

סיכום שיעור

דיברנו על העמסת פעולות. טכניקה שבעזרתה ניתן על ידי פעולות חשבון פשוטות, ניתן לבצע פעולות מורכבות על עצמים.

הדוגמה שהצגנו בכיתה הייתה פעולת חיבור של "תיבות" במחלקה Box. לאחר החלטה שרירותית שחיבור תיבות יעשה על ידי חיבור המקצועות התואמים שלהן ויצירת תיבה חדשה, קודם כתבנו מתודה בשם add שעובדת על התיבה (העצם הנוכחי) כלהלן:

```
public Box add(Box another){
    //returns a Box object and receives
    //a Box object as a parameter.
    double nlength = this.length + another.length;
    double nwidth = this.width + another.width;
    double nheight = this.height + another.height;
    Box newbox = new Box(nlength, nwidth, nheight);
    return newbox;
}
```

כאן עצרנו לרגע וחשבנו. המתודה add הרי מוגדרת בתוך המחלקה Box, ולמרות זאת היא גם מחזירה עצם מסוג Box וגם מקבלת פרמטר מסוג Box. זה לא נראה כ"כ טבעי, כיצד אפשר כבר להשתמש במחלקה Box אם עדיין לא סיימנו להגדיר אותה?

מסתבר שאפשר לעשות זאת. ממש כמו שניתן לקרוא לפונקציה מתוך עצמה (רקורסיה).

אז יש לנו את add שיכולה לחבר תיבות, כאשר היא פועלת על תיבה מסוימת ומקבלת תיבה נוספת כפרמטר בסוגריים. דוגמה להפעלת add:

```
Box a=new Box(4,6,7);
Box b=new Box(1,2,3);
Box c=a.add(b);
c.print();
```

נקבל הדפסה של:

(5; 8; 10)

לכאורה, הכל בסדר. אבל משהו לא לגמרי מאוזן. הרי יכולנו גם לכתוב
 ולהשיג אותה תוצאה.

a ו-b הן במעמד זהה של תיבות, אז מדוע על אחת פועלת המתודה בעזרת הזימון (נקודה) והשניה ניתנת כפרמטר?

כדי לשנות זאת וגם כדי לפשט את מראה הקוד של תכנות מונחה עצמים, פיתחו אפשרות להגדיר את פעולות החשבון גם בין עצמים. כך זה יראה בשפת C#:

```

public static Box operator+(Box left, Box right){
    double nlength = left.length + right.length;
    double nwidth = left.width + right.width;
    double nheight = left.height + right.height;
    Box newbox = new Box(nlength, nwidth, nheight);
    return newbox;
}

```

שם המתודה מורכב ממילת המפתח operator ולאחריה הפעולה עצמה (חיבור במקרה זה). הפעם ישנם שני פרמטרים שהן תיבות. המתודה הינה סטטית כיוון שאינה מקבלת עצם מובנה שעליו היא פועלת ולכן בעצם היא ניראית כפונקציה רגילה. מלבד זאת, הפקודות הן אותן פקודות שהיו לנו ב-add. גם הערך המוחזר ניראה אותו דבר.

נסו לכתוב מתודה דומה עבור חיסור תיבות, אשר בנוסף, תבדוק אם החיסור חוקי (אם לאחר חיסור קיבלנו מקצוע שלילי או אפס ניתן הודעת שגיאה).

את "ההיגיון" של 'חיבור' תיבות על ידי חיבור המקצועות התואמים אנחנו 'המצאנו', אבל זה לא חשוב. התוכנה מיישמת את הרעיונות שלנו באשר הם.

דבר נוסף שעליו חזרנו היה אלגוריתם המיון של מערכים, בשם: Bubble sort. ביצענו אותו על ידי שתי לולאות מקוננות, כאשר בכל מעבר על הלולאה הפנימית לפחות תא אחד 'שוקע' למקומו הטבעי במערך, ולכן הלולאה הפנימית תרוץ עד מספר התאים במערך פחות האינדקס של הלולאה החיצונית (שאומר כמה מספרים כבר הסתדרו במקומם). עוד שיכלול היה לסמן בדגל שיאמר אם הייתה איזו החלפה בין תאים, שכן במצב שלא הייתה שום החלפה, המערך כבר ממוין

```

bool sorted = false;
for(int i=0; i<10 && !sorted; ++i){
    sorted=true;
    for(int j=0; j<a.Length-i-1; ++j){
        if(a[j]>a[j+1]){
            int temp=a[j];
            Console.WriteLine("temp=", temp);
            a[j]=a[j+1];
            a[j+1]=temp;
            sorted=false;
        }
    }
}

```