# יחידות 9, 10 – מערכים

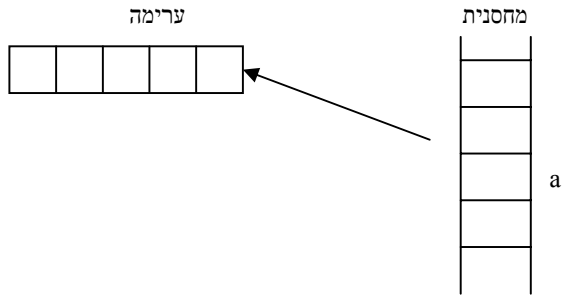
1. מערכים

לפעמים אנו זקוקים לכמות רבה של משתנים מאותו סוג בתוכנית שלנו. עד עכשיו הדרך היחידה לעשות זאת היא להגדיר באופן מפורט את כל המשתנים בתוכנית, כאשר לכל משתנה יש שם שונה. אולם כשמדובר בכמות גדולה מאוד של משתנים, העניין נהיה לא נוח ולא יעיל לתכנות.
לשם כך קיים מבנה בשפה שמאפשר לשמור כמות גדולה של משתנים, תחת שם אחד, וכך להתייחס לכולם באותו שם. המבנה הזה נקרא **מערך**.
מערך הוא בלוק של תאי זכרון מאותו סוג, שיושבים ברצף אחד אחרי השני, ויש להם שם אחד.
לדוגמא, כך נגדיר מערך של int בגודל 5:

```
int[] a = new int[5];
```

2. מערך כאובייקט

שורת ההגדרה של המערך רומזת לנו (בגלל השימוש במילה new) שיש כאן יצירה של אובייקט, ואכן, מערך הוא למעשה אובייקט שנמצא בזכרון הערימה. לאחר השורה הנ"ל נראה הזכרון כך:




שימו לב – למרות שמדובר באובייקט, אופן ההגדרה של מערך שונה מאופן ההגדרה של אובייקטים אחרים שראינו עד כה. למרות זאת, נא לא לשכוח שעדיין מדובר באובייקט, כלומר, ישנו משתנה במחסנית שמצביע לאובייקט בערימה, שם נשמר רצף תאי הזכרון המבוקש.

שאלה 9.1

מה יוגדר כתוצאה מהשורה הבאה?

```
int[] arr;
```

3. קווים לדמותו של המערך

המערך שלעיל בנוי מחמישה תאים שכל אחד מהם הוא int. לכל אחד מהתאים ישנו אינדקס, ובכדי לגשת לתא בודד במערך צריך לציין את שם המערך, ואת האינדקס של התא.
נתבונן במערך שוב –

האינדקסים של המערך ממוספרים מ-0 ועד 4. שימו לב – התא הראשון במערך הוא תא מספר 0, התא האחרון הוא תא מספר 4, וישנם חמישה תאים!

השורות הבאות מדגימות גישה לאיברי המערך:

```
a[0] = 2;    // assign 2 into the first cell
a[1] = a[0] * 3; // the second cell equals 6
System.out.println(a[0] + ", " + a[1]); // prints 2, 6
```

חשוב לזכור שכל תא במערך הוא בעצם משתנה int רגיל, רק שהגישה אליו חייבת להיעשות דרך האינדקס.

שאלה 9.2

מה יקרה כתוצאה מהשורות הבאות?

```
a[5] = 3;
a[10] = 3;
```

דוגמא:

התוכנית הבאה מגדירה מערך של double בגודל 5, וקולטת לתוכו ערכים מהמשתמש:

```
Scanner scan = new Scanner(System.in);
double[] dArr = new double[5];
for(int i = 0; i < 5; i++)
   dArr[i] = scan.nextDouble();
```

שאלה 9.3

למה תנאי סיום הלולאה הוא i<5 ולא i <= 5?

4. מערך עם גודל שנקבע בזמן ריצה

בג'אווה אנו לא חייבים לקבוע את גודל המערך בזמן קומפילציה, אלא ניתן להקצות אותו דינמית בזמן ריצה. התוכנית הבאה קולטת מהמשתמש מספר, ומקצה מערך של int בגודל שנקלט:

```
int[] a;
int n = scan.nextInt();
a = new int[n];
```

שאלה 9.4

כיצד יושפע גודל המערך אם בהמשך לתוכנית שלעיל נרשום את השורה:

```
n = 10;
```

5. גודל המערך כתכונה

ראינו שגודל המערך יכול להיקבע דינמית בזמן הריצה. גודל המערך הוא נתון מאוד חשוב לריצה תקינה של התוכנית.

לדוגמא, בהמשך התוכנית הנ"ל נוסיף לולאה שמציבה את המספר 2 בכל תאי המערך:

```
for(int i = 0; i < n; i++)
   a[i] = 2;
```

שאלה 9.5

מה הבעייתיות בלולאה הנ"ל?

לאובייקט המערך יש תכונה ציבורית שנקראת length והיא מחזיקה את גודלו של המערך.

שימו לב – גודל המערך הוא קבוע, מהרגע שנוצר ועד סוף חייו. אי אפשר להוסיף תאי זכרון, או למחוק תאי זכרון ממערך שהוקצה!

בכדי להתייחס לנתון המדויק של גודל המערך, נוכל להשתמש בתכונה length. נכתוב מחדש את הלולאה הקודמת, כך שתהיה תלויה ב-length ולא ב-n:

```
for(int i = 0; i < a.length; i++)
   a[i] = 2;
```

שימו לב – length הוא תכונה ולא שיטה!

שאלה 9.6

האם ניתן לשנות את גודל המערך ע"י שינוי תכונת ה-length, כך למשל:

```
a.length++;
```

תרגיל 1

ברצונינו לכתוב תוכנית שמקצה מערך של int בגודל 5 וקולטת מספרים חיוביים מהמשתמש, עד למספר 1- שיהווה את סוף הקלט. כתבו את הטיפול במערך כך שגם אם המערך מתמלא נוכל להמשיך לקלוט נתונים ולשמור אותם. רמז – אמנם אי אפשר לשנות את גודלו של המערך, אבל ראינו שניתן להגדיר מערכים באופן דינמי בזמן ריצת התוכנית.

6. איתחול מערך בשורת ההגדרה

ניתן לאתחל מערך בערכים גם בשורת ההגדרה, כך:

```
int[] b = {3, 1, 4, 2, 6};
```

המערך b יהיה בגודל 5, כאשר בתא הראשון שלו יהיה המספר 3, בתא השני המספר 1 וכן הלאה.

שימו לב שבצורת רישום זאת אין צורך בפקודה new, המערך מוקצה אוטומטית. ניתן להשתמש באיתחול זה רק בזמן הגדרת המערך. הפקודה הבאה גוררת שגיאת קומפילציה:

```
b = {5, 1, 3, 6, 8};
```

התוכנית הבאה מגדירה מערך, ומעתיקה אותו בסדר הפוך למערך אחר:

```
int[] a = {1, 2, 3, 4, 5};
int[] b = new int[a.length];
int i, j;
for(i=0, j=b.length-1; i < a.length; i++, j--)
   b[j] = a[i];
```

תרגיל 2

כתוב תוכנית שמגדירה מערך, והופכת את סדר האיברים שבו, בלי שימוש במערך עזר.

7. <u>מערכים ושיטות</u>

כמו כל אובייקט, גם אובייקט של מערך ניתן לשלוח לשיטה כפרמטר, ושיטה יכולה להחזיר מערך בערך חוזר. חשוב לזכור שכמו כל אובייקט, מערכים עוברים לשיטות By Reference, כלומר כל שינוי שמבצעת השיטה במערך מתבצע למעשה במערך המקורי.

דוגמא: מה מאוחסן בתא הראשון של מערך arr אחרי הפעלת השיטה?

```
public static void main()
{
   int[] arr = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   f(arr);
}
…
public static void f(int[] a)
{
   a[0] = 8;
}
```

<u>תרגילים נוספים</u> – הכנסת מספר למערך ממויין, חיפוש בינארי.

8. <u>מערכים של אובייקטים</u>

ניתן גם להגדיר מערך שהאיברים שלו הם אובייקטים. למשל, בהינתן המחלקה Point (המחלקה מוגדרת ביחידה 7 עמוד 9), נוכל להגדיר מערך של נקודות כך:

```
Point[] pArray = new Point[5];
```

<u>שאלה 9.7</u>

מה מכילים כעת תאי המערך?

כל תא במערך הוא למעשה מצביע לאובייקט מסוג נקודה.

שימו לב – יצירת אובייקט המערך לא יוצרת גם אובייקטים של נקודות! בפרט, כרגע השורה הבאה תגרום לשגיאת ריצה (למה?).

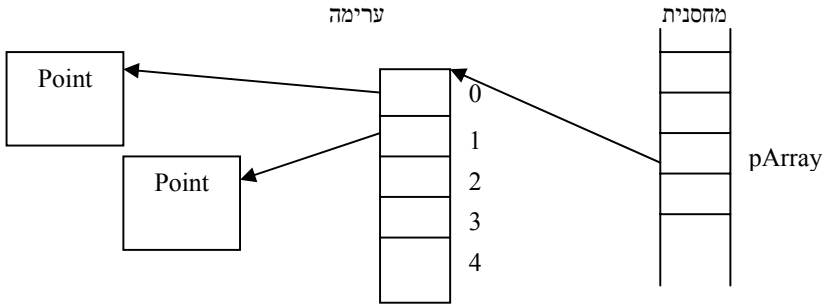
```
pArray[0].setX(3);
```

בכדי ליצור אובייקטים אמיתיים בתוך המערך, חייבים להשתמש ב-new, כך:

```
pArray[0] = new Point(1, 2);
pArray[1] = new Point(pArray[0]);
```

ושוב, כמו במשתנים רגילים, חשוב לזכור שכל תא במערך הוא למעשה אובייקט של נקודה, וניתן להשתמש בו כמו שהיינו משתמשים באובייקט רגיל (רק שצריך לציין את האינדקס של התא).

אחרי השורות הנ"ל הזכרון יראה כך:




9. <u>מערכים דו-מימדיים</u>

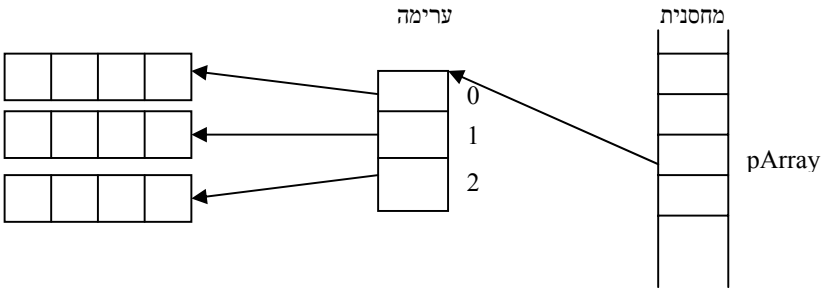
ניתן גם להגדיר מערכים דו-מימדיים, בצורה הבאה:

```
int[][] a1 = new int[3][4];
```

בצורה הזאת אנו מגדירים בסה"כ 12 משתנים מסוג `int` כאשר המערך עצמו מאורגן בשלוש שורות (המימד הראשון) וארבע עמודות (המימד השני). הגישה לתא בודד חייבת כעת לכלול את שני האינדקסים, אחד לשורה ואחד לעמודה:

```
a1[0][0] = 5; // assigns 5 into the top left cell
a1[2][3] = 2; // assigns 2 into the bottom right cell
```

מבחינת הזכרון, המצב נראה קצת יותר מסובך, אבל בכדי להבין את זה נזכור שמערך חד מימדי הוא אובייקט, ולכן מערך דו-מימדי הוא למעשה מערך חד-מימדי של מצביעים למערכים חד מימדיים... הזכרון יראה כך:




<u>שאלה 9.8</u>

מה מייצג כעת הערך `a1.length`?

שימו לב שכיוון שעכשיו ישנו מערך של מערכים, בכדי להגיע לגודל המימד השני צריך לכתוב כך:

```
a1[0].length;
```

כלומר, גודל המערך שמוצבע ע"י התא הראשון במערך החד מימדי.

<u>שאלה 9.9</u>

מה ההבדל בין השורה שלעיל לשורה הבאה? האם יתקבלו ערכים שונים?

```
a1[1].length;
```

בשל העובדה שמערך דו מימדי הוא בעצם מערך חד מימדי של מערכים חד מימדיים, קל לראות שלמעשה אנחנו יכולים להגדיר מערך דו-מימדי כאשר מספר התאים בכל שורה הוא שונה, למשל כך:

```
int[][] b = new int[3][];
b[0] = new int[5];
b[1] = new int[10];
b[2] = new int[3];
```

<u>שאלה 9.10</u>

איך יראה הזכרון כתוצאה מהשורות שלעיל?

10. <u>וזה ממשיך להסתבך – מערכים רב מימדיים</u>

ומכאן כבר לא קשה לראות שבג'אווה אין הגבלה על מספר המימדים שמערך יכול להכיל. למשל, מערך תלת מימדי, או ארבע מימדי:

```
int[][][] arr3d = new int[10][5][10];
int[][][][] arr4d = new int[10][5][10][5];
```

וכן הלאה. הגישה לתא בודד במערך חייבת להיעשות כרגיל דרך האינדקסים של התא (כאשר חייבים לפרט את כל האינדקסים בכל המימדים).

וכמובן שכל אחד מהמערכים יכול להכיל גם אובייקטים ולא רק טיפוסים פשוטים...