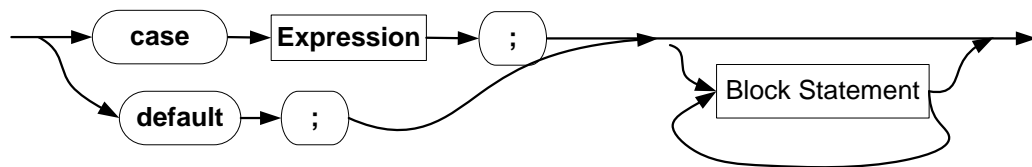
משפט switch

משפט switch הוא במידה רבה משפט if "משוכלל", משפט זה מאפשר לכנס אוסף ברירות במבנה ברור יותר מאשר מבנה הכולל מספר משפטי If ו-else-ים.

מבנה תחבירי משפט switch:



מבנה תחבירי Switch Case



```
switch ( value selector ) {
case value selector 1 : statement; break;
case value selector 2 : statement; break;
case value selector 3 : statement; break;
case value selector 4 : statement; break;
case value selector 5 : statement; break;
default: statement;
}
```

הסבר:

value selector הוא ביטוי שתוצאתו ערך סודר (ערכים סודרים הם: מספרים שלמים, תווים (משפט switch משווה בין תוצאת חישוב ערך זה עם כל אחד מהערכים המופיעים בהמשך המשפט. אם נמצאה התאמה, בקרת התוכנית תעבור לביצוע statement המופיע בהמשך לערך זה.

שים לב! המילה **break** מסיימת את זרימת ביצוע התוכנית בתום ביצוע הבחירה שנבחרה ובהמשך הביצוע לאחר משפט ה- **switch**. במידה ולא תופיע המילה **break** יתבצע הקוד של שאר הברירות, לעיתים, מצב זה מתאים למטרות התוכנית ולעיתים לא. במידה ואף אחד מהערכים לא היה כערכו של ביטוי ההשוואה יתבצע קוד ה-**default**.

המחלקה mySwitch - מחשבון לפי בחירה

למחלקה 7 שיטות כל שיטה לביצוע פעולה אריתמטית שונה. כל השיטות במחלקה סטטיות, שוב בדומה למחלקת השירות Math, המחלקה לא מגדירה אובייקט אלא מהווה מחלקת שירות המבצעת חישובים מתמטיים שונים.

השיטה main

בשיטה main נקלט קוד הפעולה, אם הפעולה המבוקשת היא פעולה אונרית (רק עם אופרנד יחיד) מתבצע קלט רק של מספר יחיד. אם קוד הפעולה הוא של פעולה בינארית (פעולה על שני אופרנדים) נקלטים שני מספרים. ההבחנה בין המקרים הללו מתבצעת עם משפט תנאי הכולל שימוש באופרטור הלוגי or שסימונו בג'אוה || .

```
import java.util.Scanner;
class mySwitch {
    public static int myMul(int x, int y ) {
        return x*y;
    }
    public static int myAdd(int x, int y ) {
        return x+y;
    }
    public static int myDivInt(int x, int y ) {
        return x/y;
    }
    public static int mySub(int x, int y ) {
        return x-y;
    }
    public static int myMod(int x, int y ) {
        return x%y;
    }
    public static int myInc(int x ) {
        return ++x;
    }
    public static int myDec(int x ) {
        return --x;
    }
    public static void main(String args[]) {
        Scanner input = new Scanner (System.in);
        int x=0;
        int y=0;
        int op;
        System.out.println(" 1 for x*y");
```

```

System.out.println(" 2 for x+y");
System.out.println(" 3 for x/y");
System.out.println(" 4 for x-y");
System.out.println(" 5 for x%y");
System.out.println(" 6 for inc x ");
System.out.println(" 7 for dec y ");
System.out.println("please enter your selection 1 -7 ");
op=input.nextInt();
if((op==6) || (op==7))
    { System.out.println("please enter A number");
      x=input.nextInt(); }
else
    { System.out.println("please enter first number ");
      x=input.nextInt();
      System.out.println("please enter second number ");
      y=input.nextInt();
    }
switch (op) {
case 1: System.out.println(x + "*" +y +"=" + myMul(x,y)); break;
case 2: System.out.println(x + "+" +y +"=" + myAdd(x,y)); break;
case 3: System.out.println(x + "/" +y +"=" + myDivInt(x,y)); break;
case 4: System.out.println(x + "-" +y +"=" + mySub(x,y)); break;
case 5: System.out.println(x + "%" +y +"=" + myMod(x,y)); break;
case 6: System.out.println(x + "inc" +x +"=" + myInc(x)); break;
case 7: System.out.println(x + "dec " +x +"=" + myDec(x)); break;
    default: System.out.println(" worng operation ");
    }
}
}

```

numSwitch המחלקה

השיטה main במחלקה numSwitch קולטת מספר שלם, המספר נבדק באמצעות פקודת הבקרה switch התוכנית מדפיסה בהתאם לערך שלם בטווח 1-9 את המילים one two three וכך הלאה.

```
import java.util.Scanner;
class numSwitch {
    public static void main (String args []) {
Scanner input=new Scanner(System.in);
        int op;
        System.out.println("please enter digit ");
        op=input.nextInt();
        switch (op) {
            case 1: System.out.println("one"); break;
            case 2: System.out.println("two"); break;
            case 3: System.out.println("three"); break;
            case 4: System.out.println("four"); break;
            case 5: System.out.println("five"); break;
            case 6: System.out.println("six"); break;
            case 7: System.out.println("seven"); break;
            case 8: System.out.println("eight"); break;
            case 9: System.out.println("nine"); break;
            default: System.out.println(" Worng digit ");
        }
    }
}
```


סיכום – שלבים בבניית מחלקה בג'אוה

נניח שברצוננו לבנות מחלקה – חתולים, נבנה את המחלקה תוך הצגה כללית של השלבים בבניית מחלקה.

השלבים בבניית מחלקה די דומים לשלבים בבניית טיפוס נתונים חדש בשפות פרוצדורליות, טיפוסים, אותם, אנו נוהגים לכנות טיפוסים נתונים מופשטים, כלומר, לא הטיפוסים הבסיסיים שהם חלק מהשפה, אלא, טיפוסים נתונים חדשים שאנו יוצרים. הגדרה של טיפוס נתונים חדש משמעותה, הגדרת התכונות של הטיפוס, וגם את הפעולות המותרות וזמינות על טיפוס הנתונים. לכן השלב הראשון בבניית מחלקה הוא שלב הדיון התיאורטי, והשלב המעשי הכולל פירוט התכונות של המחלקה ולאלו שיטות אנו זקוקים. שלב זה הוא השלב הקשה ביותר משום שהוא השלב המחבר בין האובייקטים הקיימים בעולם הבעיה שלי, לבין ההגדרה "המחשבתית" שלהם, כלומר, איך לייצג את האובייקט במחלקה שאני כותב.

מחלקה: חתול

תכונות: גזע, שם, מצב רעב, חיסון, צבע, מצב רוח

שיטות:

שיטה בונה:

חתול(גזע,שם,מצב רעב,חיסון,צבע)

שיטות

מיצי.גזע()

מיצי.מצברוח()

מיצי.רעב()

ועוד.. כיד הדמיון הטובה

בתום השלב, בו עצבנו את האובייקט, ורשמנו מה הן השיטות שלדעתנו נזדקק להם, אנו כותבים את השיטה הבונה, זו השיטה המאפשרת לנו ליצור אובייקטים, מופעים של המחלקה שהגדרנו (יתכנו כמה שיטות בונות).

עתה משיש לנו מחלקה הכוללת את הגדרת תכונותיה, ושיטה בונה "המייצרת" את האובייקטים, וכן, הגדרה כללית עבור השיטות להן נזדקק, זה הזמן לחבר הכול ולבנות את השיטות האחרות, בהתאם לתכנון הראשוני שתכננו. בשלב הזה ניתן לחזור שוב לתכונות המחלקה, לשנות, להוסיף ולתקן, זאת בתנאי, שלא נשכח לתקן בהתאם את השיטה הבונה. סיום בניית המחלקה הוא ע"י בדיקת המחלקה, הפעלת השיטות ובדיקת תקינותן ע"י כתיבת

השיטה הראשית main

לסיכום שלבי בניית מחלקה

1. הצהרה על מחלקה
2. הצהרה על התכונות
3. מימוש השיטה-הבונה
4. מימוש השיטות האחרות
5. בדיקת המחלקה

מחלקה String

בשפות תכנות רבות string מהווה משתנה בסיסי, בג'אווה String הינה מחלקה לכל דבר, מחלקה שנכתבה כחלק מממשק הפיתוח הבסיסי של ג'אווה – Java API (Application Programmer Interface).
יצירת מחרוזת והשיטה length

```
class Stringexe{
```

```
    public static void main(String args[]) {
        String st = new String("macabi tal aviv ");
        System.out.println(st);
        System.out.println( " The length of the string " + st.length() );
    } // end of main
} // end of Stringexe
```

יצירת אובייקט של המחלקה String ללא הכנסת ערך:

```
String st = new String("");
```

יצירת אובייקט של המחלקה String עם הכנסת ערך לאובייקט

```
String st = new String("maccabi tel aviv ");
```

קבלת אורך המחרוזת באמצעות השיטה length

```
System.out.println( " the length of the string is " + st.length() );
```

יצירת אובייקט מחרוזת לא באמצעות **new**

השימוש במחרוזות בתוכניות נפוץ ביותר, מסיבה זו ג'אווה מאפשרת קיצור ביצירת אובייקט מחרוזת, וניתן ליצור מחרוזת ללא **new** באופן הבא:

```
String testString = "";
```

```
String myName = "William"
```

האופרטור + - אובייקט מחרוזת

עתה נוכל להסביר טוב יותר את משפט הצגת פלט באמצעות השיטה System.out.println בתוכניות רבות השתמשנו באופרטור + בין חלקים שונים של ההדפסה, הפעולה, שבעצם בצענו, הייתה "שרשור" מחרוזות.

קלט מחרוזת

קליטת תווים למחרוזת באמצעות שיטה מהמחלקה Scanner מתבצעת באופן הבא:
נניח שברצוננו לקלט שם צבע, ההוראה הבאה קולטת תווים עד סוף השורה ומחזירה עצם מסוג מחרוזת.

```
color = in.nextLine
```

הפעולה `nextLine` קולטת את התווים בקלט עד סוף שורת הקלט ומחזירה עצם מסוג מחרוזת עם התוכן של השורה שנקלטה. אם בקלט הייתה המילה `blue` לאחר הפעולה נוצר עצם `color` עם הערך `blue`.

שאלה

כאשר השתמשנו בשיטה `System.out.println` שרשרנו בין מחרוזות ובין מספר, ולא רק בין שתי מחרוזות, הכיצד?

תשובה

האופרטור `+` הוא אופרטור "חכם", ופעולתו תלויה בטיפוס האופרנדים שהוא מחבר.

האופרטור `+` מחבר שני מספרים

אם האופרטור `+` מופעל על שני מספרים, האופרטור מחזיר את הסכום המספרי של האופרנדים, בקיצור ובמילים פשוטות, הוא פועל כפעולת חיבור רגילה.

האופרטור `+` פועל על שתי מחרוזות

האופרטור משרשר את שתי המחרוזות, ומחזיר מחרוזת אחת המורכבת משתי המחרוזות

האופרטור `+` פועל על מחרוזת ומספר

אם רק אחד מהאופרנדים הוא מחרוזת והאופרנד השני מספר, מומר המספר למחרוזת ושתי המחרוזות משורשרות. אם אחד האופרנדים הוא מסוג `boolean` יומר האופרנד למחרוזת `"false"` או `"true"` בהתאם לערך האופרנד.

שאלה

כיצד מומר אופרנד שהוא מספר למחרוזת?

תשובה

ע"י שימוש לא נראה בשיטה `toString()`

הסבר

לא אוכל לספק הסבר מלא בשלב זה, שכן מרביתו שייך לפרקים בספר, המרחיבים את נושא עצמים בג'אווה, כולל, ירושה בין מחלקות, אולם, ניתן להסביר באופן חלקי שבתוך הביטוי, הנמצא בין הסוגריים, מופעלת השיטה `toString`, למרות שאינה מופיעה במפורש. שיטה זו הופכת כל אופרנד-אובייקט למחרוזת, מה שמאפשר בסופו של תהליך לאופרנד `+` לשרשר את כל חלקי המחרוזת.

שיטות לטיפול במחרוזות

לג'אוה שיטות רבות לטיפול במחרוזות. נספח ג' בספר מתאר ומסביר את החשובות שבהן. באמצעות מספר משימות נציג את היכולות של שיטות אלו. מחרוזת לא ניתנת לשינוי! immutable, כמו כן לא ניתן להשוות "באופן ישיר" בין שתי מחרוזות כי ההשוואה תהיה בין שתי ההפניות של האובייקטים מחרוזות. כל פעולות שינוי נערך או השוואה וכל "טיפול" במחרוזת מומלץ ורצוי להפעיל באמצעות שיטות המחלקה String

משימה

נתונות שתי מחרוזות – st1, st2 אובייקטים מסוג String, ואנו מעוניינים לבדוק האם האובייקט st2 כלול בתוך האובייקט st1. דוגמא:

האובייקט st1 מכיל את הערך "I love computer games"

האובייקט st2 מכיל את הערך "computer"

התשובה לשאלה האם מכילה (st1,st2) תהייה אמת עבור הערכים שבדוגמא.

עתה נכתוב שיטה שמקבלת שני אובייקטים מסוג מחרוזת ומציגה את מיקום תת המחרוזת במחרוזת, או הודעה שתת המחרוזת לא נמצאת.

המחלקה Stringexe2 - האם מכילה מחרוזת א' את מחרוזת ב'

```
class Stringexe2
{
    public static int foundString(String s1, String s2){
        return s1.indexOf(s2); }
    public static void main(String args[]) {
        int index;
        String st1 = "I love computer games";
        String st2 = "computer";
        index = foundString(st1,st2);
        if(index != -1)
            { System.out.println("The string: " + st2 + " found in " + st1 + " in location " + index );}
        else
            { System.out.println( "The string " + st2 + " is not within " + st1 );}
    } // end of main
} // end of Stringexe class
```

המחלקה isLetOrDigit – בדיקת תו

המחלקה isLetOrDigit מזמנת שיטה של המחלקה String, ש"קוטעת" את התו הראשון של המחרוזת המתקבלת מהקלט ובודקת האם התו הוא : ספרה או אות גדולה, או אות קטנה.

השיטה עושה שימוש באופרטור הלוגי and

קליטת תו בודד (שים לב, אני משתמש בשיטה לקליטת מחרוזת)

```
st= input.nextLine();
```

"הפרדת" התו הראשון מהמחרוזת.

שים לב! בהרצת התוכנית יש להקליד תו אחד שמיקומו במחרוזת הוא 0, אני משתמש בשיטה

st.charAt(0) שמחזירה תו במחרוזת לפי מיקום מבוקש (כאן בקשתי מיקום 0)

המשך השיטה main בדיקה איזו סוג תו הוקלד.

```
import java.util.Scanner;
public class isLetOrDigit {
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        char tav;
        String st;
        System.out.println("please enter A char");
        st= input.nextLine();
        tav= st.charAt(0);
        System.out.print(" you pressed a ");
        if( tav >='A' && tav <= 'Z')
            { System.out.println(" capital letter. ");}
        else
            if( tav >='A' && tav <= 'z')
                { System.out.println(" small letter. ");}
            else
                if( tav >='0' && tav <= '9')
                    { System.out.println(" digit ");}
                else
                    { System.out.println();
                     System.out.println(" Its not a letter or a digit ");}
            } // end of main
    } // end of class
```




פרק חמישי – לולאות



בפרק זה אני מציג את המימוש בג'אווה למבנה לולאה. המבנה הסדרתי, תנאי ומבנה לולאה מהווים את אבני הלגו הבסיסים של בקרה ושליטה על זרימת תוכנית בעת ביצועה. למימוש לולאות בג'אווה אנו משתמשים ב-3 הוראות: הוראת for עבור מימוש לולאות לא מותנות, ההוראות while ו-

do while למימוש לולאות מותנות.

בפרק זה ארחיב את הרעיון של בלוק הוראות בתוכנית. בלוק הוראות כולל קבוצת הוראות המתבצעת כחלק ממבנה תנאי, או מבנה לולאה או מבנה סדרתי, השימוש במבני התכנות והכללת הוראות שונות במסגרת בלוקים של פקודות זאת כדי לפתור את הבעיה האלגוריתמית מהווים את הבסיס לפתרון בעיות באמצעות מחשב.

חשוב לשים לב לאופן כתיבת אלגוריתמים מילוליים ברורים, עם מספור הפקודות כשלב לפני מימוש האלגוריתם המילולי בסביבת העבודה. כתיבה נכונה של אלגוריתם מילולי, כחלק מניתוח הבעיה, מהווה צעד ראשון וחשוב בכתיבת תוכניות, שפותרות את הבעיות המוצגות בפנינו.

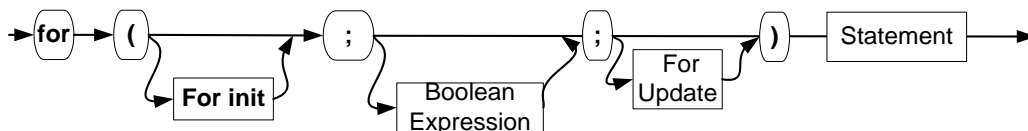
בחלק האחרון של הפרק אני מציג את רעיון קינון לולאות, זה אינו רעיון חדש, לא הוראה חדשה, אלא, הבנה מעמיקה יותר מה המשמעות של שילוב מבני הוראות במיוחד לולאה בתוך לולאה.



פרק חמישי – לולאות בג'אווה

משפט for

מבנה תחבירי משפט for



**for (start value ; Boolean-expression; step)
statement**

הסבר מבנה משפט for

המילה השמורה **for** מתחילה את משפט הלולאה

(start value) for init

הערך ההתחלתי הניתן למשתנה הלולאה for - דוגמא: $i=4$

Boolean expression

תנאי סיום הלולאה – דוגמא : $i \leq 15$

step

אופן עדכון משתנה הלולאה – דוגמא: $i++$

Statement

משפט הביצוע, המשפט יכול להיות משפט פשוט – `System.out.println(x)`

או משפט מורכב הכולל בלוק פקודות

```
{System.out.println(x)
```

```
  x= x+1;
```

```
}
```


המחלקה LotoNumber

הסבר על המחלקה

יצירת מספר אקראי עם השיטה random השיטה כלולה במחלקה Math
 Math.random() מייצרת מספר ממשי בטווח 0-0.999999 , מספר זה מוכפל ב- 49, ערך
 זה מומר לסוג שלם (ע"י אופרטור ההמרה int) והתוצאה של כל הביטוי הוא מספר מסוג שלם
 בטווח 1 – 49.

הביטוי ליצירת מספר אקראי בטווח 1 – 49

```
num= (int) (Math.random() * (49) ) + 1;
```

גוף השיטה ליצירת המספרים באמצעות לולאת for

```
for (int i = 0; i < 6; i++) {  
    num= (int) (Math.random() * (49) ) + 1;  
    System.out.println("Random number : " + num);  
}
```

מבנה ההוראה for המופיעה בשיטה:

הכרזה על i והשמת הערך 0, כערך ראשון של משתנה בקרת הלולאה. int i=0

תנאי סיום של הלולאה i<6

הוספת 1 לערך של i באמצעות i++

```
class LotoNumber {  
    public static void main(String[] args) {  
        int num;  
        for (int i = 0; i < 6; i++) {  
            num= (int) (Math.random() * (49) ) + 1;  
            System.out.println("Random number : " + num);  
        } // end of loop  
    } // end of main  
} // end of LotoNumber class
```

המחלקה Azeret - חישוב עצרת

```
import java.util.Scanner;
class Azeret {
    public static void main (String[] args){
        Scanner input = new Scanner(System.in);
        long f;
        int i;
        f=1;
        System.out.println("Please enter a number");
        int num=input.nextInt();
        for(i=1; i<=num; i++) {
            f=f*i;    }
        System.out.println(f);
    } // end of main
} // end of Azeret class
```

השיטה main

כוללת:

כותרת

קריאת מספר שלם מהקלט.

לולאת חישוב עצרת

שים לב, הערך של עצרת עבור מספרים גדולים "גדול מאוד", (לדוגמא $6! = 720$) לכן הגדרתי את f מסוג long, עובדה שתאפשר לנו לחשב עצרת עבור מספרים גדולים יחסית.

המחלקה loopChar - לולאה עם משתנה מסוג char

גם משתנה מסוג תו (char) יכול לשמש כמשתנה בקרה על לולאת for המחלקה loopChar ממחישה זאת.

```
class loopChar {
    public static void main (String[] args){
        char ch;
        for(ch='a'; ch<='z'; ch++) {
            System.out.println(ch);}
    } // end of main
} // end of loopChar class
```

המחלקה **primeNumber** - בדיקת מספר האם הוא ראשוני?

השיטה **main** במחלקה **primeNumber**, מקבלת מספר ומחזירה ערך אמת אם המספר הוא מספר ראשוני, ושקר אחרת. תזכורת: מספר ראשוני הוא מספר שלם שמתחלק רק! בעצמו ובמספר 1.

```
import java.util.Scanner;

class primeNumber {
    public static void main (String[] args){
        Scanner input = new Scanner(System.in);
        int i;
        boolean ok;
        ok= true;
        System.out.println("Please enter an int number");
        int num = input.nextInt();
        for(i=2; i<num; i++) {
            if (num % i == 0) {
                ok=false;      }   }
        if( ok) { System.out.println(num + " is prime");}
        else
            if( ok== false ) { System.out.println(num + " is not prime");}
        } // end of main
    } // end of class
```

המחלקה **allSeven** - בדיקת מספר האם מתחלק ב-7 ללא שארית

השיטה **main** במחלקה זו קולטת 100 מספרים ומדפיסה כל מספר שאינו מתחלק ב-7 ללא שארית.

```
public class allSeven {
    public static void main(String[] args){
        int num;
        for (num = 0; num <= 100; num++) {
            if ((num % 7) == 0 ) {
                System.out.println(num + " modolo 7 = 0 "); } // end if
        } // end for
    } // end of main
} // end of allseven class
```

המחלקה stringExe - פירוק מחרוזת והדפסה מדורגת

השיטה main במחלקה stringExe קולטת מחרוזת, "מפרקת" את המחרוזת ומדפיסה הדפסה

מדורגת

דוגמא:

מחרוזת הקלט: **COMPUTER**

הפלט:

COMPUTER

COMPUTE

COMPUT

COMPU

COMP

COM

CO

C

class stringExe2

{

public static void main(String args[]) {

int index;

 String st= "COMPUTER";

 index = st.length();

while(index>=1) {

 System.out.println(st);

 st = st.substring(1);

 index --; } // end of while

 System.out.println(" length of " + st.length());

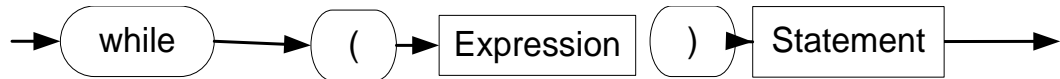
 } // end of main

} end of stringExe2

משפט while

בניגוד ללולאת **for** שהיא "לולאה ללא תנאי" הפועלת מספר מוגדר וקבוע של פעמים, לולאת **while** הינה "לולאה מותנית" והיא תמשיך לפעול כל עוד תנאי מסוים מתקיים.

מבנה תחבירי ההוראה while



מבנה משפט while

```

while ( ביטוי לוגי )
{
    בלוק פקודות
}

```

המחלקה whileExe מדגימה שימוש במשפט while

הבעיה אלגוריתמית

יש לקלוט סדרה של מספרים חיוביים, סדרת המספרים מסתיימת במספר שלילי. על האלגוריתם לחשב את הממוצע של סדרת המספרים שנקלטה.

האלגוריתם המילולי :

חשב_ממוצע()

1. סכום_מספרים = 0

2. מונה_מספרים = 0

3. קלוט(מספר)

4. כל עוד (מספר < 0)

סכום_מספרים = סכום_מספרים + מספר

קלוט(מספר)

מונה_מספרים = מונה_מספרים + 1

5. ממוצע = סכום_מספרים \ מונה_מספרים

6. החזר ממוצע

מחלקה whileExe - חישוב ממוצע

```
import java.util.Scanner;
class whileExe {
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        double sum = 0; double x,avg; int i=0;
        System.out.println("please enter first number ");
        x=input.nextDouble();
        while(x>0) {
            sum+=x; i++;
            System.out.println("please enter " + i + " number");
            x=input.nextDouble();
        }
        System.out.println("The avg is " + sum /i);
    } // end of main
} // end of class
```

חישוב ממוצע – אלגוריתם

השיטה main קולטת קבוצת מספרים שלמים ומחשבת את:

- ממוצע קבוצת המספרים
- המספר הגדול ביותר
- המספר הקטן ביותר.

אלגוריתם המחלקה FindAvg

1. קלוט (ציון)
2. מקסימום = ציון
3. מינימום = ציון
4. סכום = 0
5. מונה_מספרים = 0
6. כל עוד ציון <> 999 בצע
 1. מונה_מספרים = מונה_מספרים + 1
 2. סכום = סכום + ציון
 3. אם ציון < מקסימום אזי מקסימום = ציון
 4. אם ציון > מינימום אזי מינימום = ציון

5. קלוט(ציון)

7. אם מונה_מספרים = 0 הדפס("אין נתונים חוקיים בקלט") אחרת

7.1 הדפס(מונה_מספרים)

7.2 הדפס(ממוצע)

7.3 הדפס(מקסימום)

7.4 הדפס(מינימום)

מחלקה whileStillInput - חישוב ממוצע

מחלקה זו מחשבת ממוצע של קבוצת מספרים, קטע התוכנית הקולט מספרים ומסכמם מתבצע בלולאה עד אשר מסתיים הקלט. אנו עושים שימוש בשיטה של המחלקה Scanner שהוזכרה בקצרה בתחילת הספר. השיטה היא שיטה בוליאנית המחזירה אמת כאשר מסתיים הקלט ושקר אחרת.

עבור עצם input של המחלקה Scanner ניתן לכתוב את ביטוי התנאי:

`while(input.hasNext()`

קוד המלא של המחלקה

```
import java.util.Scanner;
class WhileStillInput {
    public static void main(String[] args) {
        double sum = 0;
        int count = 0;
        Scanner input = new Scanner(System.in);
        while (input.hasNext()) {
            sum = sum + input.nextDouble();
            count++;
        }
        System.out.println("The average is " + sum/count);
    }
}
```

קלט מספרים מתבצע כל עוד יש מספרים בקלט, בכל מחזור בלולאה זוכרים להוסיף 1 למונה המספרים הנקלטים, כך שנוכל לחשב את הממוצע.

המחלקה FindAvg – חישוב הממוצע, מקסימום, מינימום

```
// This class read numbers and compute the max , min and avg
import java.util.Scanner;
public class FindAvg {
    public static void main(String [] args ) {
        Scanner input=new Scanner(System.in);
        int grade,count = 0, sum=0, minimum,maximum;
        double avarge;
        System.out.print ("Enter the first grade ( 999 to quit ): ");
        grade=input.nextInt();
        maximum=minimum=grade;
        // read the rest of the grades
        while ( grade != 999) {
            count ++;          sum+= grade;
            if ( grade > maximum )
                maximum= grade ;
            if ( grade < minimum)
                minimum = grade ;
            System.out.print ("Enter the next grade ( 999 to quit ): ");
            grade=input.nextInt();
        } // end of the while loop
        if ( count== 0)
            System.out.println("No valid grades were entered.");
        else
            { avarge = sum / count;
            System.out.println();
            System.out.println("Total number of the students: " + count );
            System.out.println("Average grade: " + avarge);
            System.out.println("Highest grade : " + maximum);
            System.out.println("Lowest grade: " + minimum);        }
    } // end of main
} // end of class
```

משימה

קלוט מספר שלם, וחשב את סכום כל המספרים האי זוגיים בטווח 1 – מספר שנקלט (1-n)
האלגוריתם המילולי:

```

1. sum = 0
2. קלוט(num)
3. עבור i=0 עד i=num בצע
    4.1 sum = sum + i
    4.2 i = i+2
5. הדפס ( sum )
    
```

המחלקה SumEven – סכום המספרים הזוגיים

```

import java.util.Scanner;;

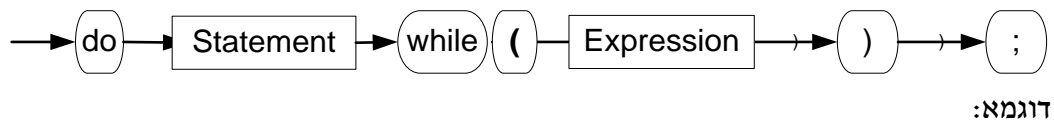
public class SumEven {
    Scanner input = new Scanner(System.in);

    public static void main( String [] args ) {

        int i,num;
        int sum = 0;
        System.out.println("Enter the number");
        num=input.nextInt();
        for(i=0; i<=num ; i=i+2 ) {
            System.out.println("The next even number is " + i);
            sum= sum+i;
        } // end of for
        System.out.println();
        System.out.println("The sum of the even number is : " + sum );
    } // end of main
} // end of SumEven class
    
```

משפט *do while*

משפט זה הממשש לולאת *while* בצורה קצת שונה אינו נפוץ וניתן לכתוב כל תוכנית מחשב ללא שימוש במשפט הזה. ההבדל בין *do while* למשפט *while* הוא בכך שבמשפט זה נבדק תנאי הלולאה רק בסוף הלולאה כלומר הלולאה תתבצע לפחות פעם אחת, לעומת משפט *while* שיתכן ולא תתבצע אפילו פעם אחת.



```

do
{
    System.out.println(i);
    i++;
}
while (i<100)
    
```

משימה – סכום המספרים 1-n

המחלקה הבאה קולטת מספר שלם *n* ומחשבת את סכום המספרים 1-n.

```

class SumD {
    public static void main(String[] args) {
        int sum = 0;
        int num=IO.readInt("please enter a number ");
        for (int current = 1; current <= num; current++) {
            sum += current;
        }
        System.out.println("Sum = " + sum);
    }
}
    
```

הסבר

משתנה *sum* עבור סכום המספרים מאותחל לערך 0, השיטה *main* קולטת מספר טבעי *n* ומבצעת לולאת *for* לסכימה של המספרים 1-n. תוצאת הסכום מוצגת בפלט מחוץ ללולאה.

המחלקה myDiv – ביצוע חלוקה שלמה ללא פעולת חילוק

השיטה main מבצעת פעולת div (חלוקה שלמה) ללא שימוש בפעולת חילוק.

השיטה כוללת לולאה המבוקרת בתנאי לוגי הכולל שימוש במשתנה מסוג **boolean** הרעיון המרכזי באלגוריתם הוא החסרה חוזרת של מחלק מהמחולק, עד לקיום התנאי.

```
import java.util.Scanner;
class myDiv {
    public static void main (String[] args){
        Scanner input =new Scanner(System.in);
        boolean sof = false;
        int mana = 0;
        System.out.println("please enter a number");
        int num1=input.nextInt();
        System.out.println("please enter a number");
        int num2=input.nextInt();
        while(sof==false ){
            num1=num1-num2;
            mana=mana+1;
            if( num1< num2)
                sof=true;
        } // end of while
        System.out.println("The mana of " + "num1" + " " + "num2"+" is " +
mana );
    } // end of main
} // end of myDiv
```

משימה

שנה את לולאת ה- while בשיטה main כדי לממש אלגוריתם שמבצע כפל בין שני מספרים ללא שימוש בפעולת כפל

משימה – סכום ספרות של מספר שלם

בפרק 3 כתבנו מחלקה DigitNum , האלגוריתם, כלל פירוק המספר התלת ספרתי לספרותיו. וחישב סכום 3 הספרות.

עתה נשתמש בלולאת while כדי לפתור בעיה כללית יותר.

היישום הבא קולט מספר שלם (אין חשיבות למספר הספרות במספר) ומדפיס כל ספרה במספר בנפרד ואת סכום ספרות המספר.

המחלקה SumDigit

```
import java.util.Scanner;

public class SumDigit {
    public static void main( String [] args ) {
        Scanner input = new Scanner(System.in);
        int i,num,nextDigit;
        int sum = 0;
        System.out.println("Enter the number");
        num= input.nextInt();
        while(num>0)
        { nextDigit= num % 10;
          sum=sum+nextDigit;
          num = num / 10;
          System.out.println(nextDigit);
        } // end of while
        System.out.println();
        System.out.println("The sum of the even number is :"+ sum ) ;
    } // end of main
} // end of SumDigit class
```

הרעיון המרכזי של האלגוריתם הוא חלוקה ב-10 בכל מחזור של הלולאה, ובכל מחזור בודקים האם המספר < 0, אם לא הסתיים פירוק המספר לספרותיו.

המחלקה MyTest

טל תלמיד מדעי המחשב החליט לסייע לאחיו הלומד בכיתה א' ללמוד ולשנן את לוח הכפל. טל החליט לכתוב שיטה שתציג בפני אחיו תרגילי כפל, תקלוט את תשובתו ותבדוק האם ענה נכון. טל החליט שהתוכנית תתרגל את אחיו 10 תרגילים בכל פעם שתופעל ובסיום יוצג הציון אותו השיג אחיו.

```
import java.util.Scanner;

public class MyTest {
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        System.out.println("Finel Math test");
        int good=0;
        for(int exe=1;exe<=10;exe++){
            int x=(int)(Math.random()*10)+1;
            int y=(int)(Math.random()*10)+1;
            System.out.println(x+"*"+y+"=? ");
            System.out.println("your answer ");
            int a=nextInt();
            if(a==x*y){
                System.out.println("right answer ");
                good++;
            }
            else
            {
                System.out.println("worng answer");
            }
            System.out.println("Your mark is "+good*10);
        }
    }
}
```

הסבר:

באמצעות השיטה random של המחלקה Math אנו "מייצרים" שני מספרים הכלולים בכל תרגיל.

```
int x=(int)(Math.random()*10)+1;
```

```
int y=(int)(Math.random()*10)+1;
```

שים לב! האופרטור (int) מבצע casting כלומר שינוי הערך שנוצר מסוג **double** לערך מסוג שלם **int**.

התשובה של המשתמש (זוכרים אחיו הצעיר של טל) נבדקת ואם התשובה נכונה מתבצע עדכון של מונה התשובות נכונות , וטל לא שכח לדאוג להודעה מעודדת המוצגת על המסך.

```
if(a==x*y){
```

```
    System.out.println("right answer ");
```

```
    good++;
```

```
}
```

משימה

שנה את השיטה כך שיוצגו תרגילי חיסור או חיבור. כמו כן שטווח המספרים יהיה שונה מהטווח 1-10 (לטל יש אח יותר גדול והוא צריך לתרגיל כפולות של מספרים דו ספרתיים)

המחלקה LetDigit

```
public class LetDigit {  
    public static void main(String [] args){  
        int i=0;  
        for(char ch='A';ch<='Z';ch++){  
            System.out.print(ch);  
            System.out.print(i);  
            i++;  
            if(i==9)  
                i=0;  
        }  
    }  
}
```

השיטה main במחלקה LetDigit מדגימה שימוש במשתנה מסוג char כמשתנה הבקרה של הלולאה. משתנה הלולאה ch מתחיל בערך A ומסתיים בערך Z.
שים לב! האופרטור ++ שמשמעות פעולתו על משתנה מסוג שלם, מבצע פעולה מקבילה על משתנה מסוג תו.
פלט השיטה הוא שילוב של הדפסת תו ולאחריו ספרה 0-9.
פלט השיטה:

AoB1C2D3E4F5G6H7I8JoK1L2M3N4O5P6Q7R8SoT1U2V3W4X5Y6Z7

המחלקה FindMaxPlace

השיטה main במחלקה זו מיישמת אלגוריתם לקליטת קבוצת מספרים, מציאת המספר הגדול בסדרת הקלט ומיקומו של המספר הגדול ביותר בסדרת הקלט. דוגמא של קלט:

23

21

45

34

11

הפלט של השיטה:

המספר הגדול ביותר בסדרה הוא: 45

מיקום המספר הגדול ביותר בסדרה הוא: 3

```
import java.util.Scanner;
public class FindMaxPlace {
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        double Max;
        double First=input.nextInt();
        Max=First;
        int PlaceOfMax =1;
        int n=10;
        for(int i=2;i<=n;i++){
            double Next=input.nextInt();
            if (Next>Max){
                Max=Next;
                PlaceOfMax=i;
            }
        }
        System.out.println("The big number is: " + Max);
        System.out.println("The place is: " + PlaceOfMax);
    }
}
```

שים לב!

לפני הלולאה נקלט המספר הראשון בסדרה, ערכו מושם כערך הגדול ביותר (עד כה) והערך 1, מושם במשתנה המייצג את מיקום הערך הגדול ביותר

```
double First=input.nextInt();
```

```
Max=First;
```

```
int PlaceOfMax =1;
```

לולאה מרכזית המבצעת קליטת שאר המספרים בסדרה ובדיקה של כל מספר האם הוא המספר הגדול ביותר.

```
for(int i=2;i<=n;i++){
```

```
    double Next=input.nextInt();
```

```
    if (Next>Max){
```

```
        Max=Next;
```

```
        PlaceOfMax=i;
```

```
}
```

משחק "נחש את המספר"

המשחק הבא מביא לידי ביטוי נושאים רבים שנלמדו עד כה, וגם מהנה. זו ההזדמנות ליישם את מה שלמדנו ואף להשתעשע.

תיאור המשחק

המחשב "חושב" על מספר ומייצר מספר אקראי בטווח (1-100), ואתה מנסה לנחש את המספר. המשחק מסתיים כאשר הצלחת לנחש את המספר. כדי שהניחושים שלך לא יהיו "פרועים" לחלוטין, התוכנית מסייעת בהנחיה. אם המספר שניחשת יותר גדול מהמספר שיש לנחש, אתה מקבל הודעה שהמספר גדול מדי. אם הוא קטן אתה מקבל הודעה שהמספר קטן. במידה והינך מצליח, לנחש הודעת ברכה מקדמת אותך וגם מידע על מס' הניחושים שנדרשו לך כדי להגיע למספר.

אלגוריתם התוכנית

1. מספר = הגרל מספר()
2. מספר_נמצא = שקר
3. מספר_ניחושים = 0
4. כל עוד מספר_נמצא ≠ שקר
 - 4.1 קלוט(מספר_שחקן)
 - 4.2 אם מספר_שחקן = מספר
 - 4.2.1 הצג (" יפה מאוד " + מס_נחושים)
 - 4.2.2 מספר_נמצא = אמת
 - 4.3 אחרת
 - 4.3.1 אם מספר_שחקן < מספר הצג("ניחושך גדול מדי") אחרת הצג ("ניחושך קטן מדי")

במחלקה game שתי שיטות: השיטה main והשיטה creatNum(). השיטה creatNum() שמזומנת מתוך השיטה main, היא שיטה של המחלקה, ולכן היא static. השיטה יוצרת מספר אקראי ומחזירה אותו ל- main שזמנה אותה. מספר זה הוא המספר "של המחשב", ואותו המשחק צריך לנחש. השיטה main מיישמת את האלגוריתם הכולל לולאת while שמסתיימת כאשר המשחק מנחש את המספר.

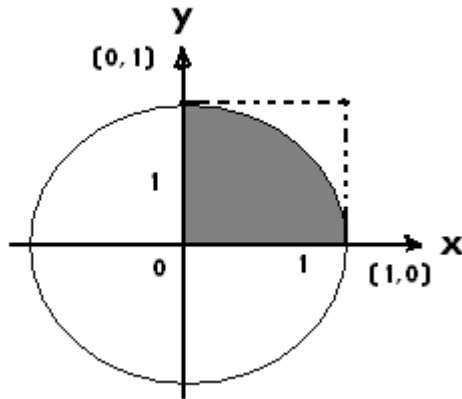
המחלקה game – משחק נחש את המספר

```
import java.util.Scanner;
class game { // start game
    public static int creatNum() { // create num
        return (int)(Math.random() *100 +1);
    } // end create num
    public static void main(String args[]) { /// start main
        Scanner input = new Scanner(System.in);
        int compnum,playnum,myTry=0;
        boolean found = false;
        compnum= creatNum();
        while (!found) {
            myTry++;
            System.out.println("please enter your number " + " attempt #" +
myTry);
            playnum= input.nextInt();
            if(playnum== compnum)
            {
                found=true;
                System.out.println(" very good you got the number after " + myTry +
                "attempts" );
            }
            else
            {
                if ( playnum < compnum )
                    System.out.println( playnum + " got it lower, enter a bigger number ");
                else
                    System.out.println( playnum + " too high! Enter a smaller number ");
            }
        } // end while
    } // end main
} // end game
```

המחלקה MonteCarlo

שיטת מונטה קרלו היא שיטה לחישוב מקורב של ערך π , השיטה מבוססת על רעיון הבא:

נתון מעגל שרדיוסו יחידה (רדיוס = 1), שטח המעגל הוא בדיוק π (3.14.....) בדוק!



שטח רבע עיגול הוא $\pi/4$ בדוק! .

עשה מבצעים ניסוי: "יורים" באמצעות חץ לתוך הריבוע שצלעו 1. החץ הפוגע בריבוע יכול לפגוע בתוך רבע העיגול או מחוץ לרבע העיגול. עשה נייר N חצים ונמנה כמה פעמים החץ פגע בתוך רבע העיגול. יחס הפגיעות בתוך רבע העיגול לבין

מספר נסיונות הירי הוא בדיוק כמו יחס השטחים בין רבע העיגול שרדיוסו 1, לבין הריבוע שצלעו 1. יחס השטחים הוא בדיוק $\pi/4$ (מדוע?).

דוגמא:

נניח ש"ירינו" 1000 חצים ופגענו 350 פעמים בתוך רבע העיגול, היחס של הפגיעות למספר הנסיונות הוא 350/1000 וזה ערך מקורב ל- $\pi/4$.

ככל שנירה יותר חצים נגיע לערך מקורב יותר של $\pi/4$. ערך זה מוכפל ב-4 ונקבל את הערך מקורב של π .

כיצד "יורים" חצים? כיצד מממשים את האלגוריתם המיוחד הזה?

המחלקה monteCarlo, מממשת את הרעיון ועבור מספר נסיונות גבוה מגיעים לקירוב טוב של π (דיוק עד המקום השלישי אחרי הנקודה)

עבור מס' זריקות = 100,000 ערך π The value of pi is: 3.14324 שהתקבל:

מימוש ירי של חצים לתוך הריבוע:

```
x=Math.random();
y=Math.random();
```

בדיקה האם החץ פוגע בתוך רבע העיגול:

```
if(Math.sqrt(x*x+y*y)<=1)
```

המחלקה monteCarlo – המימוש המלא.

```
import java.util.Scanner;
public class MonteCarlo
{
    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        double x,y,pi;
        long hits,times;
        hits=0;
        System.out.println("How many times you want to throws? ")
        times=input.nextInt();
        for(int i=1;i<=times;i++){
            x=Math.random();
            y=Math.random();
            if(Math.sqrt(x*x+y*y)<=1)
                hits++;
        }
        pi=4*((double)hits/times);
        System.out.println("The value of pi is: "+ pi);
    }
}
```

שים לב!

המשתנים hits ו- times הוגדרו מסוג **long** זאת כדי נוכל להריץ את התוכנית עם ערכים שלמים גדולים יותר מהייצוג באמצעות **int**.

לולאות מקוננות

רעיון לולאה מקוננת הוא שילוב של לולאה בתוך לולאה (ויותר), הרעיון אינו חידוש בפקודות אלא שילוב בין מבנים שתיארנו ומפאת החשיבות ותדירות השימוש ראוי להסבירו. הרעיון אינו ממומש רק באמצעות **for** בתוך **for**, אלא כול שילוב בין סוגי הלולאות בשפת ג'אווה כולל: **for** מקונן

for {

statement

for {

statement

}

statement

}

while מקונן

while {

statement

while {

statement

}

statement

}

שילוב לולאת while ולולאת for

while {

statement

for {

statement

}

statement

}

המחלקה lokefel - יצירת לוח כפל

```
class lokefel {  
import java.util.Scanner;  
class lokefel {  
    public static void main (String args[]){  
        Scanner input = new Scanner(System.in);  
        int n,m,i,j;  
        System.out.println("please enter first number ");  
        n= nextInt();  
        System.out.println("please enter second number ");  
        m= input.nextInt ();  
        for (i=1; i<=m; i++) {  
            for (j=1;j<=n; j++) {  
                System.out.print( i*j + " ");  
            }  
            System.out.println();  
        }  
    }  
}
```

המחלקה lokefel מדגימה את הרעיון של קינון לולאות, כלומר לולאה בתוך לולאה.

המחלקה stars

משימה – הדפסת "משולש כוכביות"

ברצוננו להדפיס משולש של כוכביות :

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

הרעיון המרכזי באלגוריתם

בשורה הראשונה אנו מדפיסים כוכבית אחת

בשורה השנייה אנו מדפיסים 2 כוכביות

מספר הכוכביות גדל ב-1 בכל שורה

האלגוריתם הזה ידפיס 10 שורות, ניתן לשנות את מספר השורות.

```
class stars {  
    public static void main( String [] args ) {  
        final int max_row = 10;  
        int row;  
        for ( row=1; row<= max_row; row++){  
            for (int star=1 ; star<= row; star++){  
                System.out.print ("*");  
                System.out.println();  
            } // end of for  
        } // end of main  
    } // end of stars
```

המחלקה printStar – הדפסת מלבן כוכביות

מחלקה זו מהווה "שיפור" של המחלקה star, השיטה main מאפשרת קליטת "אורך" מלבן הכוכביות, ו"רוחב" מלבן הכוכביות.

```
import java.util.Scanner;
public class printStar {
    public static void main( String [] args ) {
        Scanner input= new Scanner(System.out.println());
        int a,b,i,j;
        System.out.println("please enter width ");
        a= input.nextInt();
        System.out.println("please enter length ");
        b= input.nextInt();
        for (i=1; i<=a; i++)
            { for (j=1; j<=b; j++)
                System.out.print("*");
                System.out.println(); }
    } // end of main
} // end of printStar class
```

המחלקה ZipCode

טל (שוב טל) בקש לשלוח מכתב לדודתו, טל לא זכר את מספר תיבת הדואר, אולם, זכר שהמספר היה תלת ספרתי, ספרת העשרות ערכה היה 5, וספרת המאות ערכה גדול ב-2 מספרת היחידות וסכום ספרות המספר הוא 17. טל החליט לכתוב שיטה שתסייע לו להגיע למספר על בסיס הרמזים שזכר. המחלקה ZipCode שכתב טל אכן מסייעת לו למצוא את המספר.

```
public class ZipCode {
    public static void main (String [] args){
        for(int i=1; i<=9;i++)
            for(int j=0;j<=9;j++)
                for(int k=0;k<=9;k++){
                    if (j==5)
                        if(i-2==k)
                            if(i+j+k==17)
                                System.out.println(i*100+j*10+k);
                }
            }
    }
}
```

הסבר:

בשיטה main לולאה מקוננת (בעצם 3 לולאות) המשתנה i בלולאה החיצונית ערכו ההתחלתי 1 וערכו הסופי 9, משתנה זה מייצג את ספרת המאות. המשתנה j בלולאה הפנימית ערכו ההתחלתי 0 וערכו הסופי 9, משתנה זה מייצג את ספרת העשרות. המשתנה k בלולאה "היותר" פנימית ערכו ההתחלתי 0 וערכו הסופי 9. משתנה זה מייצג את ספרת היחידות. מבנה הלולאות שיצרנו "מייצר" את כל המספרים התלת ספרתיים מהמספר 100 ועד למספר 999. כל מספר נבדק לפי הרמזים שזכר טל. הפלט של התוכנית: 755, ואכן זה המספר התלת ספרתי היחיד המקיים את התנאים: ספרת העשרות 5, ספרת המאות גדולה ב-2 מספרת היחידות והתנאי האחרון סכום הספרות במספר הוא 17.

משימה

כתוב שיטה המבצעת את אותה מטלה, אולם עליך להשתמש רק בלולאה אחת. רמז: במקום לבנות את המספר התלת ספרתי מספרות שונות, בצע לולאה מהמספר 100 ועד 999 ופרק את המספר לספרותיו.

המחלקה pascalTriangle – הדפסת משולש פסקל

השיטה main במחלקה זו מדפיסה את משולש פסקל לפי ערך המתקבל מהקלט.
עבור קלט 6 הפלט יהיה המשולש הבא:

```

1
121
12321
1234321
123454321
12345654321

```

השיטה main מורכבת במחלקה זו ממספר לולאות, לולאה מרכזית עבור מספר שורות, ולולאות עבור הדפסת רווחים תחילת השורה, ובסוף השורה והדפסת המספרים באמצע השורה. מאוד רצוי לעקוב אחר השיטה main באמצעות סביבת Jeliot

```

import java.util.Scanner;
class pascalTriangle {
    public static void main( String [] args ) {
        Scanner input = new Scanner(System.in);
        int n,i;
        System.out.println("please enter your number ");
        n=input.nextInt();
        for (i=1; i<=n; i++) { // The big for
            for( int j=1; j<=n-i; j++){
                System.out.print(" ");}
            for ( int j=1; j<i; j++){
                System.out.print(j); }
            System.out.print(i);
            for ( int j=1; j<i; j++){
                System.out.print(i-j); }
            System.out.println();
        } // end of the big for
    } // end of main
} // end of class

```

לולאות וצבים

המחלקה Turtle שנקתבה באוניברסיטה העברית מזמנת לנו חוויה תכנותית. האובייקטים שניתן ליצור באמצעות מחלקה זו הם "צבים קטנים וחמודים", לזנבם של הצבים "מחובר עפרון". ניתן ליצור אובייקט צב ולהשתמש בשיטות הרבות שיש במחלקה.

יצירת אובייקט של המחלקה Turtle:

```
Turtle t= new Turtle ();
```

שיטות שהוגדרו במחלקה Turtle

- הרם זנב - tailup()
 - הורד זנב - taildown()
 - התקדם קדימה - moveForward(double)
 - התקדם אחורה - moveBackward(double)
 - פנה ימינה - turnLeft(double)
 - פנה אחורה - turnRight(double)
 - שנה צבע - setTailColor(String)
 - העלם את הצב - setVisible(false);
 - הצג את הצב - setVisible(true);
- לחלק מהשיטות יש ארגומנטים ולחלק אין. המחלקה Turtle מעניינת וצפויה לכל לומד ג'אוה חוויה אמיתית. שילוב לולאות עם "צבים" מספק פלט מעניין ונאה.

המחלקה turtle1

```
import TurtleLib.Turtle;
```

```
public class turtle1 {
```

```
    public static void main ( String [] args){
```

```
        Turtle t= new Turtle ();
```

```
        t.tailDown();
```

```
        int stp =10;
```

```
        for(int i=1; i<=30;i++){
```

```
            t.moveForward(stp);
```

```
            t.turnLeft(90);
```

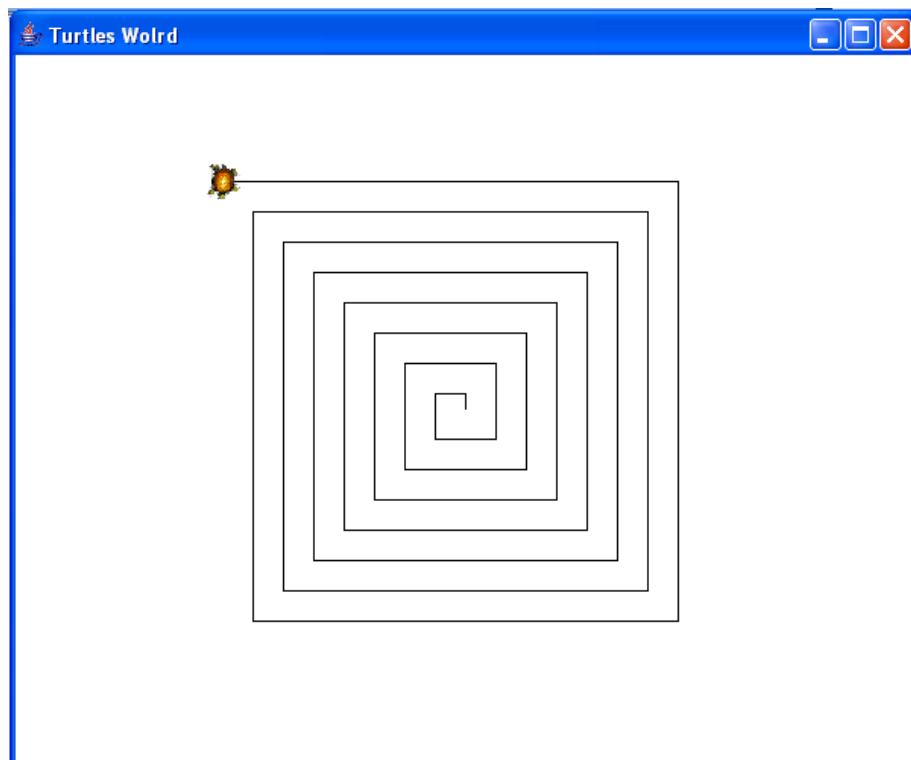
```
            stp=stp+10; }
```

```
        }
```

```
    }
```

שים לב למשפט "יבוא המחלקה" - import Turtlelib.Turtle

פלט הרצת המחלקה: turtle1



הסבר:

בתוכנית לולאה בה בכל מחזור מבצע הצב צעד ופניה שמאלה, הצעד גדל בכל מחזור של לולאה וכך אנו מקבלים "שבול".

המחלקה turtle2

```
import TurtleLib.Turtle;
public class turtle2 {
    public static void main ( String [] args){
        Turtle t1= new Turtle ();
        t1.tailDown();
        Turtle t2= new Turtle ();
        t2.tailDown();
        int i;
        i = 0;
        t1.moveForward(20);
        for (i=1;i<=30;i++){
            t2.turnLeft((int)(Math.random()*20));
            t1.turnLeft((int)(Math.random()*20));
            t2.moveForward((int)(Math.random()*20));
            t1.moveForward((int)(Math.random()*20));
        }
    }
}
```

פלט המחלקה turtle2

הרצת המחלקה מציגה יצירת שני
אובייקטים מסוג Turtle שעסוקים
במרדף.

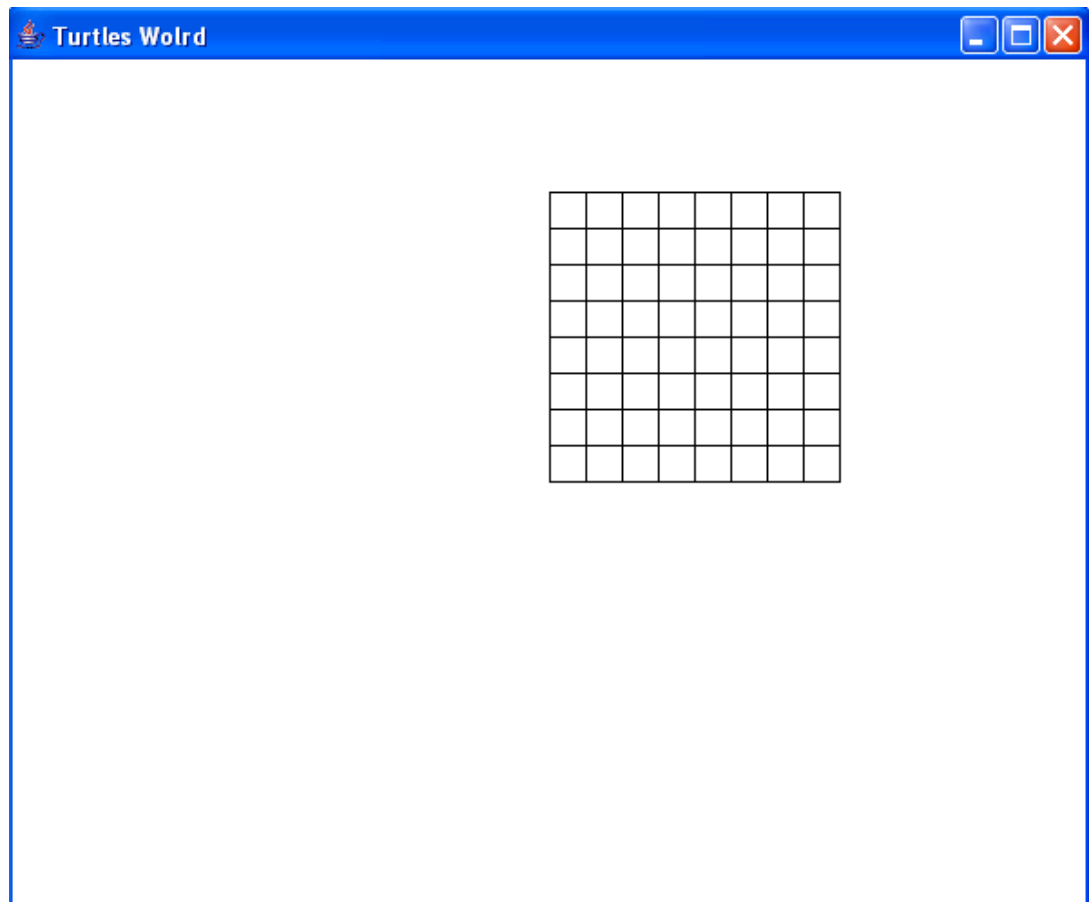


המחלקה turtle3

```
import TurtleLib.Turtle;
public class turtle3 {
    public static void main ( String [] args){
        Turtle t1= new Turtle ();
        t1.tailDown();
        int i,j;
        i=j=0;
        for(i=1;i<=8;i++){
            for(j=1;j<=8;j++){
                for(int k=1;k<=4;k++){
                    t1.moveForward(20);
                    t1.turnRight(90);
                }
                t1.tailUp();
                t1.tailDown();
                t1.turnRight(90);
                t1.moveForward(20);
                t1.turnLeft(90);
            }
            t1.turnLeft(90);
            t1.moveForward(160);
            t1.turnRight(90);
            t1.moveForward(20);
        }
        t1.setVisible(false);
    }
}
```

המחלקה try3 ממחישה שימוש בלולאות מקוננות. לולאה פנימית המציירת ריבוע, לולאה חיצונית ללולאה זו שתפקידה למקם את הצב במיקום חדש ולצייר שורת ריבועים. ולולאה חיצונית שמבצעת בכל מחזור את ההוראות לציור שמונה ריבועים. המחלקה ממחישה באופן ויזואלי ויפה את מבנה לולאות מקוננות, חשיבות המיקום של הוראות בתוך לולאה, לפני הלולאה, בתום הלולאה.

פלט המחלקה try3



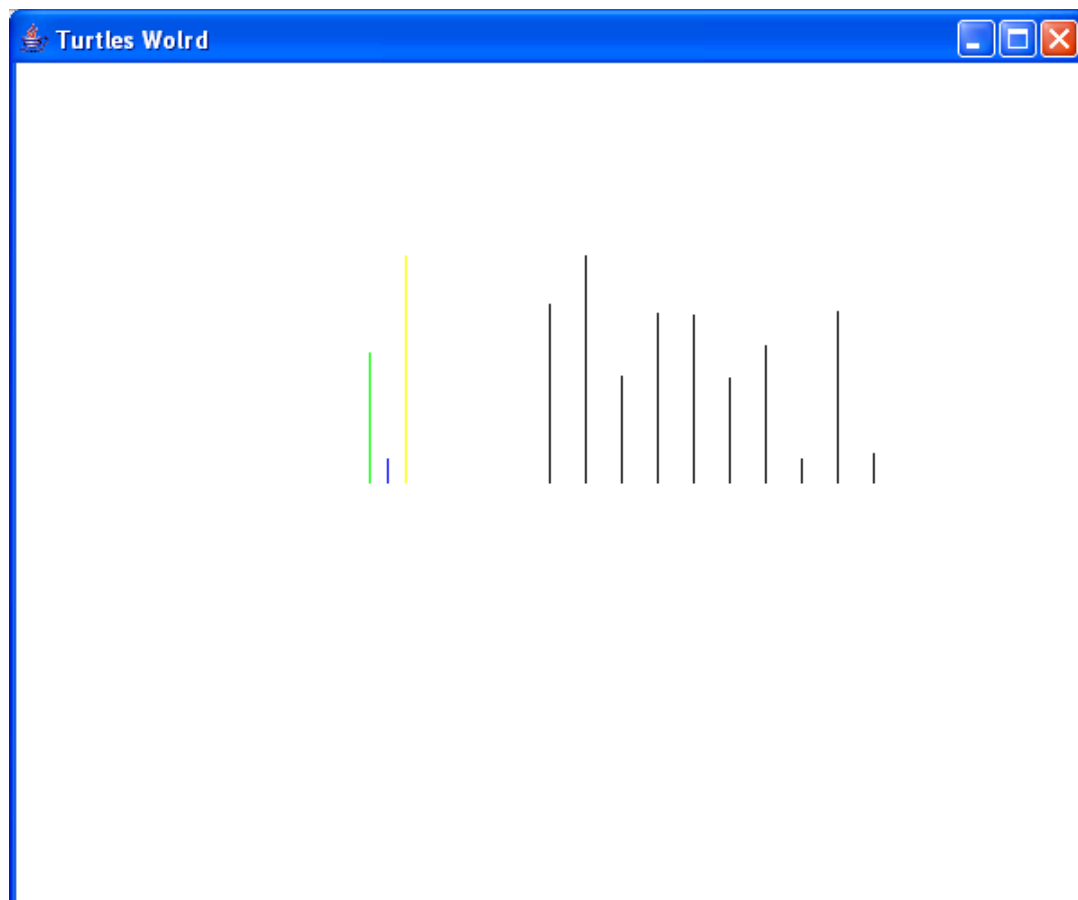
המחלקה turtle4

המחלקה turtle4 ממחישה בצורה ויזואלית את האלגוריתם לחישוב ממוצע, מקסימום ומינימום. הקוד יוצר 10 מספרים אקראיים אשר מוצגים בצורת דיאגרמה על חלון הפלט. הקוד מחשב את המינימום, המקסימום והממוצע של 10 המספרים ומציג את הביטוי הגרפי של גודל מספרים אלו.

```
import TurtleLib.Turtle;
public class turtle4 {
    public static void main ( String [] args){
        Turtle t1= new Turtle ();
        t1.tailDown();
        int i=0,j=0;
        double num=Math.random()*120+10;
        double avg=num,sum=0,big=num,min=num;
        for (i=1;i<=10;i++){
            t1.tailDown();
            t1.moveForward(num);
            t1.moveBackward(num);
            t1.tailUp();
            t1.turnRight(90);
            t1.moveForward(20);
            t1.turnLeft(90);
            num=Math.random()*120+10;
            if (num<min) min=num;
            if (num>big) big=num;
            sum=sum+num;
        }
        t1.setTailColor("red");
        t1.turnLeft(90);
        t1.tailUp();
        t1.moveForward(300);
        t1.turnRight(90);
        t1.tailDown();
        avg=sum/10;
    }
}
```

```
t1.setTailColor("green");
t1.moveForward(avg);
t1.moveBackward(avg);
t1.tailUp();
t1.turnRight(90);
t1.moveForward(10);
t1.turnLeft(90);
t1.tailDown();
t1.setTailColor("blue");
t1.moveForward(min);
t1.moveBackward(min);
t1.tailUp();
t1.turnRight(90);
t1.moveForward(10);
t1.turnLeft(90);
t1.tailDown();
t1.setTailColor("yellow");
t1.moveForward(big);
t1.moveBackward(big);
t1.tailUp();
t1.turnRight(90);
t1.moveForward(10);
t1.turnLeft(90);
t1.tailDown();
t1.setVisible(false);
    }
}
```

turtle4 פלט המחלקה



המחלקה fibonacciGraph

המחלקה הזו יוצרת את האיברים בסדרת פיבונצ'י ומציגה גרסה גרפית של הסדרה, זאת ע"י ציור קווים באורך הערך של האיבר.

ערך האיבר ה- a_n בסדרה פיבונצ'י מוגדר לפי הנוסחה הבאה:

$$a_n = a_{n-1} + a_{n-2}$$

המחלקה fibonacciGraph יוצרת עבור איברי הסדרה שערך האיבר הראשון $a_1=1$ והאיבר השני $a_2=1$ את האיברים הבאים בסדרה (בתוכנית חישוב 10 איברי הסדרה) כדי להציג את התנהגות הסדרה בחרתי "לצייר" את איברי הסדרה זאת באמצעות שיטות המחלקה Turtle.

משימות:

שנה את התוכנית כך שתיצור סדרה אחרת ותצייר את אברי הסדרה , דוגמאות לסדרות:

1,2,3,4,5,6,7,8,8,10

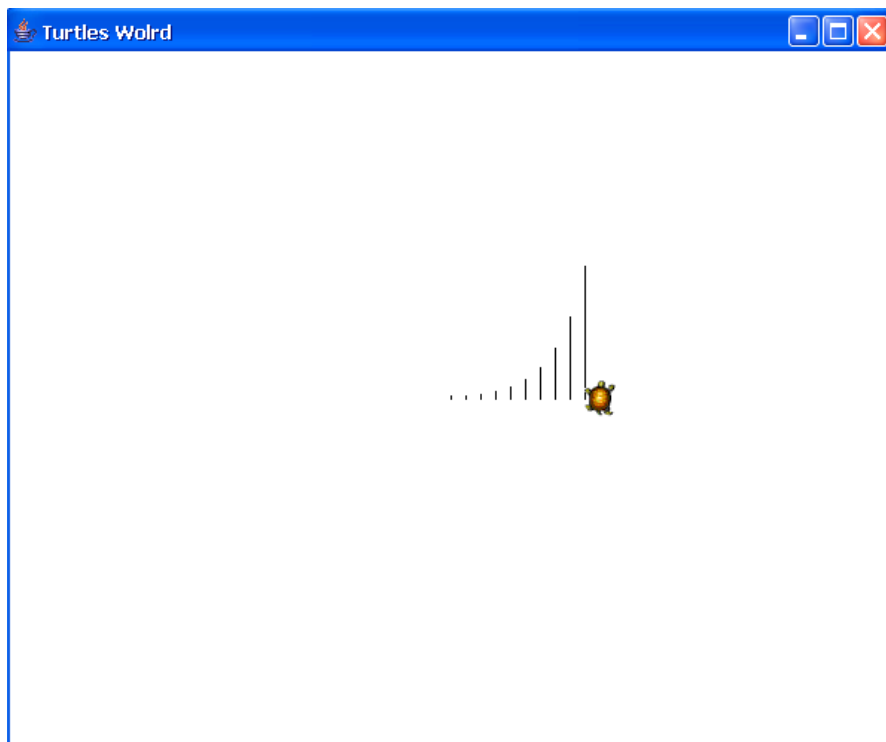
2,4,6,8,10,12,14,16

1,4,9,16,25,36,49,64,100

```
import unit4.turtleLib.Turtle;
import unit4.ioLib.*;
public class fibonacci{
    public static void main(String[] args) {
        Turtle t=new Turtle();
        t.tailDown();
        int a1=1;
        int a2=1;
        int next;
        next=a1+a2;
        for(int i=1;i<=10;i++){
            t.moveForward(next);
            t.tailUp();
            t.moveBackward(next);
            t.turnRight(90);
            t.moveForward(10);
            t.turnLeft(90);
            next=a1+a2;
```

```
a1=a2;  
a2=next;  
t.tailDown();  
}  
}  
}
```

פלט התוכנית:



תבניות

כאשר אנו בוחנים בעיות שונות ואת האלגוריתמים שפתחנו וממשנו לעיתים אנו רואים שהפתרונות "דומים", חולקים את אותו מבנה לצורך המחשה הרעיון נניח שנתונות שתי הבעיות הבאות:

בעיה א

לידור קיבל מאביו דמי כיס במשך שנה. כל סוף חודש קבל לידור דמי כיס לא קבועים (בהתאם להישגיו בלימודים...) פתח וממש אלגוריתם שיקלוט את סכום דמי הכיס שקיבל לידור בכל חודש ויחשב וידפיס את סכום דמי הכיס שקיבל לידור במהלך כל השנה.

בעיה ב

לידור החליט לשפר את כושרו הגופני והחליט ללכת בכל יום בשעת בוקר מוקדמת מספר דקות. לידור התמיד בכך 10 ימים ורשם בכל יום את המרחק שהלך. פתח וממש אלגוריתם שיקלוט את מס' הקמ' שהלך לידור בכל יום ויחשב וידפיס כמה קמ' הלך לידור במשך 10 ימי ההליכה.

המשותף לשתי הבעיות

בשתי הבעיות אנו נדרשים לחשב סכום, הרעיון הבסיסי של האלגוריתם הוא ביצוע לולאה בה כלול משפט קלט וסכימה של המספר הנקלט לסכום. הסכום ההתחלתי הוא כמובן 0 (למה?). ההבדל בין שתי הבעיות נעוץ בסיפור הכללי, במספר הקלטים. הבדלים נוספים בין הבעיות יבוא לידי ביטוי בצורת הקלט לעיתים המספרים נקלטים ולעיתים נוצרים באמצעות השיטה ליצירת מספרים אקראיים. הבדל חשוב יבוא לידי ביטוי אם הבעיה כוללת סימן שאלה לגבי כמות המספרים בקלט, בעיה עם מספר ידוע מראש של מספרים בקלט תמומש באמצעות לולאת for, בעיה עם מספר לא ידוע של מספרים בקלט תמומש באמצעות לולאת while כל הבעיות שהאלגוריתם המפותח הוא **סכום** ניתן לאחד תחת "תבנית" כללית שהיא **תבנית סכום**.

שאלה

למה חשוב לזהות מהי התבנית המסתתרת בפתרון הבעיה?

תשובה

זיהוי התבנית שעומדת בבסיס פתרון הבעיה מסייע לפתור את הבעיה ולנסח את אלגוריתם ובשלב שני לממש את הפתרון. אולם, חשוב לא לאמץ את הרעיון של זיהוי תבנית הפתרון כשיטה לפתרון בעיות כוללת. לעיתים יש לפתח אלגוריתם מקורי, רעיון "חדש" ולא שגרתי ולכן לצד שימוש בתבניות ככלי עזר בפתרון בעיות אל תשכחו להיות "יצירתיים".

תבנית הסכום

סכום = 0

בצע n פעמים

$x =$ קלוט את המספר הבא // ניתן גם ליצור ערכים אקראיים או סריקה ממערך נתון

סכום = סכום + x

סוף

הדפס סכום

```
import java.util.Scanner;
public class BasicSum {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("How many number");
        int n=input.nextInt();
        double sum=0,avg,num;
        for(int i=1;i<=n;i++){
            System.out.println("Enter the next number ");
            num=input.nextDouble();
            sum=sum+num;
        }
        System.out.println("The sum is " + sum);
    }
}
```


תבנית צבירת מכפלה

multy = ערך התחלתי // שים לב ערך זה חייב להיות שונה מאפס (למה!?)

בצע n פעמים

x = קלוט ערך חדש

multy = multy * x

סוף

הדפס (multy)

```
import java.util.Scanner;
public class Multy {
    public static void main(String[] args){
        Scanner input= new Scanner(System.in);
        System.out.println("      How many number ");
        int n=input.nextInt();
        double finel=1;// The value set to 1 but can any value but not zero
        double num;
        for(int i=1;i<=n;i++){
            System.out.println("Enter the next number ");
            num=input.nextDouble();
            finel=finel*num;
        }
        System.out.println("The finel is " + finel);
    }
}
```

תבנית מניה

מונה = 0

בצע n פעמים

x = קלוט את המספר הבא // ניתן גם ליצור ערכים אקראיים או סריקת מערך נתון

אם x מקיים את התנאי (_____)

אז מונה = מונה + 1

הדפס מונה

```
import java.util.Scanner;
public class BasicCount{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println(" How many number " );
        int n=input.nextInt();
        int num_ok=0;
        double num;
        for(int i=1;i<=n;i++){
            System.out.println("Enter the next number ");
            num=input.nextDouble();
            if (num>56)
                num_ok++;
        }
        System.out.println(" There is " + num_ok+" greater then 56");
    }
}
```

תבנית ממוצע

סכום = 0

בצע n פעמים

num = קלוט את המספר הבא

סכום = סכום + num

סוף

ממוצע = סכום / n

```
import java.util.Scanner;
public class BasicAvg{
    public static void main(String[] args){
        Scanner input= new Scanner(System.in);
        System.out.println("How many numbers ");
        int n = input.nextInt();
        double sum=0,avg,num;
        for(int i=1;i<=n;i++){
            System.out.println("Enter the next number ");
            num = input.nextDouble();
            sum=sum+num;
        }
        avg=sum/n;
        System.out.println("The average is " + avg);
    }
}
```

כל התבניות זהות במקרה של לולאת while ההבדל היחיד הוא באיזה מקרה הלולאה מסתיימת וישנם מקרים שונים אותם אדגים:

תבנית ממוצע לולאת while

```
import java.util.Scanner;
public class CountWhile{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        int num_ok=0;
        double sum=0,num;
        boolean more_number=true;
        while(more_number){
            System.out.println("Enter the next number ");
            num=input.nextDouble();
            if (num>=0){
                if (num>56)
                    num_ok++;
            }
            else more_number=false;
        }
        System.out.println("The counter of the right number is " + num_ok);
    }
}
```

שים לב!

אין מידע ללולאת הבקרה כמה מספרים יש בקלט.

הלולאה ממשיכה כל עוד המספרים שקולטים עומדים בתנאי עצירת הלולאה במקרה

הזה כל עוד המספר שנקלט גדול מ-0

תבנית מניה לולאת while

```
import java.util.Scanner;

public class WhileCountAndAvg{
    public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        double sum=0,num,n=0;
        boolean more_number=true;
        while(more_number){
            System.out.println("Enter the next number ");
            num=input.nextDouble();
            if (num>=0){
                sum=sum+num;
                n++;
            }
            else more_number=false;
        }
        if (n>0)
            System.out.println("The avarge is " + sum/n);
    }
}
```

שים לב!

כמות המספרים החוקית נקבעת ל- $n=0$

אם המספר שנקלט "חוקי" מסכמים אתו לסכום ומוסיפים 1 למונה המספרים שנקלטו.

שאלה

מדוע בודקים האם $n>0$ לפני שמחשבים את הממוצע?

תרגיל :

התרגיל הבא משתמש בתבנית מניה.

מטרת התרגיל לספור כמה פעמים מופיעה האות A במחרוזת (קבוע מחרוזת בתוכנית או הנקלטת מהקלט)

```
public class HowManyA{
    public static void main(String[] args){
        String str = new String();
        str="MAKE LOVE NOT WAR";
        int counterA = 0;
        for(int i=0;i<=str.length()-1;i++){
            if (str.charAt(i)=='A')
                counterA++;
        }
        System.out.println("In the string "+ str +" We have "+ counterA+ " A" );
    }
}
```

נסכם כמה רעיונות:

בתבנית סכום או מניה המונה או הצובר - המשתנה המכיל את הסכום מקבלים ערך 0 לפני תחילת הלולאה המרכזית.

התרגיל הבא מציג שימוש בתבנית סכום ותבנית מניה. הרעיון הבסיסי של תבניות אלו מיושם על בעיה הכוללת 3 סכומים ו-3 מונים.

דוגמא:

בחנות כלבו 3 מחלקות:

מחלקת הלבשה – מחלקה מס' 1

מחלקת מזון – מחלקה מס' 2

מחלקת כלי בית – מחלקה מס' 3

בסוף כל יום מנהל הסופר מכניס למחשב את המכירות לפי מחלקות ומעוניין לקבל

בסוף כל יום עבודה עבור כל מחלקה את סכום הפדיון של המחלקה וכמה מכירות

התבצעו בכל מחלקה.

שאלה

כמה מכירות בסה"כ יש בכל יום?

נממש את התרגיל בשתי צורות:

צורה א' : שיודעים מראש כמה מכירות יש

צורה ב' : שכאשר קולטים מס' מחלקה = 0 מפסיקים את הקלט.

נממש את התבנית עם מספר ידוע מראש של מכירות:

```

import java.util.Scanner;
public class BasicDepartment{
    public static void main(String[] args){
        Scanner input = new Scanner (System.in);
        double sumdep_1=0,sumdep_2=0,sumdep_3=0;
        int sales_1=0,sales_2=0,sales_3=0;
        int num_of_sales=10;
        for(int i=1;i<=num_of_sales;i++){
            System.out.println("Enter num of dep ");
            int dep=input.nextInt();
            System.out.println("Enter the payment ");
            double sale= input.nextDouble();
            switch(dep){
                case 1: sales_1++; sumdep_1=sumdep_1+ sale; break;
                case 2: sales_2++; sumdep_2=sumdep_2+sale; break;
                case 3: sales_3++; sumdep_3=sumdep_3+sale; break;
            }
        }
        System.out.println("The sum of dep 1 is " + sumdep_1);
        System.out.println("The num of dep 1 is " + sales_1);
        System.out.println("The sum of dep 2 is " + sumdep_2);
        System.out.println("The num of dep 2 is " + sales_2);
        System.out.println("The sum of dep 3 is " + sumdep_3);
        System.out.println("The num of dep 3 is " + sales_3);
    }
}

```

נממש את התבנית עם מספר לא ידוע של מכירות. בדוגמא זו אם מקישים מספר מחלקה שלילי מסתיימת התוכנית.

```

import java.util.Scanner;
public class Department {
    public static void main(String[] args){
        Scanner input= new Scanner(System.in);
        double sumdep_1=0,sumdep_2=0,sumdep_3=0;
        int sales_1=0,sales_2=0,sales_3=0;
        boolean more_sale=true;
        while(more_sale){
            System.out.println("Enter num of dep ");
            int dep=input.nextInt();
            System.out.println("Enter the payment ");
            double sale= input.nextDouble();
            if (dep>=1 && dep<=3){
                switch(dep){
                    case 1: sales_1++; sumdep_1=sumdep_1+ sale; break;
                    case 2: sales_2++; sumdep_2=sumdep_2+sale; break;
                    case 3: sales_3++; sumdep_3=sumdep_3+sale; break;
                }
            }
            else
                more_sale=false;
        }
        System.out.println("The sum of dep 1 is " + sumdep_1);
        System.out.println("The num of dep 1 is " + sales_1);
        System.out.println("The sum of dep 2 is " + sumdep_2);
        System.out.println("The num of dep 2 is " + sales_2);
        System.out.println("The sum of dep 3 is " + sumdep_3);
        System.out.println("The num of dep 3 is " + sales_3);
    }
}

```


תבניות מציאת מינימום ומקסימום

מימוש תבנית מציאת מקסימום עבור מס' ידוע מראש של מספרים

```
import java.util.Scanner;

public class FindMax{

    public static void main(String[] args){
        Scanner input= new Scanner(System.in);
        System.out.println("How many numbers ");
        int n = input.nextInt();
        System.out.println("Enter num ");
        double max = input.nextDouble();
        for (int i=2;i<=n;i++){
            System.out.println("Enter num ");
            double max = input.nextDouble();
            if(num > max)
                max=num; } // end loop
        System.out.println("The max number is "+max);
    }
}
```

מימוש אותה תבנית עבור המספר הקטן ביותר

```
import java.util.Scanner;

public class FindMin{

    public static void main(String[] args){
        Scanner input= new Scanner(System.in);
        System.out.println("How many numbers ");
        int n = input.nextInt();
        double max = input.nextDouble();
        for (int i=2;i<=n;i++){
            System.out.println("Enter num ");
            double max = input.nextDouble();
            if(num < min)
                min=num; } // end loop
        System.out.println("The min number is "+min);
    }
}
```

התבנית מציאת המקסימום \ מינימום כולל מיקום המספר בסדרת המספרים

עבור מקסימום

```
import java.util.Scanner;
public class bigNumber{
    public static void main(String[] args){
        Scanner input= new Scanner(System.in);
        System.out.println("How many numbers ");
        int n = input.nextInt();
        double num;
        System.out.println("Enter num ");
        double max = input.nextDouble();
        int place = 1;
        for (int i=2;i<=n;i++){
            System.out.println("Enter num ");
            num = input.nextDouble();
            if(num > max)
                max=num;
            place=i;
        }
        System.out.println("The biggest5 number is "+max);
        System.out.println("The place of the number is "+place);
    }
}
```

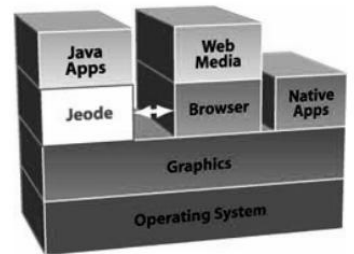
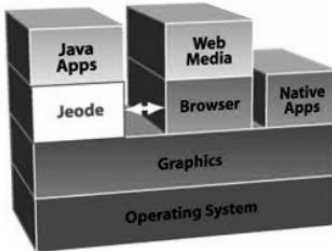
עבור מינימום

```
import java.util.Scanner;
public class smallNumber{
    public static void main(String[] args){
        Scanner input= new Scanner(System.in);
        System.out.println("How many numbers ");
        int n = input.nextInt();
        double num;
        System.out.println("Enter num ");
        double min = input.nextDouble();
        int place = 1;
        for (int i=2;i<=n;i++){
            System.out.println("Enter num ");
            num = input.nextDouble();
            if(num < min)
                min=num;
            place=i;
        }
        System.out.println("The biggest5 number is "+min);
        System.out.println("The place of the number is "+place);
    }
}
```

משימת סיכום שילוב תבניות מציאת מקסימום, מקסימום, ממוצע

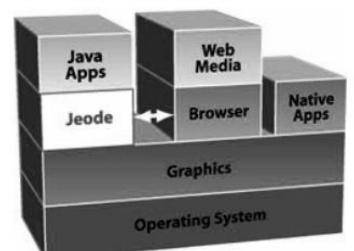
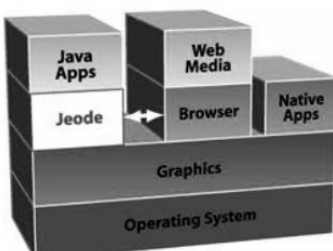
```
import java.util.Scanner;;

public class FindAll {
    public static void main(String [] args ) {
        Scanner input = new Scanner (System.in);
        int grade,count = 0, sum=0, minimum,maximum;
        double avarge;
        System.out.println ("Enter the first grade ( 999 to quit ): ");
        System.out.println("The first number ");
        grade=input.nextInt();
        maximum=minimum=grade;
        while ( grade != 999) {
            count ++;
            sum+= grade;
            if ( grade > maximum )
                maximum= grade ;
            if ( grade < minimum)
                minimum = grade ;
            System.out.print ("Enter the next grade ( 999 to quit ): ");
            System.out.println("Enter the next grade ( 999 to quit ): ");
            grade=input.nextInt();
        } // end of the while loop
        if ( count== 0)
            System.out.println("No valid grades were entered.");
        else
            { avarge = sum / count;
            System.out.println();
            System.out.println("Total number of the students: " + count );
            System.out.println("Average grade: " + avarge);
            System.out.println("Highest grade : " + maximum);
            System.out.println("Lowest grade: " + minimum);        }
    } // end of main
} // end of class
```



פרק שישי – מערכים

הפרק מציג את אופן ההכרזה על מערכים בג'אווה.
שילוב בין מערכים ולולאות המאפשר סריקה של מערכים בצורה יעילה.
התרגיל האחרון, תרגיל הסוכנים מהווה תרגיל מסכם המממש את כל מה שנלמד בספר, כולל:
עצמים, לולאות, מערכים, קלט פלט



פרק שישי – מערכים בשפת ג'אווה

מערך – הגדרה כללית

מערך (array) הוא סדרה של אובייקטים או של נתונים מסוגים בסיסיים. כל הסדרה של אובייקטים ונתונים הם מאותו הסוג, וכלולים "במסגרת" אותו שם. כל מערך בג'אווה שאנו יוצרים הוא **עצם**, ואינו משתנה מטיפוס בסיסי. תאי המערך ממוספרים על פי מיקומם. מיקומו של התא הראשון הוא 0, מיקומו של התא השני הוא 1, וכך הלאה מיקומו של התא האחרון הוא $n-1$. לכל מערך שם המייחד אותו, בדומה, לכל משתנה, שיטה או אובייקט בג'אווה.

הגדרת מערך בג'אווה

ההגדרה של מערך נעשית באמצעות סוגריים מרובעים הנקראים: **אופרטור אינדקס** (indexing operator) `[]`. כדי להגדיר מערך. יש להוסיף סוגריים מרובעים `[]` מיד לאחר שם הטיפוס שהגדרת. דוגמא: הגדרת `a1` כמערך מטיפוס שלם – `int [] data;` אין חשיבות לסדר הופעת הסוגריים המרובעים, וגם הגדרה זו תקפה: `int data []`

סוגים שונים (Type) של מערך

טיפוס המערך	בשפת ג'אווה
מערך של מספרים שלמים	<code>int[]</code>
מערך של תווים	<code>char[]</code>
מערך של מספרים עשרוניים	<code>double[]</code>
מערך של ערכים בוליאנים	<code>boolean[]</code>
מערך של עצמים מסוג String	<code>String[]</code>
מערך של עצמים מסוג Point	<code>Point[]</code>

שאלה

מה גודל המערך, כלומר כמה תאים מכיל המערך?

תשובה

ג'אוה מאפשרת הגדרת מערך והשמת ערכים באותה הוראה. `int [] num = {9,5,0,3};` ההגדרה גם מכריזה על המערך, גם מבצעת השמה של ערכים במערך והפעולה מגדירה בכך את גודל המערך – 4 תאים. ניתן להגדיר את המערך ללא השמת ערך, ההגדרה כוללת גם את "יצירת האובייקט" מערך באמצעות `new`:

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

הערך 3 מציין את מספר התאים, המיקום הראשון מתחיל במיקום 0,0 ומספר השורות 3 ומספר העמודות 3. גודל המערך נקבע בעת ההגדרה ויצירת המערך, ואינו ניתן לשינוי בעת ריצת התוכנית.

אתחול המערך data

```
data = {4,7,9,3,12}
```

ניתן לשלב את הגדרת המערך ואיתחולו בהוראה אחת: `int[] data = {1,2,3,4,5,8,-6}`. בעת איתחול המערך מוגדר גם אורכו, וניתן לגשת לתכונה של אורך המערך באמצעות התכונה `length`. מהדר השפה אינו מאפשר גישה לתא במערך שמחוץ להגדרת תחומו. מכאן שאם אורכו `length` והמיקום הראשון מתחיל ב-0, הרי שהמיקום האחרון הוא `length-1`. הגדרת מערך מטיפוס שלם, השמת ערכים והדפסת המערך

```
class intarray {
    public static void main(String args[]) {
        int [] data = {1,2,3,4,5,6,7};
        int i;
        for ( i=0; i<data.length; i++ ) {
            System.out.println(data[i]); } // end of for
        } // end of main
    } // end of intarray class
```

השמת ערכים באמצעות יצירת מספרים אקראיים

```

class intarray2 {
    public static void main(String args[] ) {
        int [] data = {1,2,3,4,5,6,7}; // השמת ערכים
        int i;
        for ( i=0; i<data.length; i++ ) {
            System.out.println(data[i]); } // הדפסת ערכי המערך
        for ( i=0; i<data.length; i++ ) {
            data[i] = (int) (Math.random() * 100); } // השמת ערכים
// אקראיים
        for ( i=0; i<data.length; i++ ) {
            System.out.println(data[i]); } // הדפסת ערכי המערך
        } }

תרגיל הגדרת המערך, יצירה של 10 תאים, השמת 10 הערכים משנה באופן דינמי את ערכו של
length המכיל את ערכו של אורך המערך.

int [] data = {1,2,3,4,5,6,7,8,9,10};
int i;
System.out.println(" length = " + data.length );
for ( i=0; i<data.length; i++ ) {
    data[i] = (int) (Math.random() * 100); }
for ( i=0; i<data.length; i++ ) {
    System.out.println(data[i]); }
    }
    
```


לולאה לחישוב סכום המערך והדפסת הסכום מחוץ ללולאה:

```
for ( i=0; i<data.length-1; i++ ) {  
    sum = sum + data[i]; }  
System.out.println(" The sum is " + sum);
```

הגדרת ויצירת מערך באמצעות new

```
data = new int [10];
```

הגדרת מערך מסוג double

```
data = new double [100];
```

התרגיל הבא מציג פקודות ומימוש מעניין של פקודות אלו

```
class intarray3 {  
    public static void main(String args[] ) { // start main  
        char [] data ;  
        data = new char [100];  
        int i;  
        for ( i=0; i<data.length; i++ ) {  
            data[i] = (char) ((Math.random() * 26) + 65); }  
        for ( i=0; i<data.length-1; i++ ) {  
            System.out.println(data[i]); }  
        // how many 'A' we have in the array  
        int c=0;  
        for ( i=0; i<data.length-1; i++ ) {  
            if(data[i]=='A')  
                { c++;}  
            else  
                {}  
        }  
        System.out.println("we have " + c + " in the array ");  
    } // end main  
}
```

הסבר

הגדרת המערך:

```
char [] data ;  
data = new char [100];
```

יצירת מספרים בטווח 65 – 90 שהם קודי ה-ASCII של האותיות Z-A.

```
for ( i=0; i<data.length-1; i++ ) {  
data[i] = (char) ((Math.random() * 26) + 65); }
```

בדיקה, כמה מופעים של האות A יש במערך:

```
int c=0;  
for ( i=0; i<data.length-1; i++ ) {  
    if(data[i] == 'A')  
        { c++;}  
    else  
        {}
```

משימה

כתוב אלגוריתם "המייצר" באופן אקראי 300 תווים A-Z על האלגוריתם לאחסן תווים אלו ולבדוק כמה מופעים הוגרלו מכל תו. פלט האלגוריתם יהיה מהצורה הבאה:

A=6

B=12

C=0

..

Z=2

המחלקה הבאה מיישמת את פתרון הבעיה, מאחר והפתרון מורכב נציג את כל קוד התוכנית ונסביר את הקוד לפי מרכיביו

```

class countChar {
    public static void main(String args[] ) { // start main
        char [] data ;
        data = new char [300];
        int [] countChar;
        countChar = new int[26];
        int i;
        for ( i=0; i<data.length-1; i++ ) {
            data[i] = (char) ((int)(Math.random() * 26) + 65)); }

        for ( i=0; i<countChar.length-1; i++ ) {
            countChar[i]= 0;    }

        for ( i=0; i<data.length; i++ ) {

            switch (data[i])
            {
                case 'A': countChar[0]++ ; break;
                case 'B': countChar[1]++ ; break;
                case 'C': countChar[2]++ ; break;
                case 'D': countChar[3]++ ; break;
                case 'E': countChar[4]++ ; break;
                case 'F': countChar[5]++ ; break;
                case 'G': countChar[6]++ ; break;
                case 'H': countChar[7]++ ; break;
                case 'I': countChar[8]++ ; break;
                case 'J': countChar[9]++ ; break;
                case 'K': countChar[10]++ ; break;
                case 'L': countChar[11]++ ; break;
                case 'M': countChar[12]++ ; break;
                case 'N': countChar[13]++ ; break;
                case 'O': countChar[14]++ ; break;
                case 'P': countChar[15]++ ; break;
                case 'Q': countChar[16]++ ; break;
            }
        }
    }
}

```

```

        case 'R': countChar[17]++; break;
        case 'S': countChar[18]++; break;
        case 'T': countChar[19]++; break;
        case 'U': countChar[20]++; break;
        case 'V': countChar[21]++; break;
        case 'W': countChar[22]++; break;
        case 'X': countChar[23]++; break;
        case 'Y': countChar[24]++; break;
        case 'Z': countChar[25]++; break;

        default: System.out.println("NOT IN A-Z ");
    } // END SWITCH */
} // END OF FOR LOOP

for ( i=0; i<countChar.length-1; i++ ) {
    System.out.println((char) (i+65) + " " + countChar[i]);
}

} // END OF MAIN
} // END OF CLASS

```

ההכרזה והקצאת שני מערכים:

מערך לאחסון 300 התווים שהוגרלו והוא מערך בשם data וטיפוס הנתונים הוא char
 מערך לאחסון מניית הפעמים שהופיע כל תו במערך הנתונים והוא מערך countChar וטיפוס
 הנתונים הוא **int**

```

char [] data ;
data = new char [300];
int [] countChar;
countChar = new int[26];

```

לולאה להגרלת 300 התווים למערך data

```

for ( i=0; i<data.length-1; i++ ) {
    data[i] = (char) ((int)((Math.random() * 26) + 65)); }

```

שים לב: לולאה מוגבלת לפי מס' התאים במערך, מספר התאים נמצא בתוך התכונה `data.length`, התא הראשון במיקום 0 והתא האחרון במיקום `data.length - 1`.

איתחול ל-0 של מערך המונים

```
for ( i=0; i<countChar.length-1; i++ ) {  
    countChar[i]= 0;    }
```

הלולאה המרכזית "סורקת" את מערך `data` ובודקת באמצעות משפט `switch` כל תו מה ערכו ובהתאם לכך מעדכנת את מונה התו במערך המונים.

לסיכום מודפס מערך המונים לפי דרישות האלגוריתם:

```
for ( i=0; i<countChar.length-1; i++ ) {  
    System.out.println((char) (i+65) + " " +  
countChar[i]);    }
```

מערכים דו ממדיים

מערך דו-ממדי הוא מערך שאבריו מטיפוס מערך = < מערך של מערכים

יצירת מערך דו ממדי:

```
<data type> [][] <array name> = new <data type >
[rows][columns]
```

הסבר:

< טיפוס האיברים > [][] שם המערך = new < טיפוס האיברים > [מספר שורות] [מספר
עמודות]
דוגמא:

```
double[][] sales = new double [12][6];
```

התכונה length של מערך דו-ממדי מתארת את מספר השורות בו.
מאחר וכל שורה היא מערך חד-ממדי, ה-length שלה מתאר את מספר העמודות באותה שורה.

לדוגמה, עבור המערך sales לעיל הפקודה: System.out.println(sales.length) תדפיס 12
(מספר השורות במערך הדו מימדי sales).

והפקודה: System.out.println(sales[1].length) תדפיס 6, מס' העמודות בשורה
sales[4] שהוא מערך חד ממדי

משימה:

חברה בת 5 סוכנים מעוניינת לבחון את ההכנסות של החברה כל שליש שנה (4 חודשים) יש
לכתוב תוכנית המגדירה מערך : double [5][4] ולקלוט למערך את נתוני המכירות ולהדפיס
את סכום המכירות בכל חודש.

```

import java.util.Scanner;
class sales {
    public static double[][] sale = new double[5][4];
    public static void readSale() {
        Scanner input= new Scanner(System.in);
        int i,j;
        for(j=0;j<=4;j++){
            System.out.println(" read the next salesman sale " + j );
            for(i=0;i<=3;i++)
                System.out.println("please enter next sale in month "+ j);
                sale[j][i] =input.nextDouble();
            }
        }
    public static void writeSale() {
        int m,n;
        for(m=0;m<=4;m++) {
            for(n=0;n<=3;n++) {
                System.out.println("sale" + n + "in month " + m + " " + sale[m][n]);
            }
        }
    }
    public static void sumSale() {
        int m,n;
        double sum = 0;
        System.out.println(" the finel sales ");
        for(m=0;m<=4;m++) {
            for(n=0;n<=3;n++) {
                sum = sum + sale[m][n];
            }
            System.out.println(" the sale in month " + m + " is " + sum );
            sum=0;
        }
    }
}

```

```
public static void main(String args[]){
    readSale();
    writeSale();
    sumSale();          }
}
```

הסבר:

הגדרת המערך

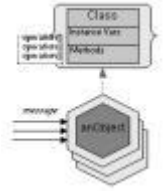
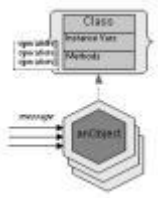
```
public static double[][] sale = new double[5][4];
```

לולאת קליטה של נתוני המכירות למערך, שים לב! אנו מבצעים לולאה מקוננת. לולאה מקוננת נדרשת, כי יש לסרוק כל שורה ועבור כל שורה יש לסרוק את העמודה. סריקה של מערך דו ממדי באמצעות לולאה מקוננת מהווה, לעיתים, "פתח לצרות" זאת אם מדובר במערך שמספר השורות אינו שווה למספר העמודות, טעות באינדקסים של הסריקה יכולה לגרום לחריגה מגבולות המערך.

```
for(j=0;j<4;j++){
    System.out.println(" read the next salesman sale " + j );
    for(i=0;i<3;i++ )
        System.out.println("please enter next sale in month "+ j);
        sale[j][i] =input.nextDouble();
}
```

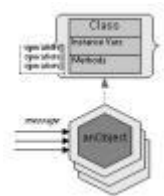
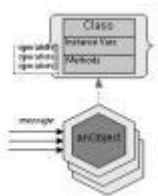
סיכום שורות המערך- סכום ההכנסות בכל חודש

```
int m,n;
double sum = 0;
System.out.println(" the finel sales ");// הודעה המודפסת לפני ביצוע הלולאות
for(m=0;m<=4;m++ ) { // לולאה חיצונית, סריקת שורות
    for(n=0;n<=3;n++ ) { // לולאה פנימית הכוללת את ביטוי הסיכום לשורה
        sum = sum + sale[m][n];
    }
    System.out.println(" the sale in month " + m + " is " + sum );
    sum=0; // איפוס משתנה הסכום כהכנה לסכום השורה הבאה
}
```

פרק שביעי – תכנות מונחה עצמים

בפרק זה אני מציג סיכום והעמקה של נושא תכנות מונחה עצמים. ניתן בצורה מלאכותית ללמד את כל נושאי הלימוד בתוכנית יסודות תורת המחשב, ללא הצגת רעיון תכנות מונחה עצמים (אובייקטים), אולם כתיבת תוכנית בג'אווה ללא תיאור של נושא תכנות מונחה אובייקטים, ללא הסבר כיצד אנו משתמשים בקלט פלט, כיצד מזמנים שיטות ליצירת מספרים אקראיים, או כיצד מגדירים מערכים ומשתמשים במערכים, אינה אפשרית. תכנות מונחה אובייקטים "לכד" את כל המתכנתים דווקא בגלל פשטות הרעיון. העולם שלנו בנוי מעצמים, חלק מהעצמים הם עצמים ממשיים וחלקם עצמים מופשטים. כל תוכנית מחשב מתייחסת לחלק מהעולם שלנו, העולם הממשי, העולם המתמטי, או כל עולם דמיוני שאנו יוצרים. יצירת העולמות הללו וייצוגם במחשב באמצעות אובייקטים מהווה דרך טבעית ומהנה לתכנת ולפתור בעיות.



פרק שביעי – תכנות מונחה עצמים

על אובייקטים תכונות ושיטות

אנו אומרים על שפת ג'אווה שהיא שפה מונחת עצמים, מה ההבדל בין שפה "רגילה" לשפה מונחת עצמים.

בשפה רגילה – פרוצדורלית אנו מגדירים משתנים, מחלקים את התוכנית למשימות ומשימות משנה וכותבים פרוצדורות ופונקציות. בשפה מונחת עצמים אנו שואלים מי הם "השחקנים" של התוכנית, מה הפעולות על "שחקנים" אלו, אלו תכונות יש ל"שחקנים" הללו. הדיון על מבנה התוכנית מתחיל בשאלה מי הם העצמים שהתוכנית משרתת. תכנות בשפה מונחת עצמים טבעי ו"קרוב" יותר למציאות משפה פרוצדורלית. תכנות בשפה פרוצדורלית גורם למתכנת לחשוב במושגים של מבנה מחשב: אלו משתנים אני צריך, כמה מקום "תופס" משתנה מסוים, האם התוכנית תעבוד במחשב של הלקוח, כלומר, המתכנת חושב במונחים של "מרחב הבעיה", לעומת זאת, בתכנות מונחה עצמים המתכנת חושב במונחים של עולם הפתרון, אם הוא יוצר תוכנה לניהול חשבונות יוצר המתכנת את האובייקטים המייצגים הזמנה, חשבון, קבלה, לקוח. לכל עצם יש תכונות, העצמים קשורים ומעבירים מידע אחד לשני, וכך עולם הפתרון נוצר מתוך המציאות וזאת הודות לרמת ההפשטה הגבוהה שמאפשרת ובעצם מעודדת את התפיסה של תכנות מונחה עצמים. בתכנות מונחה עצמים ניתן להתייחס לכל אובייקט מעין "אובייקט המחשב את עצמו", לכל אובייקט יש מצבים ותכונות וישנם פעולות שאפשר לבקש ממנו לבצע.

תוכנית בשפה מונחת עצמים

תוכנית בשפה מונחת אובייקטים היא אוסף של אובייקטים, אובייקטים אלו "אומרים" זה לזה מה לעשות. אובייקטים מסוימים קשורים וחולקים תכונות עם אובייקטים אחרים ובכך ניתן ליצור מודל של מציאות מורכבת על-ידי יצירת אובייקטים פשוטים והרכבת אובייקטים מורכבים יותר מאובייקטים אלו.

כל אובייקט הוא "מופע" – instance של המחלקה שלו – class.

אוסף המחלקות, האובייקטים, השיטות (מה אובייקט "יודע לעשות" ואיזה "מידע" ניתן "לקבל" מאובייקט), התכונות והקשר בין האובייקטים והמחלקות, מהווים את הבסיס לעולם תכנות מונחה עצמים.

האופרטור new

יצירת מופע של המחלקה

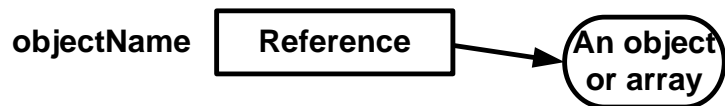
כדי ליצור מופע של המחלקה אנו משתמשים באופרטור new

האופרטור new יוצר מופע של המחלקה ע"י הקצאת זיכרון למופע ויצירת הפנייה לאובייקט.

יצירת מופע של מחלקה נקרא בג'אווה: instantiates a class

ביצוע של האופרטור new יוצר את האובייקט של המחלקה, מקצה זיכרון ויוצר הפניה לאובייקט

האיור הבא מציג את משמעות ביצוע הפעולה new



כל שימוש באופרטור new וזימון שיטה בונה יוצר אובייקט, מקצה זיכרון לאובייקט והפנייה

באמצעות השם שנתנו למופע שייצרנו באיור שם האובייקט הוא: **objectName**

נציג את המחלקה point

```
class Point {  
    private int x,y; }  
}
```

שאלה

אם אנו נכתוב את השורה הבאה: האם אנו יוצרים אובייקט?

```
Point pointUpLeft;
```

תשובה

לא, אין אנו יוצרים אובייקט אלא מכריזים על הפניה pointUpLeft שתצביע על עצם מסוג

Point. בהכרזה לא נוצר עצם\מופע של המחלקה Point

נגדיר שיטה בונה למחלקה Point

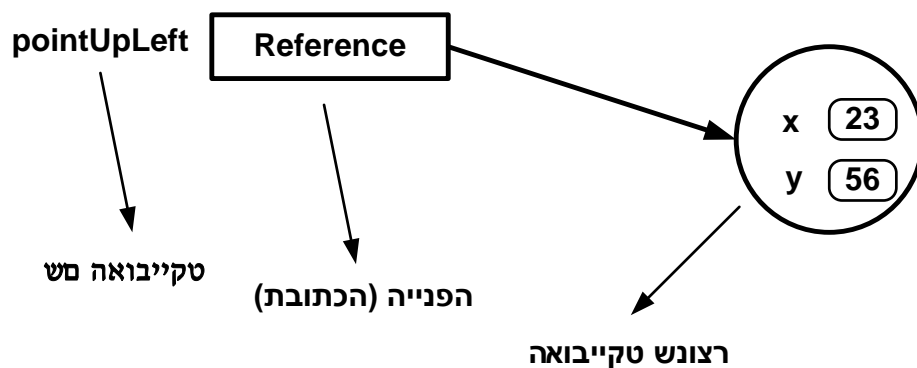
```
Public Point ( int x, int y) {
    this.x = x;
    this.y=y;
}
```

הפעלה האופרטור new ויצירת אובייקט מסוג Point

```
Point pointUpLeft = new Point(23,56);
```

הפעלת **new** יצרה מופע של המחלקה Point זאת ע"י האופרטור new. תוצאת הפעולה היא הקצאת זיכרון למופע המחלקה והפנייה למופע זה.

תיאור גרפי של הביצוע:



מעתה כל התייחסות לתכונות האובייקט וקריאה לשיטות של האובייקט מתבצעות ע"י שם האובייקט ואופרטור הנקודה.

כללי: `objectReference.variableName`

תרגום בעברית פשוטה: שם_הפנייה.שם_משתנה_אובייקט

דוגמא:

הדפסת ערכי הנקודה pointUpLeft

```
System.out.println(pointUpLeft.x + pointUpLeft.y)
```

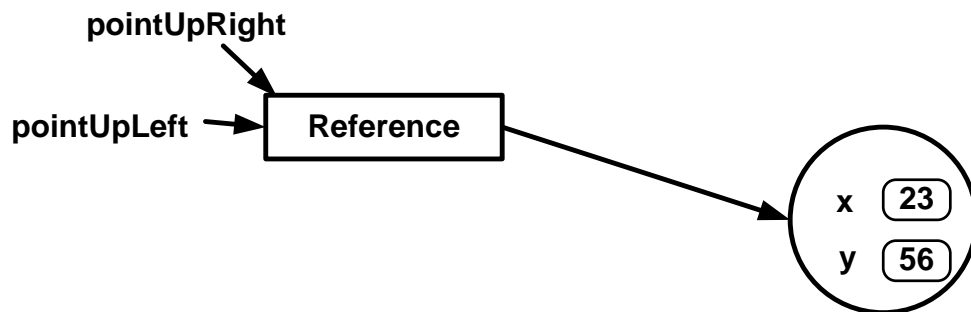
שאלה

מה תהיה התוצאה של ביצוע ההוראות הבאות?

```
Point pointUpRight ;  
pointUpRight= pointUpLeft;
```

תשובה

מאחר ולא יצרנו מופע של הנקודה באמצעות **new** לא הוקצה זיכרון מיוחד לנקודה `pointUpLeft` ולא נוצרה הפנייה חדשה. המשתנה `pointUpLeft` מקבל את ערך ההפניה של המשתנה `pointUpRight` והאיור הבא מסביר את תוצאת ביצוע ההוראות.



כל שימוש בייחוס נקודה וגישה לתכונה של אחת הנקודות ושינוי ערכים, ישנה את הערכים גם עבור האובייקט השני. שני ההפניות מתייחסות לאותו אובייקט.

דוגמאות לעצמים

דוגמא 1: בבית מלון ישנם חדרים.

חדר במלון

לחדר במלון יש תכונות כולל:

גודל החדר

מס' המיטות בחדר

האם יש טלוויזיה בחדר האם יש מיזוג בחדר

האם יש מזגן בחדר

האם החדר מאוכלס

דוגמא 2: בבית מלון ישנם אורחים.

אורח במלון

שם האורח

באיזה חדר האורח מתארח

כמה האורח צריך לשלם

באיזה תאריך האורח הגיע

מתי האורח עוזב

בשפה פרוצדורלית נקודת המבט של המתכנת היא מה הם הפעולות שיש לבצע. בתכנות מונחה עצמים נקודת המבט של המתכנת, מה הם העצמים בתוכנית מה הן התכונות של העצמים ואלו פעולות יש לבצע. גם התכונות והפעולות קשורים ישירות לעצם.

העצם – חדר במלון

roomHotel

לחדר התכונות הבאות:

isFree - תכונה לוגית האם החדר תפוס או פנוי

roomSize - גודל החדר

bedNumber - מס' מיטות

בתכנות מונחה אובייקטים אנו מגדירים את המחלקה של עצם – אובייקט ובדוגמא שלנו ההגדרה תראה כך.

class roomHotel

```
{ private Boolean isFree;  
  private double roomSize;  
  private int bedNumber; }
```

למחלקה יש תכונות אשר משותפות לכל העצמים של אותה מחלקה ויש שיטות , כל שיטה הכלולה במחלקה מבצעת פעולה, או בודקת מידע על עצמים של המחלקה (מלבד שיטות המחלקה שאינם מופעלות ע"י עצמים של המחלקה אלא ע"י המחלקה עצמה).

דוגמא 3: רובוט מכסח דשא .

האובייקט הבא הוא רובוט מכסח דשא שפותח לאחרונה:

תכונות הרובוט:

מצב פעולה (דולק, לא דולק)

מצב נסיעה (עצירה, נסיעה איטית, נסיעה מהירה)

מצב סכין כיסוח (מצב עליון, מצב תחתון)

צבע

דגם

שנת ייצור

פעולות שהאובייקט רובוט מכס דשא יודע לעשות

קבלת מצב נסיעה של הרובוט

רובוט. נסיעה()

קבלת מצב הפעולה של הרובוט

רובוט.פעולה()

קבלת מצב הסכין

רובוט.סכין()

קבלת צבע הרובוט

רובוט.צבע()

האובייקט רובוט מזמן את השיטה צבע המחזירה את צבע הרובוט.

קבלת דגם הרובוט

רובוט.דגם()

האובייקט רובוט מזמן את השיטה דגם המחזירה את סוג הדגם.

רובוט.שנת_ייצור()

האובייקט רובוט מזמן את השיטה שנת_ייצור המחזירה את שנת הייצור שלו.

רובוט.מצב_פעולה (קוד)

האובייקט רובוט מזמן את השיטה מצב_פעולה לשינוי מצב הפעולה שלו.

רובוט.מצב_נסיעה (קוד)

האובייקט רובוט מזמן את השיטה מצב_נסיעה המשנה את מצב הנסיעה שלו

רובוט.מצב_להב(קוד)

האובייקט רובוט מזמן את השיטה מצב_להב לשינוי מצב הלהב שלו

דוגמא "לתוכנית" עבור מכסח הדשא

רובוט = צור אובייקט מהמחלקה רובוט כיסוח דשא ()

רובוט.צבע(ירוק)

רובוט.נסיעה(שקר)

רובוט.להב(שקר)

הצג (רובוט.מצב_פעולה())

הצג (מצב_נסיעה(רובוט.סע()))

חתימת השיטה main

```
public static void main(String [] args){  
}
```

השיטה main היא השיטה שחייבת להיכלל בכל מחלקה.

הכותרת של שיטה נקראת חתימה של השיטה ומכילה מידע על אופן פעולת השיטה, מידע שחלק ממנו נסביר כאן.

המילה public

שיטה יכולה להיות מוגדרת כ- **public** או **private**, אם השיטה מוגדרת כציבורית (**public**), נוכל להשתמש בה גם במחלקות אחרות שאינן המחלקה בה מוגדרת השיטה. אם השיטה מוגדרת כפרטית (**private**) לא נוכל להשתמש בה ממחלקות אחרות.

המילה void

המילה **void** מציינת שהשיטה אינה מחזירה ערך, כלומר אין חישוב בשיטה שתוצאתו מועברת מחוץ לשיטה. אם במקום המילה **void** נכתב **int** או **double** וכו.. הרי זה מגדיר את סוג הערך המוחזר ע"י השיטה. בשפה פרוצדורלית רגילה תת תוכנית, שאינה מחזירה ערך, נקראת פרוצדורה או שגרה, ואילו תת תוכנית המחזירה ערך, נקראת פונקציה.

המילה static

שיטה המוגדרת **static** היא שיטת מחלקה ולא שיטת-מופע, כלומר השיטה אינה קשורה למופע מסוים של המחלקה וניתן להפעיל את השיטה גם ללא יצירת (ע"י new) של מופע המחלקה. אם השיטה אינה סטטית לא ניתן להפעילה אלא אם יצרנו מופע של המחלקה. לדוגמא, השיטה main היא static זו השיטה המופעלת ראשונה בטרם נוצר מופע כלשהו של אובייקט. שים לב! לא תוכל לקרא לשיטה שאינה סטטית מתוך שיטה סטטית.

מה זה (String args[])

בשלב זה לא נשלים את ההסבר על המשפט המופיע בתוך הסוגריים של חתימת השיטה main, ההסבר החלקי הוא שכל שיטה מכילה בתוך הסוגריים של חתימתה את הארגומנטים שהיא מקבלת, והשיטה main מקבלת כארגומנט מערך של מחרוזות args, ולכן מצויין הארגומנט String args[] המציין מערך מחרוזות. מערך מחרוזות זה משמש לקליטת נתונים מהמשתמש המריץ את התוכנית.

סיכום מושגים – אובייקטים

בשפת ג'אווה	הסבר	דוגמא
Class	מילה הפותחת הגדרת מחלקה	class Point
return	החזר – ההוראה להחזרת ערך משיטה שאינה מוגדרת כ- void	return y
אין החזרת ערך	Void	Public void main(String args[])
new	יצירה של מופע של מחלקה	new klaf();
public	הגדרה של משתנה, שיטה כציבורי	public int sum();
private	הגדרה של משתנה שיטה כפרטי ולא מוכר מחוץ למחלקה או השיטה	private double sum(int x);
dot notation	כל הרעיון של חיבור אובייקט לשיטותיו ותכונותיו, בין מחלקה לתכונות המחלקה	point.left

הגדרת מחלקה חדשה

```
public class triangle {  
  
}
```

הגדרת שיטה

```
public int big( double num1, double num2);
```

חתימת השיטה

כותרת השיטה נקראת **חתימת השיטה** אשר מהווה חלק חשוב בזהות והגדרת השיטה :

```
public int big( double num1, double num2);
```

public - מגדיר את תחום ההכרה של השיטה, - ציבורי (**private** - פרטי)

int מגדיר את סוג הערך שהשיטה מחזירה (**void** במקרה שלא מוחזר ערך)

big שם השיטה

() סוגריים מציינים שמדובר בשיטה, אם אין ארגומנטים שמועברים לשיטה יופיעו הסוגריים,

אם יש ארגומנטים הם יופיעו בין הסוגריים

מחלקה KLAF

הגדרת מחלקה עם שני משתנים שלמים מקומיים. המחלקה מממשת טיפוס נתונים מופשט קלף של דומינו.

```
class klaf{  
private int left;  
private int right;
```

השיטה הבונה

```
public klaf ( int x, int y ) {  
this.left = x;  
this.right = y;  
}
```

הכנסת ערכים בתוך השיטה הבונה

מאחר והשיטה הבונה "לא יודעת" איזה עצם של המחלקה יזמן אותה ואנו רוצים להתייחס לשדות right ו- left ולכן נרשום בשיטה הבונה **this.left** ו- **this.right**

this

פירושו האובייקט שזמן את השיטה. בעת כתיבת שיטה לא ידוע איזה אובייקט יזמן אותה ולכן אנו משתמשים ב- **this**, מעין תבנית פתוחה, שמקבלת בעת הביצוע את שם האובייקט שזמן את השיטה. במקרה של זימון השיטה הבונה יקבל האובייקט את הערכים שיוכנסו לתוך תכונות האובייקט. הסבר על **this** אינו פשוט ואני אדגים את השימוש ואפרט יותר את ההסברים בהמשך.

השיטה הבונה klaf מקבלת שני מספרים שלמים ויוצרת מופע של המחלקה klaf

הפעלה השיטה הבונה ליצירת מופעים של קלפים

```
klaf c3 = new klaf(6,8);
```

הסבר על השורה:

klaf c3

הכרזה בלבד! על c3 שיכיל הפניה לאובייקט של המחלקה klaf

```
new klaf(6,8);
```

החלק השני של השורה מפעיל את השיטה הבונה שיוצרת את האובייקט שההפניה c3 מצביעה עליו. ומכניסה את הערכים לתוך האובייקט.

דרך שנייה ליצירת מופע של klaf והכנסת ערכים ניתנת לביצוע בשני שלבים ובשלוש שורות קוד.

```
klaf c1= new klaf();
```

השורות המבצעות הכנסת ערכים

```
c1.right = 5;
```

```
c1.right = 8;
```

המחלקה klaf

כותרת המחלקה המתארת אובייקט קלף דומינו עם שני משתנים מקומיים מסוג שלם.

```
public class klaf {  
    private int left;  
    private int right;
```

השיטה הבונה

```
public klaf(int x, int y) {  
    this.left = x;  
    this.right = y;  
}
```

השיטה getLeft

```
public getLeft() {  
    return this.left  
}
```

השיטה getRight

```
public getRight() {  
    return this.left  
}
```

השיטה sum

השיטה מקבלת אובייקט מסוג klaf ומחזירה ערך שלם המהווה סכום הערכים של שני צידי הקלף שהתקבל.

```
public int sum() {  
  
    return this.getLeft()+ this.getRight();  
}
```

השיטה המרכזית main
השיטה יוצרת 3 מופעים של המחלקה klaf ומכניסה ערכים ל- 3 המופעים בשתי דרכים שונות כפי שתוארו לעיל.
בהמשך נבדוק איזה ערך יותר גדול בין שני מופעים של המחלקה.

```
public static void main (String[] args) {  
    klaf c3 = new klaf(6,8);  
    klaf c1= new klaf(5,4);  
    klaf c2 = new klaf(3,2);  
    if(c1.getRight() > c2.getRightt())  
        System.out.println(c1.getRight() + " c1.right > c2.right ") else  
        System.out.println(c2.getRigth() + " c2.right > c1.right ");  
}
```

מחלקה Point

```
public class Point {  
    private int x;  
    private int y;
```

השיטה הבונה של המחלקה Point

```
public Point( int x, int y) {  
    this.x = x;  
    this.y=y;    }
```

במחלקה Point הגדרתי 4 שיטות

השיטה getX המקבלת נקודה ומחזירה את ערך X של הנקודה.

השיטה getY המקבלת נקודה ומחזירה את ערך Y של הנקודה.

השיטה setX מקבלת ערך שלם ומשנה את ערך ה-X של הנקודה

השיטה setY מקבלת ערך שלם ומשנה את ערך ה-Y של הנקודה

המחלקה Point

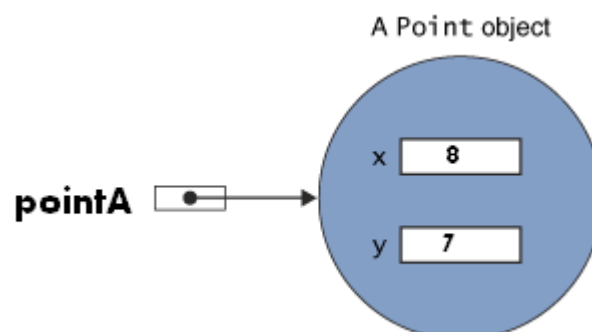
```
public class Point {  
    private int x=0;  
    private int y=0;  
    public Point( int x, int y) {  
        this.x = x;  
        this.y= y;    }  
    public double getX() {  
        return this.x; }  
    public double getY() {  
        return this.y; }  
    public void setX(int x){  
        this.x=x;    }  
    public void setY(int y){  
        this.y=y;    }  
}
```

המחלקה testPoint

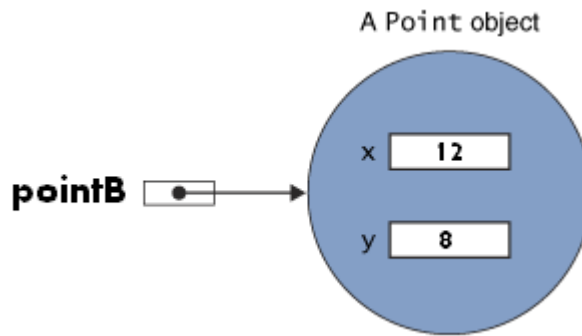
המחלקה משתמשת במחלקה Point ויוצרת שני מופעים של המחלקה Point, ומשתמשת בשיטות שהוגדרו במחלקה Point כדי לבצע פעולות על האובייקטים שנוצרו pointA ו-pointB

```
import java.lang.*;
class testPoint {
public static void main( String[] args){
    double deltaX,deltaY;
    Point pointA = new Point(8,7);
    Point pointB = new Point (12,8);
    System.out.println( "x1 = " + pointA.getX());
    System.out.println( "x2 = " + pointB.getX());
    System.out.println( "Y1 = " + pointA.getY());
    System.out.println( "Y2 = " + pointB.getY());
    pointA.setX(12);
    pointB.setY(34);
    System.out.println( "x1 = " + pointA.getX());
    System.out.println( "x2 = " + pointB.getX());
    System.out.println( "Y1 = " + pointA.getY());
    System.out.println( "Y2 = " + pointB.getY());
    deltaX = pointA.getX() - pointB.getX();
    deltaY = pointA.getY() - pointB.getY();
    System.out.println( "deltaX = " + Math.abs(deltaX));
    System.out.println( "deltaY = " + Math.abs(deltaY));
}
}
```

מה התרחש כתוצאה מהפקודה : **Point pointA = new Point(8,7);**



מה התרחש כתוצאה מהפקודה **Point pointB = new Point (12,8);**



”בקשה” מהאובייקטים שיצרנו להחזיר את ערכי ה-X ו-Y והצגתם בפלט

```
System.out.println("x1 = " + pointA.getX());  
System.out.println("x2 = " + pointB.getX());  
System.out.println("Y1 = " + pointA.getY());  
System.out.println("Y2 = " + pointB.getY());
```

שינוי ערכי הנקודות ע”י שימוש בפעולות `setX` ו-`setY`

```
pointA.setX(12);  
pointB.setY(34);
```

חישוב ערכים של הפרשי קורדינטות X-ים והפרשי קורדינטות Y-ים

```
deltaX = pointA.getX() - pointB.getX();  
deltaY = pointA.getY() - pointB.getY();
```

הדפסת ההפרשים ושימוש בשיטה `abs` של המחלקה `Math`

```
System.out.println("deltaX = " + Math.abs(deltaX));  
System.out.println("deltaY = " + Math.abs(deltaY));
```


שיטה בונה מעתיקה – Copy Constructor

ראינו שאם מבצעים השמה ערך של משתנה המכיל הפניה למשתנה אחר שתי ההפניות מצביעות על אותו עצם.

נניח ש- p1 מכיל הפני לעצם מסוג Point אנו מבקשים ש-p2 יכיל הפניה לעצם מסוג Point עם אותם ערכים אבל לא הצבעה על אותו עצם, כלומר להעתיק את העצם ש-p1 מצביע עליו. המחלקה Point המעודכנת מדגימה כיצד אנו מוסיפים עוד שיטה בונה למחלקה, שיטה המבצעת העתקה של עצם ונקראת שיטה-בונה מעתיקה Copy Constructor

```
public class Point {
    private int x=0;
    private int y=0;
    public Point( int x, int y) {
        this.x = x;
        this.y= y;
    }
    public Point (Point p){
        this.x=p.x;
        this.y=p.y;
    }
    public int getX() {
        return this.x;
    }
    public int getY() {
        return this.y;
    }
    public void setX(int x){
        this.x=x;
    }
    public void setY(int y){
        this.y=y;
    }
    public static void main (String[] args ) {
        Point p1=new Point(5,8);
        Point p2 =new Point(p1);
        System.out.println(p1.getX());
    }
}
```

```

System.out.println(p2.getX());
p2.setX(12);
System.out.println(p1.getX());
System.out.println(p2.getX());
}
}

```

השיטה הבונה "הרגילה" במחלקה Point

```

public Point( int x, int y) {
    this.x = x;
    this.y= y;
}

```

זימון השיטה ליצירת עצם מסוג Point

```

Point p1=new Point(5,8);

```

שים לב, השיטה מקבלת כפרמטר שני מספרים שלמים זאת כדי ליצור נקודה שאלו הקורדינטות שלה.

השיטה – בונה מעתיקה במחלקה Point

```

public Point (Point p){
    this.x=p.x;
    this.y=p.y;
}

```

השיטה מקבלת כפרמטר את כל העצם מסוג Point בבת אחת ולא תכונות נפרדות של העצם. בכך מתאפשר מנגנון ההעתקה של עצם לעצם חדש.

זימון השיטה-בונה מעתיקה

```

Point p2 =new Point(p1);

```

שים לב! p1 הוא עצם מסוג Point מאותחל כי אחרת לא תתבצע פעולת ההעתקה.

העובדה שמדובר בשני עצמים שונים עם הפניות שונות באה לידי ביטוי בהוראה המשנה תכונה בעצם אחד ללא השפעה על ערכה של תכונה זו בעצם השני.

```

System.out.println(p1.getX());
System.out.println(p2.getX());
p2.setX(12);
System.out.println(p1.getX());
System.out.println(p2.getX());

```

המנגנון המאפשר כתיבת שתי שיטות בונות ויותר הוא מנגנון העמסת שיטות Overloading עליו נרחיב בהמשך.

מערך של אובייקטים

בדוגמא הבאה מציגה מערך של אובייקטים מסוג Point, המחלקה עושה שימוש במחלקה Point וגם במחלקה Turtle שילוב זה ממחיש את היכולות של שפת ג'אווה ליצירת מבנה מורכב ושימוש במחלקות ציבוריות. המחלקה Point נכתבה בספר זה, המחלקה Turtle נכתבה ע"י האוניברסיטה העברית.

המחלקה Point

בדוגמא זו הוספתי למחלקה Point את השיטה toString המחזירה אובייקט מסוג String הכולל את ערכי הנקודה.
השיטה toString

```
public String toString(){
    String str = "<" + (int)this.x + ">,<" + (int)this.y + ">";
    return str;
}
```

המחלקה testPointTurtle הסבר:

יצירת הפניה ואובייקט מסוג מערך של Point

```
Point [] points = new Point[10];
```

שים לב!

יצרנו הפניה ואובייקט מסוג מערך של אובייקטים מסוג Point, 10 האובייקטים מסוג Point במערך לא נוצרו, יש רק הפנייה למערך מסוג Point כדי ליצור את האובייקטים במערך מסוג Point כתבתי את הלולאה הבאה:

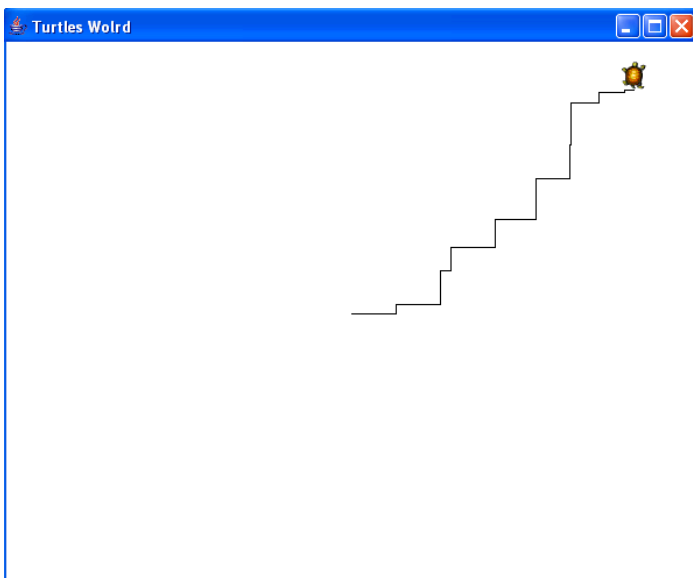
```
for(int i=0;i<=9;i++){
    double x=(Math.random()*20);
    double y=(Math.random()*20);
    points[i]= new Point(x,y);
}
```

שים לב: משתנה הלולאה i בכל מחזור יוצר אובייקט חדש של נקודה עבור כל הפניה של אובייקט במערך:

```
points[i]= new Point(x,y);
```

המחלקה testPointTurtle משלבת את השימוש במחלקה Turtle ע"י מעבר על הנקודות שנוצרו ויוצרת קוים בהתאם לערכי הנקודות.

```
import TurtleLib.Turtle;
public class testPointTurtle {
    public static void main(String args[]){
        Point [] points = new Point[10];
        for(int i=0;i<=9;i++){
            double x=(Math.random()*20);
            double y=(Math.random()*20);
            points[i]= new Point(x,y);
        }
        Turtle t = new Turtle();
        t.tailDown();
        for(int i=0;i<=9;i++){
            t.turnRight(90);
            t.moveForward(points[i].getX());
            t.turnLeft(90);
            t.moveForward(points[i].getY());
            System.out.println(("point " + i + " = " +
points[i].toString()));
        }
    }
}
```



פלט המחלקה: testPointTurtle
 "מדרגות" שאורך הצלעות בהתאם לערכי
 x ו-y של כל נקודה:

המחלקה dirot

בניין דירות כולל מספר קומות, בכל קומה ישנם מספר דירות. לכל דירה יש מס' תכונות:

מס' דירה, מס' חדרים, מס' הקומה, כיוון הדירה, מחיר הדירה.

המחלקה dirot מגדירה את המחלקה ותכונות המחלקה.

```
public class dirot
{
    private int numApp;
    private double room;
    private double price;
    private int floor;
    private char side;

    public dirot (int nA, double sR, double sP, int sF, char sS)
    {
        this.numApp=nA;
        this.room= sR;
        this .price = sP;
        this.floor = sF;
        this.side = sS;
    }

    public void setApp(int n) {
        this.numApp=n; }

    public void setRoom(double r) {
        this.room=r; }

    public void setPrice(double p) {
        this.price=p; }

    public void setFloor(int f) {
        this.floor=f; }

    public void setSide(char c) {
        this.side=c; }

    public double getApp() {
        return this.numApp; }

    public double getRoom() {
```

```

        return this.room;  }
    public double getPrice() {
        return this.price;  }
    public int getFloor() {
        return this.floor;  }
    public char getSide() {
        return this.side;  }

}

```

תכונות המחלקה

```

private int numApp;
private double room;
private double price;
private int floor;
private char side;

```

תכונות המחלקה (member) מוגדרים כ- private זאת כדי שלא ניתן לשנות את הערכים של אובייקט רק באמצעות השיטות המוגדרות במחלקה. כלומר דרך "ממשק" המחלקה.

השיטה הבונה

```

public dirot (int nA, double sR, double sP, int sF, char sS)
{
    this.numApp=nA;
    this.room= sR;
    this .price = sP;
    this.floor = sF;
    this.side = sS;
}

```

השיטה הבונה של המחלקה מבצעת השמת הערכים לתוך חברי אובייקט של המחלקה. שיטות איחזור- השיטות לאיחזור של תכונות אובייקטים של המחלקה.

```

public void setApp(int n) {
    this.numApp=n;  }
public void setRoom(double r) {
    this.room=r;  }
public void setPrice(double p) {
    this.price=p;  }
public void setFloor(int f) {

```

```

        this.floor=f;    }
    public void setSide(char c) {
        this.side=c;    }

```

השיטות להחזרת ערכים של אובייקטים של המחלקה.

```

    public double getApp() {
        return this.numApp;    }
    public double getRoom() {
        return this.room;    }
    public double getPrice() {
        return this.price;    }
    public int getFloor() {
        return this.floor;    }
    public char getSide() {
        return this.side;    }

```

השיטות המוגדרות במחלקה dirot, כוללות שיטות לעדכון תכונות אובייקטים של המחלקה, והחזרת ערכי תכונות מאובייקטים של המחלקה.

המחלקה projectDirot : מחלקה המשתמשת במחלקה dirot יוצרת מופעים, עצמים של המחלקה, ומבצעת שינויים בתכונות באמצעות שימוש בשיטות המחלקה. במחלקה הוגדרה שיטה המבצעת חישוב של עלות דירה בשקלים.

```

public class projectDirot
{
    public static final double dolorRate = 4.65;
    public static double computeShekel()
    {
        return dolorRate * this.getPrice();
    }
    public static void main (String args []) {
        dirot dira01= new dirot(4,4.5,125000.0,7,'w');
        System.out.println("num of room in app " + dira01.getApp()
+ "is "+dira01.getRoom());
        System.out.println("The price for the app in shekel is " +
dira01.computeShekel());
    }
}

```

תיאור המחלקה projectDiro

יצירת מופע המחלקה dirot

```
dirot dira01= new dirot(4,4.5,125000.0,7,'w');
```

יצרתי מופע מחלקה- dira01 דירה זו כוללת את התכונות הבאות:

- מספר הדירה
- מספר החדרים
- מחיר הדירה
- קומת הדירה
- כיוון הדירה

השיטה computeShekel

שיטה זו מקבלת ארגומנט אובייקט מסוג דירה, ומחזירה את מחיר הדירה בשקלים.

```
public static double computeShekel(diro d)
{ return dolarRate * d.getPrice();}
```

הסבר הביטוי :

```
return dolarRate * d.getPrice();
```

dolarRate - קבוע שיש בו ערך של שער הדולר

getPrice() שימוש בשיטה המחלקה לאחזור מחיר הדירה

תוצאת הביטוי מחיר הדירה שקלים.

שיטת חישוב מחיר הדירה בשקלים נכתבה כשיטה סטטית. מימוש זה נעשה מטעמים פדגוגיים, ההמלצה היא תמיד לכתוב שיטות לא סטטיות, כלומר כתיבת שיטה המופעלת על ידי אובייקט של המחלקה כלומר תכנות מונחה עצמים "אמיתי".

המחלקה Dia

המחלקה Die מממשת קוביה בעל 6 פאות

```
public class Die {
    private int num;
    public Die(int n) {
        this.num=n;
    }
    public void roll() {
        this.num= (int)(Math.random()*6)+1;
    }
    public int getNum(){
        return this.num;
    }
}
```

השיטה הבונה יוצרת עצם מסוג קוביה ומאתחלת עם הערך שהשיטה מקבלת. ניתן לקבוע שאתחול קוביה וקביעת הערך של הפאה העליונה יהיה 0.

```
public Die(int n) {
    this.num=n;
}
```

השיטה roll "מגלגלת" את הקוביה וקובעת את ערך הפאה העליונה

```
public void roll() {
    this.num= (int)(Math.random()*6)+1;
}
```

השיטה getNum מאפשרת גישה לתכונת האובייקט-הקוביה ומחזירה ערך זה.

```
public int getNum(){
    return this.num;
}
```

המחלקה DiscGame

המחלקה DiscGame מבצעת הדמייה של זריקת שתי קוביות. הקוד עוקב אחר תוצאות הזריקה של שתי הקוביות וממתין לאירוע ההיסתברותי של קבלת שתי תוצאות שוות ל-6 על פני שתי הקוביות. המחלקה מדפיסה תוצאות כל זריקה ואת מספר הזריקות שבוצעו עד לקבלת האירוע המבוקש.

```
public class DiscGame {
    public static void main(String[] args) {
        Die cubeA = new Die (0);
        Die cubeB = new Die (0);
        int round = 0;
        while((cubeA.getNum()!=6)|| (cubeB.getNum()!=6)){
            cubeA.roll();
            cubeB.roll();
            round++;
            System.out.println(cubeA.getNum()+ " " + cubeB.getNum());
        }
        System.out.println("The num of rounds " + round);
    }
}
```

המחלקה Forms

```

import unit4.turtleLib.*;
import unit4.ioLib.*;
public class Forms {
    private int sides;
    private int size;
    public Forms (int sides,int size){
        this.sides=sides;
        this.size=size;
    }
    public int getSide(){
        return this.sides;
    }
    public int getSize(){
        return this.sides;
    }
    public void drawShape(){
        Turtle t = new Turtle();
        t.tailDown();
        for(int i=1;i<=this.sides;i++){
            t.moveBackward(this.size);
            t.turnLeft(360/this.sides); }
        t.setVisible(false);
    }
    public int parimeter (){
        int s=0;
        for(int i=1; i<=this.sides;i++){
            s=s+this.size; }
        return s;
    }
}

```

המחלקה Form מגדירה אובייקט מצולע משוכלל שתכונותיו הם: אורך צלע המצולע ומספר צלעותיו.

במחלקה Form שיטה בונה היוצרת מצולע לפי מספר צלעות ואורך הצלע. במחלקה שיטות נוספות כולל:

השיטה `getSide` שיטה המחזירה את מספר צלעות המצולע

```
public int getSide(){
    return this.sides;
}
```

השיטה `getSize` שיטה המחזירה את אורך צלע המצולע

```
public int getSize(){
    return this.sides;
}
```

השיטה `DrawShape` מבצעת "משימה מעניינת". השיטה באמצעות שיטות המחלקה `Turtle` מציירת את המצולע על המסך.

```
public void drawShape(){
    Turtle t = new Turtle();
    t.tailDown();
    for(int i=1;i<=this.sides;i++){
        t.moveBackward(this.size);
        t.turnLeft(360/this.sides); }
    t.setVisible(false);
}
```

השיטה מחזירה את היקף המצולע, היקף המחושב בהתאם לאורך צלע המצולע (ראוי להזכיר כי מדובר במצולע משוכלל, כלומר כל צלעותיו שוות)

```
public int parimeter (){
    int s=0;
    for(int i=1; i<=this.sides;i++){
        s=s+this.size;
    }
    return s;
}
```

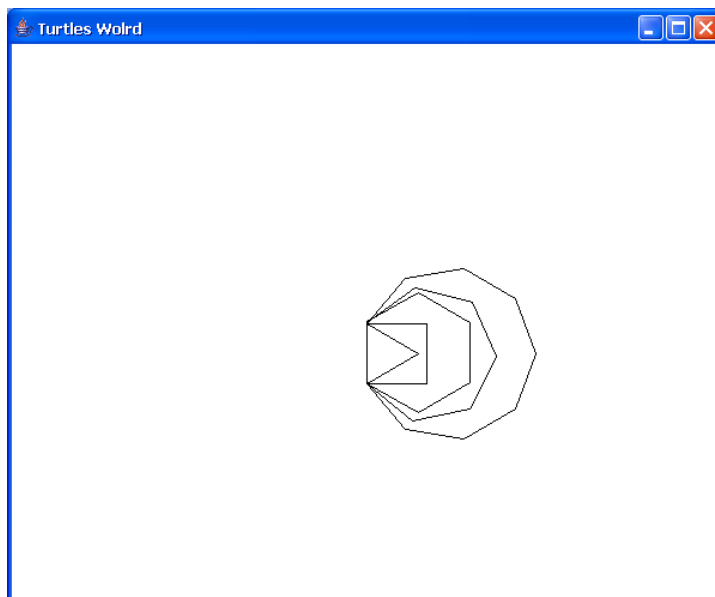
המחלקה DrawForm

המחלקה DrawForm עושה שימוש במחלקה Form ובמחלקה חיצונית Turtle. במחלקה אני מגדיר מערך שסוג האיברים שלו הוא מסוג Form כלומר כל איבר במחלקה הוא מצולע. חשוב!

שים לב! העובדה שהמערך הוגדר מסוג Form עדיין לא יוצרת אובייקטים מסוג זה במערך אלא רק הפניות, יש צורך בלולאה שסורקת את איברי המערך ובאמצעות האופרטור **new** נוצרים המצולעים בכל תא במערך זאת כפי שניתן לראות במחלקה. באמצעות שיטות המחלקה Turtle מצויירות כל המצולעים במערך ומחושב היקפו של המצולע.

```
public class DrawForm{
public static void main(String[] args) {
    Forms []f = new Forms [10];
    for(int j=0;j<=9;j++){
        f[j]=new Forms((int)(Math.random()*7)+3,50);
        f[j].drawShape();
        System.out.println("The number of sides "+ f[j].getSide() + "
            the parimeter is "+f[j].parimeter());
    }
}
}
```

פלט הרצת המחלקה:



משחק "מלחמה"

משחק קלפים "מלחמה" הוא משחק פשוט, לכל שחקן ערימת קלפים התחלתית (מספר שנקבע בין השחקנים). בכל משחקון מציג כל שחקן את הקלף הראשון בערימה, השחקן שהערך של הקלף שלו גדול יותר זוכה בקלף של היריב, לוקח את שני הקלפים, שלו ושל יריבו ומניח אותם בתחתית הערימה שלו וכך המשחק נמשך עד אשר אחד השחקנים מפסיד את קלפיו או שכאשר אחד השחקנים או שניהם משתעמם.

המחלקות הבאות מבצעות הדמייה של משחק, באמצעות 2 אובייקטים ושלוש מחלקות.

המחלקה Card

```
public class Card{
    private int power;
    public Card (int p){
        this.power =p;  }
    public Card (){
    }
    public int getPower(){
        return this.power; }
    public void setPower(int p){
        this.power=p;
    }
}
```

במחלקה Card שתי שיטות בונות, שיטות עם השמת ערך בקלף, שיטה ללא השמת ערך.

במחלקה שתי שיטות נוספות:

השיטה getPower שיטה המחזירה את הערך של הקלף

```
public int getPower(){
    return this.power;
}
```

השיטה setPower שיטה שמקבלת מספר שלם ומעדכנת את ערך הקלף לפי הערך שהתקבל כארגומנט.

```
public void setPower(int p){
    this.power=p;
}
```

שים לב!

תכונת power במחלקה Card היא private כלומר לא ניתן לגשת לתכונה מחוץ למחלקה. השיטות הם public ומאפשרות גישה לשליפת הערך של power ועדכון הערך power

המחלקה Player

```

public class Player {
    private Card []hand=new Card[50];
    private int numCard;
    public Player() {
        for(int j=0;j<=49;j++){
            this.hand[j]= new Card(0);}
        this.numCard=0;
    }
    public void addCard ( int c){
        this.numCard++;
        for(int i=this.numCard-1;i>=1;i--){
            hand[i].setPower(hand[i-1].getPower());}
        this.hand[0].setPower(c);
    }
    public void printHand(String s){
        System.out.println("The hand of player "+ s);
        for(int i=this.getNumCard();i>=0;i--){
            System.out.print(" "+this.hand[i].getPower());}
        System.out.println();
    }
    public void subCard (){
        this.numCard--;
    }
    public int getNumCard(){
        return this.numCard;
    }
    public void setNumCard(int n){
        this.numCard=n;
    }
    public Card nextCard(int next){
        return hand[next]; }

} // end of class

```

המחלקה Player מגדירה שחקן שמשחק במשחק. לשחקן שתי תכונות:
מספר הקלפים שברשותו, והערכים של הקלפים בערימה. הנחת הבסיס שלשחקן יש בתחילת המשחק 12 קלפים.
תכונות המחלקה:

```
private Card []hand=new Card[50];  
private int numCard ;
```

שים לב!

"ערימת" הקלפים היא מערך, המערך נוצר כאובייקט ע"י האופרטור new, אולם המערך הוא מערך של עצמים מסוג Card, בשלב זה המערך כולל רק הפניות לאובייקט מסוג Card.

השיטה הבונה:

```
public Player() {  
    for(int j=0;j<=49;j++){  
        this.hand[j]= new Card(o);}  
    this.numCard=0;  
}
```

השיטה בונה יוצרת לכל שחקן 50 אובייקטים מסוג Card, מספר הקלפים נקבע ל-0. יצירת האובייקטים מסוג Card נועדה לאפשר הוספת קלפים לשחקן תוך כדי משחק זאת ע"י שימוש בשיטות המחלקה Card.

השיטה printHand מדפיסה את ערכי הקלפים של שחקן.

```
public void printHand(String s){  
    System.out.println("The hand of player "+ s);  
    for(int i=this.getNumCard();i>=0;i--){  
        System.out.print(" "+this.hand[i].getPower());}  
        System.out.println();  
}
```

שים לב!

המחלקה מקבלת כארגומנט מחרוזת שערכה הוא one או two וזאת כדי ליצור פלט ברור המציג גם למי שייכת ערימת הקלפים לשחקן one או שחקן two כיצד מודפס הערך של קלף בערימה של השחקן?
המשפט : **this**.hand[i].getPower() מציג ערך של קלף באופן הבא:
this – המשמעות התייחסות לאובייקט שזמן את השיטה (אובייקט "השחקן")

this.hand – גישה למערך המייצג את ערימת הקלפים.

this.hand[i] – גישה לאיבר במערך המייצג את הערימה שמשמעותו גישה לאוביקט מסוג

Card

מאחר ו- **this.hand[i]** זה קלף בערימה נוכל להשתמש בשיטה מהמחלקה Card כדי לשלוף

את הערך של הקלף וזו השיטה `getPower()`. ולכן גישה לקלף בתוך הערימה היא ההוראה

הכוללת:

this.hand[i].getPower()

המחלקה game

המחלקה game מבצעת הדמייה של המשחק זאת ע"י הפעלת השיטות של המחלקות Player

ו- Card. ניהול המשחק הוא ללא "התערבות אדם". המשחק יוצר שני שחקנים, לכל שחקן

נוצרת ערימת קלפים ומתנהלים משחקונים. מאחר והמשחק מסתיים כאשר אחד השחקנים

מפסיד יצרתי יתרון לשחקן אחד זאת ע"י הגדלת הערכים של הקלפים שלו לעומת יריבו.

```
public class game {
    public static void main(String[] args)
    {
        Player p1 = new Player();
        Player p2 = new Player();
        for(int k=0;k<12;k++){
            p1.nextCard(k).setPower((int)(Math.random()*7)+1);
        }
        p1.setNumCard(12);
        for(int k=0;k<12;k++){
            p2.nextCard(k).setPower((int)(Math.random()*20)+1);
        }
        p2.setNumCard(12);

        while ((p1.getNumCard())>0 && (p2.getNumCard())>0){
            int c1 = p1.nextCard(p1.getNumCard()-1).getPower();
            int c2 = p2.nextCard(p2.getNumCard()-1).getPower();
            System.out.println("c1 = "+c1);
            System.out.println("c2 = "+c2);
            if (c1>c2){
                p1.subCard();
            }
        }
    }
}
```

```
p1.addCard(c2);
p1.addCard(c1);
p2.subCard();
System.out.println("Player one win with card "+c1 );
}
else
if(c2>c1){
    p2.subCard();
    p2.addCard(c2);
    p2.addCard(c1);
    p1.subCard();
    System.out.println("Player two win with card "+c2 );
}
else
    if(c2==c1){
        p1.subCard();
        p2.subCard();
        p1.addCard(c1);
        p2.addCard(c2);
    }
    p1.printHand("one");
    p2.printHand("two");
    System.out.println("Player one have "+ p1.getNumCard() + "
    cards");
    System.out.println("Player two have "+ p2.getNumCard() + "
    cards");
}
}
}
```

RaceTurtle המחלקה

המחלקה RaceTurtle מהווה תרגיל סיום נאות ואתגרי לתוכנית הלימודים. המחלקה יוצרת 10 אובייקטים מסוג Turtle (צב), הצבים "עומדים" על קו הזינוק ויוצאים למירוץ. הצב הראשון שחוצה את קו הגמר "משתולל" משמחה.

```
import unit4.turtleLib.*;

public class RaceTurtle {
    private Turtle tzavim[]=new Turtle[10];
    private int postion[] =new int [10];
    public RaceTurtle(){
        int i;
        for (i=0;i<=9;i++){
            this.tzavim[i]= new Turtle();
            this.postion[i]=0;
        }
    }
    public void moveToPos(){
        for(int i=0;i<=9;i++){
            this.tzavim[i].tailUp();
            this.tzavim[i].moveForward(i*20);
            this.tzavim[i].turnLeft(90);
            this.tzavim[i].setVisible(true);
        }
    }
    public void race(){
        int finish=100;
        boolean win=false;
        while (win==false){
            for(int i=0;i<=9;i++){
                this.tzavim[i].tailUp();
                int move=(int)(Math.random()*20);
                this.tzavim[i].moveForward(move);
                this.postion[i]=this.postion[i]+move;
                if (this.postion[i]>finish){
                    win=true;
                }
            }
        }
    }
}
```

```

        for(int w=1;w<20;w++){

            this.tzavim[i].turnLeft((int)(Math.random()*120));

            this.tzavim[i].turnRight((int)(Math.random()*120));

        }

    }

    public static void main(String args[]){
        Turtle t=new Turtle();
        t.setVisible(false);
        t.turnLeft(90);
        t.tailUp();
        t.moveForward(100);
        t.turnRight(90);
        t.tailDown();
        t.moveForward(200);
        t.setVisible(false);
        RaceTurtle race1= new RaceTurtle();
        race1.moveToPos();
        race1.race();
    }
}

```

הסבר:

הגדרת המחלקה:

```

private Turtle tzavim[]=new Turtle[10];
private int postion[] =new int [10];

```

תכונות המחלקה:

מערך של אבויקטים מסוג Turtle

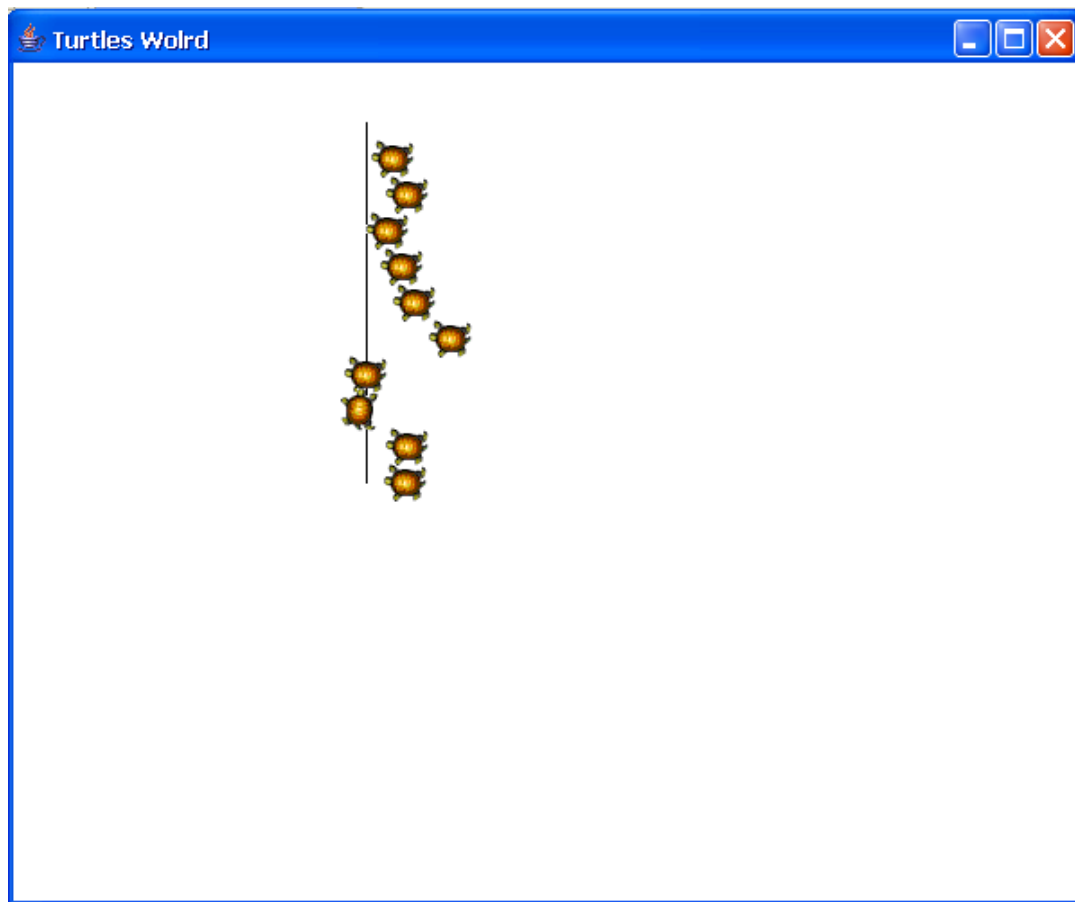
מערך של מספרים שלמים – המערך שומר את מיקומו של כל צב על מסלול "המירוצים"

השיטה הבונה:

השיטה הבונה כוללת יצירה של האובייקטים מסוג Turtle בכל אחד מתאי המערך הראשון tzavim. ואיפוס מיקום הצבים בכל אחד מתאי המערך position .

```
public RaceTurtle(){  
    int i;  
    for (i=0;i<=9;i++){  
        this.tzavim[i]= new Turtle();  
        this.postion[i]=0;  
    }  
}
```

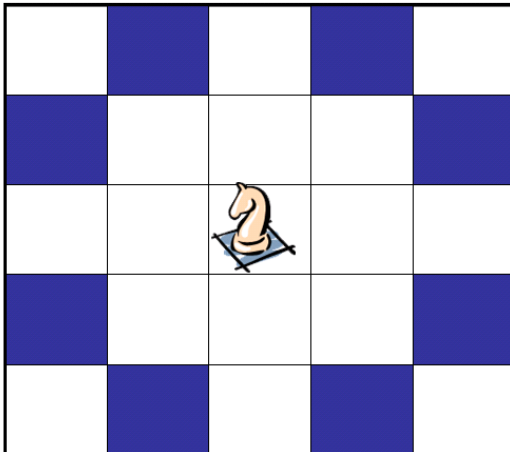
פלט המחלקה RaceTurtl צבים בקו הסיום.



המחלקה Objectsus

סוסופרש במשחק שח נמצא במקום (X,Y) על לוח השח. הסוס יכול לנוע ל-8 מיקומים אפשריים על הלוח.

לוח פרש



המחלקה Objectsus ממחישה את תנועת הסוס באמצעות הדמייה. התוכנית מגרילה את המיקום הראשוני של הפרש, ממיקום זה מבצע הפרש "קפיצות" למיקום הבא הפנוי. הפרש ממשיך "לקפץ" כל מיקום על הלוח שאליו קופץ הפרש מסומן כמיקום תפוס זאת ע"י סימון מס' הצעד.

ההדמיה מסתיימת כאשר הפרש מגיע למיקום שממנו לא ניתן להמשיך "לקפץ" – כלומר כל 8 המקומות מסביבו תפוסים.

המחלקה נכתבה בתוכנית מונחית עצמים בצורה

חלקית, ויכולה להוות תרגיל מסכם שיש בו מרכיבים של תכנות מונחה עצמים וחלק תכנות "רגיל". ניתן לתת לתלמידים לבצע "הסבה" מלאה של המחלקה לתכנות מונחה עצמים מלא.

```
public class Objectsus{
    private int [][]tabel = new int[12][12];
    private int [][] option=new int[8][2];
    private int px=(int)(Math.random()*10)+2;
    private int py= (int)(Math.random()*10)+2;
    private int step;
    public Objectsus(){
        for (int i=0;i<=11;i++)
            for (int j=0;j<=11;j++)
                this.tabel[i][j]=1;
        for (int i=2;i<=9;i++)
            for (int j=2;j<=9;j++)
                this.tabel[i][j]=0;
        this.option[0][0]=-2; this.option[0][1]=-1;
        this.option[1][0]=-1; this.option[1][1]=-2;
        this.option[2][0]=1; this.option[2][1]=-2;
        this.option[3][0]=2; this.option[3][1]=-1;
        this.option[4][0]=2; this.option[4][1]=1;
    }
}
```

```

this.option[5][0]=1; this.option[5][1]=2;
this.option[6][0]=-1; this.option[6][1]=2;
this.option[7][0]=-2; this.option[7][1]=1;
this.step=1;
    tabel[px][py]=step;
}

public void showTabel(){

    System.out.println("gen num " + this.step);
    for(int k=2;k<=9;k++){
        for(int l=2;l<=9;l++ )
            if(this.tabel[k][l]==0)
                System.out.print("- ");
            else
                System.out.print(this.tabel[k][l]+" ");
        System.out.println();
    }
}

public static void main(String[] args){
    Objectsus game=new Objectsus();
    game.showTabel();
    boolean more = false;
    boolean nextMove=true;
    while(nextMove){
        more=false;
        for (int k=0;k<=7;k++){
            int dx=game.option[k][0];
            int dy=game.option[k][1];
            if(game.tabel[game.px+dx][game.py+dy]==0){
                game.step++;
                game.px=game.px+dx;
                game.py=game.py+dy;
                game.tabel[game.px][game.py]=game.step ;
                more = true;
            }
        }
    }
}

```

```
        k=8;
    }
}
if (more==false)
    nextMove=false;
game.showTabel();

}

}

}
```


הסבר

המחלקה כוללת את לוח השח


```
private int [][]tabel = new int[12][12];
```

לוח שח הוא לוח 8X8 כאן בחרתי לוח 12X12, הסיבה נועדה לאפשר סריקה של האפשרויות של הצעד הבא של הסוס גם בגבולות הלוח כלומר בעמודה 1 או עמודה 2 וכן בשורה 1 או 8. הלוח עם 12 שורות ו-12 עמודות מאפשר לקבוע את הלוח הפנימי עם שתי שורות ושתי עמודות כמסגרת חיצונית.

מערך העידכונים קובע איך אנו מחשבים את הצעדים האפשריים של הסוס הנמצא במיקום X,Y.

אם הסוס נמצא במיקום X,Y שמונה המקומות האפשריים ביחס למיקום זה הם:

לוח פרש

	X-2,Y-1		X-2,Y+1	
X-1,Y-2				X-1,Y+2
		 X,Y		
X+1,Y-2				X+1,Y+2
	X+2,Y-1		X+2,Y+1	

מערך העידכון יכול את הערכים שיש להוסיף למיקום הנוכחי כדי לקבל את 8 המיקומים האפשריים.

```
this.option[0][0]=-2; this.option[0][1]=-1;
this.option[1][0]=-1; this.option[1][1]=-2;
this.option[2][0]=1; this.option[2][1]=-2;
this.option[3][0]=2; this.option[3][1]=-1;
this.option[4][0]=2; this.option[4][1]=1;
this.option[5][0]=1; this.option[5][1]=2;
this.option[6][0]=-1; this.option[6][1]=2;
this.option[7][0]=-2; this.option[7][1]=1;
```

לולאת הסריקה מבצעת סיכום אינדקסים של מיקום הנוכחי של הסוס עם הוספת הערכים הנמצאים במערך האפשרויות, בכך אנו מקבלים סריקה של כל הצעדים האפשריים, עבור כל צעד נבדק אם הוא "פנוי", הסוס מבצע "קפיצה" אל המקום הפנוי הראשון. לולאת while מתבצעת כל עוד הסוס "לא נתקע"

```

boolean nextMove=true;
    while(nextMove){
        more=false;
        for (int k=0;k<=7;k++){
            int dx=game.option[k][0];
            int dy=game.option[k][1];
            if(game.tabel[game.px+dx][game.py+dy]==0){
                game.step++;
                game.px=game.px+dx;
                game.py=game.py+dy;
                game.tabel[game.px][game.py]=game.step ;
                more = true;
                k=8;
            }
        }
        if (more==false)
            nextMove=false;
        game.showTabel();
    }

```

הרצת המחלקה מציגה את הפלט הבא (הצגה חלקית בלבד):

הצעד הראשון – המיקום נבחר אקראית.

המיקום הראשון נבחר : 6,2 שים לב נקבע על הלוח הערך 1, כלומר צעד ראשון

```
gen num 1
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
- 1 - - - -
- - - - -
- - - - -
```

"נדלג" לצעד ה-15

```
gen num 15
5 14 - - - - -
- - 6 15 - - - -
7 4 13 - - - -
2 - - - - -
- 8 3 12 - - - -
- 1 - - - - -
9 - - - 11 - - -
- - 10 - - - -
```

ולצעד האחרון- 46 בו הסוס "נתקע"

```
gen num 46
5 14 19 40 - 38 - -
20 17 6 15 36 41 - -
7 4 13 18 39 - 37 -
2 21 16 27 42 35 - -
29 8 3 12 33 - - -
22 1 28 43 26 - 34 -
9 30 45 24 11 32 - -
46 23 10 31 44 25 - -
```

בכל הרצה הסוס יתקע במיקום אחר, מספר הצעדים המקסימלי שהסוס יכול לבצע הוא כמובן 63, נסה והרץ את התוכנית מספר פעמים אולי הסוס שלך יזכה "במירוץ" משימה נוספת בתרגיל זה הוא לארגן מחדש את התוכנית כך שגם המרכיב העיקרי של האלגוריתם שנכתב ב- main ובתכנות "רגיל", יוחלף לתכנות מונחה עצמים. התרגיל הבא מציג עוד בעיה מעניינת ובתרגיל זה כל התוכנית מבוססת תכנות מונחה עצמים.






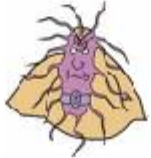
המחלקה LifeGame

החיידיקים חיים במושבה, דור אחר דור מתפתחת המושבה, חיידיקים מתים ונולדים. במושבת החיידיקים שלנו חיידיקים מתים מצפיפות ונולדים כאשר יש מקום ויש מסביב כמה "חברים לתמיכה"

משחק החיים התפרסם מאוד בקרב אנשי המחשבים והישומים של הרעיון היו בתחומים רבים כולל כלכלה, מודל של אוטומט תאי ועוד. המחלקה LifeGame מממשת את הרעיון של המשחק עם הכללים הבאים:

חיידיק מת מצפיפות כאשר יש יותר מ-4 שכנים מסביבו
חיידיק נולד כאשר 3 חיידיקים נמצאים מסביב למקום פנוי.

השכנים של החיידיק מוצגים באיור הבא:

	משבצת שכן	משבצת שכן	משבצת שכן
		 חיידיק במיקום X,Y	משבצת שכן
	משבצת שכן	משבצת שכן	
			

ביישום המחלקה בחרתי ביישום הכולל שמירה של דורות המושבה, מערך של אובייקטים הכולל 30 דורות, כל אובייקט במערך מייצג דור של מושבת חיידיקים. כל המחלקה מיושמת בתכנות מונחה עצמים מלא ומהווה דוגמא מסכמת לתוכנית הלימודים בספר זה.

```

public class LifeGame
{
    private char [][]world = new char [22][22];
    private int generation;
    private LifeGame(){
        int i,j;
        for(i=0;i<=21;i++)
            for (j=0;j<=21;j++)
                this.world[i][j]='-';
        this.generation=0;
    }
    private void start(){
        int x,y;
        for(int bug=1;bug<=100;bug++){
            x=(int)(Math.random()*22);
            y=(int)(Math.random()*22);
            this.world[x][y]='*';
        }
    }
    private void updateGame(LifeGame L){
        int i,j,k,l;
        for( i=1;i<=20;i++){
            for( j=1;j<=20;j++){
                int b=0;
                if (this.world[i][j]=='*'){
                    for(k=i-1;k<=i+1;k++)
                        for (l=j-1;l<=j+1;l++)
                            if (this.world[k][l]=='*')
                                b++;
                }
                if(b>4)
                    L.world[i][j]='-';
            }
        }
    }
}

```

```

    }
}
for( i=1;i<=20;i++){
    for( j=1;j<=20;j++){
        if (this.world[i][j]=='-'){
            int b=0;
            for(k=i-1;k<=i+1;k++)
                for (l=j-1;l<=j+1;l++)
                    if (this.world[k][l]=='*')
                        b++;
            if(b>2)
                L.world[i][j]='*';
        }
    }
}

private void printWorld(){
    int i,j;
    System.out.println("The generation No: "+this.generation);
    System.out.println();
    for(i=1;i<=20;i++){
        for (j=1;j<=20;j++)
            System.out.print(this.world[i][j]);
        System.out.println();
    }
}

private void setGen(int g){
    this.generation=g;
}

private int getGen(){
    return this.generation;
}

```

```
public static void main(String[] args)
{
    LifeGame []play = new LifeGame[30];
    for (int nextGen=0;nextGen<=29;nextGen++)
        play[nextGen]=new LifeGame();
    play[0].start();
    play[0].printWorld();
    for (int nextGen=1;nextGen<=29;nextGen++){
        play[nextGen-1].updateGame(play[nextGen]);
        play[nextGen].setGen(nextGen);
        play[nextGen].printWorld();
    }

}

}
```

השיטות החשובות במחלקה:

private LifeGame()

השיטה LifeGame היא השיטה הבונה, יוצרת את האובייקט מושבת החיידקים של דור מסוים.

private void start(){

השיטה start מאתחלת את הדור הראשון ויוצרת 100 חיידקים המהווים את הדור הראשון.

private void updateGame(LifeGame L)

השיטה update מחשבת את הדור הבא. השיטה מופעלת ע"י מושבה מדור קודם ומעדכנת את האובייקט המייצג את הדור הבא.

private void printWorld()

השיטה printWorld מציגה את מצב מושבת החיידקים בדור מסוים.

.

הרצת התוכנית:

הדור ה-5

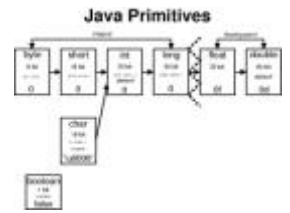
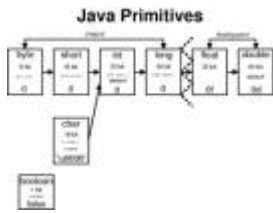
The generation No: 5

```
-----
-----
-----
--*-----
*-*-----
--*-----*--
-----*-*-----*-*
-----*-----*--**
-----*--**-----*--*--
-----***-----****
-----*-*-*-----*-----
-----**-*-----*--*--*
-----*-----**-*--*--
-----***-----*--*--
-----**-*-----*--*--
--*-----*--*--**
-----*--*--*--
***-----*--**--*--
-----*--*--
-----
```

הדור ה-6

The generation No: 6

```
-----
-----
-----
--*-----
--*-*-----
--*-----*--
-----*-----*--*
-----*--**-----*--*--
-----**-----*--**
-----*--**-----*-----
-----*--*--*-----****
-----*--*-----*--*--*
-----***-----*--*--
-----**-*-----*--*--*
-----*-----*--**
-----*--**-----*--*--
-----*--*--*-----*--*--
***-----*--**--*--
-----*--*--*
--*-----*--*--
-----
```

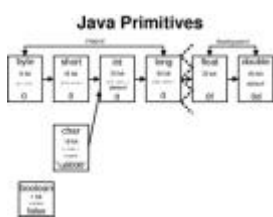



פרק שמיני – הורשה, כימוס ועצמים מורכבים

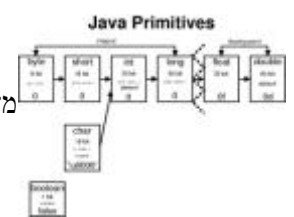
פרק זה הוא פרק ניסוי, הפרק נכתב כבסיס לכתיבה מורחבת על הנושא בחלק ב' של הספר. הנושאים הכלולים בפרק זה אינם מהווים חלק מנושאי הלימוד של תוכנית הלימודים יסודות תורת המחשב.

הפרק כולל נושאים מתקדמים בתכנות מונחה עצמים כולל הורשה המאפשרת ליצור עצמים חדשים היורשים תכונות ושיטות של עצמים בסיסיים יותר. כימוס- הדרך ליצור קוד יותר מוגן בפני שינויים בלתי מורשים ובאותה העת לאפשר גישה לשיטות מחלקה מסויימת ע"י מחלקות אחרות. וכן דוגמאות הממחישות יצירת "עולמים מורכבים"- כלומר אובייקטים שתכונה אחת או יותר שלהם מהווה אף היא אובייקט בפני עצמו.

סטודנטים המשתמשים בספר זה ככלי עזר בקורס מדעי המחשב מבוא לתכנות באוניברסיטאות ומכללות ימצאו בפרק בסיס להבנת נושאים מתקדמים אלו.



מבוא לתכנות מונחה עצמים



פרק שמיני – הורשה כימוס ועצמים מורכבים

בפרק זה אציג את הרעיונות המרכזיים של תכנות מונחה עצמים: הורשה, כימוס וכיצד לשלב בין מחלקות (עצמים מורכבים)

- Encapsulation - כימוס
- Inheritance - הורשה
- הרכבת עצמים

עקרון הכימוס Encapsulation

בדרך כלל נשתדל להגדיר את השיטות פומביות **public** ואת התכונות פרטיות. הסיבה היא שהגדרת השיטות פומביות מאפשרות לשאר המחלקות להשתמש בשיטות אלו, אולם באותה העת מונעות גישה ישירה לתכונות ושינוי התכונות. כאשר משמשים בעצם חייבים להכיר את הפעולות \ שירותים שניתן לבקש מהאובייקט לבצע או מידע שאנו מבקשים לקבל. האופן בו האובייקט מממש את הפעולות נסתר ממבקש השירות.

עקרון האנקפסולציה - הכמסה מהווה עקרון חשוב בתכנות מונחה עצמים. עקרון הכימוס אינו ייחודי לשפת ג'אווה או לתכנות מונחה אובייקטים, אלא, רעיון כללי המאפשר לכתוב מודולים של תוכנה כאשר הממשק של הפונקציות (שיטות) של המודול הזה "חשופות" וידועות למשתמש באותם פעולות, אך באותה העת לא יכול המשתמש לשנות או לפגוע באובייקט שלא דרך הממשק. יצירה מעין "קפסולה" – "כמוסה" של קוד חסינה ל"פגיעות" מבחוץ יוצרת קוד טוב יותר וסיכוי לשגיאות קטן יותר במיוחד בפרויקט תוכנה גדול עליו עובדים מספר אנשים.

רעיון הכימוס מעלה שאלה חשובה. מי יכול לגשת לאיברים ביישום שאנו כותבים? בג'אווה קיימות הרשאות שונות.

פרטי – private

כאשר לפני הגדרת איבר מופיעה המילה **private**, ניתן לגשת לאיבר רק מתוך המחלקות/תחום בו הוא מוגדר ולא ניתן לגשת אליו ממחלקות אחרות/ תחומים אחרים.

פומבי – public

כאשר לפני הגדרת האיבר מופיעה המילה **public**, פירושו של דבר שניתן לגשת לאיבר זה מכל המחלקות ביישום/פרויקט.

מתי נגדיר במחלקה איברים כ- **private**, ומתי נגדירם כ- **public**?

כאשר מגדירים איברים בעלי הרשאה גישה – **private**, אנו "שומרים" על המחלקה. כל איבריה הם פרטיים, המחלקה תהיה "חסינה" בפני ניסיונות גישה "מבחוץ", כל האיברים שלה מוגנים. אולם, באותה העת היא תהיה חסרת ערך לכל המחלקות האחרות, כי לא ניתן לגשת לאיבריה.

כאשר מחלקה תוגדר כציבורית, ניתן יהיה להשתמש בשיטות המחלקה ולעשות שימוש מועיל בכל היישום, אולם יש לשים לב שהגישה לאובייקטים של המחלקה תשמור על עקרון הכימוס, כלומר גישה לאובייקט ותכונותיו תתאפשר רק באמצעות שיטות-ממשק המחלקה.

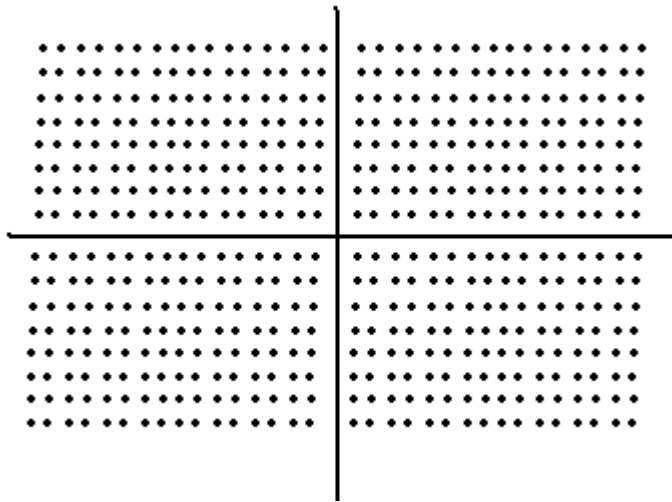
שיתוף־הרכבה של מחלקות שהוגדרו

רעיון עצמים מורכבים הוא הרעיון הפשוט. רעיון זה אינו ייחודי לתכנות מונחה עצמים ובעצם מהווה חלק בלתי נפרד מכל תכנות כולל בשפות תכנות פרוצדורליות. מימוש של רעיון זה הוא ביצירה של מחלקות חדשות תוך כדי שימוש במחלקות קיימות (הגדרת כ- **public**) שנכתבו בידי אחרים או אתה כתבת ובדקת. שימוש ברעיון זה הכולל שיתוף קוד של מחלקות ציבוריות הוצג בדוגמאות בפרקי הספר השונים בדוגמא הבאה אני מסביר את הרעיון בצורה "מסודרת". הדוגמא של המחלקה Malben מציגה את מימוש הרעיון. כדי להגדיר את המחלקה Malben אנו משתמשים במחלקה שהגדרנו Point.

על נקודות ומלבנים

כל מלבן בעולם המלבנים בתרגיל זה צלעותיו מקבילות לצירים.
 לנקודה על המישור יש תכונות: ערך X וערך Y , על המישור אינסוף נקודות, באמצעות שתי נקודות נוכל להגדיר מלבן. פשוט נציין את הנקודה שמאלית התחתונה ואת הנקודה הימנית העליונה. שתי נקודות אלו מגדירים מלבן על המישור באופן חד חד ערכי.

המישור

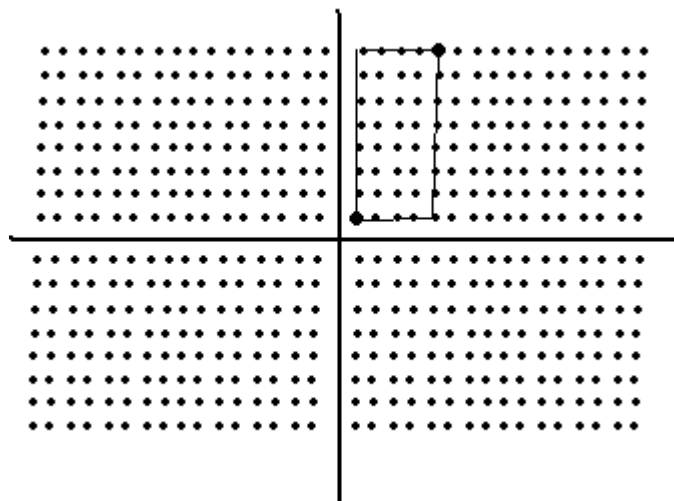


נגדיר מלבן באמצעות שתי נקודות: נקודה שמאלית תחתונה, נקודה ימנית עליונה. נבחר בערכים הבאים:

נקודה שמאלית תחתונה: 1,1

נקודה ימנית עליונה: 5,8

המלבן שנקבל:



שתי הנקודות מגדירות את המלבן המסוים.

לסיכום בנינו מלבן, בניית המלבן התבססה על בניית יחידה בסיסית והיא נקודה, הגדרת המלבן מבוססת על הגדרת הנקודה.

עתה ברצוננו לקבל מלבן ולחשב על-פי שתי הנקודות שמגדירות אותו את אורך המלבן, רוחב המלבן, היקף המלבן ושטח המלבן.

נניחם באמצעות שיטות בג'אוה את המודל של נקודות במישור המגדירות אינסוף מלבנים על המישור באמצעות שתי נקודות (נקודות של קודקודים נגדיים), כולל, חישוב אורך, רוחב המלבן היקף ושטח המלבן

המחלקות בפרויקט

המחלקה Point מגדירה נקודה עם שתי תכונות ערך X של הנקודה וערך Y של הנקודה המחלקה Malben מגדירה מלבן באמצעות אבן הבניין הבסיסית נקודה. לפי הייצוג שבחרנו למלבן ערכי שתי הנקודות הם של קודקודים מנוגדים.

המחלקה Point

```
public class Point {
    private int x=0;
    private int y=0;
    public Point( int x, int y) {
        this.x = x;
        this.y= y;    }
    public int getX() {
        return this.x; }
    public int getY() {
        return this.y; }
    public void setX(int x){
        this.x=x;    }
    public void setY(int y){
        this.y=y;    }
    public static void main (String[] args ) {

    }
}
```

אני משתמש במחלקה Point שהגדרתי כדי ליצור את המחלקה Malben

המחלקה Malben

```
public class Malben {  
    private Point bottomLeft, topRight;  
    Malben(Point p1, Point p2){  
        this.bottomLeft=p1;  
        this.topRight = p2; }  
    public Point getbL(){  
        return this.bottomLeft; }  
    public Point gettR(){  
        return this.topRight; }  
}
```

המחלקה מלבן כוללת שני "חברים" – member שהן שתי נקודות, כלומר מלבן מוגדר באמצעות שתי נקודות, הנקודה "פינה שמאלית תחתונה", ונקודה שנייה "ימנית למעלה", ניתן להגדיר מלבן באמצעות נקודה אחת ואורך ורוחב או הגדרה לא "חסכונית" באמצעות ארבע הנקודות.

המחלקה כוללת שתי שיטות המחזירות את ערכי הנקודות. שים לב! השיטה מחזירה נקודה ולא קורדינטה x או קורדינטה y, אין אנו זקוקים לכך, כי כדי לקבל ערכים אלו נשתמש בשיטות של המחלקה Point.

המחלקה malbenProject

```

public class malbenProject {
public static int length( Malben m){
return Math.abs(m.getbL().getX() - m.gettR().getX());}
public static int width( Malben m){
return Math.abs(m.bottomLeft.getY() - m.topRight.getY());}

public static int area (Malben m){
return length(m) * width(m);    }
public static int perimeter (Malben m){
return 2*(length(m) + width(m));    }
public static void main(String args[]){
int x1,y1,x2,y2;
x1=IO.readInt("bottomLeft x ");
y1= IO.readInt ("bottomLeft y ");
x2= IO.readInt ("topRight  x ");
y2= IO.readInt ("topRight  y ");
Point p1 = new Point(x1,y1);
Point p2 = new Point(x2,y2);
Malben m1 = new Malben(p1,p2);
System.out.println("The value of p1.x is: "+ p1.getX() );
System.out.println("The value of p1.y is: "+ p1.getY() );
System.out.println("The value of p2.x is: "+ p2.getX() );
System.out.println("The value of p2.Y is: "+ p2.getY() );
p1.setX(2);
p1.setY(5);
p2.setX(20);
p2.setY(24);
System.out.println("The value after changing using Point method" );
System.out.println("The value of p1.x is: "+ p1.getX() );
System.out.println("The value of p1.y is: "+ p1.getY() );
System.out.println("The value of p2.x is: "+ p2.getX() );
System.out.println("The value of p2.Y is: "+ p2.getY() );

```

```
System.out.println("The length of rectangle is " + length(m1) );
System.out.println("The width of rectangle is " + width(m1) );
System.out.println(" The area is : " + Math.abs(area(m1)));
System.out.println(" The perimeter is : " + Math.abs(perimeter (m1)));
}
}
```

שיטות במחלקה malbenProject

השיטה length מקבלת כארגומנט מלבן ומחזירה את אורך המלבן.

```
public static int length( Malben m){
return Math.abs(m.getbL().getX() - m.gettR().getX());}
```

חשוב: השיטה "ניגשת" לנקודות המלבן "ישירות" ללא שימוש בשיטות המוגדרות במחלקה Malben, זאת בגלל שהמשתנים – חברים במחלקה Malben הוגדרו כגישה public, כלומר ניתן לגשת לתכונות המחלקה גם ממחלקות אחרות. דרך זו אינה מומלצת אך אני משתמש בה כדי להציג את ההבדל בין הרשאת private להרשאת public, לעומת זאת התכונות של המחלקה Point הוגדרו כ-private ולכן לא ניתן לגשת לערכי x ו-y רק באמצעות השיטות של המחלקה Point וכך זה מתבצע במחלקה malbenProject. השיטה מוצגת גם כדוגמא לתכנות "דמוי" פרוצדורלי ואינה מופעלת ע"י אובייקט כמו בתכנות מונחה עצמים. גם כאן מסיבה פדגוגית בחרתי כמה דוגמאות ליישם ולכתוב שיטות סטטיות. מומלץ להמעיט ולהקטין למינימום צורת תכנות זו, מרבית הספר מיישם תכנות מונחה אובייקטים "טהור", כמה דוגמאות מציגות תכנות "רגיל" כולל שיטות סטטיות, שיטות שאינן מופעלות ע"י אובייקט אלא אובייקט מועבר לשיטה כארגומנט דרך זו לכאורה יותר "קצרה" אולם אינה מומלצת ויש להמעיט ככל האפשר בתכנות "פרוצדורלי".

השיטה **width** מקבלת כארגומנט מלבן ומחזירה את רוחב המלבן.

```
public static int width( Malben m){  
    return Math.abs(m.bottomLeft.getY() - m.topRight.getY());}
```

יצירת שני מופעים של המחלקה Point

```
Point p1 = new Point(x1,y1);  
Point p2 = new Point(x2,y2);
```

יצירת מופע של המחלקה Malben

```
Malben m1 = new Malben(p1,p2);
```

ביצוע שינויים בערכי הנקודות באמצעות שיטות המחלקה Point

```
p1.setX(2);  
p1.setY(5);  
p2.setX(20);  
p2.setY(24);
```

חשוב

גישה לתכונות המחלקה רק באמצעות שיטות המחלקה נקראת "כימוס" כלומר מנגנון המאפשר אי ביצוע שינויים בערכי מופעי המחלקה ממחלקה אחרת, אלא, רק באמצעות שיטות המחלקה. מנגנון "הכימוס" מאפשר שמירה על "בטיחות" הנתונים ואי ביצוע שינויים לא מורשים. אופן גישה זה בא לידי ביטוי גם בפקודות "הפלט" המופיעות בהמשך המחלקה malbenProject

```
System.out.println("The value of p1.x is: "+ p1.getX() );  
System.out.println("The value of p1.y is: "+ p1.getY() );  
System.out.println("The value of p2.x is: "+ p2.getX() );  
System.out.println("The value of p2.Y is: "+ p2.getY() );
```

גישה לערכי הנקודה באמצעות
שיטות המחלקה Point

Class methods – שיטות מחלקה - דוגמאת מימוש

המחלקה Sons

המחלקה Sons מציגה מימוש של משתנה מחלקה.
המחלקה מגדירה אובייקט ילד במשפחה, התכונות עבור כל ילד :
שם הילד
גיל הילד
ומיקום הילד בסדר הילדים במשפחה

משתנה המחלקה הוא מס' הילדים במשפחה

```
private static int numChildren=0;
```

המלה static שמופיעה בהגדרת התכונה numChildren, תפקידה להורות שהתכונה היא תכונה של המחלקה. התכונה numChildren נוצרת עם הגדרת המחלקה ולא בעת יצירת מופע של המחלקה.
ניתן לפנות אליו על-ידי פנייה למחלקה: Sons.numChildren

השיטה הבונה כוללת עדכון "מיקומו" של הילד במשפחה.

```
public Sons(String name,int Age){  
    this.name=name;  
    this.place=numChildren+1;  
    this.Age=Age;  
    numChildren++;  
}
```

התכונה הפרטית של האובייקט החדש place מייצגת את מיקום הילד במשפחה. כל יצירה של אובייקט "ילד" התכונה הפרטית מקבלת את הערך של מס' ילדים במשפחה. ערך זה מעודכן (הוספה של 1) לפעם הבאה שניצור אובייקט "ילד" חדש.
תכונת המחלקה numChildren מעודכנת בכל יצירת אובייקט: numChildren++;

הקוד המלא של המחלקה

```
public class Sons  
{  
    private static int numChildren=0;  
    private int place;  
    String name;  
    int Age;  
    public Sons(String name,int Age){
```

```

    this.name=name;
    this.place=numChildren+1;
    this.Age=Age;
    numChildren++;
}
public String toString(){
    String str="My name is: "+ this.name +" Im in place "+ this.place+ " Age
"+ this.Age;
    return str;
}

public static void main(String[] args) {

}
}

```

שימוש במחלקה Sons

```

public class useSons{
public static void main(String[] args) {
    Sons boy1,boy2,boy3;
    boy1=new Sons("Tal",16);
    boy2=new Sons("Gil",12);
    boy3=new Sons("Ron",7);
    System.out.println(boy1.toString());
    System.out.println(boy2.toString());
    System.out.println(boy3.toString());
    System.out.println("The number of children in the family is: "
+Sons.numChildren);
}
}

```

משימה

הריצו את התוכנית ורשמו את הפלט.

"הוסיפו" שני ילדים למשפחה ובדקו מה הפלט של המשפט:

```

System.out.println("The number of children in the family is: "
+Sons.numChildren);

```

טבלת סיכום והשוואה בין תכונת מופע ותכונת מחלקה

תכונות מופע ותכונות מחלקה

תכונות מופע	תכונות מחלקה
מאפיינות מופע מסוים של המחלקה.	מאפיינות את המחלקה באופן כללי.
לכל מופע עותק משלו של התכונה שנוצר עם יצירת מופע מסוים מטיפוס המחלקה.	קיים עותק אחד בזיכרון שנוצר מיד עם הגדרת המחלקה, עוד לפני יצירת מופע כלשהו.
ניתן לפנות אליהן על ידי פנייה לעצם שינוי ערך התכונה של מופע מסוים אינו משפיע על ערכי התכונה במופעים אחרים	ניתן לגשת אליהם על ידי פנייה למחלקה אם מופע מסוים שינה את ערך התכונה, ערכו ישתקף בכל המופעים
	בהגדרת התכונה תופיע המלה static
מוגדרות תמיד כ- private , פרטיות	יכולות להיות מוגדרות גם כ- public , אבל בצירוף static final , כלומר: ציבוריות של המחלקה אבל קבועות, כך שאי אפשר לשנות אותן.

פעולות מופע ופעולות מחלקה

פעולות מופע	פעולות מחלקה
פעולות שמבצע מופע מסוים של המחלקה ניתן להפעיל אותן רק על מופע קיים.	פעולות שמבצעת המחלקה. ניתן להפעיל אותן גם מבלי לבנות מופעים של המחלקה.
זימון: () שיטה-המופע	זימון: () שיטה-המחלקה

הטבלאות נכתבו ע"י אווי גורןולד ומופיעים בספר באישורה.

הורשה - Inheritance

הורשה היא חלק מרכזי משפות תכנות מונחות עצמים, אולי לא ידעת אבל בכל מחלקה שיצרת השתמשת בהורשה וההורשה הייתה מהמחלקה Object שהיא מחלקה תקנית של שפת ג'אווה. תחביר השיתוף ושימוש במחלקות ע"י שימוש בקוד שלהם הוא פשוט, אולם בהורשה יש לציין את שם המחלקה החדשה, וכן איזו מחלקה המחלקה החדשה יורשת.

תהליך ההגדרה מתחיל בציון שם המחלקה החדשה, המילה extends (המציינת הרחבה) ולאחר מכן שם מחלקת העל- המחלקה ממנה יורשים.

דוגמא:

```
public class Engineer extends Employee{
```

הסבר "טכני" :

המחלקה **Engineer** יורשת את המחלקה **Employee**

הסבר "פשוט":

יש לנו עובדים במפעל, ומהנדס הוא בעצם סוג של עובד, ולכן לאובייקט מהנדס יהיו כל התכונות שיש לעובד במפעל, וכל הפעולות (שיטות) שאנו מבצעים "עם" עובד אפשר לבצע "עם" מהנדס כמו: קידום בדרגה, העלאה במשכורת או לצערנו פיטורין, לעומת זאת, למהנדס יש תכונות נוספות, כמו, איזה מהנדס הוא (מהנדס חשמל, מהנדס מכונות וכדומה), באיזו מעבדה הוא עובד.

ההורשה אם כן מאפשרת לנו לממש את הרעיון הטבעי, שאובייקטים "בעולם האמיתי" שייכים לאובייקטים הכוללים אותם או שאובייקט הוא סוג של אובייקט אחר (לדוגמא פרה היא סוג של יונק), והורשה מהווה מכשיר יעיל למימוש קשרים אלו בעולם הפתרון שלנו.

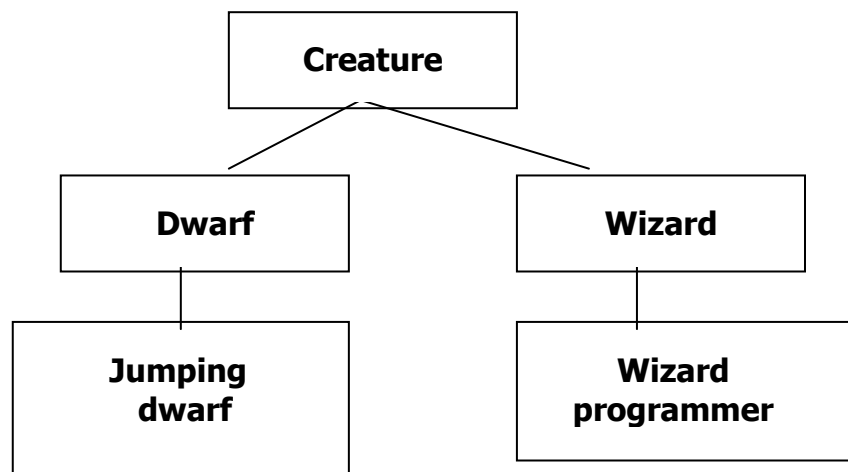
דוגמא: ארץ האגדות *fairy tale*

בארץ "עולם האגדות" יצורים רבים. היצורים שונים זה מזה אך יש להם גם כמה תכונות משותפות.

יכולים להיות זכרים או נקבות ולכולם ממש לכולם, יש שם שמייחד אותם בארץ "עולם האגדות". קבוצת המכשפים היא הקבוצה החשובה והגדולה ביותר ולמכשפים מלבד שם ומין יש גם עוצמה. החלשים ביותר עוצמתם 1, והחזקים עוצמתם 10. בין המכשפים יש את מומחי מחשבים, גזע מיוחד שיודע שפת תכנות ג'אווה, או סי שרפ והם מלבד מעשה קסמים גם מתכנתים בשעות הפנאי.

קבוצת הגמדים היא הקבוצה השנייה בחשיבותה בעולם האגדות והם פשוט נמדדים בגובה, ככל שאתה נמוך יותר אתה גמד מבריק יותר. בין הגמדים יש קבוצת גמדי הקפיצות, גזע מיוחד, שפיתח פיצוי לקומה הנמוכה ע"י קפיצות לגובה. בעולם האגדות, די ברור, שאם נפגשים שני קוסמים, מיד בודקים מי עוצמתו גדולה יותר, ואם נפגשים גמדים קופצניים מיד בודקים מי קופץ גבוה יותר. אבל בעולם האגדות משתדלים לשדך בין היצורים, ולכל נקבה מחפשים שידוך נאה, לא חשוב, אם הוא קוסם או ננס העיקר שיהיה זכר.

The world of fairy tale



אנו רואים הקוסמים והגמדים כולם יצורים של עולם האגדות. והמתכנתים הם הקוסמים המשכילים ואילו הגמדים הקופצניים הם באמת גמדים מדליקים. לכל היצורים יש שתי תכונות:

שם היצור

מין היצור

לקוסמים יש התכונות הבסיסיות של היצורים בעולם האגדות, אולם יש להם גם תכונת העוצמה, ולקוסמים המתכנתים יש הידע בתכנות שפת ג'אווה או סי שרפ.

הגמדים פשוט מדליקים, כי בראש ובראשונה יש להם שם ואפשר לשאול אותם, האם הם גמד או ננס אישה, וגם גמדים הקופצנים הם כאלה מוכשרים ומלבד שם ומין אפשר לשאול אותם לאיזה גובה הם קופצים.

לסיכום תכונות יצורים שלנו בעולם האגדות:

יצורים: שם, מין

מכשף רגיל: שם, מין, עוצמה

מכשף מתכנת: שם, מין, עוצמה, שפת תכנות

גמד רגיל: שם, מין, גובה

גמד קופצני: שם, מין, גובה, קפיצה

השאלה עתה, מה לעולם האגדות ולספר על ג'אוה? והתשובה מאוד פשוטה, כשם שבעולם האגדות, וגם בעולם המציאות שלנו, ישנם אובייקטים רבים וחלק מהאובייקטים, לעיתים, שייכים למשפחה שלמה של אובייקטים, אלא, שיש להם עוד תכונות, וישנה היררכיה של עצמים שמקיימים יחסי הכלה בין האובייקטים.

גמד קופצני הוא סוג של גמד, וגמד הוא סוג של יצור בעולם האגדות.
כך גם מכונית ספורט היא סוג של מכונית ומכונית היא סוג של כלי תחבורה.

inheritance – הורשה

בג'אוה ושפות תכנות מונחות עצמים אחרות קיים מנגנון הורשה, מנגנון זה מאפשר יצירת היררכיה של עצמים. כאשר יש לנו עצם ניתן להגדיר עצם חדש אשר "יורש" את תכונות העצם וגם את כל השיטות של העצם ובנוסף יש לו משלו תכונות נוספות ושיטות משלו.
בניה של עצמים והורשה בין עצמים יוצר עץ היררכי, עץ ירושה של מחלקות.

מחלקה היורשת ממחלקה אחרת נקראת תת-מחלקה (subclass).

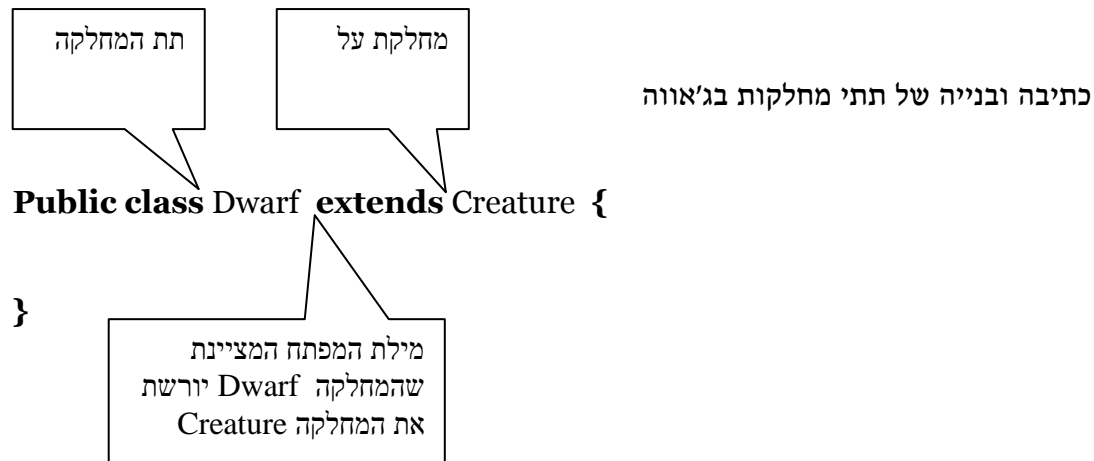
תת המחלקה יורשת את כל החברים שיש למחלקה המורשת. המחלקה המורשת נקראת

מחלקת-על (super class)

כל מחלקה בעץ הירושה היורשת מחלקת על, נקראת צאצא של כל אחת מהמחלקות שמעליה.

(descendant)

בג'אוה לכל מחלקה יכולה להיות רק מחלקת-על אחת. (ישנם שפות מונחות עצמים המאפשרות יותר מחלקות על למחלקה). אין מגבלה על "עומק" עץ ההיררכיה כלומר ניתן ליצור עץ ירושה עם כמה רמות. בדוגמא שלנו "עולם האגדה" יש 3 רמות בעץ הירושה.



השיטה הבונה של תתי מחלקות
תת מחלקה יורשת את כל האיברים של מחלקת העל, כולל השיטות של מחלקת העל.

שאלה

האם תת המחלקה יורשת גם את השיטה הבונה של מחלקת העל?

תשובה

התשובה לא, השיטה הבונה נושאת את אותו השם של המחלקה למחלקה Creature יש שיטה בונה Creature שיוצרת אובייקטים לפי מפרט התכונות של המחלקה, ואינה יכולה לעבור בירושה לתת המחלקות שלה.

שאלה

כיצד יוצרים שיטה בונה של תת המחלקה?

תשובה

מילת המפתח **super** מאפשרת בנייה של שיטה בונה עבור תתי מחלקות. מאחר ובג'אווה אין לאף מחלקה, יותר מאשר, מחלקת-על אחת. ההתייחסות למחלקת על של תת מחלקה היא חד חד ערכית, ושפת ג'אווה מנצלת פשטות זו באמצעות מילת המפתח **super** המחלקה Creature והשיטה הבונה של המחלקה Creature

```
public class Creature {
    private String name;
    private String gender;
    Creature(String n , String g ) {
        this.name = n;
        this.gender = g;
    }
}
```


תת המחלקה Wizard והשיטה הבונה של תת המחלקה Wizard

```
public class Wizard extends Creature {
    private int power;
    public Wizard (String name, String gender, int power ){
        super (name,gender);
        this.power = power;
    }
```

שים לב!

מילת המפתח **super** משתמשת ומפעילה את השיטה הבונה של מחלקת העל, ולשיטה זו מועברים שם היצור (name) ומין היצור (gender) שם תכונות המחלקה Creature ובנוסף החבר השלישי המופיע בכותרת השיטה הבונה של Wizard מקבל ערך ע"י המשפט

this.power = power;

הגדרת תת המחלקה Wizard והשיטה הבונה של תת המחלקה Wizard

```
public class WizardProgrammer extends Wizard {
    private String language;
    public WizardProgrammer (String name, String gender, int power , String
    language) {
        super (name,gender,power);
        this.language = language; }
    public void main(String args[]) {
    }
}
```

שים לב!

מילת המפתח **super** משתמשת ומפעילה את השיטה הבונה של מחלקת העל שבמקרה זה היא Wizard, ולשיטה זו מועברים שם היצור (name) ומין היצור (gender) ועוצמת הכוח של המכשף (power), ובנוסף החבר הרביעי המופיע בכותרת השיטה הבונה של WizardProgrammer מקבל ערך ע"י המשפט **this.language = language;** שהיא שפת התכנות בה מתמחה המכשף חובב המחשבים. תהליך דומה של ירושה מתקיים עבור הגמד המקפץ שהוא גמד וגם יצור בעולם האגדות שלנו.

המחלקה גמד

```
public class Dwarf extends Creature{
    private double height;
    public Dwarf (String name, String gender, double height ){
        super (name,gender);
        this.height = height;    }
    public void main (String args[]){
        } // end of main
    } end of Dwarf class
```

המחלקה הגמד המקפץ

```
public class DwarfJump extends Dwarf{
    private double heightJump;
    public DwarfJump (String name, String gender, double height , double
heightJump ){
        super (name,gender,height);
        this.heightJump = heightJump;    }
    public void main (String args[]){
        }
    } // end of DwarfJump class
```

לצורך הבהרת נושא הירושה, לא הרחבתי את המחלקות שבדוגמא עולם האגדות, ולכן לא צרפתי שיטות למחלקות. נציג עתה את היישום המלא של עולם האגדות. שים לב לשימוש בשיטות של מחלקת-על בתתי המחלקות של מחלקות אלו.

שאלה

מאיזו מחלקה יורשת המחלקה creature?

תשובה

כל מחלקה מעולם האגדות שיצרנו יורשת ממחלקת על, השאלה מאיזו מחלקה יורשת המחלקה Creature, שהרי אין בכותרת המחלקה את המילה השמורה extends. התשובה לשאלה: מחלקת העל של כל המחלקות שאנו יוצרים בג'אווה היא המחלקה Object, מחלקה זו נמצאת בשורש ההיררכיה של כל המחלקות שאנו יוצרים ביישומים שאנו כותבים בג'אווה. המחלקה Object היא כלולה ב- Java API והיא כמובן יורשת את כל השיטות של Java API.

השיטה toString()

השיטה toString() הכלולה במחלקה object מקבלת כאופרנד אובייקט, והופכת אותו למחרוזת. השיטה מופעלת באופן אוטומטי במשפט הפלט System.out.println עבור כל מרכיב במשפט שאנו רוצים לשרשר באמצעות + ולהדפיסו.

דוגמא: קווי נסיעה

הדוגמא הבאה מציגה מימוש נוסף של רעיון ההורשה. קווי נסיעה בתחבורה ציבורית בערים גדולות בעולם כוללים מגוון גדול. בעיר תל אביב ישנם קווי נסיעה בין שתי תחנות מרכזיות. קו הנסיעה כולל כמה תכונות ומגוון תצורות:

תכונות קו נסיעה:

1. מספר הקו
2. תחנת מוצא
3. תחנת יעד

המחלקה Line

המחלקה Line מממשת את העצם קו נסיעה בתל אביב

```
public class Line {
    protected int lineNum;
    protected String firstStation;
    protected String lastStation;
    public Line(int nLine, String first, String last) {
        this.firstStation=new String(first);
        this.lastStation=new String (last);
        this.lineNum=nLine;
    }
    public int getLine(){
        return this.lineNum;
    }
    public String getFirst(){
        return this.firstStation;
    }
    public String getLast(){
        return this.lastStation;
    }
    public static void main(String[] args) {

    }
}
```

הסבר:

הגדרת תכונות המחלקה:

```
protected int lineNum;  
protected String firstStation;  
protected String lastStation;
```

protected המילה השמורה

למדנו על הרשאות הגישה **public** ו-**private**. השימוש בהרשאות גישה מאפשר לנו לממש את הרעיונות הסתרת מידע וכימוס. באמצעות ההרשאה **private** מונעים גישה ישירה לתכונות המחלקה ממחלקות אחרות, ומאידך גיסא, הרשאת הגישה **public** מאפשרים שימוש בשיטות המחלקה ע"י מחלקות אחרות. כאשר אנו מממשים את רעיון ההורשה אנו מעוניינים לגשת לאיברי המחלקה המורשת מתוך המחלקה היורשת וכאן יש לנו בעיה!.

ניתן לפתור את בעיית הגישה ע"י הרשאת **public** לתכונות, אולם בכך אנו מאפשרים "פגיעה" לא מורשית ואפשרות לשינוי ערכי התכונות מכל מחלקה. אם נגדיר את ההרשאה כ-**private** לא נוכל לגשת מהמחלקה היורשת לתכונות המחלקה המורשת. פתרון הבעיה הוא באמצעות ההרשאה **protected** הרשאה זו היא בין ההרשאה **private** להרשאה **public**. הרשאת גישה זו מאפשרת גישה משני מקומות:

1. מתוך המחלקה בה הוגדר האיבר של המחלקה
2. מכל תת-מחלקה (מחלקה יורשת) של המחלקה בה האיבר הוגדר בה.

השיטה הבונה של המחלקה Line

```
public Line(int nLine, String first, String last) {  
    this.firstStation=new String(first);  
    this.lastStation=new String (last);  
    this.lineNum=nLine;  
}
```

השיטה הבונה יוצרת את העצם- קו נסיעה, מאתחלת את שם תחנת המוצא ושם תחנת היעד וכן את מספר קו הנסיעה.

משימה

השלם כמה שיטות המיועדות לשליפה ואיחזור תכונות המחלקה.

המחלקה BusLine

המחלקה BusLine יורשת את המחלקה Line. הרעיון הוא שקו הנסיעה מסויים כולל כמה סוגי תחבורה המשרתים את ציבור הנוסעים כולל: מונית, אוטובוס, רכבת קלה ועוד. כל אחד מכלי התחבורה הללו יש לו את התכונות הבסיסיות של קו הנסיעה שהם: מספר הקו, תחנת המוצא, תחנת היעד (תחנה סופית). ולכן קווי הנסיעה הללו מהווים תתי מחלקות של המחלקה Line. כאן בחרתי ליצור את תת המחלקה BusLine, אתה הלומד מוזמן ליצור תתי מחלקות יורשות עבור קו נסיעה מונית, קו נסיעה רכבת קלה עם התכונות הנוספות היחודיות (לדוגמא ברכבת קלה תהייה תכונה של מס' הקרונות שיש ברכבת).

```
public class BusLine extends Line {
    private int numPass;
    public BusLine(int nLine, String first, String last,int passnger) {
        super(nLine, first, last);
        this.numPass=passnger;
    }
    public int getPassnger(){
        return this.numPass;
    }
    private String toString() {
        return "NumLine #" +this.lineNum+ " First station " +this.firstStation +
        " Last station " + this.lastStation;
    }
    public static void main(String[] args) {
        BusLine bus = new BusLine(34,"Arlozorov","Habima",45);
        System.out.println(bus.toString());
    }
}
```

הסבר:

המחלקה BusLine יורשת את המחלקה Line

```
public class BusLine extends Line
```

השיטה הבונה של המחלקה BusLine כולל שימוש במשפט **super** משפט המאפשר יצירת עצם של המחלקה תוך שימוש בשיטה של מחלקה העל Line בנוסף כוללת השיטה הבונה השמת ערך של מספר נוסעים המקסימלי של האוטובוס שזו תכונה ייחודית של תת המחלקה BusLine ולכן השיטה הבונה נכתבה כך:

```
public BusLine(int nLine, String first, String last,int passnger) {
    super(nLine, first, last);
    this.numPass=passnger;
}
```

הגדרתי במחלקה היורשת שיטה לאיחזור מס' נוסעים, והשיטה toString לקבלת מחרוזת הכוללת את תכונות העצם. אתה הלומד מוזמן להרחיב את המחלקה בשיטות מועילות נוספות.

המחלקה MeasefBus

ידוע שקווי נסיעה באוטובוס בעלי איפיון שונה. ישנם קווי נסיעה הנקראים "אקספרס", קוויים אלו מתחילים בתחנת המוצא ואינם מורידים או אוספים נוסעים בדרך, אלא, עוצרים רק פעם אחת בתחנת היעד. לעומת זאת קו "מאסף" הוא קו נסיעה שמתחיל בתחנת היעד ולאורך קו הנסיעה עוצר בתחנות בניימ עד להגעה לתחנת היעד הסופית. ברור שקו מאסף כולל את כל התכונות של קו נסיעה באוטובוס עם תכונה נוספת שהיא "מספר התחנות לאורך קו הנסיעה". רעיון זה ממומש במחלקה MeasefBus תת מחלקה של המחלקה LineBus

המחלקה MeasefBus יורשת את המחלקה LineBus

```
public class MeasefBus extends BusLine
```

התכונה הנוספת של המחלקה היורשת היא מספר התחנות

```
int numStation;
```

השיטה הבונה של המחלקה MeasefBus

```
public MeasefBus(int nLine, String first, String last, int passnger,int
NumStation)
{
    super(nLine, first, last,passnger);
    this.numStation=NumStation;
}
```

קוד המחלקה המלא

```

public class MeasefBus extends BusLine {
    int numStation;
    public MeasefBus(int nLine, String first, String last, int passnger,int
        NumStation)
    {
        super(nLine, first, last,passnger);
        this.numStation=NumStation;
    }
    public String printLine()
    {
        return "NumLine #"+this.lineNum+ " First station "+this.firstStation + "
        Last station "+ this.lastStation + " NumStation "+this.numStation;
    }
    public static void main(String[] args)
    {
        MeasefBus bus = new MeasefBus(34,"Afeka","Ramat Aviv",34,8);
        System.out.println(bus.printLine());
    }
}

```


פולימורפיזם - polymorphism

פולימורפיזם מאפשר לנו להתייחס לעצמים מטיפוסים שונים באופן אחיד. במפגש משפחתי נפגשו שני אבות, אח, דוד, סבא, אחיין, נכד ושני בנים אבל הם הסתפקו ב- 4 כסאות וכל אחד ישב על כסא.

האם זה יתכן?

התשובה כן!

בפגישה השתתפו סבא, אחיו, בן של הסבא ונכד הסבא. אולם האח של הסבא הוא הדוד של בנו. הסבא ואחיו הם שני אחים, הסבא הוא גם אבא, וגם בנו הוא אבא וכו... בעצם ניתן להסתכל על כמעט כל אחד ממשפטי המפגש בכמה צורות. פולימורפיזם מאפשר לנו להתייחס אל אובייקט בצורות שונות. האובייקט עצמו אינו משתנה אלא רק נקודת המבט שלנו. באופן הזה אנו יכולים לזמן שיטה המוגדרת מחדש (overriding) במחלקות הגזרות (בתת-המחלקות) ואע"פ שאנו מזמנים את השיטות באופן אחיד כל אובייקט יבצע את השיטה כפי שהיא מוגדרת בתת המחלקה בה הוא נוצר.

נציג דוגמא של מחלקה כללית form – צורה, מחלקה זו היא מחלקה עבור צורות הנדסיות, נגדיר מחלקות נגזרות ריבוע, עגול וטרפז וניצור מערך כללי של אובייקטים מסוג צורה, אולם לא ניצור את האובייקטים אלא בשעת ריצה ואז ניצור אובייקטים שונים מהמחלקות הנגזרות וזה אפשרי כי פולימורפיזם מאפשר לי להתייחס לכל הצורות של המחלקות הנגזרות ללא תלות בצורה או ליתר דיוק האפשרות הזו נובעת משני עקרונות הפולימורפיזם:

עיקרון 1: refernce בתורשה – reference למחלקת בסיס יכול בפועל להצביע על עצם ממחלקה נגזרת.

עיקרון 2: נוכל ליישם שיטות וירטואליות, כלומר כאשר נקרא לשיטה שהיא קיימת ונקראת בכל אחת מהמחלקות הנגזרות, השיטה הנקראה היא שיטה וירטואלית, אולם בעת הקריאה לשיטה, היישום המעשי יהיה קריאה לשיטה בהתאם לסוג המחלקה היורשת.

מחלקת האב – Form

```
public class Form
{ private String name;
public Form(String n) {
    this.name=n;
}
public double area() {
    return 1;
}
public double perimeter(){
    return 1;
}
public void print(){
    System.out.println(this.name);
    System.out.println("The print method from the super class");
}
}
```

החברים במחלקה Form

תכונה שם הצורה מסוג String

שתי שיטות המשמשות לשוב שטח והיקף, במחלקה Form קבעתי שיחזירו ערך "סתמי" 1.

```
public double area() {
    return 1;
}
public double primeter() {
    return 1;
}
```

השיטה print במחלקת הארב מדפיסה את התכונה היחידה של המחלקה שם האובייקט.

```
public void print(){
    System.out.println(this.name);
}
```

עתה נציג את המחלקות הנגזרת של Form, כולל מערך צורות שבזמן הרצת התוכנית מגדיר את סוג האובייקט שכל הפנייה של המערך תצביע בסופו של דבר.

הגדרה מחדש של שיטות (Method's Overriding)

גישה לשיטה מוסתרת באמצעות super

הגדרה מחדש של שיטה אינה מבטלת את השיטה של מחלקת האב, וניתן לגשת להגדרה המקורית של השיטה של מחלקת האב דרך המופעים (האובייקטים) של מחלקת האב וגם בתוך הקוד של המחלקה היורשת באמצעות המילה השמורה super. לדוגמא שימוש בשיטה print בדוגמא של עובדים בארגון שתוצג בהמשך בה יש שימוש בשיטה של מחלקת האב וגם

overriding – הגדרה מחדש של שיטות במחלקות הגוזרות.

ההגדרה מחדש של שיטות זה מנגנון של ג'אווה בדומה לשפות מונחות עצמים אחרות המנגנון מאפשר לאובייקטים של תת מחלקה לפעול באופן שונה ממחלקת האב (כל מחלקה "מעל" המחלקה הגוזרת), זהו מנגנון המאפשר לאובייקטים להתנהג אחרת מהתנהגות אובייקטים של מחלקות האב.

אם ברצוננו לממש את ההגדרה מחדש עלינו לודא שחתימות השיטות זהות (וזאת בשונה למנגנון העמסת שיטות) המימוש יהיה שונה. חשוב לציין שניתן לשנות את הרשאת הגישה בשיטה החדשה זאת בגלל העבודה שהרשאת הגישה אינה מהווה חלק מחתימת שיטה, אולם לא ניתן להקבוע הרשאת גישה במחלקה היורשת יותר מצומצמת מהרשאת הגישה במחלקת האב. (אם במחלקת האב ההרשאה הייתה private ניתן להגדרה במחלקה הגוזרת כ- public

או protected ולא לכיוון השני שהרשאת הגישה במחלקת האב היתה public לא נוכל

להגדירה במחלקה הגוזרת כ-private או protected.

בדוגמאות ראינו את השימוש במילה השמורה super המאפשרת לנו עדיין למרות שהוגדרה שיטה חדשה במחלקה הגוזרת שהסתירה את השיטה המקבילה במחלקת האב נוכל להשתמש בשיטה כפי שהוגדרה במחלקת האב וזאת ע"י super.

שימוש בשיטה מוסתרת toString של שיטה במחלקת האב: super.toString();

המחלקה היורשת Circle מהמחלקה Form

```
public class Circle extends Form{
    private int radius;
    public Circle (int r) {
        super("cccc");
        this.radius=r; }
    public int getR(){
        return this.radius; }
    public double area()
        super.print(); {
        return Math.PI*Math.pow(this.radius,2);
        public double perimeter (){
            super.print();
            return this.radius*2*Math.PI;        }
    }
```

השיטה הבונה, משתמשת בקריאה לשיטה הבונה של המחלקה Form, שים לב קריאת השיטה הבונה של מחלקת האב חייבת להיות ההוראה הראשונה בשיטה הבונה של המחלקה היורשת.

```
public circle (int r)
{   super("cccc"); // קריאה לשיטה הבונה של מחלקת האב
    this.radius=r;
}
```

השיטות של חישוב ההיקף וחישוב השטח של המחלקה הגוזרת דורסות (overriding) את השיטות של מחלקת האב Form.

```

public double area()
{
    super.print(); // שימוש בשיטה של מחלקת האב
    return Math.PI*Math.pow(this.radius,2);
}

public double perimeter ()
{
    super.print();
    return this.radius*2*Math.PI;
}

```

המחלקה **Square** היורשת מהמחלקה **Form**

בדומה למחלקה circle גם מחלקה זו כוללת שתי שיטות לחישוב שטח והיקף הדורסות את השיטות של מחלקת האב Form, שים לב לשיטה הבונה של Square העושה שימוש בשיטה בונה של מחלקת האב Form עם המילה השמורה super, והפעלת השיטה הבונה של מחלקת האב חייבת להיות הפקודה הראשונה בשיטה.

```

public class Square extends Form
{
    private int size;
    public Square (int x){
        super("ssss");
        this.size=x
    }
    public double area()
    {
        super.print();
        return this.size*this.size; }
    public double primeter ()
    {
        super.print();
        return this.size*4; }
}

```

המחלקה Trapez היורשת מהמחלקה Form

```
public class Trapez extends Form {  
    private int bigBase;  
    private int smallBase;  
    private int high;  
    public Trapez(int b, int s, int h)  
    { super("Trapez");  
        this.bigBase=b;  
        this.smallBase=s;  
        this.high=h; }  
    public double area(){  
        super.print();  
        return (this.bigBase+this.smallBase)*this.high/2;  
    }  
}
```

המחלקה Worker

דוגמא זו לקוחה מהספר ג'אווה על כוס קפה.

```
public class Worker
{
    String    name;
    int       ID;
    float     salary;

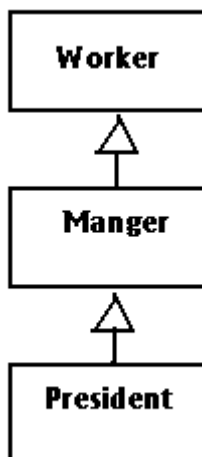
    public Worker (String n,int id, float sal) {
        name=n;
        ID= id;
        salary = sal;
    }

    public void print(){
        System.out.print(name + ", ");
        System.out.println("ID =" + ID + "salary= " + salary);
    }
}
```

לאובייקט במחלקה 3 תכונות: שם, מספר מזהה, ושכר. כלומר עבור כל עובד בארגון יש רישום של 3 המאפיינים הללו ללא תלות במקצועו או תפקידו. המחלקה כוללת את השיטה print המדפיסה את פרטי העובד.

המחלקות היורשות של Worker

העובדים שהם מנהלים - `public class Manger extends Worker`
 העובד שהוא נשיא החברה והוא גם מנהל בחברה וגם עובד ולכן מחלקה זו יורשת מהמחלקה `public class President extends Manger - Manger`



עץ הירושה הזה מוצג בתרשים הבא:
 כל האנשים בארגון הם עובדים, חלק מהעובדים הם מנהלים, ועובד אחד שהוא גם מנהל הוא הנשיא של הארגון.

המחלקה Manger היורשת מהמחלקה Worker

למנהל כל התכונות של עובד, אולם יש לו תכונה נוספת והיא רשימת עובדים עליה הוא ממונה, ולכן בשיטה הבונה אנו רואים שימוש בפקודה `super` ובכך יוצרת אובייקט עובד, ובהמשך מתווספת תכונה נוספת של מנהל והיא רשימת העובדים שלו.

```
public Manger(String n, int id, float sal, Worker [] wlist) {  
    super(n, id, sal);  
    worker_list= wlist; }
```

למחלקה Manger יש גם שיטה `print`, שיטה זו דורסת – `overriding` את השיטה `print` של מחלקת האב ובך מגדירה אותה מחדש. ההגדרה כוללת קריאה לשיטה `print` של מחלקת האב עם ההנראה `super.print()` והמשך הוראות שמטרתם להדפיס את התכונה הנוספת של המנהל והיא הדפסת רשימת העובדים שלו.

public class Manger extends Worker

```
{
    Worker [] worker_list;
    // constructor
    public Manger(String n, int id, float sal, Worker [] wlist) {
        super(n, id, sal);
        worker_list= wlist; }
    public void print ()
    { super.print(); // call Worker.print
      System.out.println();
        System.out.println(" The manager ");
        System.out.println("Worker list: ");
        for (int i=0;i<worker_list.length;i++){
            worker_list[i].print(); // call worker.print
        }
        System.out.println();
    }
}
```

נשיא החברה הוא גם מנהל בחברה וגם עובד – מחלקת נשיא החברה, לנשיא תכונות של מנהל בחברה, ותכונה נוספת רשימת המנהלים עליה אחראי הנשיא.

פוליפורמיזם מבוסס על המרות כלומר כאשר מסתכלים על עובד במפעל כבן אדם או מבצעים המרה כלפי מעלה, כאשר מסתכלים על העובד כמנהל במפעל או מבצעים המרה כלפי מטה. או "נעים" למעלה ולמטה **בעץ הירושה**. בדוגמא שלנו הסתכלות על מנהל כעובד היא המרה כלפי מעלה, לעומת זאת הסתכלות על עובד כנשיא הארגון זו המרה כלפי מטה. עצמים מטיפוסים שונים יכולים להיות מועברים לאותה שיטה ללא צורך לבדוק מהו הטיפוס של הפרמטר שמועבר. אם יש לנו שיטה שמקבלת כפרמטר אובייקט מסוג עובד או יכולים להעביר לה מהנדס, מנהל, נשיא בחברה שכולם עובדים בחברה ולכן אין בעיה להעביר כפרמטר כל אחד מהאובייקטים הללו שהם תתי מחלקות של המחלקה **עובד** לשיטה שנניח מדפיסה את פרטי העובד הבסיסיים.

```
public class President extends Manger
{
    Manger [] manger_list;
public President(String n, int id, float sal, Worker[] wlist, Manger []
mlist) {
        super(n, id, sal, wlist);
        manger_list=mlist;
    }
public void print(){
    super.print();
        System.out.println(" Manager list: ");
        for (int i=0; i<manger_list.length; i++){
            manger_list[i].print();
        }
        System.out.println();
    }
}
```

בשיטה הבונה יש קריאה לשיטה הבונה של מחלקת האב – Manger
super(n, id, sal, wlist);
השיטה print המגדירה מחדש את השיטה הזו יש קריאה לשיטה print של המחלקה
Manger ושם כזכור יש קריאה של השיטה print של המחלקה Worker וכך מודפסים
לבסוף פרטי הנשיא, פרטי המנהלים שתחתיו ופרטי העובדים.
המחלקה WorkerApp, ממחישה את המכלול ע"י יצירת אובייקט נשיא, כמה אובייקטים של
מנהלים ועובדים בארגון.
המחלקה אף מדפיסה ע"י קריאה של print של המחלקה נשיא את כל עובדי הארגון. גם
במחלקת יישום זה הוגדר מערך כללי של אובייקטים של המחלקה Worker אולם כמו בדוגמא
עם מערך Form רק בהמשך עם הרצה הוגדרו חלק מההפניות הכלולות במערך שיצביעו
במקום על אובייקט של מחלקת האב, יצביעו על אובייקט של מחלקה נגזרת.

המחלקה - WorkerApp

```

public class WorkerApp
{
    Worker    marketing[] = new Worker[5];
    Worker    engineering[] = new Worker[4];
    Worker    office[] = new Worker[2];
    Manger managers [] = new Manger[3];
    President president;

    public WorkerApp(){
        marketing[0] =new Worker ("moshe",123,2345);
        marketing[1] =new Worker ("david",124,3345);
        marketing[2] =new Worker ("tal",125,2545);
        marketing[3] =new Worker ("gil",126,4345);
        marketing[4] =new Worker ("ron",127,5645);
        engineering[0]=new Worker ("mor",128,3454);
        engineering[1]=new Worker ("ran",131,3354);
        engineering[2]=new Worker ("miki",141,3494);
        engineering[3]=new Worker ("haim",151,3424);
        office[0]=new Worker ("yael",345,4546);
        office[1]=new Worker ("yafa",346,4556);
        managers[0] = new Manger ("or",23,123456,marketing);
        managers[1] = new Manger ("orit",24,233456,engineering);
        managers[2] = new Manger ("natan",25,253456,office);
        president= new President("ben",1,567877,marketing,managers);
    }

    public static void main(String args[]){
        WorkerApp factory= new WorkerApp();
        System.out.println(" The president ");
        System.out.println();
        factory.president.print();    }
}

```

הגדרה מחדש של שיטות overriding

גישה לשיטה מוסתרת באמצעות super

הגדרה מחדש של שיטה (דריסה – overriding) אינה מבטלת את הגדרת השיטה במחלקת – העל לאחר ההגדרה מחדש עדיין ניתן להשתמש בשיטה המקורית: או דרך המופעים (האובייקטים) של המחלקה המקורית או בתוך הקוד של המחלקה היורשת באמצעות המילה השמורה super אבל לא ניתן לפנות לשיטה המקורית ממחלקות אחרות(מחלקות לא יורשות) גם דרך ובאמצעות מופעים של המחלקה היורשת (כי לא ניתן להגיע באמצעות המילה השמורה super).

בשתי הדוגמאות ראינו שימוש ברעיון ה- overriding בדוגמא של צורות הנדסיות כתבנו שית שיטות לחישוב שטח והיקף הצורה ההנדסית. הגדרנו במחלקת האב Form את השיטות והגדרנו את השיטות מחדש בכל אחת מהמחלקות הגוזרות. ביישום של עובדים בארגון הגדרנו את השיטה print המדפיסה את פרטי העובד. שיטה זו הוגדרה מחדש במחלקות הגוזרות Manger ו- President. אולם עדיין השתמשנו גם במחלקות הגוזרות בשיטה המקורית ע"י המילה השמורה super באופן הבא:

```
super.print();
```

וגם ע"י זימון השיטה המקורית באמצעות מופעים של המחלקה המקורית באופן הבא:

```
worker_list[i].print();
```

קריאה לשיטה המקורית print של המחלקה Worker באמצעות המופע של המחלקה Worker

```
manger_list[i].print();
```

קריאה לשיטה המקורית print של המחלקה Manger

באמצעות מופע של של המחלקה Manger

המרות – דוגמא מפורטת

מחסן חברת "אלוני" למוצרי קרמיקה מנהל את המלאי, במחסן מוצרים מגוונים. הוחלט להגדיר את המערכת והוגדרו מחלקות שונות המייצגות את מוצרי החברה. קבוצה מרכזית של מוצרי החברה היא מרצפות חיפוי מלבניות. התכונות המשותפות לכל מרצפות אלו הם: אורך המרצפת, רוחב המרצפת.

המחלקה **caramic**

מרצפת מלבנית בסיסית במחסן "אלוני"

```
public class caramic {
    private int length;
    private int width;
    public caramic(int a, int b){
        this.length=a;
        this.width=b;
    }
    public void setWidth(int a){
        this.width=a;
    }
    public void setL(int a){
        this.length=a;
    }
    public int getw(){
        return this.width;
    }
    public int getl(){
        return this.length;
    }
    public void print(){
        System.out.println("This printing in caramic class");
        System.out.println("The width is " + this.width);
        System.out.println("The lenght is " + this.length);
    }
}
```

המחלקה כוללת שיטה print להדפסת תכונות המרצפות וכן שיטות לשליפה ושינוי ערכי התכונות.

המרצפות במחסן מחולקות לשני סוגים מרכזיים, מרצפות המשמשות לחיפוי רצפה, ומרצפות המשמשות לחיפוי קירות.

המחלקה wallcarmic

ממשת את קבוצת המרצפות במחסן זה.

```
public class wallCarmic extends caramic{
    public int color;
    public char type;
    public wallCarmic(int a, int b, int c, char t){
        super(a,b);
        this.color=c;
        this.type=t;
    }
    public char getType(){
        return this.type ;
    }
    public int getColor(){
        return this.color;
    }
    public void setColor(int c){
        this.color=c;
    }
    public void setType(char c){
        this.type=c;
    }
    public void print(){
        System.out.println("This printing in wallCramic class");
        super.print();
        switch (this.color){
            case 1: System.out.println("green"); break;
            case 2: System.out.println("yellow"); break;
```

```

case 3: System.out.println("blue"); break;
case 4: System.out.println("red"); break;
case 5: System.out.println("brown"); break;
default: System.out.println(" Worng color ");
}
System.out.println();
switch (this.type) {
case 'f': System.out.println("portzlan"); break;
case 's': System.out.println("stone"); break;
case 'g': System.out.println("glass"); break;
case 'p': System.out.println("plastic"); break;
default: System.out.println(" Worng type ");
}
}
}

```

המחלקה כוללת תכונות מחלקת האב אורך ורוחב ותכונות נוספות צבע המרצפת וסוג החומר. השיטה הבונה כולל "הפעלת" השיטה הבונה של המחלקה ceramic עם המילה השמורה super, פקודה זו חייבת להיות הפקודה הראשונה בשיטה הבונה של המחלקה היורשת. המחלקה כוללת שיטה print שיטה זו דורסת ומגדירה מחדש את השיטה print של מחלקת האב. שים לב לשימוש בשיטה print של מחלקת האב ceramic באמצעות המילה השמורה super. השיטה עושה שימוש בהוראה switch כדי לייצג תכונות המהוות קודים בהדפסה של מחרוזת מובנת יותר.

המחלקה item

```
public class item extends wallCarmic {
    int catalogNum;
    double price;
    public item(int a,int b,int c,char type,int cat,double p) {
        super(a, b, c,type);
        this.catalogNum=cat;
        this.price=p; }

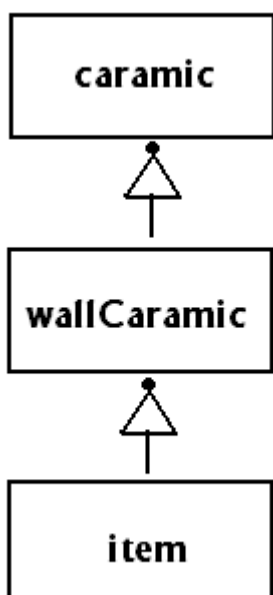
    public void print(){
        super.print();
        System.out.println("The catalog number is:"+ this.catalogNum);
        System.out.println("The price is:" + this.price);
    }
}
```

המחלקה item יורשת את המחלקה wallcarmic ומייצגת פריט במחסן עם תכונות של מספר קטלוגי ומחיר הפריט.

השיטה הבונה משתמשת בשיטה הבונה של מחלקת wallcarmic

```
public item(int a,int b,int c,char type,int cat,double p) {
    super(a, b, c,type);
    this.catalogNum=cat;
    this.price=p; }
```

המחלקה כוללת את השיטות לגישה ושינוי מספר קטלוגי ומחיר, אין צורך להגדיר שיטות לגישה ושינוי של שאר התכונות כי ניתן להשתמש בשיטות של מחלקות האב זאת בסיוע המילה השמורה super.



עץ ההורשה של המחלקות

מחלקת העל ceramic
מחלקות גוזרות:

wallCaramic
item

על עץ הירושה ניתן לנוע לשני הכיוונים, לבצע המרה כלפי מטה שפירושו למחלקות גוזרות וכלפי מעלה למחלקות אב. כאשר יש לנו הפנייה ניתן להציע המרה בטוחה כלפי מעלה כי יש לנו מחלקת אב לעומת זאת חייבים לבדוק כאשר עושים המרה כלפי מטה האם ש לנו מחלקה גוזרת. המחלקה carmicApp מציגה שימוש בהמרות וכן באופרטור הבוליאני instanceof אופרטור זה מאפשר לנו בזמן ריצה האם הפנייה מסויימת היא מטיפוס מסוים ומידע זה בעת ריצת תוכנת יכול לסייע למנוע שגיאה של המרה כלפי מטה.

האופרטור instanceof

באופן כללי השימוש באופרטור instanceof יראה כך:

<object> instaceof <class>

האופרטור יחזיר ערך "אמת" אם העצם הנתון מקיים את אחד התנאים הבאים:

- מופע של המחלקה הנתונה
- מופע של מחלקה הנגזרת מהמחלקה הנתונה
- מופע של מחלקה המממשת את המחלקה הנתונה

המחלקה carmicApp

```
public class carmicApp {
    public static void main (String args[]){
        carmic c=new carmic(6,8);
        wallCarmic cwall= new wallCarmic(12,5,4,'s');
        item diana = new item(12,5,4,'s',34735,34.56);
        c.print();
        cwall.print();
        diana.print();
        System.out.println("Printing after changing reference");
        System.out.println("=====");
        // casting up
        carmic newc= (carmic)diana;
        newc.print();
        // casting down
        wallCarmic newWallStone =(wallCarmic) newc;
        newWallStone.print();
        // The instanceof operator
        if (newc instanceof carmic)
            System.out.println("newc is carmic object ");
        if (newc instanceof wallCarmic)
            System.out.println("newc is not wallCarmic object");
    }
}
```

המרות

המשתנה Diana מכיל הפניה לעצם של המחלקה item, בהוראה:

```
caramic newc= (caramic)diana;
```

אנו מבצעים המרה כלפי מעלה ויוצרים הפניה לאובייקט מהמחלקה ceramic, יצירת הפנייה זו מאפשרת לנו נקודת הסתכלות שונה על העצם עצמו ויכולת להשתמש באופן ישיר בשיטות של המחלקה ceramic.

```
System.out.println("The width is: " + newc.getw());
```

פלט ההוראה: The width is: 5

ההמרות מממשות את עקרון הפולימורפיזם שהוא שינוי נקודת ההסתכלות על עצם, העקרון מאפשר לנו באמצעות מנגנון ההמרות להסתכל על עצם מנקודת מבט שונה. המרה כלפי מטה

Newc הפניה לעצם של מחלקת האב ceramic בהמרה הבאה אנו רוצים לשנות את נקודת ההסתכלות על העצם מנקודת מבט שונה.

```
wallCarmic newWallStone =(wallCarmic) newc;
```

```
newWallStone.print();
```

שינוי נקודת המבט מאפשר לנו להשתמש בשיטה print של מחלקה מסוג wallCarmic

העמסת שיטות – Overloading

שיטות בעלות שם זהה אך עם מספר פרמטרים, או פרמטרים מסוג שונה, יכולות להימצא באותה מחלקה. לעיתים אנו נדרשים לאותה שיטה אך עם מספר פרמטרים או סוג פרמטרים וסדר הופעתם שונה וג'אוה מאשרת מצב זה הנקרא "העמסת שיטות", כי החתימה של השיטות שונה.

דוגמא:

```
public int big(int sum, int tax );  
public double big ( double sum , double tax ) ;
```

שתי השיטות מקבלות שני מספרים ומחזירות את הערך הגדול, אולם שיטה אחת מקבלת שני מספרים מטיפוס int, והשיטה השנייה מקבלת שני מספרים מסוג double

דוגמא: השיטה round הכלולה במחלקה Math

[round](#)(double)

Returns the closest long to the argument.

[round](#)(float)

Returns the closest int to the argument.

שתי שיטות בעלי שם זהה אך עם חתימה שונה מממשים את המנגנון שנקרא "העמסת שיטות"

אנו רואים שתי שיטות עם אותו השם – round כאשר חתימת השיטות שונה. אנו מרבים להשתמש ברעיון של העמסת שיטות בעת כתיבת שיטה בונה, במקרים רבים נכתוב יותר משיטה בונה אחת מצב המאפשר לנו גמישות ביצירה ואתחול של אובייקטים גם שיטה בונה מעתיקה מהווה העמסת שיטה.

תרגילים

בתרגילים של פרקים 3-6 בחרתי לכתוב את כל המחלקות עם שיטות הקלט של המחלקה IO, התרגילים קצרים ומטרתם לתרגול רעיונות פשוטים והחלטתי לא לכלול הגדרה של אובייקט המחלקה Scanner וכו.. כל לומד יכול לי בחירתו להעתיק פתרון התרגיל ולשנות את צורת הקלט. יש בכך יתרון זו דרך נוספת לתרגל נושא של מחלקה, הגדרת עצם ושימוש בשיטות העצם.

מבחר תרגילים לפרקים 3-6, בעקרון כל התרגילים המופיעים במגוון הספרים כולל ספרי מדעי המחשב בסביבת פסקל ניתן לפתור בשינויים של נסוח ושימוש במונחים ורעיונות הקשורים לעולם שפת ג'אווה ותכנות מונחה עצמים.

תרגילים – פרק 3

1. פתח ויישם אלגוריתם שיקלוט למשתנה ציון תלמיד ויוסיף לתלמיד 15% לציונו.
2. פתח ויישם אלגוריתם שיקלוט מחיר מוצר ואחוז הנחה, השיטה main תחשב ותדפיס את מחיר המוצר לאחר ההנחה.
3. פתח ויישם אלגוריתם שיקלוט מספר ממשי וידפיס את החלק השלם של המספר
4. פתח ויישם אלגוריתם שיקלוט ערך למשתנה num1 ערך למשתנה num2 על השיטה main להחליף את הערכים בין המשתנים.
5. נתונה המחלקה הבאה:

```
class ch3E5 {  
    public static void main (String[] args){  
        int num1;  
        int num2,x;  
        double sum;  
        num1=5; num2=12;  
        sum = num1 + num2;  
        System.out.println(sum);  
        sum = num1/num2;  
        System.out.println(sum);  
        x= num1/num2;  
        System.out.println(x);  
        System.out.println(x+1);  
    }  
}
```

```
System.out.println(x+5);
```

```
} // end of main
} // end of ch3E5
```

מה יהיה פלט השיטה main? הרץ בטבלת מעקב

6. פתח ויישם אלגוריתם שיקלוט משקל של משתתף בתוכנית הרזיה. במשך 3 ימים מדד המשתתף את משקלו ומצא שביום הראשון ירד ב- 5 ק"ג, ביום השני ירד ב- 6 ק"ג, ביום השלישי ירד ב- 7 ק"ג. על השיטה main להדפיס את משקל משתתף תוכנית ההרזיה בכל יום.
7. פתח ויישם אלגוריתם שקולט שני מספרים ומחשבת את המספר העוקב של סכום שני המספרים
8. פתח ויישם אלגוריתם שקולט את העומק במטרים שנמצא צוללן ומחשב ומדפיס את הלחץ האטמוספירי שבו נמצא הצוללן. שים לב, כל צלילה לעומק של 10 מטר מגדילה את הלחץ האטמוספירי ב-1 אטמוספירה. בגובה פני הים הלחץ הוא 1 אטמוספירה.
9. פתח ויישם אלגוריתם שיקלוט משקל של גוף בטונות וידפיס את המשקל בק"ג.
10. פתח ויישם אלגוריתם שיקלוט רדיוס מעגל וידפיס את שטח המעגל והיקפו.
11. פתח ויישם אלגוריתם שיקלוט מספר ומדפיס את ההופכי של המספר, את המספר הנגדי של המספר.
12. פתח ויישם אלגוריתם מספר ממשי ומדפיס את החלק העשרוני של המספר.
13. נתונה המחלקה הבאה:

```
class ch3E14 {
    public static void main (String[] args){
        int x = -36;
        System.out.println(Math.abs(x));
        System.out.println(Math.sqrt(Math.abs(x)));
    } // end of main
} // end of ch3E14
```

מה יהיה פלט השיטה main

14. פתח ויישם אלגוריתם שיקלוט את אורכי שני ניצבים במשולש ישר זווית ויחשב את אורך היתר במשולש. רמז: יישם באלגוריתם את משפט פיתגורס.

תרגילים – פרק 4

1. פתח ויישם אלגוריתם שיקלוט 3 מספרים וידפיס את המספרים מהקטן לגדול.
2. פתח ויישם אלגוריתם שיקלוט מספר וידפיס את הערך המוחלט של המספר ללא שימוש בשיטה `abs`.
3. פתח ויישם אלגוריתם אלגוריתם שיקלוט שני מספרים וידפיס הודעה האם שני המספרים עוקבים.
4. מה מבצעת המחלקה `ch4E4`.

```
class ch4E4 {
    public static void main (String[] args){
        double x,y,z;
        x= IO.readDouble ("enter the first number");
        y= IO.readDouble ("enter the second number");
        z= IO.readDouble ("enter the third number");
        if (x>y)
        {if( x>z)
        {
            System.out.println(x);
            if(y>z)
            {
                System.out.println(y);
                System.out.println(z); }
            else
            { System.out.println(z);
              System.out.println(y); }
        }
        else
        {
            System.out.println(z);
            if(x>y)
            {System.out.println(x);
              System.out.println(y); }
            else
            {System.out.println(y);
              System.out.println(x); }
```



```

    }
}
else
{ if (y>z)
  { System.out.println(y);
    if(x>z)
    {System.out.println(x);
      System.out.println(z);}
    else
    {System.out.println(z);
      System.out.println(x);} }
else
{ System.out.println(z);
  if (y>x)
  {System.out.println(y);
    System.out.println(x); }
  else
  {System.out.println(x);
    System.out.println(y); }
}
}
} // end of main
} // end of ch4E4

```

5. אם פתרת את תרגיל 4, ודאי ברור לך שהפתרון מסורבל. השתמש באופרטורים הלוגיים and ו-or כדי לממש את פתרון הבעיה האלגוריתמית של תרגיל 4 בצורה פשוטה יותר.
6. פתח ויישם אלגוריתם שיקלוט 3 מספרים, האלגוריתם יחשב את הממוצע ויבדוק אם הממוצע > 56 , יש להדפיס הודעה "נכשלת", אם הציון בתחום 56-80 תודפס הודעה "אתה משתפר", אם הציון < 90 תודפס הודעה "יפה מאוד". (אפשר להדפיס באנגלית)
7. כתוב שיטה main שקולטת 3 מספרים ומדפיסה הודעה אם כל המספרים שווים
8. כתוב שיטה main שקולטת מספר ומדפיסה הודעה אם המספר זוגי
9. כתוב שיטה main הקולטת 3 מספרים, אם סכום המספרים שווה למכפלתם יש להדפיס הודעה מתאימה
10. כתוב שיטה main הקולטת מספר בטווח 0-99 ומדפיסה את המספר העוקב, אם המספר < 99 השיטה תדפיס את המספר 0.

11. כתוב שיטה main הקולטת מספר שלם תלת ספרתי ומדפיסה הודעה "שוות" אם כל ספרות המספר שוות, והודעה "לא שוות" אם לא.
12. בעל חנות בגדים פרסם את המודעה הבאה:
מחיר חולצה 120 ₪
אם קונים 1-5 חולצות מקבלים 5% הנחה
אם קונים 5-10 חולצות מקבלים 10% הנחה
אם קונים יותר מ-10 חולצות מקבלים 20% הנחה
כתוב שיטה main שקולטת את כמות החולצות שרכש לקוח ומדפיסה את התשלום.
13. מספר תלת ספרתי נקרא "זהב" אם סכום ספרותיו = למכפלת הספרות, כתוב שיטה main שקולטת את המספר ומדפיסה הודעה אם המספר הוא "זהב"
14. מספר נקרא פילנדרום אם ערכו מימין לשמאל = לערכו משמאל לימין. לדוגמא המספר 131 הוא פילנדרום. כתוב שיטה main שקולטת מספר שלם תלת ספרתי ומדפיסה הודעה "פילנדרום" אם אכן המספר עונה להגדרה.
15. כתוב שיטה main שקולטת מספר שלם ומדפיסה הודעה אם המספר מתחלק ב-7 ללא שארית.
16. תלמיד החליט לקרא ספר קריאה במשך שלושה ימים. מספר העמודים בספר 230 עמוד. כתוב שיטה main הקולטת כמה עמודים קרא התלמיד בכל יום, ומדפיסה הודעה האם התלמיד סיים לקרא את הספר בתום שלושת הימים.
17. כתוב שיטה main שקולטת מחרוזת ומספר. יש להדפיס הודעה אם מס' התווים במחרוזת גדול מהמספר שנקלט.
18. כתוב שיטה main שקולטת שתי מחרוזות ומדפיסה את סכום אורך שתי המחרוזות.
19. כתוב שיטה main הקולטת מספר שלם ומדפיסה הודעה "מתחלק גם וגם" אם המספר מתחלק ללא שארית ב-5 וב-2.
20. כתוב שיטה main שקולטת 4 מספרים ומדפיסה את המספר הגדול ביותר.
21. קבוצת נוסעים הגיעה לנמל כדי לצאת למסע ימי. לרשות הקבוצות עומדות ספינות קטנות המסיעות 50 נוסעים כל אחת, וספינות גדולות המסיעות 70 נוסעים. מחיר השכרה של ספינה קטנה הוא 1200 ₪ וספינה גדולה הוא 1500 ₪. הקבוצה רשאית לבחור ספינות רק מסוג אחד. כתוב שיטה main הקולטת את מספר הנוסעים בקבוצה ומדפיסה את ההודעה "ספינות קטנות" אם כדאי לקבוצה לבחור את סוג הספינה הקטנה או "ספינות גדולות" אחרת.

22. מה מבצעת המחלקה ch4E22

```
class ch4E22 {
    public static void main (String[] args){
        int x;
        int n1,n2,n3;
        int i,j,k;
        x= IO.readInt ("enter the number");
        n1=x / 100;
        n2=(x / 10 ) % 10;
        n3=x % 10;
        System.out.println( n1 + " " + n2 + " " + n3);
        if ((n1==n2) && ( n2==n3))
            System.out.println(" yes "); else
            System.out.println(" no ");
        } // end of main
    } // end of ch4E22
```

23. מה מבצעת המחלקה ch4E23

```
class ch4E23 {
    public static void main (String[] args){
        int x;
        int n1,n2,n3;
        x=IO.readInt("enter the number");
        n1=x / 100;
        n2=(x / 10 ) % 10;
        n3=x % 10;
        if ((n1 * n2 * n3) == ( n1+n2+n3))
            System.out.println("Yes");
        else
            System.out.println("No");
        } // end of main
    } // end of ch4E23
```

תרגילים – פרק 5

1. פתח ויישם אלגוריתם שקולט 2 מספרים שלמים num1 ו- num2 ומדפיס את num1 בחזקת num2 ללא שימוש בשיטה של המחלקה Math
2. חובב סביבוני פנה אליך ובקש תוכנית מחשב שתדמה את סיבוב הסביבון, על התוכנית לבצע הדמייה של סיבוב הסביבון n פעמים. על התוכנית לחשב כמה פעמים הסביבון נעצר על האו "נ" כמה על האות "ג" כמה על האות "ה" וכמה על האות "פ".
3. כתוב שיטה main המחשבת ומדפיסה את 10 האיברים של טור הערכים הבא: 1,4,9,16....
4. כתוב שיטה main המחשבת ומדפיסה את 10 האיברים של טור הערכים הבא: 1,3,5,7....
5. כתוב שיטה main המחשבת ומדפיסה את 10 האיברים של טור הערכים הבא:
$$1/1, 1/3, 1/5, \dots$$
6. הנוסחא הבאה מחשבת את קירוב לערך של $i/2^p$. השתמש בנוסחא לחישוב ערך של pi. הנוסחא : $1/2^p \pi = 2/1^2 - 2/3^2 + 4/3^4 - 6/5^2 + 6/5^4 - 8/7^2 + 8/7^4 - 9/9^2 + \dots$
7. כתוב שיטה main שתקלוט 10 מחרוזות ותדפיס את המחרוזת עם מספר התווים הגדול ביותר.
8. בכתה י' 30 תלמידים. מחנכת הכיתה הציעה לתלמידים לבחור הצגה מתוך מבחר של 3 הצגות. פתח ויישם אלגוריתם שיקלוט עבור כל תלמיד את בחירתו (1 או 2 או 3) בסיום קליטת בחירת התלמידים יש להדפיס עבור כל הצגה כמה תלמידים בחרו אותה.
9. כתוב שיטה main המחשבת את סכום המספרים האי זוגיים בתחום 1-100.
10. כתוב שיטה main שקולטת מחרוזת ומדפיסה את כל התווים במחרוזת הנמצאים במיקום זוגי.
11. תלמיד כתה י' למד במהלך ההשנה 10 מקצועות. בכל מקצוע התלמיד נבחן ב- 6 בחינות. פתח ויישם אלגוריתם שיקלוט את ציוני התלמיד במקצועות השונים וידפיס את הממוצע של התלמיד בכל מקצוע ואת הממוצע הכללי.
12. שני תלמידים החליטו לשחק את המשחק "אבן נייר או מספריים", פתח ויישם אלגוריתם שידמה את המשחק. יש לאפשר במשחק לבחור את 'מס' סיבובי המשחק. כלול במשחק סיכומי נקודות לגבי כל שחקן בתום סיבובי המשחק ששחקו.
13. פתח ויישם אלגורים הקולט סדרה של מספרים תלת ספרתיים ומחשב את סכום כל הספרות של ספרת היחידות, סכום הספרות של ספרת העשרות וסכום הספרות של ספרת המאות. האלגוריתם יסיים לקלט מספרים כאשר יקלט מספר שמספר ספרותיו אינו 3.
14. בבית החולים ממליצים ליולדת לשקול את הרך הנולד בכל יום. פתח ויישם אלגוריתם שיקלטו את משקל התינוק במהלך 30 הימים הראשונים. על האלגוריתם לחשב את ממוצע המשקל של התינוק במהלך 30 הימים, את כל המדידות בהם עלה התינוק ביותר מ-50 גרם בהשוואה ליום הקודם.

15. תלמיד התערב עם חברו שאם ישליך 2 קוביות 360 פעמים הוא בודאי יקבל יותר מ-10 פעמים את התוצאה 6 בשתי הקוביות. פתח וממש אלגוריתם שיבדוק האם התלמיד יזכה בהתערבות..
16. חידה: חיפושית נפלה לבור שעומקו 13 מטרים. החיפושית האומללה טפסה בכל יום 3 מטרים במעלה הבור, אולם, בלילה בעייפותה כי רבה הדרדרדה החיפושית המסכנה 2 מטרים במורד הבור. פתח ויישם אלגוריתם שיחשב וידפיס את היום בו החיפושית תמצא על פני האדמה (מחוץ לבור).
17. כתוב שיטה main הקולטת תו ובודקת האם התו הוא אות, אם כן השיטה תדפיס את כל התווים מהתו A ועד לתו שנקלט
18. כתוב שיטה main שתחשב את סכום המספרים בטווח 1-1000 שמתחלקים ב-3 ו-5 ללא שארית.
19. במרכז הספורט במלון באילת החלו למלא את הבריכה לקראת חופשת הקיץ. ברז המילוי ממלא את הבריכה בקצב הולך וגדל, בכל יום הברז מכפיל את תפוקתו. ביום הראשון המערכת ממלאת 1 מ"ק מים, ביום השני 2 מ"ק ביום הרביעי 4 מ"ק וכך הלאה. כתוב שיטה main שקולטת את נפח הבריכה ומדפיסה את היום בו הבריכה תתמלא.
20. כתוב שיטה main הקולטת מחרוזת ומדפיסה את המחרוזת הפוך. לדוגמא אם נקלטה המחרוזת: COMPUTER הפלט יהיה "RETUPMOC".
21. כתוב שיטה main הקולטת מחרוזת ומדפיסה הודעה אם המחרוזת היא פילנדרום, להזכירך מחרוזת פילנדרום היא מחרוזת שניתן לקרוא מימין לשמאל ומשמאל לימין. דוגמא למחרוזת פילנדרום: "sus".
22. מה מבצעת המחלקה ch5E23

```

class ch5E23 {
    public static void main (String[] args){
        int t,n1=1,n2=1;
        int i,n;
        n= IO.readInt ("enter the number");
        System.out.println(n1);
        System.out.println(n2);
        for(i=1; i<=n;i++)
        { System.out.println(n1+n2);
          t=n2;
          n2=n1+n2;
          n1=t; }
        } // end of main
    } // end of ch5E23

```

24. כתוב שיטה main המדפיסה את המשולש הבא:

```

1
12
123
1234
12345
.....
.....
123456....n
    
```

הערך n יקלט מהקלט.

25. מה הפלט של המחלקה הבאה, בנה טבלת מעקב שתעקוב אחר הביצוע.

```

class ch5E25 {
    public static void main (String[] args){
        int i,j,n;
        n= IO.readInt ("enter the number");
        for(i=1; i<=n;i++){
            for (j=1; j<=n; j++){
                System.out.print(i*j+ " ");
                System.out.println(); }
            } // end of main
    } // end of ch4E25
    
```

תרגילים – פרק 6

שים לב! האינדקס הראשון במערך הוא 0. המקום האחרון במערך הוא $\text{length}-1$.

1. עבור ההגדרה הבאה:

```
int [] data = {3,5,-7,9,12,12,9,-7,5,3};
```

בצע את הסעיפים הבאים.

- כתוב לולאה להשמת הערך 0 בכל אחד מאיברי המערך.
 - כתוב לולאה להכפלה ב-2 של ערכו באיברי המערך.
 - כתוב לולאה להצגה כפלט של חצי מערכו של כל איבר במערך.
 - שיטה שמחשבת את סכום המערך
 - שיטה שמדפיסה את המספר הגדול ביותר במערך
 - שיטה המחזירה אמת אם כל האיברים במערך שונים מ-0 ושקר אחרת.
 - שיטה שתחשב ותדפיס את סכום כל האיברים הזוגיים במערך
 - שיטה שתחשב ותדפיס את סכום כל האיברים הנמצאים במיקום זוגי במערך.
 - שיטה שמחזירה את המיקום של מספר x במערך
 - שיטה שמחליפה את הערך בכל תא במערך במספר שהוא הנגדי לערך בתא
2. נתונה ההגדרה הבאה:

```
int [][] data = new int [10][10];
```

בהנחה שהוצבו במערך מספרים שלמים בצע את הסעיפים הבאים.

- שיטה המחשבת ומדפיסה את סכום כל אחת מהשורות
 - שיטה המחשבת ומדפיסה את סכום כל אחת מהעמודות
 - שיטה המחשבת ומדפיסה סכום כל אחד משני אלכסוני המערך
 - שיטה הבודקת האם המערך הוא מערך סימטרי. מערך הוא סימטרי אם הוא מקיים את השוויון: $\text{data}[i][j] = \text{data}[9-i][9-j]$ לכל i, j בתחום 0-9.
 - שיטה הקולטת מספר x ומחזירה כמה פעמים המספר נמצא במערך.
3. נתונה רשימת קלט של 30 מספרים שמהווים טמפרטורה שנמדדה בהר חרמון בחודש ינואר. ציין האם נחוץ מערך לביצוע כל אחד מן החישובים הבאים:
- מספר הימים שנמדדה בהם טמפרטורה מתחת ל-0
 - הטמפרטורה הממוצעת שנמדדה בחודש ינואר.
 - מספר הימים בהם הטמפרטורה שנמדדה הייתה נמוכה מהטמפרטורה ביום האחרון של החודש.
 - מספר הימים בהם נמדדה טמפרטורה מתחת ל-0.
 - מספר הימים בהם נמדדה הטמפרטורה 0 מעלות.
4. בכתה י' בצעו ניסוי, כל אחד מ-30 תלמידי הכיתה זרק 10 פעמים קוביה. נתוני זריקות הקוביה של כל אחד מהתלמידים הוכנסה למערך. עבור כל סעיף כתוב קטע שיטה למימוש.
- הנח שנתונה ההגדרה הבאה: `int cube[][] = new int [30][10]`

- א. כמה פעמים הופיעה התוצאה 6.
- ב. כמה פעמים הופיעה כל אחת מהתוצאות (1-6)
- ג. אם תלמיד קבל נקודת זכות עבור כל תוצאה 6, מהו מספרו של התלמיד שקיבל את מספר הנקודות הגדול ביותר.
5. נתונה ההגדרה הבאה: `int bin [][] = new int[9][9]` במערך הושם הערך 0 או 1 בכל התאים. בצע את המשימות הבאות:
- א. כתוב קטע קוד הבודק האם יש באחת משורות המערכת את סדרת הערכים 101.
- ב. מדען מחשבים ספר לך שהמערך מייצג קוד סודי, כל אחת מהשורות מייצגת מספר. כל שורה מהווה מספר בינארי בן 10 ספרות. עליך לחשב את הערך העשרוני של כל אחת מהשורה.
6. מה מבצעת המחלקה `ch6E6`, רשום טבלה דו מימדית 3×3 המייצגת את ערכי המערך כאשר הפלט הוא one וטבלה המייצגת את ערכי המערך כאשר הפלט הוא two
- ```
class ch6E6 {
 public static void main (String[] args){
 int [][] data = new int[3][3];
 int i,j;
 int n=2;
 for (i=0; i<=n; i++){
 for (j=0;j<=n; j++)
 { data[i][j]=InputRequestor.requestInt("next number");}
 int s=0;
 for(i=0; i<=n;i++) {
 if (data[i][i]==1)
 s++;
 }
 if (s==3)
 System.out.println("one");
 s=0;
 for(i=0; i<=n;i++) {
 if (data[i][2-i]==1)
 s++;
 }
 if (s==3)
 System.out.println("two");
 } // end of main
 } // end of ch6E6
}
```





## נספחים

### נספח א – ספריית מחלקות

ספריית מחלקות פותחו כדי לתמוך בעבודה בסביבת ג'אווה, חלק מהספריות מהוות חלק מסביבת הפיתוח ( ספריות שנכתבו ע"י עובדי Sun ). שאר הספריות נכתבו ומשווקות ע"י חברות צד שלישי תמורת תשלום או ע"י אנשים "טובים", וניתנות להורדה בחינם או במחיר נמוך.

המחלקה String לדוגמא אינה מהווה משפת ג'אווה, אלא נכתבה ע"י עובדי חברת Sun. ספריית מחלקות מאורגנות באשכול של מחלקות הקשורות זו בזו ונהוג לקרוא להם Java APIs - Application Programmer Interface לדוגמא Java Swing API שכוללת אוסף מחלקות ליצירת גרפיקה ועיצוב ממשקי משתמש ביישומים. אוסף מחלקות סטנדרטיות ב-Java גם הן מאוגדות לחבילת מחלקות הנקראת Package , לדוגמא המחלקה String מהווה חלק מחבילת מחלקות הנקראת java.lang ,

#### טבלת חבילות המחלקות הזמינות במרבית סביבות הפיתוח

| החבילה – Package | החבילה תומכת בפעולות                                                                                      |
|------------------|-----------------------------------------------------------------------------------------------------------|
| java.awt         | תומכת בפעולות גרפיות לעיצוב ממשקי תוכנה                                                                   |
| java.applet      | יוצרת תוכניות שניתן להבעיר בין מחשבים ברשת האינטרנט                                                       |
| java.beans       | מגדירה מרכיבי תוכנה כך שניתן לשלבם בקלות ביישומים שונים                                                   |
| java.io          | תומכת בפעולות קלט פלט מגוונות                                                                             |
| java.leng        | חבילה בסיסית ונדרשת לכל יישום, אין צורך לצרפה לישום באופן מיוחד אלא מצורפת באופן אוטומטי ע"י סביבת הפיתוח |
| java.math        | כוללת שיטות רבות לביצוע פעולות מתמטיות ( מספר אקראים, sin,cos וכו.. )                                     |
| java.net         | תמיכה בפעולות ברשת                                                                                        |
| java.sql         | תמיכה בעבודה מול בסיסי נתונים                                                                             |
| java.text        | עיצוב טקסטים המיועדים להדפסה                                                                              |
| java.util        | תמיכה בעזרים כללים                                                                                        |
| java.rmi         | תמיכה ביצירת תוכניות שניתן לבזרם על מספר מחשבים                                                           |
| Java.security    | קביעת נוהלי בטיחות נתונים                                                                                 |

אוסף החבילות בטבלה זמין במרבית סביבות הפיתוח של java כדי לתמוך בפיתוח יישומים מורכבים. חלקם נדרש גם ביישומים פשוטים כמו מרבית התוכניות בספר זה, וחלק מהספריות תשתמש רק אם תכתוב יישומים מורכבים ומסחריים.

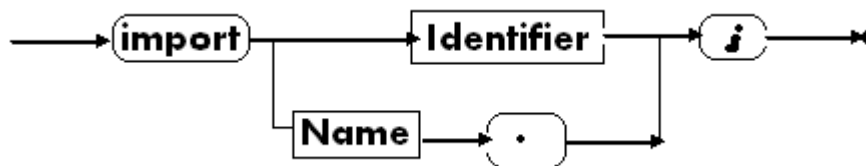
כדי להשתמש באחת מחבילות הספרייה יש ל"חבר" אותה ליישום שאתה כותב, וזאת באמצעות משפט מיוחד המופיע בתחילת היישום, שאתה כותב- import declaration. משפט זה מגדיר באופן מדויק את המחלקות שהתוכנית שלך תשתמש בהם. לדוגמא:

## **import java.util.Random**

משפט זה משמעותו "יבוא" של השיטה Random לתוכנית כך שתוכל להשתמש בה. במידה וברצונך להשתמש בכל השיטות הכלולות בחבילה util עליך לציין זאת עם הסימן כוכבית באופן הבא:

## **import java.util.\***

משפט זה מאפשר לך לגשת לכל המחלקות הכלולות ( וכמובן לשיטות הכלולות במחלקות אלו ( בחבילה util מבנה משפט import



דוגמאות:

זו החבילה כוללת את המחלקה Scanner בה אנו משתמשים לקלט

```
import java.util.*;
```

```
import cs1.Keyboard;
```

```
import java.lang.*;
```

החבילה Keyboard אינה חלק מהמחלקות הסטנדרטיות של java ויתכן שאינה כלולה בכל סביבות הפיתוח.



## נספח ב- שיטות הספרייה Math

העשרה והרחבה – המחלקה Math  
במרבית פרקי הספר השתמשתי בשיטות השייכות למחלקה Math, השיטה random, השיטה  
abs ושיטות אחרות.  
להלן קבוצת השיטות השייכות למחלקה Math. שים לב! כל שיטות המחלקה Math הן  
שיטות סטטיות.

### [abs\(double\)](#)

Returns the absolute value of a double value.

השיטה מקבלת מס' מספר מסוג double ומחזירה מספר מסוג double המהווה את הערך  
המוחלט של המספר שהתקבל.

### [abs\(float\)](#)

Returns the absolute value of a float value.

שיטה זוהי בפעולתה, הארגומנט המתקבל והערך המוחזר מסוג float

### [abs\(int\)](#)

Returns the absolute value of an int value.

שיטה זוהי בפעולתה, הארגומנט המתקבל והערך המוחזר מסוג int

### [abs\(long\)](#)

Returns the absolute value of a long value.

שיטה זוהי בפעולתה, הארגומנט המתקבל והערך המוחזר מסוג long

### [acos\(double\)](#)

Returns the arc cosine of an angle, in the range of 0.0 through  $\pi$ .

**asin**(double)

Returns the arc sine of an angle, in the range of  $-pi/2$  through  $pi/2$ .

**atan**(double)

קבוצת שיטות זו  $\arccos$ ,  $\arcsin$ ,  $\arctan$  מקבלת ארגומנט שערכו ברדיאנים בטווח  $0 - \pi$  ומחזירה את הערך של הארגומנט בהתאמה לסוג השיטה

**cos**(double)

Returns the trigonometric cosine of an angle

השיטה מקבלת מספר מסוג double ומחזירה את  $\sin$  הזווית שמספר זה מייצג

**exp**(double)

Returns the exponential number  $e$  (i.e., 2.718...) raised to the power of a double value.

השיטה מקבלת מספר מסוג double ומחזירה את  $e$  (מספר אי רציונלי שערכו 2.718..) בחזקת מספר זה

**log**(double)

Returns the natural logarithm (base  $e$ ) of a double value.

השיטה מקבלת מספר מסוג double ומחזירה את  $\log$  המספר לפי הבסיס  $e$

Returns the greater of two double values.

**max**(float, float)

Returns the greater of two float values.

**max**(int, int)

Returns the greater of two int values.

**max**(long, long)

Returns the greater of two long values.

קבוצת שיטות אלו מקבלות שני מספרים ( מסוג float או int או long ) ומחזירות את המספר הגדול.

[min](#)(double, double)

Returns the smaller of two double values.

[min](#)(float, float)

Returns the smaller of two float values.

[min](#)(int, int)

Returns the smaller of two int values.

[min](#)(long, long)

Returns the smaller of two long values.

קבוצת שיטות אלו מקבלות שני מספרים ( מסוג float או int או long ) ומחזירות את המספר הקטן

[pow](#)(double, double)

Returns of value of the first argument raised to the power of the second argument.

השיטה מחזירה את הערך של הארגומנט הראשון בחזקה של הארגומנט השני.

[random](#)()

Returns a random number between 0.0 and 1.0.

שיטה שמייצרת מספרים אקראיים בטווח 0-1 ( לא כולל 1 ) שים לב ניתן כפי שבצענו בתרגילים רבים ליצור מספרים ואף תווים בטווחים שונים ע"י יצירת ביטויים שונים בהתאם לצורך ושימוש ב- char ו- int להפיכת המספר לתו או לשלם בהתאמה.

[round](#)(double)

Returns the closest long to the argument.

[round](#)(float)

Returns the closest int to the argument.

שיטות המקבלות ארגומנט ומחזירות את השלם הקרוב למספר שהתקבל.

[sin](#)(double)

Returns the trigonometric sine of an angle.

[tan](#)(double)

Returns the trigonometric tangent of an angle.

שיטות אלו מחשבות את הערך של הפונקציות הטריגונומטריות  $\tan$ ,  $\sin$  של הזווית שהתקבלה

[sqrt](#)(double)

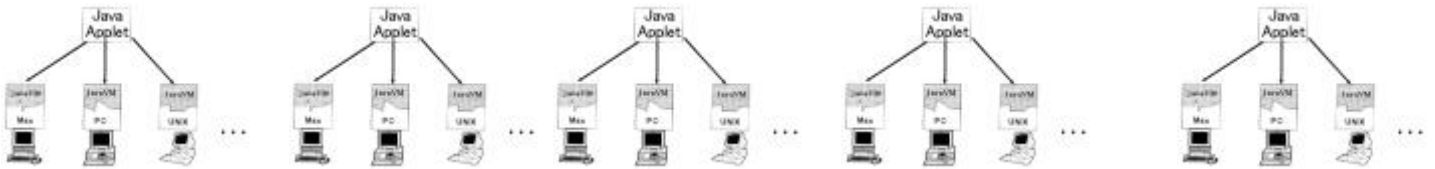
Returns the square root of a double value.

השיטה מחזירה את השורש הריבועי של הארגומנט שהתקבל.

[sqr](#)(double)

Returns the square root of a double value.

השיטה מחזירה את ריבוע הארגומנט שיתקבל.



## נספח ג – המחלקה String - שיטות המחלקה

בנספח זה נציג חלק מהשיטות של המחלקה String. שים לב, השיטות מופעלות ע"י עצמים של המחלקה, כלומר מחרוזות שהוגדרו כאובייקטים של המחלקה String.

### • charAt(int)

Returns the character at the specified index

השיטה מחזירה את התו במיקום המבוקש במחרוזת  
דוגמא:

```
String st = new String("macabi tal aviv ");
```

```
System.out.println(st.charAt(5));
```

הערך שיוחזר הוא i, שים לב i אינו התו ה-5 במחרוזת אלא התו ה-6, זאת משום שהתו הראשון נמצא במיקום 0,

### • compareTo(String)

Compares two strings lexicographically.

השיטה מבצעת השוואה בין שתי מחרוזות, ההשוואה מתבצעת לפי סדר מילוני לקסיקוגרפי, כלומר השוואה של התו הראשון במחרוזת אחת לתו הראשון במחרוזת השנייה, התו השני במחרוזת האחת לתו השני במחרוזת השנייה וכך הלאה.  
דוגמא:

```
String st1 = new String("dogs");
```

```
String st2 = new String("cats");
```

```
String st = "william";
```

```
System.out.println(st1.compareTo(st2));
```

### • concat(String)

Concatenates the specified string to the end of this string.

השיטה מבצעת שרשרת שתי מחרוזות, בדומה לאופרנד +

דוגמא:

```
String st1 = "william ";
String st2 = "farjun ";
System.out.println(st1.concat(st2));
```

### endsWith(String)

**Tests if this string ends with the specified suffix.**

השיטה מחזירה ערך בוליאני, הערך יהיה true אם מחרוזת String מהווה חלק האחרון של מחרוזת הראשונה.  
דוגמא:

```
String st1 = "william farjon";
String st2 = "farjon";
boolean yes;
yes = st1.endsWith(st2);
System.out.println(yes);
```

עבור ערכים אלו של st1 ו-st2 יוחזר ערך שקר

```
String st1 = "william farjon";
String st2 = "farjon";
boolean yes;
yes = st1.endsWith(st2);
System.out.println(yes);
```

עבור ערכים אלו של st1 ו-st2 יוחזר ערך אמת.

### hashCode()

**Returns a hashcode for this string.**

השיטה hashCode מבצעת חישוב הנקרא hashing חישוב זה מקבל מחרוזת ומבצעת על המחרוזת חישוב אשר יוצר קוד עבור המחרוזת. השימוש בשיטה זו יעיל כאשר מעוניינים לאחסן אוסף מחרוזות במערך ומעוניינים לאחסן מחרוזות בצורה מיידיית. לצורך זה ניתן לאחסן מחרוזות במיקום שיהיה הערך המחושב בשיטה hashCode  
דוגמא:

```
int code;
String st1 = "william farjon";
String st2 = "farjun";
```



```
code = st1.hashCode();
System.out.println("The hashing code for: " + st1 + " " +code);
```

### indexOf(String, int)

Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

השיטה `indexOf(String)` מחזירה את המיקום של תת מחרוזת `String` המעבירה כארגומנט לשיטה  
דוגמא:

```
int index;
String st1 = "william farjon";
String st2 = "far";
index = st1.indexOf(st2);
System.out.println(" index " + index);
```

הערך שיוחזר מהפעלת השיטה הוא 8, וזה בדיוק תחילת המיקום של המחרוזת `st2 (far)` במחרוזת `st1 (william farjon)`  
אם המחרוזת `st2` אינה כלולה כתת מחרוזת במחרוזת `st1` הערך המוחזר יהיה -1.

השיטה `indexOf(String,int)` מבצעת פעולה דומה לשיטה `indexOf(String)` ההבדל הוא שלשיטה זו מועברים שני פרמטרים, תת המחרוזת שיש לחפש ומספר שלם שממנו יש להתחיל לחפש את תת המחרוזת.

### length()

Returns the length of this string.

השיטה `length()` מחזירה את אורך המחרוזת כלומר כמה תווים יש במחרוזת  
דוגמא:

```
int len;
String st1 = "william farjon";
len= st1.length();
System.out.println("In the string " + st1 + " we have " + len + "
characters");
```

הערך שיוחזר ע"י השיטה הוא 14, שזה מס' התווים במחרוזת st1) שים לב זה כולל כל תו וגם תו רווח (

### replace(char, char)

Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.

השיטה replace(char, char) מבצעת חילוף של כל המופעים של הארגומנט הראשון שהוא מסוג תו, בארגומנט השני כלומר בתו אחר.

דוגמא:

```
String st1 = "abc abc abc abc";
String st2;
st2=st1.replace('b','m');
System.out.println(st1 + " " + st2);
```

המחרוזת st2 היא עתה "amc amc amc amc"

### startsWith(String)

**Tests if this string starts with the specified prefix**

השיטה startsWith(String) היא שיטה בוליאנית המחזירה אמת אם המחרוזת עליה מופעלת השיטה מתחילה במחרוזת המתקבלת כארגומנט.

דוגמא:

```
String st1 = "William farjun";
String st2 = "Wil";
System.out.println(st1.startsWith(st2));
```

### startsWith(String, int)

**Tests if this string starts with the specified prefix.**

השיטה startsWith(String, int) שהיא שיטה מועמסת של השיטה הקודמת מבצעת את אותה פעולה אולם הפרמטר int מגדיר מאיזה מקום להתחיל צריך להתחיל לבדוק.

### substring(int)

**Returns a new string that is a substring of this string.**

השיטה `substring(int)` מבצעת החסרה של חלק ממחרוזת החל ממקום מסוים המוגדר ע"י `int`

```
String st1 = "William farjun";
System.out.println(st1.substring(8));
```

קטע התוכנית ידפיס `farjun` כי זה "נשאר" במחרוזת לאחר קיצוץ המחרוזת החל מהמקום ה-8.

### substring(int, int)

**Returns a new string that is a substring of this string.**

שיטה מועמסת ומבצעת את אותה הפעולה, קטע המחרוזת שמחזירים מתחיל מהמקום לפי ערך הארגומנט הראשון ועד המקום לפי ערך הארגומנט השני.

### trim()

**Removes white space from both ends of this string.**

השיטה `trim` מקצצת מהמחרוזת משני צידי המחרוזת רווחים אם קיימים.





## נספח ד- מילים שמורות בג'אווה

### Keywords in the Java Language

|          |          |            |           |              |
|----------|----------|------------|-----------|--------------|
| abstract | continue | for        | new       | switch       |
| assert   | default  | if         | package   | synchronized |
| boolean  | do       | implements | private   | this         |
| break    | double   | import     | protected | throw        |
| byte     | else     | instanceof | public    | throws       |
| case     | enum     | int        | return    | transient    |
| catch    | extends  | interface  | short     | try          |
| char     | final    | long       | static    | void         |
| class    | finally  | native     | strictfp  | volatile     |
|          | float    |            | super     | while        |





## נספח ה- שיטות המחלקה IO

קבוצת השיטות קלט/פלט, הכלולות במחלקה IO - בפיתוח פרופ' מוטי בן ארי מכון וייצמן

קליטת מחרוזת עם טקסט מלווה

**readString(String prompt)**

קליטת מחרוזת ללא הופעת טקסט מלווה

**readString()**

קליטת תו עם טקסט מלווה

**readChar(String prompt)**

קליטת תו ללא טקסט מלווה

**readChar()**

קליטת ערך בוליאני עם טקסט מלווה

**readBoolean(String prompt)**

קליטת ערך בוליאני ללא טקסט מלווה

**readBoolean()**

קליטת מספר שלם עם טקסט מלווה

**readInt(String prompt)**

קליטת מספר שלם ללא טקסט מלווה

**readInt()**

קליטת מספר שלם מסוג Long עם טקסט מלווה

**readLong(String prompt)**

קליטת מספר שלם מסוג Long ללא טקסט מלווה

**readLong()**

קליטת מספר ממשי מסוג Float עם טקסט מלווה

**readFloat(String prompt)**

קליטת מספר ממשי מסוג Float ללא טקסט מלווה

**readFloat()**

קליטת מספר ממשי מסוג Double עם טקסט מלווה

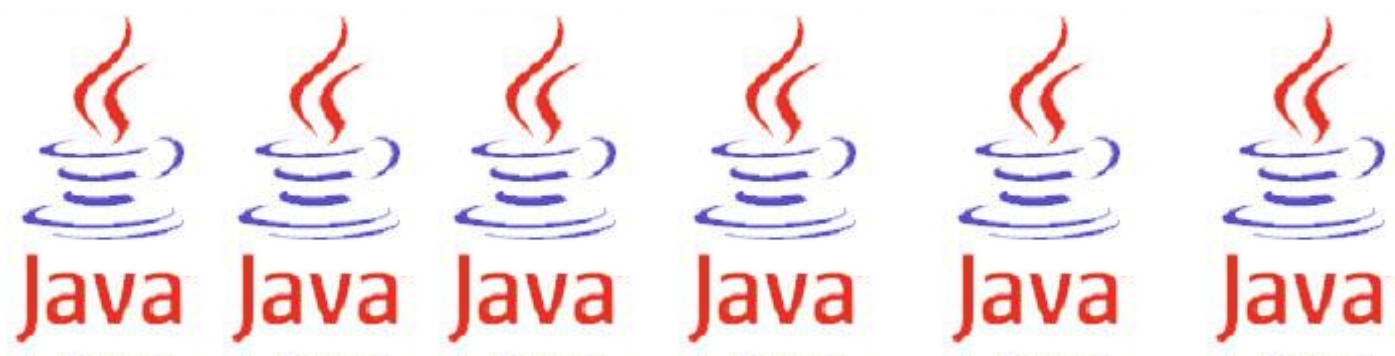
**readDouble(String prompt)**

קליטת מספר ממשי מסוג Double ללא טקסט מלווה

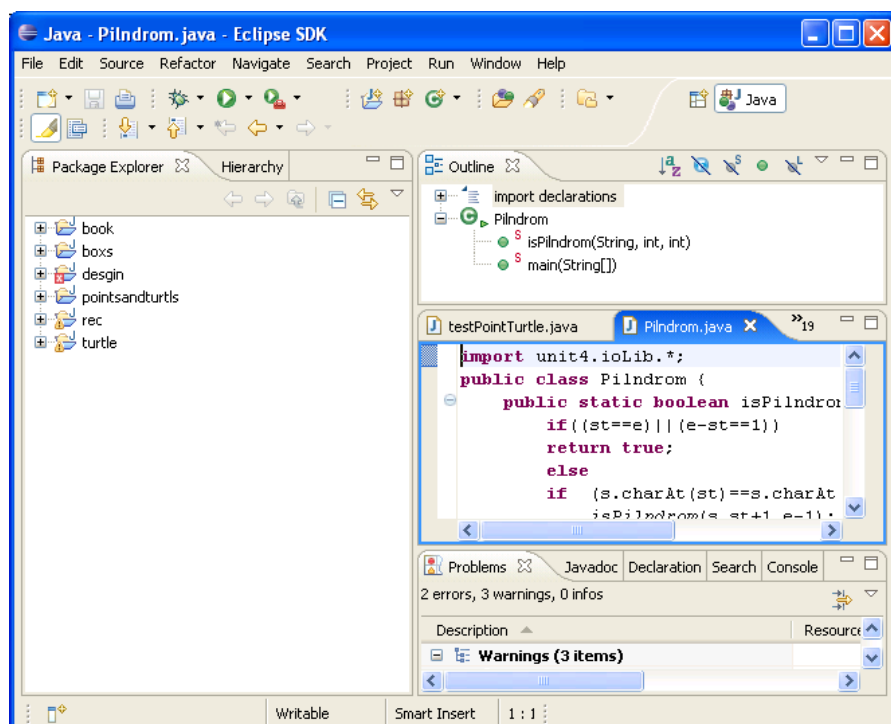
**readDouble()**

דוגמא: כתיבת הוראת קלט לקליטת מספר שלם עם טקסט מלווה:

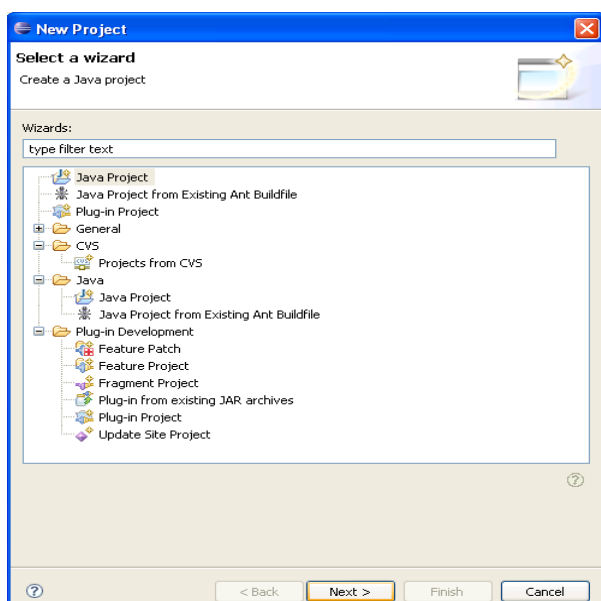
**X= IO.readInt("Enter integer number")**



## נספח ו – עבודה עם ג'אווה סביבת Eclipse



המסך מחולק לחלונות כולל:  
חלון עריכה  
חלון תיקיות הפרויקטים  
חלון הפלט \ שגיאות  
תחביריות



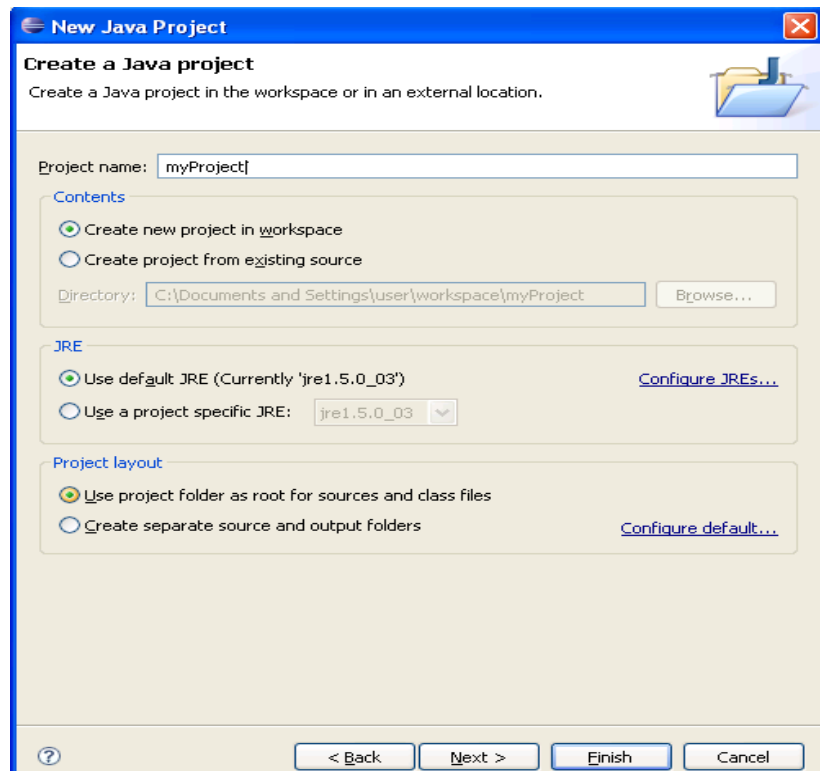
יצירת פרויקט ומחלקה  
חדשה

בחירה בתפריט **File** את  
האופציה **New** ויצירה של  
פרויקט חדש:

בחירה בלחצן **NEXT**:

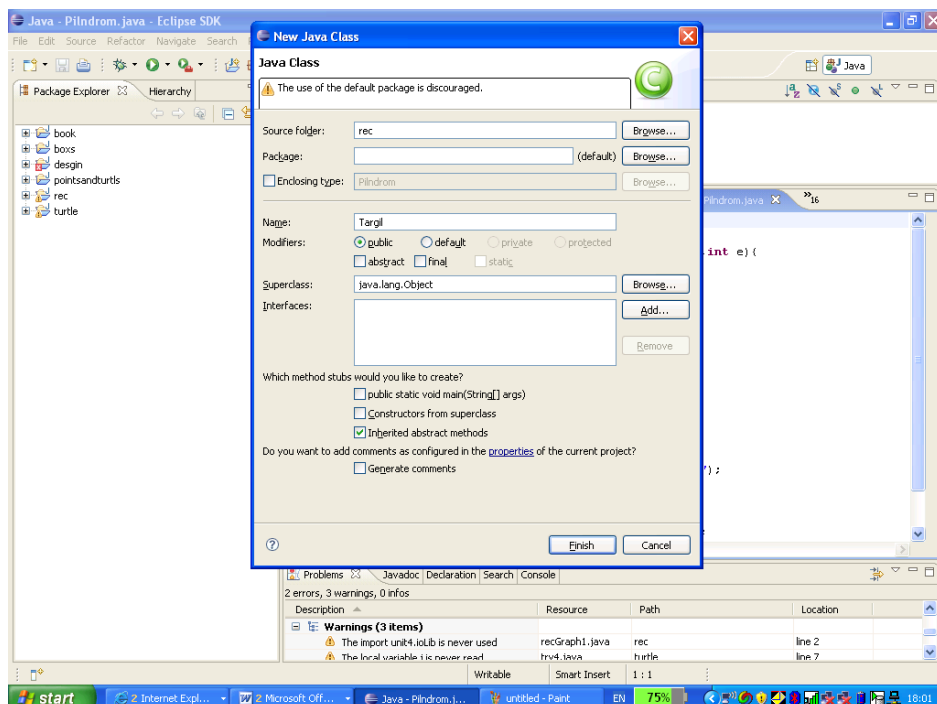


ואנו מקבלים את המסך הבא בו אנו מקלידים את שם הפרויקט בשדה המתאים ואפשר לאשר



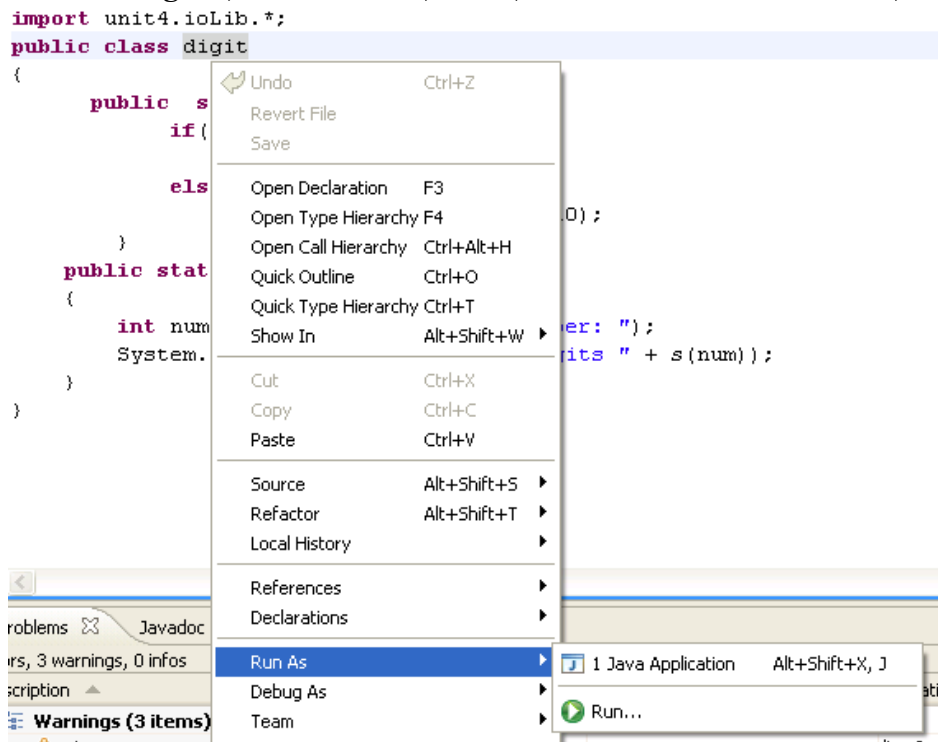
**יצירת מחלקה חדשה:**

בתפריט File >> New >> Class נבחר עתה יצירת מחלקה חדשה לפי File >> New >> Class שים לב! אם המחלקה מוגדרת כמחלקת ציבורית שם המחלקה חייב להיות זהה לשם הקובץ הנשמר בתיקיית הפרויקט. אם השם לא יהיה זהה לא תוכר המחלקה כמחלקה ציבורית בפרויקט

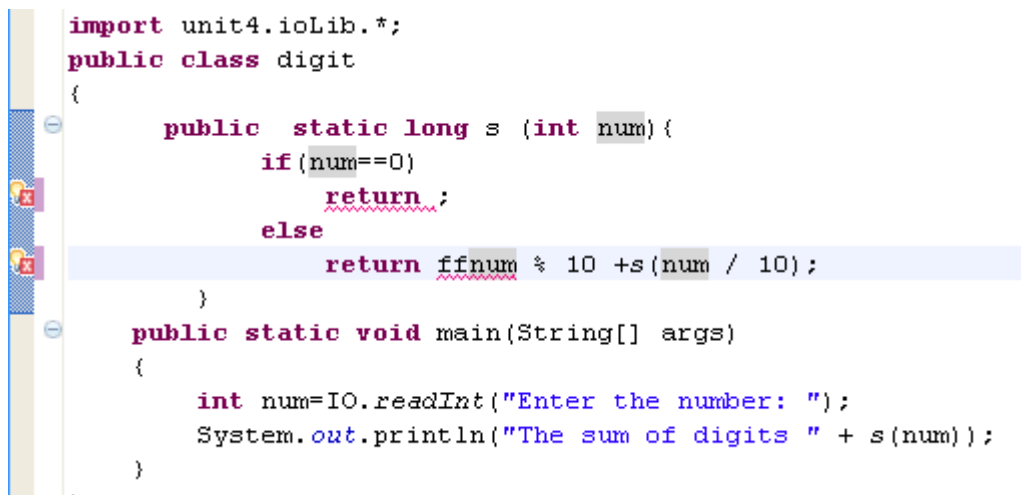


## ביצוע קומפילציה – הידור

הדרך הקצרה והנוחה ביותר לביצוע הרצה של מחלקה היא באמצעות לחיצה על לחצן ימני בעכבר כאשר הסמן של העכבר נמצא על שם המחלקה בחלון העריכה (המחלקה digit)



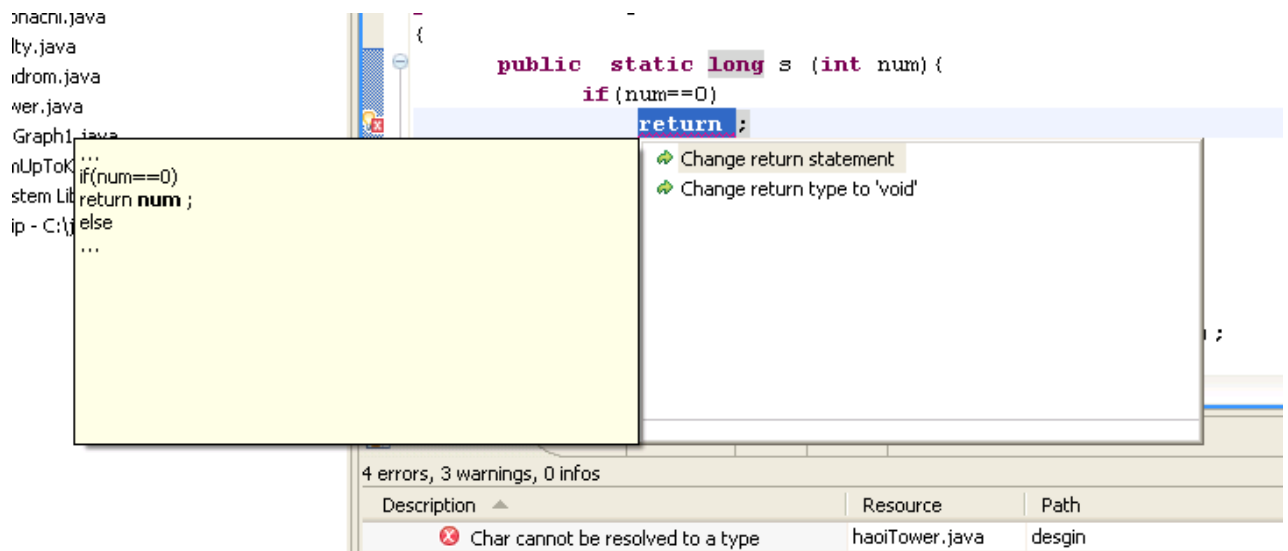
הצגת שגיאות קומפילציה - הידור



סביבת Eclipse מסייעת בזיהוי שגיאות תחביר ואלו מוצגות בתהליך כתיבת הקוד ללא ביצוע קומפילציה (הידור) שים לב, באיור האחרון מוצגים שני סימני הזהרה בשתי שורות תוכנית, שמשמעותן יש שגיאת תחביר בשורה.

## פירוט השגיאות

לחיצה עם העכבר על סימן השגיאה המופיע בתחילת השורה מסייע לזיהוי השגיאה וסביבת העבודה אף מציעה הצעות לתיקון השגיאה, מידע אשר מסייע באיתור ותיקון מהיר של שגיאות תחביריות.







## נספח ז' – עבודה בסביבת Jeliot

### Jeliot3 - סביבת אנימציה להמחשת מושגים במדעי המחשב ובתכנות

#### מבוא

הסביבה מציגה מודל הכולל אנימציה המאפשרת מעקב אחר כל פעולה, כולל הגדרת משתנים, יצירת אובייקטים למחלקות, ביצוע של פקודות השמה, תנאי, לולאות ואף פעולות על מערכים. את הסביבה ניתן להוריד באתר הבית של Jeliot בכתובת הבאה:

<http://cs.joensuu.fi/jeliot/downloads/jeliot3.5.1/Jeliot3.5.1-JRE.exe>.

[http://cs.joensuu.fi/jeliot/files/ronit\\_examples.zip](http://cs.joensuu.fi/jeliot/files/ronit_examples.zip)

#### תיאור הסביבה Jeliot3

הסביבה מאפשרת הרצת קוד של הלומד, כל תוכנית שכתבת ניתן לטעון לסביבה ללא כל שינוי בקוד שנכתב. (יש לשנות את הפקודות עם שיטות קלט/פלט). ניתן לשלוט על קצב האנימציה, לבצע חזרה לאחור, לשלוט על הרצת התוכנית.

#### תכנים שניתן להדגים בעזרת הסביבה

הסביבה מתאימה לכל טיפוס תיבה בעלת צבע שונה, כך התלמיד ידע להבחין בין סוגי המשתנים, כמו-כן הסביבה מציינת בכל תיבה את סוג המשתנה. מערך מיוצג כאוסף של תאים בעלי מספר סידורי.

חישובים (אריתמטיים ולוגיים): הסביבה מציגה את תהליך החישוב באופן מפורט ביותר, מראה תוצאות חישובי ביניים, ומודיעה על תוצאת החלטותיה בעבור חישובים לוגיים – למשל Choosing then branch בעבור ביטוי לוגי עליו התקבל הערך True בתוך משפט if.

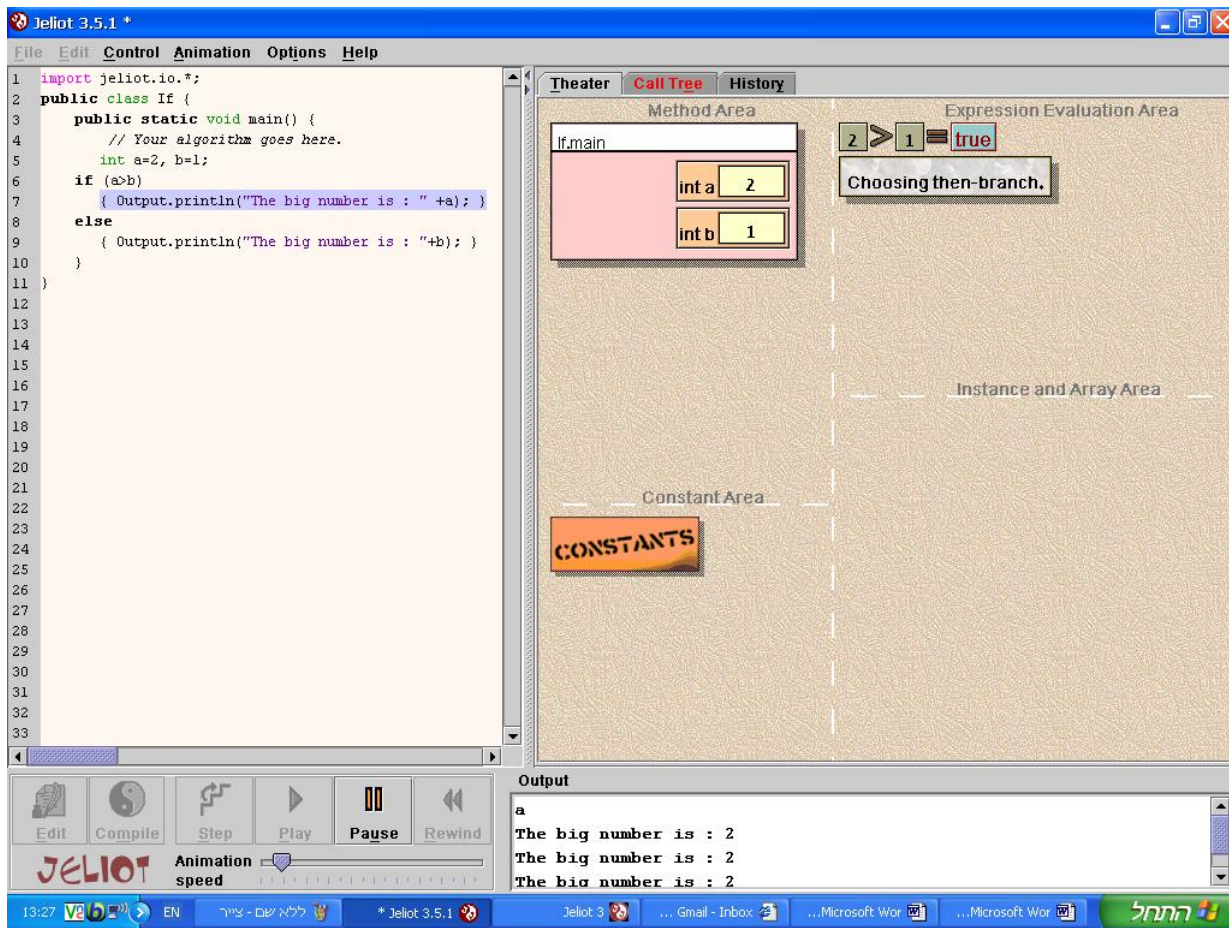
פעולת קלט: נפתח חלון מיוחד בו מתבקש המשתמש להכניס ערכים מהסוג המתאים. פעולת פלט: מופיעה יד הגוררת את העתק תוכן התא לאזור הפלט. ניתן לטייל בין הפלטים שהתקבלו בעזרת כפתור גלילה וניתן לנקות את אזור הפלט. פעולות על מערכים: בפניה לתא נוצר קשר משם המערך לתא המתאים – כך התלמיד יכול להבחין בין תוכן לבין מקום במערך.

לולאות: מוצגת גישה בה מתקבלות הודעות על כניסה לולאה, המשך ביצוע הלולאה ויציאה ממנה בהתאם לביטוי הרשום בה. בלולאת for התנאי קשור לגבול העליון של משתנה הלולאה.

#### מיומנויות שניתן להקנות בעזרת הסביבה

התלמידים יכירו את האופן בו המחשב מבצע כל אחת מהפעולות אותן הם לומדים בכיתה. ההדמיה הויזואלית עוזרת לתלמיד לבנות לעצמו מודל לאופן פעולת המחשב, כך שתהליך של ניתוח תוכנית כתובה, או שימוש בפעולות באופן נכון, שהן המיומנויות הבסיסיות אותן על התלמיד לרכוש, יהפכו לפעולות טבעיות וברורות לגביו. השפה מספקת גם אוצר מילים שמשמש תקשורת טובה בין התלמידים לבין עצמם ובין התלמידים למורה. תלמיד יכול להסביר עצמו טוב יותר. שים לב לשוני בשיטות הקלט, פלט בין סביבת JCreator לבין סביבת Jeliot

## מסך סביבת Jeliot



## דוגמא:

מחלקה If ( שים לב, האות I היא גדולה, ולכן המילה If אינה זהה למילה if שהיא מילה שמורה)  
השיטה main משימה שני ערכים בשני משתנים שלמים ומתבצעת בדיקה באיזה משתנה הושם הערך הגדול יותר, בהתאם מודפס שמו של המשתנה וערכו.

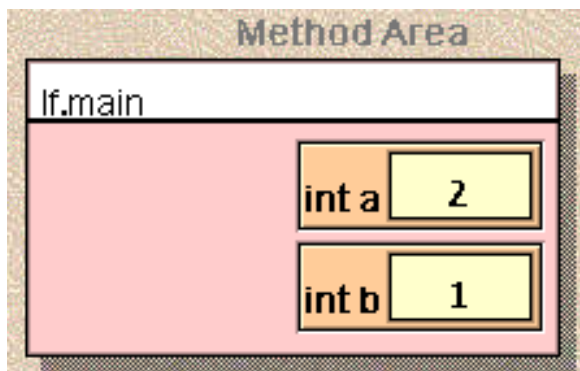
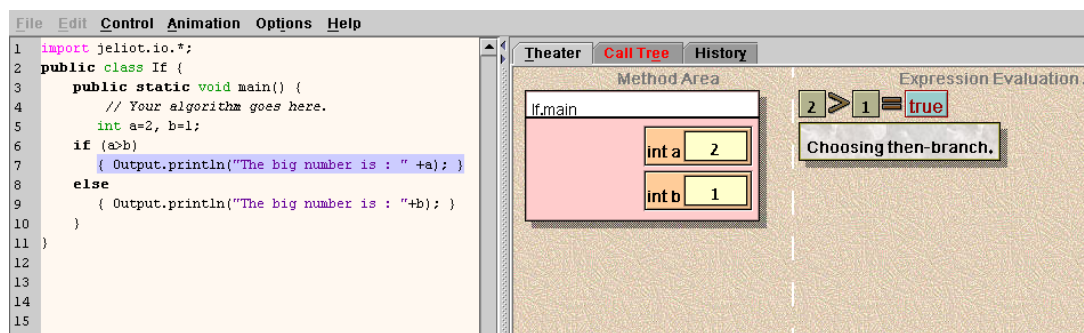
## התוכנית

```
import jeliot.io.*;
public class If {
 public static void main() {
 // Your algorithm goes here.
 int a=2, b=1;
 if (a>b)
 { Output.println("The big number is : " +a); } // שים לב לשוני
 בשיטה
 else
 { Output.println("The big number is : "+b); }
 }
 }
}
```

הסבר:

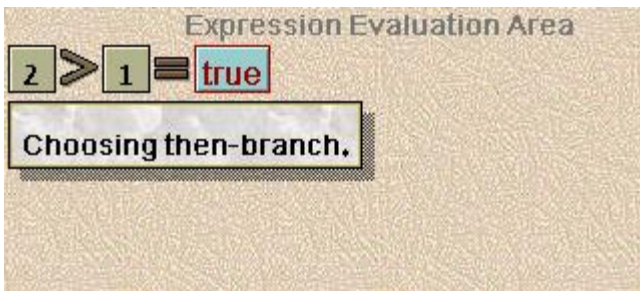
שים לב, התוכנית מבצעת יבוא של המחלקה io של הסביבה Jeliot ( השיטות שונות מסביבות אחרות של Java ). ולאובייקט המחלקה Output יש כמה שיטות, כולל, `println` השיטה `print` השיטה גם תחביר הכתיבה של שיטות אלו שונה מהשיטות בהן אנו משתמשים בספר.

## הרצת התוכנית בסביבה



הסבר

באזור המוקצה למשתני של השיטה `main` במחלקה `If` אנו רואים את ההקצאה של שני המשתנים, והערך שמשתנים אלו קבלו לאחר הכרזתם והשמת ערך.



באזור "החישובים" אנו רואים את ביצוע חישוב התנאי הלוגי, והערך שהחישוב קבל ערך true

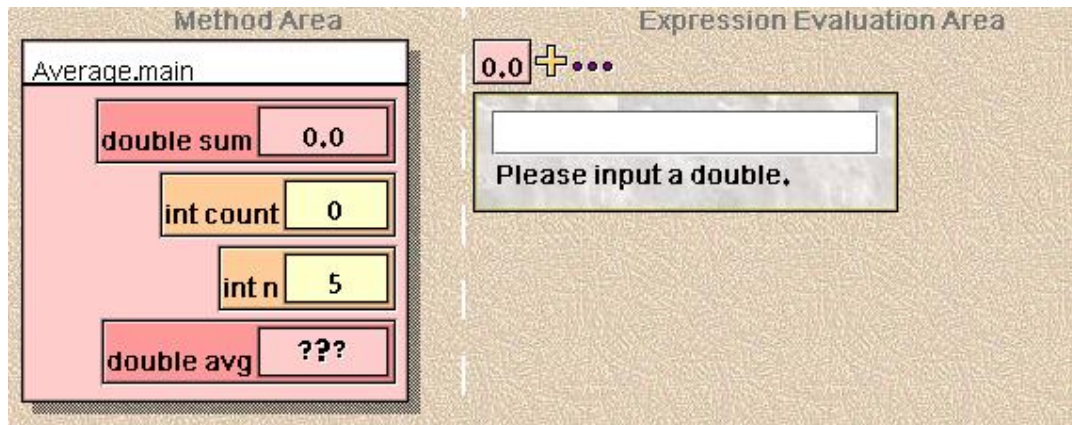
דוגמא

הדוגמא הבאה עושה שימוש בשיטות לביצוע קלט, שים לב השיטות מקבילות לשיטות שהופיעו בספר ומתייחסות לסביבת JCreator, אולם, אינן זהות בתחביר.

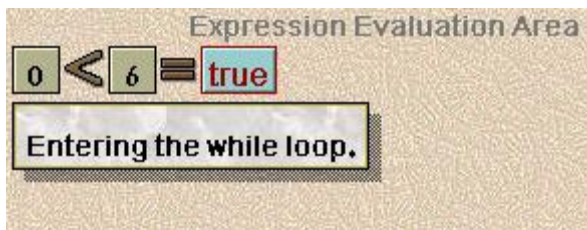
```
import jeliot.io.*;
public class Average {
 public static void main() {
 double sum = 0;
 int count = 0;
 int n;
 double avg;
 n = Input.readInt();
 while (count < n) {
 sum = sum + Input.readDouble(); // שים לב לשוני בשיטה לקליטת
מספר
 count++;
 }
 avg = sum / n;
 Output.println(avg);
 }
}
```



הרצה בסביבה



בדיקת תנאי לולאת while באזור החישוב:



## שיטות לביצוע קלט בסביבת Jeliot

ההוראה לקליטת מספר ממשי

**Input.readDouble()**

ההוראה לקליטת מספר שלם

**Input.readInt()**

ההוראה לקליטת מחרוזת

**Input.readString()**

ההוראה לקליטת תו

**Input.read.Char()**

שים לב

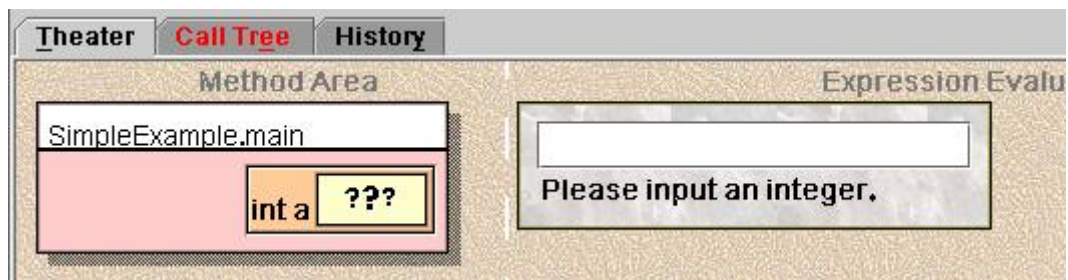
כאשר הינך מריץ תוכנית בסביבת Jeliot עליך להשתמש בשיטות אלו לקבלת קלט מהמשתמש

דוגמא

התוכנית הבאה כוללת הכרזה על 2 משתנים מסוג int, קליטת ערכים, חישוב תוצאת הכפלת שני הערכים והצגת התוצאה.

```
import jeliot.io.*;
class SimpleExample {
 static void main() {
 int a = Input.readInt();
 int b = Input.readInt();
 int c = a * b;
 Output.println(c);
 }
}
```

הרצת התוכנית בסביבת Jeliot  
שלב ביצוע קלט



שלב חישוב תוצאת ההכפלה



לסיכום

סביבת Jeliot מהווה סביבת למידה רבת עוצמה. מומלץ בשלבים הראשונים של לימוד Java להריץ כל תוכנית בסביבה זו.



## נספח ח' – מדריך מקוצר לפעולות ב-Eclipse

- פרויקט חדש – תפריט File ← New ← Java Project ← כתיבת שם הפרויקט.
- יצירת מחלקה חדשה – הקלקה ימנית על שם פרויקט ← New ← Class ← כתיבת שם המחלקה. (בזמן כתיבת שם המחלקה אפשר גם לשנות Package).
- השלמה אוטומטית של מלה – Ctrl+Spacebar (אם יש מספר אפשרויות ייפתח חלון עם כולן, לבחור על-ידי הקלקה).
- הצגת עזרה + הצגת ה-API של מחלקה / פעולה מסוימת – לחיצה על Ctrl+F1 (בזמן לאחר הקלקה שמאלית על מופע של המחלקה / שם המחלקה / שם הפעולה בקוד).
- ביצוע Import של קובץ ספרייה חיצוני לפרויקט (בלי העתקה לספרייה) – הקלקה ימנית על שם הפרויקט ← Properties ← בחירת הטאב Java BuildPath בצד שמאל ← בחירת הטאב Libraries בצד העליון של החלון ← בחירת כפתור Add External Jar מימין ← בחירת קובץ ה-Jar או ה-Zip.
- ביצוע Import של קובץ לתוך הפרויקט (כולל העתקה לתוך ספריית הפרויקט) – הקלקה ימנית על שם הפרויקט ← Import ← File System ← בחירת שם הספרייה בה נמצא הקובץ על-ידי Browse ← סימון ב-V של כל הקבצים בהם מעוניינים ו-Finish.
- הרצת פרויקט – מקליקים על שם המחלקה בה נמצא ה-Main ← Run as ← Java application.
- סגירת פרויקט (הפיכתו ל"לא-פעיל" במרחב העבודה) – הקלקה ימנית על שם הפרויקט ← Close Project.
- פתיחה מחדש של פרויקט סגור – הקלקה ימנית על שם הפרויקט ← Open Project.

- שינוי מרחב עבודה (Workspace) – תפריט File ← Switch Workspace ← בחירת ספרייה בה נמצא מרחב העבודה המבוקש (אם לא נמצא שם מרחב עבודה, יוצר מרחב עבודה באופן אוטומטי).
- שינוי שמות מהיר (למחלקה, פעולה, פרויקט וכו') – הקלקה ימנית על השם המבוקש ← בחירת Refactor ← Rename – ובחירת השם החדש. (יש להקפיד ש- Update References יהיה מסומן ב- "V" כדי שכל יתר המופעים ישתנו בהתאם).
- יצירת **JavaDoc** – תפריט פרויקט ← Generate JavaDoc ← בחירת ספרייה בה יוצר התיעוד האוטומטי (רצוי לבחור בספריית doc) ← Finish.
- ניווד פרויקט (כקובץ מכוון – Jar/Zip) – הקלקה ימנית על שם הפרויקט ← Export ← Archive file ← בחירת הקבצים לניוד, ושם קובץ הארכיב (המכוון). הקובץ יוצר בספרייה הראשית של מרחב העבודה!
- ייבוא פרויקט קיים על הדיסק לתוך מרחב העבודה (לא יועתק פיסית לתוך ספריית מרחב העבודה בדיסק, אבל יופיע כפרויקט במרחב העבודה) – תפריט File ← Import ← Existing Project into Workspace ← בחירת הספרייה בה נמצא הפרויקט.
- הוספת חלון כלשהו (שאבד, או חסר) למראה סביבת העבודה – תפריט Window ← Show View ← שם החלון, אם לא מופיע אז בחירת Other ← ואז שם החלון.
- הוספת/הסרת BreakPoint עבור ה- Debugger – הקלקה כפולה עד השטח האפור שמשמאל למספר השורה ב-Editor.

## הערה חשובה:

סביבת Eclipse "נבחרה" כסביבת העבודה העדיפה על המוסדות המפתחים את יחידות הלימוד יסודות ועיצוב בסביבת ג'אווה, הסביבה ומדריך מלא מופיעים באתרים הרשמיים של מוסדות אלו וניתנים להורדה, מטעמי זכויות יוצרים לא מופיע המדריך המלא בספר זה.



## נספח ט' – מילון מונחים

### תקציר מונחים ומוסכמות להוראת מדעי המחשב בסביבת השפות Java ו- C# לשימוש בהוראה ובפיתוח חומרי למידה

נכתב על ידי צוותי פיתוח תכניות הלימודים "יסודות מדעי המחשב" ו "עיצוב תוכנה" במוסדות: מכון ויצמן למדע - רחובות, האוניברסיטה העברית בירושלים, אוניברסיטת תל-אביב ובאישורו של המפמ"ר למדעי המחשב וטכנולוגיות מידע.

#### מונחים בתכנות מונחה עצמים

- class - מחלקה.
- object - עצם (או לחלופין אובייקט).
- attribute - תכונה.
- method - פעולה (או שיטה). השורה הראשונה בפעולה היא "כותרת הפעולה".
- instance of... - מופע (מופע של...).
- constructor - פעולה בונה
- פעולות פנימיות וחיצוניות – פעולה פנימית היא פעולה השייכת לממשק של מחלקה. פעולה חיצונית היא פעולה (לרוב סטטית) המקבלת עצם מטיפוס מחלקה ופועלת עליו.
- header - כותרת (של מחלקה או פעולה).
- this - מציין את העצם הנוכחי.
- overloading - העמסה.
- overriding - הגדרה מחדש.
- polymorphism - פולימורפיזם, ובעברית: רב-צורתיות.
- inheritance - ירושה.
- access specifiers - הרשאות גישה: private – פרטי, public – פומבי.
- composed object - עצם מורכב.
- interface - ממשק.
- הפעולה main/Main נקראת "הפעולה הראשית".

## ממשק עברי

רעיון הממשק העברי בוטל, ושמות הפעולות יופיעו רק בכתוב המדויק שלהן בשפת התכנות.

## אלגוריתמים

בתום האלגוריתם, ועל פי הצורך, יצוין מהו מופע הטיפוס המוחזר.

**ביסודות** – ישארו האלגוריתמים בנוסח המקובל כולל מספור השורות, פרט לשינויים הבאים:

העצם המפעיל את הפעולה (זה שעליו מבוצע האלגוריתם), לא יופיע ברשימת הפרמטרים של הפעולות.

**בעיצוב תוכנה** – האלגוריתמים יכתבו באופן שוטף ופחות פורמלי. שורות האלגוריתמים לא ימוספרו. האלגוריתם רק יתווה את כיוון הפתרון ולא יתאר את השלבים המדויקים לביצוע הפתרון.

העצם המפעיל את הפעולה (זה שעליו מבוצע האלגוריתם), לא יופיע ברשימת הפרמטרים של הפעולות, אם מדובר בפעולה שתוגדר בתוך המחלקה. אם מדובר בפעולה חיצונית, הפועלת על עצם מטיפוס המחלקה, יופיע העצם בין הפרמטרים.

**דוגמה:** (פעולה חיצונית הפועלת על רשימה)

אחזר-מקום (x, lst)

{*טענת כניסה:* הפעולה מקבלת רשימה של מספרים שלמים lst, ומספר שלם x}

{*טענת יציאה:* הפעולה מחזירה את המקום הראשון של x ברשימה אם x מופיע, אחרת מחזירה

{null

עבור על חוליות הרשימה lst מתחילתה

אם ערך החוליה הנוכחית ברשימה שווה ל-x, החזר את מקומה.

אם הגעת לסוף הרשימה החזר null.

## עקרונות כתיבה ועריכה כלליים:

- באלגוריתמים יהיה שימוש במשפט "החזר".
- תיעוד ייכתב מתחילת הלימוד על פי עקרונות התיעוד המקובלים.
- הערה לממשק תופיע בין /\* ... \*/ (בג'אווה) ולאחר /// בסישרפ, בעוד שהערה למתכנת עצמו תופיע לאחר //.
- הסוגריים בהגדרת בלוק הוראות ירשמו לבד בשורות נפרדות, מיושרים עם כותרת הפעולה.
- קבועים ירשמו באותיות גדולות, בין מילים יפריד קו תחתון.
- שמות (פעולות/שיטות, תכונות,...) ירשמו ברצף ללא קוים תחתונים. יתחילו באות קטנה וכל מילה חדשה תתחיל באות גדולה.
- שם מחלקה יתחיל באות גדולה.

- מילים שמורות יכתבו בספר הלימוד ב- **bold**.
  - הפעולה הראשית (main/Main) תוגדר בתוך מחלקה שתכיל אותה בלבד אלא אם כן נדרשות שיטות סטטיות נלוות.
  - יפותח ממשק קלט/פלט אחיד לכל היחידות. בג'אוה יעשה בינתיים שימוש במחלקת IO שפיתח פרופ' מוטי בן ארי.
  - ספרי הלימוד לא יתבססו על סביבת פיתוח מסוימת אלא יהיו כלליים.
- התאמה בין שפות הלימוד ג'אוה ו-C#

כדי להאחיד עד כמה שאפשר את הוראת השפות והבחינה עליהן, ילמד רק הבסיס המשותף לשתי השפות.

הנחיות מפורטות (רשימה זו תתעדכן מידי פעם):

- properties ב-C# לא ילמדו. השימוש יהיה בפעולות get ו-set בודדות בלבד.
- פרמטרים יועברו לפעולות רק בהפניה.
- לא ילמדו רשומות ב-C#.
- שמות פעולות יתחילו באות גדולה כמקובל בסישרפ.

נוסחי שאלות

המונח "ייצג" לגבי מחלקה או טיפוס נתונים – משמעותו: כתוב יצוג מלא של תכונות המחלקה והמופעים בסביבת העבודה.

נספח זה מופיע ללא שינוי כפי שפורסם באישור צוות הפיתוח האוניברסיטה העברית ירושלים ואני מודה על אישור זה.

הערה: בספר נעשה שימוש בשני סוגי מחלקות קלט/פלט, בסביבה Jeliot, ובמחלקת IO שפותחה במכון וייצמן ע"י פרופ' מוטי בן ארי. יש לכלול את המחלקה בתוך הפרויקט, ללא הכללה זו לא יוכרו השיטות, או לבצע את פעולת build path בסביבת eclipse וכתובת ההוראה import לפני השורה הראשונה של ההכרזה על המחלקה ( class... ) לפי התיאור במדריך המקוצר ובמדריך המלא של קבוצת הפיתוח באוניברסיטה העברית והמופיע באתרים הרשמיים, בנספח יג' מופיעה המלצה מעודכנת של ועדת המקצוע על מחלקת קלט פלט חדשה שיתרונה הוא התאמתה לרוח תוכנית הלימודים ולתכנות מונחה עצמים. לא השתמשתי במחלקה זו בספר מאחר ואין אישור סופי והנחת הבסיס שבשנה הקרובה ניתן להשתמש בכל מחלקה במיוחד המחלקת IO ואני מעריך שמצב זה לא ישתנה גם בעתיד.







נספח י' – פתרון תרגילים

### תרגילי פרק 3

#### תרגיל 1

```
public class ch3E1 {
 public static void main(String args[]) {
 double mark;
 mark=IO.readInt("Please enter the mark");
 System.out.println("The mark befor the chang is " + mark);
 System.out.println("The mark befor the chang is " + mark*1.15);
 } // end of main
} // end of class
```

#### תרגיל 2

```
class ch3E2 {
 public static void main(String args[]) {
 double price,dis;
 price=IO.readFloat("Please enter the price");
 dis=IO.readFloat("Please enter the discount");
 System.out.println("The discount is: " + price * dis/100);
 System.out.println("The finel price is: " + (price-(price* dis/100)));
 } // end of main
} // end of class
```

### תרגיל 3

פתרון התרגיל- קיצוץ החלק הלא שלם.

```
public class ch3E3 {
 public static void main(String args[]) {
 double number;
 number=IO.readFloat("Please enter the number");
 System.out.println("The new number is: " + (int) number);
 } // end of main
} // end of class
```

אם הדרישה ל"עגל" את המספר, התוכנית הבאה משתמשת בשיטה round השייכת למחלקה

Math

```
public class ch3E3 {
 public static void main(String args[]) {
 double number;
 number=IO.readFloat("Please enter the number");
 System.out.println("The new number is: " + Math.round(number));
 } // end of main
} // end of class
```

#### תרגיל 4

```
public class ch3E4 {
 public static void main(String args[]) {
 double num1,num2,temp;
 num1=IO.readDouble("Please enter the number");
 num2=IO.readDouble("Please enter the number");
 System.out.println("num1= "+ num1 +" num2= " + num2);
 temp=num1;
 num1=num2;
 num2=temp;
 System.out.println("num1= "+ num1 +" num2= " + num2);
 } // end of main
} // end of class
```

#### תרגיל 5

```
public class ch3E5 {
 public static void main(String args[]) {
 int num1;
 int num2,x;
 double sum;
 num1=5; num2=12;
 sum= num1+num2;
 System.out.println(sum);
 x=num1/num2;
 System.out.println(x);
 System.out.println(x+1);
 System.out.println(x+5);
 } // end of main
} // end of class
```

הפלט לתוכנית בתרגיל 5 הוא:

17  
0  
1  
5

- 17 כי הסכום של 12 ו-5 הוא 17
  - 0 כי כאשר מחלקים שני מספרים שלמים וההשמה למשתנה שלם, התוצאה היא המנה השלמה ובתוכנית זו 5 לחלק ב-12 התוצאה השלמה היא 0
  - 1 כי הוספנו ל- $x$  1 במשפט הפלט, שים לב המשתנה  $x$  לא משנה את ערכו ונשאר 0
  - 7 כי הוספנו ל- $x$  5 במשפט הפלט, שים לב  $x$  עדיין עם ערך 0, לאחר הפעולה ערכו של  $x$  לא משתנה.
- הרץ מחלקה זו בסביבת Jeliot עקוב אחר ערכי המשתנים. שים לב אל תשכח לשנות את משפטי הקלט פלט.

## תרגיל 6

```
class ch3E6 {
 public static void main(String args[]) {
 float weight;
 float day1=5,day2=6,day3=7;
 weight=IO.readFloat(" Enter the Weight ");
 weight = weight-day1;
 System.out.println("The Weight in the first day is:"+ (weight-day1));
 weight = weight-day2;
 System.out.println("The Weight in the second day is:"+ (weight-day2));
 weight = weight-day3;
 System.out.println("The Weight in the third day is: "+ (weight-day3));
 } // end of main
} //end of class
```

## תרגיל 7

```
class ch3E7 {
 public static void main(String args[]) {
 double num1,num2;
 num1=IO.readDouble("Enter the first number");
 num2=IO.readDouble("Enter the second number");
 System.out.println("num1+num2+1 = "+(num1+num2+1));
 } // end of main
} //end of class
```

## תרגיל 8

```
class ch3E8 {
 public static void main(String args[]) {
 int deep;
 deep=IO.readInt("Enter the deep");
 System.out.println("The press is "+ (deep/10 +1));
 } // end of main
} //end of class
```

## תרגיל 9

```
class ch3E9 {
 public static void main(String args[]) {
 int ton;
 ton=IO.readInt("Enter the ton");
 System.out.println("The press is "+ (ton * 1000));
 } // end of main
} //end of class
```

## תרגיל 10

```
class ch3E10 {
 public static void main(String args[]) {
 double radius;
 radius=IO.readDouble("Enter the radius ");
 System.out.println("The perimeter is "+ (radius * 2 * 3.14));
 System.out.println("The area is "+ (radius*radius * 3.14));
 } // end of main
} //end of class
```

## תרגיל 11

```
class ch3E11 {
 public static void main(String args[]) {
 int num;
 num=IO.readInt("Enter the num ");
 System.out.println(-num);
 System.out.println((double)1/num);
 } // end of main
} //end of class
```

## תרגיל 12

```
class ch3E12 {
 public static void main(String args[]) {
 double num;
 int newnum;
 num=IO.readDouble("please enter the num");
 newnum=(int)num;
 System.out.println(num - newnum);
 } // end of main
} // end of class
```

## תרגיל 13

הפלט: משורת פלט ראשונה נקבל 36, זהו הערך המוחלט של -36  
 משורת פלט שנייה נקבל 6, זהו השורש הריבועי של 36, שים לב להפעלת השיטה abs  
 לפני הפעלת השיטה sqrt.

```
class ch3E13 {
 public static void main(String args[]) {
 int x=-36;
 System.out.println(Math.abs(x));
 System.out.println(Math.sqrt(Math.abs(x)));
 } // end of main
} // end of class
```

## תרגיל 14

```
class ch3E14 {
 public static void main(String args[]) {
 double n1,n2;
 double hypotenuse;
 n1=IO.readFloat("Please enter the first Perpendicular");
 n2=IO.readFloat("Please enter the second Perpendicular");
 hypotenuse= Math.sqrt(n1*n1+n2*n2);
 System.out.println("The length of hypotenuse is: " + hypotenuse);
 } // end of main
} // end of class
```

## תרגילי פרק 4

### תרגיל 1

```

class ch4E1 {
 public static void main(String args[]) {
 float num1,num2,num3;
 float big=0, m=0,small=0;
 num1=IO.readFloat(" please enter the first number ");
 num2=IO.readFloat(" please enter the first number ");
 num3=IO.readFloat(" please enter the first number ");
 if ((num1>num2) && (num1>num3))
 big=num1;
 if ((num2>num1) && (num2>num3))
 big=num2;
 if ((num3>num2) && (num3>num1))
 big=num3;
 if ((num1<num2) && (num1<num3))
 small=num1;
 if ((num2<num1) && (num2<num3))
 small=num2;
 if ((num3<num2) && (num3<num1))
 small=num3;
 if ((num1!=big)&& (num1!=small)) m=num1;
 if ((num2!=big)&& (num2!=small)) m=num2;
 if ((num3!=big)&& (num3!=small)) m=num3;
 System.out.println("The number in right order " + small + " "+m + " "
+ big)
 } // end of main
} // end of class

```



## תרגיל 2

```
class ch4E2 {
 public static void main(String args[]) {
 float num;
 num=IO.readFloat(" please enter the number ");
 if (num>=0)
 System.out.println(num);
 else
 System.out.println(-num);
 } // end of main
} // end of class
```

## תרגיל 3

```
class ch4E3 {
 public static void main(String args[]) {
 int num1,num2;
 num1=IO.readInt("please enter the first number ");
 num2=IO.readInt("please enter the second number ");
 if (num1+1==(num2))
 System.out.println("Yes");
 } // end of main
} // end of class
```

## תרגיל 4

תרגיל 4 מבצע את המטלה שמבצע תרגיל בשינוי קטן:1:  
טענת כניסה: התוכנית קולטת 3 מספרים  
טענת יציאה: התוכנית מדפיסה את 3 המספרים בסדר יורד.  
בפתרון תרגיל 1 השתמשי באופרטורים הלוגים, לעומת זאת בפתרון תרגיל 4 אין שימוש  
באופרטורים לוגים ולחילופין ישנו שימוש בקינון משפטי תנאי.

## תרגיל 5

המטלה של תרגיל 5, מחברת בין תרגילים 1,4

## תרגיל 6

```
class ch4E6 {
 public static void main(String args[]) {
 double num1,num2,num3,avg;
 num1=IO.readDouble("please enter the first number ");
 num2=IO.readDouble("please enter the second number ");
 num3=IO.readDouble("please enter the third number ");
 avg=(num1+num2+num3)/3;
 if(avg<56)
 System.out.println("you faild the test");
 else
 if((avg>=56) && (avg<=80))
 System.out.println(" you doing well");
 else
 System.out.println(" verygood");
 } // end of main
} // end of class
```

## תרגיל 7

דרך א' לפתור את המטלה

```
class ch4E7 {
 public static void main(String args[]) {
 double num1,num2,num3,avg;
 num1=IO.readDouble("please enter the first number ");
 num2=IO.readDouble("please enter the second number ");
 num3=IO.readDouble("please enter the third number ");
 if((num1==num2)&&(num2==num3))
 System.out.println("yes");
 else
 System.out.println(" no");
 } // end of main
} // end of class
```

דרך ב' לפתור את המטלה

```
class ch4E7b {
 public static void main(String args[]) {
 double num1,num2,num3,avg;
 num1=IO.readDouble("please enter the first number ");
 num2=IO.readDouble("please enter the second number ");
 num3=IO.readDouble("please enter the third number ");
 if(num1==num2)
 if(num2==num3)
 System.out.println("yes");
 else
 System.out.println(" no");
 else
 System.out.println(" no");
 } // end of main
} // end of class
```

איזו דרך אתה מעדיף?

## תרגיל 8

```
class ch4E8 {
 public static void main(String args[]) {
 int num;
 num=IO.readInt("please enter the number ");
 if(num % 2 ==0)
 System.out.println("yes");
 else
 System.out.println(" no");
 } // end of main
} // end of class
```

## תרגיל 9

```
class ch4E9 {
public static void main(String args[]) {
 int num1,num2,num3;
 num1=IO.readInt("please enter the number ");
 num2=IO.readInt("please enter the number ");
 num3=IO.readInt("please enter the number ");
 if(num1+num2+num3 == num1*num2*num3)
 System.out.println("yes");
 else
 System.out.println(" no");
 } // end of main
}
```

## תרגיל 10

```
class ch4E10 {
public static void main(String args[]) {
 int num;
 num=IO.readInt("please enter the number ");
 if((num>=0) && (num <99))
 System.out.println(num+1);
 else
 System.out.println(0);
 } // end of main
} // end of class
```

## תרגיל 11

```
class ch4E11 {
public static void main(String args[]) {
 int num;
 num=IO.readInt("please enter the number with 3 digit ");
 int a1,a10,a100;
 a1= num%10;
 num= num/10;
```

```

a10= num%10;
a100= num/10;
System.out.println(a100+" "+a10+" "+a1+" ");
if((a1==a10)&&(a10==a100))
 System.out.println("yes");
else
 System.out.println("no");
} // end of main
} // end of class

```

## תרגיל 12

```

class ch4E12 {
 public static void main(String args[]) {
 int tsheart;
 tsheart=IO.readInt("please enter the number tsheart ");
 double pay=0;
 if((tsheart>=1)&&(tsheart<=5))
 pay=tsheart*120*0.95;
 else
 if((tsheart>5)&&(tsheart<=10))
 pay=tsheart*120*0.90;
 if(tsheart>=10)
 pay=tsheart*120*0.80;
 System.out.println("You have to pay "+ pay);
 } // end of main
} // end of class

```

## תרגיל 13

```

class ch4E13 {
 public static void main (String[] args) {
 int num;
 num=IO.readInt("please enter the number with 3 digit ");
 int a1,a10,a100;
 a1= num%10;
 num= num/10;
 }
}

```

```

a10= num%10;
a100= num/10;
if((a1+a10+a100)==(a1*a10*a100))
 System.out.println("A gold number");
else
 System.out.println("Not a gold number");
} // end of main
} // end of class

```

#### תרגיל 14

```

class ch4E14 {
public static void main (String[] args) {
 int num;
 num=IO.readInt("please enter the number with 3 digit ");
 int a1,a10,a100;
 a1= num%10;
 num= num/10;
 a10= num%10;
 a100= num/10;
 if(a1==a100)
 System.out.println("A pilndrom");
 else
 System.out.println("Not a gold pilndrom");
 } }

```

#### תרגיל 15

```

class ch4E15 {
 public static void main(String[] args){
 int num;
 num=IO.readInt("please enter the number ");
 if(num % 7 ==0)
 System.out.println("Yes");
 else
 System.out.println("No");
 } // end of main}

```

## תרגיל 16

```
class ch4E16 {
 public static void main(String[] args){
 int pageday1,pageday2,pageday3;
 pageday1=IO.readInt("please enter the num of pages in day1 ");
 pageday2=IO.readInt("please enter the num of pages in day2 ");
 pageday3=IO.readInt("please enter the num of pages in day3 ");
 if(pageday1+pageday2+pageday3==230)
 System.out.println("Yes");
 else
 System.out.println("No");
 } // end of main
 }
```

## תרגיל 17

```
class ch4E17 {
 public static void main(String[] args){
 int num;
 String word;
 num=IO.readInt("enter the number of chars in the string ");
 word=IO.readString("please enter word ");
 if(word.length()>num)
 System.out.println("Yes");
 else
 System.out.println("No");
 } // end of main
 } // end of class
```

שים לב, בהרצת התוכנית יש בקליטת מחרוזת להכניסה בין גרשיים.

## תרגיל 18

```
class ch4E18 {
 public static void main(String[] args){
 int num;
 String word1,word2;
 word1=IO.readString("please enter word ");
 word2=IO.readString("please enter word ");
 System.out.println(word1.length()+word2.length());
 } // end of main
}
```

## תרגיל 19

```
class ch4E19 {
 public static void main(String[] args) {
 int num;
 num=IO.readInt("please enter the number");
 if((num%2==0)&&(num%5==0))
 System.out.println("Yes");
 else
 System.out.println("No");
 } // end of main
} // end of class
```

## תרגיל 20

ישנם דרכים רבות לפתרון תרגיל זה, קרא בעיון ונסה דרך נוספת.

```
class ch4E20 {
 public static void main(String[] args){
 float num1,num2,num3,num4;
 float b1,b2,big;
 num1=IO.readInt("please enter the number ");
 num2=IO.readInt("please enter the number ");
 num3=IO.readInt("please enter the number ");
 num4=IO.readInt("please enter the number ");
 if(num1>=num2)
```



```

b1=num1;
else
b1=num2;
if(num3>=num4)
b2=num3;
else
b2=num4;
if(b1>=b2)
big=b1;
else
big=b2;
System.out.println("The biggest number is " + big);
} // end of main

}

```

## תרגיל 21

```

class ch4E21 {
 public static void main(String[] args){
 int passnger,boat;
 float smallBoat,bigBoat;
 passnger=IO.readInt("How many passnger ");
 smallBoat=passnger/50;
 if (smallBoat*50< passnger)
 smallBoat+=1;
 bigBoat=passnger/70;
 if (bigBoat*70< passnger)
 bigBoat+=1;
 System.out.println("The number of small boat " + smallBoat);
 System.out.println("The number of big boat " + bigBoat);
 System.out.println("If you take small boat you pay " +
smallBoat*1200);
 }
}

```

```
System.out.println("If you take big boat you pay " +
bigBoat*1600);
```

```
 if (bigBoat*1600 > smallBoat*1200)
 System.out.println(" We take the small boat");
 else
 System.out.println(" We take the big boat");

} // end of main
} // end of class
```

## תרגילי פרק 5

### תרגיל 1

```
public class ch5E1 {

 /**
 * @param args
 */
 public static void main(String[] args) {
 int num1= IO.readInt("Enter the first number");
 int num2= IO.readInt("Enter the second number");
 long finel=1;
 int i;
 for (i=1; i<=num2 ; i++)
 {
 finel = finel * num1 ;
 }
 System.out.println();
 System.out.println(num1 + " power " + num2 + " = " + finel);
 }
}
```

### תרגיל 2

```
public class ch5E2 {
 public static void main(String args[]) {
 int ness=0,gadol=0,haya=0,po=0;
 int x,n,sevivon;
 n=IO.readInt("please enter the first number of tries ");
 for (x=1;x<=n;x++){
 sevivon = (int)(Math.random()*4) +1;
 switch (sevivon) {
 case 1: ness++; break;
 }
 }
 }
}
```

```

 case 2: gadol++; break;
 case 3: haya++; break;
 case 4: po++; break;
 } // end of switch
} // end of for
System.out.println("ness= "+ness);
System.out.println("ness= "+gadol);
System.out.println("ness= "+haya);
System.out.println("ness= "+po);
} // end of main
} // end of class ch5E2

```

### תרגיל 3

```

class ch5E3 {
 public static void main(String args[]) {
 int x ;
 double nextNumber=1;
 for (x=1;x<=10;x++){
 System.out.println(nextNumber*nextNumber);
 nextNumber=nextNumber+1;
 } // end of the for
 } // end of main
} // end of class ch5E3

```

### תרגיל 4

```

class ch5E4 {
 public static void main(String args[]) {
 int x ;
 double nextNumber=1;
 for (x=1;x<=10;x++){
 System.out.println(nextNumber);
 nextNumber=nextNumber+2;
 } // end of the for
 }
}

```

## תרגיל 5

```
class ch5E5 {
 public static void main(String args[]) {
 int x ;
 double nextNumber=1;
 for (x=1;x<=10;x++){
 System.out.println(1/nextNumber);
 nextNumber=nextNumber+2;
 } // end of the for
 }
}
```

## תרגיל 6

```
class ch5E6 {
 public static void main(String args[]) {
 double pi= 1;
 double mone = 2;
 double me= 1;
 int n= IO.readInt(" Enter num loops");
 int i;
 for(i=1; i<=n; i++) {
 pi= pi* (mone/me)*(mone/(me+2));
 mone=mone+2; me=me+2;
 }
 System.out.println("The pi number = "+ pi*2);
 } // end of main
} // end of class
```

## תרגיל 7

```
class ch5E7 {
 public static void main(String args[]) {
 int i ;
 int maxLen=0;
 String maxS="";
```

```
String nextStr="";
maxS=IO.readString("Next String?");
maxLen=maxS.length();
for (i=1; i<10; i++) {
 nextStr=IO.readString("Next String?");
 if(nextStr.length()> maxLen){
 maxS=nextStr;
 maxLen=maxS.length();
 } // end of if
} // end of for
System.out.println("The longest String is "+ maxS + " " + "length " +
maxLen);
} // end of main
} // end of class
```

## תרגיל 8

```
public class ch5E8 {
 public static void main(String[] args) {
 int student;
 int s1,s2,s3,s = 0;
 s1=s2=s3=0;
 for(student=1;student<=10;student++){
 s=IO.readInt("student No ");
 switch (s) {
 case 1: s1++; break;
 case 2: s2++; break;
 case 3: s3++; break;
 }
 }
 System.out.println("show No 1 " + s1);
 System.out.println("show No 1 " + s2);
 System.out.println("show No 1 " + s3);
 }
}
```

```

public class ch5E9 {
 public static void main(String[] args) {
 int i,site,site1,site2,site3,site4;
 int site5,site6,site7,site8,site9,site10;
 site1=site2=site3=site4=site5=site6=site7=site8=site9=site10=0;
 for (i=1; i<=30; i++) {
 site=IO.readInt(" Enter the num of site " + "Student-"+ i);
 switch(site){
 case 1: site1++; break;
 case 2: site2++; break;
 case 3: site3++; break;
 case 4: site4++; break;
 case 5: site5++; break;
 case 6: site6++; break;
 case 7: site7++; break;
 case 8: site8++; break;
 case 9: site9++; break;
 case 10: site10++; break;
 default: System.out.println(" worng num of site");
 } // end of switch
 } // end of for
 System.out.println("For site number 1 " + site1 + " students");
 System.out.println("For site number 2 " + site2 + " students");
 System.out.println("For site number 3 " + site3 + " students");
 System.out.println("For site number 4 " + site4 + " students");
 System.out.println("For site number 5 " + site5 + " students");
 System.out.println("For site number 6 " + site6 + " students");
 System.out.println("For site number 7 " + site7 + " students");
 System.out.println("For site number 8 " + site8 + " students");
 System.out.println("For site number 9 " + site9 + " students");
 System.out.println("For site number 10 " +site10 + " students")
 } // end of main
} // end of class

```

## תרגיל 10

```
public class ch5E10 {
 public static void main(String[] args) {
 int i=1;
 int sum=0;
 for(i=1; i<=100; i=i+2){
 sum=sum+i;
 }
 System.out.println(" The sum is: "+sum);
 }
}
```

## תרגיל 11

```
import unit4.ioLib.*;
public class ch5E11 {
 public static void main(String[] args) {
 String st="";
 int i;
 st=IO.readString("Enter the string");
 for (i=0;i<st.length(); i=i+2){
 System.out.println(st.charAt(i)); }
 }
}
```

## תרגיל 12

```
public class ch5E12 {
 public static void main(String[] args) {
 // I run the program for 10 students
 int stu, test, mark, sum;
 for(stu=1;stu<=10;stu++){
 sum=0;
 for (test=1; test<=6; test++){
 mark=IO.readInt("Enter the mark " +test + " for student No " +
stu + " ");
 sum=sum+mark; }
 }
}
```



```

 System.out.println("The average is :"+ (double) (sum/test));
 }
}
}

```

### תרגיל 13

```

public class ch5E13 {
 public static void main(String[] args) {
 int games,i,p1=0,p2=0,t1,t2;
 /* 1 for scissor
 * 2 for paper
 * 3 for stone
 */
 games= IO.readInt(" How many games you want to play");
 for(i=1; i<=games; i++){
 t1=(int)(Math.random()*3)+1;
 t2=(int)(Math.random()*3)+1;
 switch (t1) {
 case 1: System.out.print("Player one got a Scissor ");
 break;
 case 2: System.out.print("Player one got a paper ");
 break;
 case 3: System.out.print("Player one got a stone ");
 break;
 }
 switch (t2) {
 case 1: System.out.println("Player two got a Scissor ");
 break;
 case 2: System.out.println("Player two got a paper ");
 break;
 case 3: System.out.println("Player two got a stone ");
 break;
 }
 if (t1!=t2){

```

```

 if ((t1==1)&& (t2==2)){// Scissor vs paper
 p1++;} else
 if ((t1==1)&& (t2==2)){
 p2++; }
 if ((t1==3)&& (t2==1)){// stone vs Scissor
 p1++;} else
 if ((t1==1)&& (t2==3)){
 p2++; }
 if ((t1==2)&& (t2==3)){// stone vs paper
 p1++;} else
 if ((t1==3)&& (t2==2)){
 p2++; }
 }
 System.out.println("The point fot player one is "+ p1);
 System.out.println("The point fot player two is "+ p2);

}
}
}

```

## תרגיל 14

```

import unit4.ioLib.*;
public class ch5E14 {
 public static void main(String args[]){
 int num;
 int sum1=0;
 int sum2=0;
 int sum3=0;
 num=IO.readInt("Enter numver with 3 digits");
 while ((num>=100)&&(num<=999)){
 int x;
 x=num % 10;
 sum1=sum1+x;
 x=(num/10)/10;

```

```
sum2=sum2+x;
x=num /100 ;
sum3=sum3+x;
num=IO.readInt("Enter number with 3 digits");
}
System.out.println(sum1);
System.out.println(sum2);
System.out.println(sum3);
}
}
```

## תרגילי פרק 6

### תרגיל 1

הערה: הפתרון "מדלג" על סעיף א. הפתרון אינו פותר את הסעיפים בכתיבת שיטה נפרדת לכל סעיף, פתור את התרגיל בכל דרך שתבחר

```
public class ch6ex1 {
 public static void main (String Args[]){
 int [] data ={3,5,-7,9,12,12,9,-7,5,3};
 int i;

 // א סעיף
 for(i=0;i<=data.length-1;i++){
 data[i]=data[i]*2;
 }

 // ב סעיף
 for(i=0;i<=data.length-1;i++){
 System.out.println((double)data[i]/2);
 }

 // ג סעיף
 int sum=0;
 for(i=0;i<=data.length-1;i++){
 sum=sum+data[i];
 }

 // ד סעיף
 int big= data[0];
 for(i=0;i<=data.length-1;i++){
 if(data[i]>big){
 big=data[i];
 }
 }

 System.out.println("The biggest number is: " + big);
 }
}
```

```
// ה סעיף
 boolean zero=true;
 for(i=0; i<=data.length-1;i++){
 if (data[i]==0){zero=false;}
 }
// ו סעיף
 int sum =0;
 for(i=0;i<=data.length-1;i++){
 if (data[i]%2==0){
 sum= sum+data[i];
 }
 }
// ז סעיף
 for(i=0;i<=data.length-1;i++){
 if(i%2==0){
 System.out.println(data[i]);
 }
 }
// ח סעיף
 int x=8;
 int place=-1;
 for(i=0;i<data.length;i++){
 if (data[i]==x){
 place=i;
 }
 }
// ט סעיף
 for(i=0;i<=data.length;i++){
 data[i]= data[i]*-1;
 }
}

}
```

## תרגיל 2

```

public class ch6ex2 {
 public static void main (String Args[]){
 int [][] data =new int[10][10];
 for(int i=0;i<=9;i++){
 for(int j=0;j<=9;j++){
 data[i][j]=(int)(Math.random()*100)+1;
 }
 }
 // א סעיף
 for(int i=0;i<=9;i++){
 int sum=0;
 for(int j=0;j<=9;j++){
 sum=sum+data[i][j];
 }
 System.out.println(sum);
 }
 // ב סעיף
 for(int i=0;i<=9;i++){
 int sum=0;
 for(int j=0;j<=9;j++){
 sum=sum+data[j][i];
 }
 System.out.println(sum);
 }
 // ג סעיף
 int suma=0;
 int sumb=0;
 for(int i=0;i<=9;i++){
 for(int j=0;j<=9;j++){
 if (i==j){
 suma=suma+data[i][j];}
 if(i+j==11){
 sumb=sumb+data[i][j];}
 }
 }
 }
}

```

```

 }
}

System.out.println("suma= "+suma);
System.out.println("sumb= "+sumb);

// ד סעיף
 boolean sim=true;
 for(int i=0;i<=9;i++){
 for(int j=0;j<=9;j++){
 if (data[i][j]!=data[9-i][9-j]){
 sim=false;
 }
 }
 }

// ה סעיף
 int n=0;
 int x=(int)(Math.random()*100)+1;
 for(int i=0;i<=9;i++){
 for(int j=0;j<=9;j++){
 if (data[i][j]==x){
 n++;
 }
 }
 }

}

}

}

```

### תרגיל 3

- א. לא נחוץ
- ב. לא נחוץ
- ג. נחוץ
- ד. לא נחוץ
- ה. לא נחוץ





## נספח יא' – רקורסיה

הרקורסיה הינה כל תכנותי רב עוצמה וחשוב, כל תוכנית ניתנת לכתיבה הן באופן איטרטיבי והן באופן רקורסיבי. באופן כללי על אף האופי הפחות יעיל של אלגוריתם רקורסיבי לעומת מקבילו האיטרטיבי השימוש ברקורסיה רחב במיוחד בבעיות מורכבות אשר הניסוח הרקורסיבי של פתרונם קצר פשוט וקריא.

נושא הרקורסיה אינו נכלל בתוכנית הלימודים יסודות, אולם, פטור בלא כלום אי אפשר ולכן הוספתי את הנושא כנספח בספר. אין הכתוב כאן מחליף את ההסבר המלא על נושא רקורסיה המופיע בספרים שונים ובמיוחד בספרים הכוללים את נושא עיצוב תוכנה. בנספח זה מופיעים כמה בעיות ופתרונם באופן רקורסיבי. ללומדים מספר זה את תוכנית יסודות ישמש נספח זה כהעשרה.

## מחלקה sum1\_To\_K

המחלקה מחשבת עבור  $k$  מספר הטבעי את הסכום:  $1+2+3+.....(k-2)+(k-1)+k$  החישוב מבוצע באמצעות מימוש אלגוריתם רקורסיבי.

האלגוריתם:

חשב\_סכום( $k$ )

אם  $k=0$  החזר 0 אחרת

חשב\_סכום =  $k$  + חשב\_סכום( $k-1$ )

```
import unit4.ioLib.*;
public class sum1_To_K
{
 public static int sum (int k){
 if(k==0)
 return 0;
 else
 return k+sum(k-1);
 }

 public static void main(String[] args)
 {
 int k=IO.readInt("Enter the number: ");
 System.out.println("The sum is:" + sum(k));
 }
}
```

## מחלקה power

המחלקה מחשבת את החזקה של שני מספרים שלמים,  $b$  בחזקת  $e$  ( $b^e$ )

האלגוריתם:

חשב\_חזקה( $b, e$ )

אם  $e=0$  החזר 1 אחרת

חשב\_חזקה =  $b * \text{חשב\_חזקה}(b, e-1)$

```
import unit4.ioLib.*;
public class power
{
 public static long p (int b, int e){
 if(e==0)
 return 1;
 else
 return b*p(b,e-1);
 }
 public static void main(String[] args)
 {
 int base=IO.readInt("Enter the base: ");
 int e=IO.readInt("Enter the e: ");
 System.out.println("The sum is:" + p(base,e));
 }
}
```

## מחלקה digit

המחלקה מחשבת את סכום הספרות של מספר שלם num

האלגוריתם:

חשב\_סכום\_ספרות(num)

אם num=0 החזר 0 אחרת

חשב\_סכום\_ספרות = num % 10 + חשב\_סכום\_ספרות(num / 10)

```
import unit4.ioLib.*;
public class digit
{
 public static long s (int num){
 if(num==0)
 return 0;
 else
 return num % 10 +s(num / 10);
 }
 public static void main(String[] args)
 {
 int num=IO.readInt("Enter the number: ");
 System.out.println("The sum of digits " + s(num));
 }
}
```

## מחלקה fact

המחלקה מחשבת את העצרת של מספר שלם k

האלגוריתם:

חשב\_עצרת(k)

אם k=0 החזר 1 אחרת

חשב\_עצרת = n \* חשב\_עצרת(n-1)

```
import unit4.ioLib.*;
public class fact
{
 public static long f (int k){
 if(k==0)
 return 1;
 else
 return k*f(k-1);
 }
 public static void main(String[] args)
 {
 int k=IO.readInt("Enter the number: ");
 System.out.println("The sum is:" + f(k));
 }
}
```

## fibonacci מחלקה

המחלקה מחשבת את הערך של האיבר ה- $n$  בסדרת פיבונצ'י. כידוע איבר בסדרת פיבונצ'י שווה לסכום שני האיברים הקודמים לו, האיבר הראשון והשני נקבעו כערך 1.  $a_n = a_{n-1} + a_{n-2}$

האלגוריתם:

חשב\_איבר\_הבא( $n$ )

אם  $n=1$  or  $n=2$  החזר 1 אחרת

חשב\_איבר\_הבא = חשב\_איבר\_הבא( $n-1$ ) + חשב\_איבר\_הבא( $n-2$ )

```
import unit4.ioLib.*;
public class fibonacci
{
 public static long fib (int n){
 if((n==1)|| (n==2))
 return 1;
 else
 return fib(n-1)+fib(n-2);
 }
 public static void main(String[] args)
 {
 int n=IO.readInt("Enter the position: ");
 System.out.println("The value is:" + fib(n));
 }
}
```

## מחלקה multy

כפל הוא בעצם חיבור חוזר, על רעיון זה מבוסס האלגוריתם הבא שמבצע כפל num1 ב-

num2

האלגוריתם:

חשב\_כפל(num1, num2)

אם num2=0 החזר 0 אחרת

חשב\_כפל = num1 + חשב\_כפל(num1, num2-1)

```
import unit4.ioLib.*;
public class multy
{
 public static long m (int num1, int num2){
 if(num2==0)
 return 0;
 else
 return num1 + m(num1,num2-1);
 }
 public static void main(String[] args)
 {
 int num1=IO.readInt("Enter the first num ");
 int num2=IO.readInt("Enter the second num ");
 System.out.println("The sum is:" + m(num1,num2));
 }
}
```

## מחלקה moneDigit

המחלקה מחשבת את מספר הספרות של מספר שלם num

האלגוריתם:

חשב\_מספר\_ספרות(num)

אם num=0 החזר 0 אחרת

חשב\_מספר\_ספרות = 1 + חשב\_מספר\_ספרות(num / 10)

```
import unit4.ioLib.*;
public class moneDigit
{
 public static long s (int num){
 if(num==0)
 return 0;
 else
 return 1 +s(num / 10);
 }
 public static void main(String[] args)
 {
 int num=IO.readInt("Enter the number: ");
 System.out.println("The num of digits " + s(num));
 }
}
```

הרעיון המרכזי שמקרה פשוט הוא מספר ללא ספרות ולכן תנאי העצירה מחזיר 0 עבור מספר ללא ספרות, אחרת אנו מוסיפים 1 למספר הספרות ומפעילים את הרקורסיה כאשר "מחסירים" ספרה מהמספר כלומר זו ההפעלה המפשטת את המספר. בכל הפעלה של הרקורסיה המספר "קטן" בספרה אחת עד שאנו מגיעים למספר ללא ספרות.



## מחלקה sum\_digit\_zugi

המחלקה מחשבת את סכום הספרות הזוגי של מספר שלם num

האלגוריתם:

חשב סכום ספרות זוגי(num)

אם num=0 החזר 0 אחרת

אם num % 10 הוא מספר זוגי

חשב סכום ספרות זוגי = num % 10 + חשב סכום ספרות (num / 10)

```
import unit4.ioLib.*;
public class sum_digit_zugi
{
 public static long s (int num){
 if(num==0)
 return 0;
 else
 if(num % 10 % 2 == 0)
 return num % 10 + s(num / 10);
 else return s(num / 10);
 }
 public static void main(String[] args)
 {
 int num=IO.readInt("Enter the number: ");
 System.out.println("The sum of zug digits " + s(num));
 }
}
```

השונה בדוגמא זו בהשוואה לרקורסיה ליישוב סכום ספרות מספר הוא שלפני שאנו מחשבים ומחברים את ערך הספרה הבאה לסכום הכולל אנו בודקים האם הספרה זוגית (מתחלקת ב-2 ללא שארית) ורק לאחר מכן אם התנאי חיובי מחברים את הספרה הנוכחית, אחרת "מדלגים" על הספרה ולא מחברים אותה לסכום הכולל.

## מחלקה sortNumber

המחלקה הזו בודקת האם ספרותיו של מספר ממוינות, השיטה במחלקת מחזירה אמת אם הספרות ממוינות ושקר אחרת.

דוגמאות

עבור המספר 97421 יוחזר 'אמת'

עבור המספר 3427 יוחזר 'שקר'

מקרה פשוט: כאשר המספר מכיל ספרה בודדת הרי ברור שהמספר ממויין מקרי קצה: אם שתי ספרות ימניות אינם מקיימות את התנאי אזי ברור שיש להחזיר 'שקר' ואין טעם להמשיך לבדוק כמו עבור המספר 3457 בו נספר 5 גדולה מהספרה 7, אבל אם התנאי מתקיים עבור שתי ספרות ימניות עדיין אי אפשר להחזיר 'אמת' ווצריך לבדוק את המשכו השמאלי של המספר. לכן כאשר שתי הספרות הימניות מקיימות את התנאי אנו צרכים "לקצץ" את הספרה הימנית ולהמשיך לבדוק את הספרה השמאלית עם המשך המספר כלומר עם הספרה הבאה:

התוכנית המלאה בג'אוה

```
import unit4.ioLib.*;
public class sortNumber{
 public static boolean s (long num){
 if(num<10)
 return true;
 else
 if (num<100)
 if (num %10 < num /10)
 return true;
 else
 return false;
 else
 if (num % 10 >= num /10)
 return false;
 else return s(num/10);
 }
 public static void main(String[] args) {
 long num=IO.readLong("Enter the number: ");
 if (s(num))
 System.out.println("The digit in the number are sort ");
 else
 System.out.println("The digit in the number are not sort ");
 }
}
```

## מחלקה divider

מציאת מחלק משותף הגדול ביותר של שני מספרים שלמים.

האלגוריתם של מציאת מחלק משותף הגדול ביותר נקרא האלגוריתם של אוקלידס,

המקרה הפשוט: שני המספרים שווים ולכן אחד מהם הוא מחלק משותף הגדול ביותר.

המקרה המורכב: אם המספרים שונים הרי אם הממ"ג המשותף מחלק את שני המספרים הוא מחלק גם את ההפרש בין המספרים ולכן בכל שלב נבצע החלפה של המספר הגדול יותר בהפרש.

האלגוריתם

ממ"ג ( מספרא, מספרב )

אם מספרא=מספרב החזר מספרא

אחרת

אם מספרא<מספרב

ממ"ג(מספרב, מספרא-מספרב)

אחרת

ממ"ג(מספרא,מספרב-מספרא)

```
import unit4.ioLib.IO;
public class divider
{
 public static long div (int x, int y){
 if(x==y)
 return y;
 else
 if(x>y)
 return div(y,x-y);
 else
 return div(x,y-x);
 }
 public static void main(String[] args)
 {
 int a=IO.readInt("Enter the number: ");
 int b=IO.readInt("Enter the number: ");
 System.out.println("The divider " + div(a,b));
 }
}
```

## מחלקה sumarray

האלגוריתם הבא מחשב את סכום איברי המערך שמספר איבריו הוא N בחישוב רקורסיבי.

חשב\_סכום\_מערך(מערך, n)

אם (n=0) החזר 0

אחרת

החזר חשב\_סכום\_מערך(מערך, n-1) + מערך(n)

```
public class sumArray {
 public int []num = {4,3,2,8,9,4,6,6,4,2};
 public int sum (int n){
 if (n<0)
 return 0;
 else
 return this.num[n]+sum(n-1);
 }
 public static void main(String args[]){
 sumArray data = new sumArray();
 System.out.println(data.sum(9));
 }
}
```

הגדרתי מחלקה הכוללת מערך. בשיטה main יצרתי אובייקט של המחלקה הכולל חבר אחד שהוא המערך.

השיטה לחישוב הסכום מופעלת באמצעות אובייקט של המחלקה sumArray והאלגוריתם המילולי שנכתב למעלה מיושם. בכל הפעלה רקורסיבית אנו מחסירים 1 והמערך "קטן" עד למצב בו מנסים לגשת לתא במערך -1 שאינו קיים.

## מחלקה bigArray

האלגוריתם הבא מחשב את המספר הגדול ביותר במערך שמספר איבריו הוא  $N$  בחישוב רקורסיבי.

מצא\_הגדול\_ביותר(מערך,  $n$ )

אם ( $n=0$ ) החזר תא 0

אחרת

החזר הכי גדול((מצא\_הגדול\_ביותר(מערך( $n-1$ ), תא  $n$  במערך)

הרעיון המרכזי באלגוריתם הוא :

מקרה פשוט: כאשר המערך הוא "בגודל 1" ( כלומר זה המספר האיברים שלא בדקנו ) אזי האיבר הגדול ביותר הוא האיבר היחיד

המקרה המורכב:

כאשר מספר התאים לבדיקה הוא  $n > 1$  אזי נמצא את האיבר המקסימלי בקריאה רקורסיבית של מערך "בגודל  $n-1$ " בהשוואה לאיבר המקסימום.

האלגוריתם ימומש באופן הבא:

```
public class bigArray {
 public double []data = {4,3,13,8,9,4,6,6,4,8};
 public double big (int n){
 if (n==0)
 return this.data[0];
 else
 return Math.max(this.data[n-1],big(n-1));
 }
 public static void main(String args[]){
 bigArray data = new bigArray();
 System.out.println(data.big(9));
 }
}
```

## נספח יב' – ממשק קלט פלט מעודכן – עדכון אחרון של ועדת המקצוע

הקלט הסטנדרטי יעשה ע"י המחלקה Scanner הנמצאת בספריה הסטנדרטית (java.util) החל מגירסא 1.5. מחלקה זו קוראת מהקלט "מילים" (tokens) המופרדות זו מזו ע"י רווחים. המתודות הבאות בלבד של המחלקה Scanner עשויות להופיע בבחינות:

**String next()** – מחזיר את ה המילה (token) הבאה בקלט

**int nextInt()** – מחזיר את המספר השלם הבא בקלט

**double nextDouble()** – מחזיר את המספר הממשי הבא בקלט

**boolean hasNext()** – האם יש עוד token בקלט?

שימוש במחלקת הקלט הסטנדרטי:

1. יש לייבא את המחלקה Scanner:

```
import java.util.Scanner;
```

(המשפט יופיע לפני כותרת המחלקה המשתמשת).

2. יש לייצר (פעם אחת) עצם מהמחלקה Scanner הקורא מהקלט הסטנדרטי:

```
Scanner s = new Scanner (System.in);
```

3. יש להשתמש במתודות לעיל על-פי הטיפוס הנקלט. למשל, עבור קליטת מספר שלם נרשום:

```
int num = s.nextInt();
```

דוגמא:

התוכנית הבאה מחשבת את הממוצע של ערכי הקלט:

```
import java.util.Scanner;
```

```
class Average {
```

```
public static void main(String[] args) {
```

```
double sum = 0;
```

```
int count = 0;
```

```
Scanner input = new Scanner(System.in);
```

```
while (input.hasNext()) {
```

```
sum = sum + input.nextDouble();
```

```
count++;
```

```
}
```

```
System.out.println("The average is " + sum/count);
```

```
}
```

```
}
```

## נספח יב' – ממשק קלט פלט מעודכן – עדכון אחרון של ועדת המקצוע

הקלט הסטנדרטי יעשה ע"י המחלקה Scanner הנמצאת בספריה הסטנדרטית (java.util) החל מגירסא 1.5. מחלקה זו קוראת מהקלט "מילים" (tokens) המופרדות זו מזו ע"י רווחים.

המתודות הבאות בלבד של המחלקה Scanner עשויות להופיע בבחינות:

**String next()** – מחזיר את ה המילה (token) הבאה בקלט

**int nextInt()** – מחזיר את המספר השלם הבא בקלט

**double nextDouble()** – מחזיר את המספר הממשי הבא בקלט

**boolean hasNext()** – האם יש עוד token בקלט?

שימוש במחלקת הקלט הסטנדרטי:

1. יש לייבא את המחלקה Scanner:

```
import java.util.Scanner;
```

(המשפט יופיע לפני כותרת המחלקה המשתמשת).

2. יש לייצר (פעם אחת) עצם מהמחלקה Scanner הקורא מהקלט הסטנדרטי:

```
Scanner s = new Scanner (System.in);
```

3. יש להשתמש במתודות לעיל על-פי הטיפוס הנקלט. למשל, עבור קליטת מספר שלם נרשום:

```
int num = s.nextInt();
```

דוגמא:

התוכנית הבאה מחשבת את הממוצע של ערכי הקלט:

```
import java.util.Scanner;
```

```
class Average {
```

```
 public static void main(String[] args) {
```

```
 double sum = 0;
```

```
 int count = 0;
```

```
 Scanner input = new Scanner(System.in);
```

```
 while (input.hasNext()) {
```

```
 sum = sum + input.nextDouble();
```

```
 count++;
```

```
 }
```

```
 System.out.println("The average is " + sum/count);
```

```
 }
```

```
}
```