

קורס מונחה עצמים בשפת JAVA

מרצה: עסאקלה שאדי

פתרון דוגמא 1

```
import java.util.Scanner;
public class Program {
    public static void main(String[] args) {
        int x,y, ind;
        Scanner input = new Scanner (System.in);
        System.out.println("Enter the first number:");
        x = input.nextInt();
        System.out.println("Enter the second number:");
        y=input.nextInt();
        if(x<y)
        {
            for(ind=x;ind<=y;ind++)
                if(ind%2==0)
                    System.out.println(ind);
        }
        else
        {
            for(ind=y;ind<=x;ind++)
                if(ind%2==0)
                    System.out.println(ind);
        }
    }
}
```

פתרון דוגמא 2

```
import java.util.Scanner;
public class Program {
    public static void main(String[] args) {
        int num,count=0;
        float avg=0;
        Scanner input = new Scanner (System.in);
        System.out.println("Enter a number:");
        num = input.nextInt();
        while(num!=-1)
        {
            count++;
            avg+=num;
            System.out.println("Enter a number:");
            num = input.nextInt();
        }
        if(count!=0)
            avg=(float)avg/count;
        System.out.println("The average is:"+avg);
    }
}
```

הגדרת משתנים כקבועים

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

```
public class Program {
```

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

```
final double pi=3.14;
```

```
double r,res;
```

```
System.out.println("Enter the radius ");
```

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

```
r= input.nextDouble();
```

```
res = pi*r*r;
```

```
System.out.println("Result is: " + res);
```

```
pi = pi+1;
```

```
}
```

```
}
```

בעזרת final המשתנה π יהיה קבוע ולא ניתן לשנות את ערכו.

תהיה שגיאת קומפילציה

משתנים שמוגדרים כ static

- משתנה משותף לכל המופעים של המחלקה – (לכל האובייקטים) – יורחב בהמשך.
- יתרון:
- ניתן להגדיר משתנה סטטי שהוא גם קבוע, למשל PI.
- ניתן להגדיר משתנה סטטי שערכו משתנה לפי הגדרות מסוימות של המחלקה, שהשינוי הוא בעל משמעות למשל: כמה פעמים קראנו לפונקציה מסוימת.
- ניתן לפנות למשתנה סטטי וציבורי ללא הגדרה של אובייקט, אלא דרך שם המחלקה.
- הערה: מקובל ואף מומלץ להגדיר משתנים סטטיים וקבועים עם שמות בעלי משמעות ועם אותיות גדולות.

שירותי מחלקה - פונקציה

- כמו בשפות אחרות גם ב JAVA ניתן להגדיר פונקציות בתוך המחלקה.

- פונקציות במחלקה נקראות גם שירותי מחלקה

- שירותים כאלה מוכרזים על ידי מילת המפתח **static**

- שיטה סטטית היא שיטה הנכתבת בתוך מחלקה, אך אין צורך לייצר אובייקט על מנת להפעיל אותה

- התחביר של הגדרת שרות הוא:

```
<modifiers> <type> <method-name> ( <paramlist> ) {  
    <statements>  
}
```

- **<modifiers>** הם 0 או יותר מילות מפתח מופרדות ברווחים (למשל `public` `static`)

- **<type>** מציין את טיפוס הערך שהשרות מחזיר

- **void** מציין שהשרות אינו מחזיר ערך

- **<paramlist>** רשימת הפרמטרים הפורמליים, מופרדים בפסיק, כל אחד מורכב מטיפוס הפרמטר ושמו – גם אם שני פרמטרים או יותר מאותו סוג.

שירותי מחלקה – החזרת ערך

- כמו בשפות אחרות גם ב JAVA ניתן להגדיר פונקציות שמחזירות ערך.
- החזרת ערך מתבצעת על ידי המילה השמורה `.return`.
- דוגמא שירות (פונקציה) שמחזיר סכום המספרים מ 1-10:

```
public class Program {  
    public static int Sum()  
    {  
        int i,sum=0;  
        for(i=1;i<=10;i++)  
            sum+=i;  
        return sum;  
    }  
  
    public static void main(String[] args) {  
        int sum;  
        sum=Sum();  
        System.out.println(sum);  
    }  
}
```

שירותי מחלקה – פונקציה שמקבלת פרמטרים

- דוגמא – פונקציה אשר מקבלת פרמטר שהוא רדיוס של מעגל ומחזירה את שטח המעגל:

```
import java.util.Scanner;
public class Program {
    public static double circle(double rad)
    {
        double res;
        res = rad*rad*3.14;
        return res;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        double rad,res;
        System.out.println("Enter radius plz:");
        rad = input.nextDouble();
        res = circle(rad);
        System.out.println(res);
    }
}
```


שירותי מחלקה

- יש מחלקות שרק נותנים שירות, הן מכילות רק שיטות ללא תכונות
- במחלקה כזו כל השיטות הן סטטיות, אין משמעות לאובייקט כי אין תכונות.
- דוגמא, המחלקה Math מכילה שיטות לחישובים מתמטיים שימושיים למתכנתים.

משתנים שאינם יסודיים (non primitive variables)

■ פרט ל- 8 הטיפוסים היסודיים, כל שאר הטיפוסים ב Java אינם פרימיטיביים

■ הספרייה התקנית של Java מכילה יותר מ- 3000 טיפוסים (!) ואנו כמתכנתים נשתמש בהם ואף ניצור טיפוסים חדשים

■ מערכים ומחרוזות אינם טיפוסים יסודיים

■ משתנה מטיפוס שאינו יסודי נקרא הפנייה (reference type)

■ כל המשתנים הבסיסיים נוצרים על שטח זיכרון הנקרא stack והם מועברים לפונקציות by value

■ בניגוד לשפת C, ב- JAVA כל האובייקטים נוצרים על שטח הזיכרון הנקרא heap ומוקצים ע"י new, ומועברים לפונקציות אוטומטית by reference

הפניות ומשתנים יסודיים

- ביצירת משתנה מטיפוס יסודי אנו יוצרים מקום בזיכרון בגודל ידוע שיכול להכיל ערך מטיפוס מסוים
- ביצירת משתנה הפנייה אנו יוצרים מקום בזכרון, שיכול להכיל כתובת של מקום אחר בזכרון שם נמצא תוכן כלשהו

חשוב: בשפת JAVA יש מחלקת מעטפת לכל טיפוס בסיסי למשל Integer הם מוגדרים כ-immutable, משמע כל שינוי יוצר אובייקט חדש ולכן אינו משפיע על המשתנה המקורי, כלומר, גם אם נשלח אותו לפונקציה ויהיה שינוי בפונקציה הערך של המשתנה נשאר ללא שינוי. דוגמא בשקף הבא:

מחלקת מעטפת - דוגמא

```
public class Program {  
    public static void func(Integer num)  
    {  
        num=100;  
    }
```

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

```
        Integer x=50;
```

```
        func (x);
```

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

```
    }
```

```
}
```

למרות הקריאה לפונקציה, עדיין
יודפס 50 הערך המקורי של x

שימוש למחלקת מעטפת

- המחלקה מכילה מתודות רבות שעשויים לעזור למתכנת.
 - למשל המתודה `parseInt` אשר ממירה מחרוזת למספר שלם.
- דוגמא:

```
public class Program {
```

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

```
        int res;
```

```
        res = Integer.parseInt("20");
```

```
        System.out.println(res);
```

```
    }
```

```
}
```



יודפס 20

הפניות ועצמים

■ ביצירת משתנה מסוג הפנייה, אנו בעצם יוצרים מקום בזיכרון שמכיל כתובת של מקום אחר שמכיל תוכן כלשהו.

■ מחרוזת ומערך הם מסוג הפנייה

בהגדרה הבאה: `String str="good";`

המשתנה `str` נקרא הפנייה, התוכן שעליו הוא מצביע נקרא עצם (object)

■ אזור הזיכרון שבו נוצרים עצמים שונה מאזור הזיכרון שבו נוצרים משתנים מקומיים והוא מכונה Heap (זיכרון ערימה)

המחלקה String

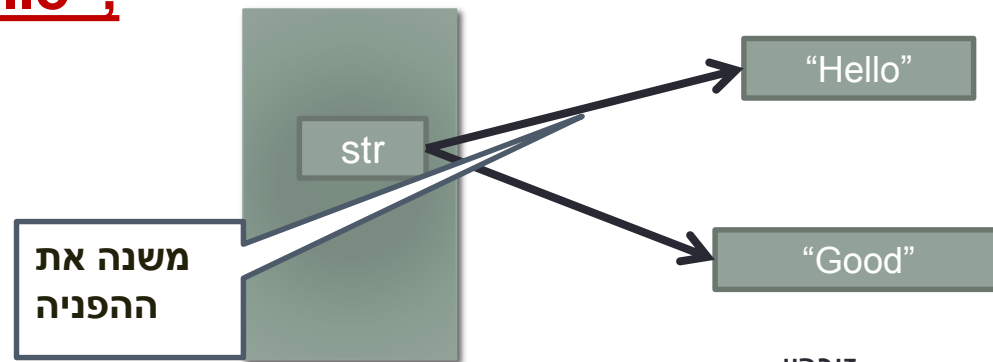
- המחלקה String הינה מחלקה של Java.
- לשים לב להבדל בין שפת C לבין JAVA. כאן זה לא הגדרת מערך של תווים.
- המחלקה מייצגת טיפוס נתונים מסוג מחרוזת, כאשר ההשמה תבוצע בצורת ליטרל או בצורת אובייקט.
- כאשר משנים את ערכו של משתנה מסוג String מבוצעת ע"י java הקצאת מקום חדשה

- **String str="Hello";**

- :

- :

- **Str = "Good";**



מחסינית התכנית

הגדרת משתנה מסוג String – ליטרל או אובייקט!

- ביצוע השמה כליטרל לאותו ערך לשני משתנים שונים, שניהם יצביעו (הפניה) על אותו מקום בזיכרון.
- ביצוע השמה כאובייקט, כל אובייקט יצביע על אזור חדש בזיכרון.

String s1="abc";

String s2="abc";

s1,s2 מצביעים על אותו אזור בזיכרון.

String s3= new String("abc");

String s4= new String("abc");

s1,s2 מצביעים על אזורים חדשים ושונים בזיכרון.

פעולות על הפניות

■ בהגדרה הבאה:

```
public static void main(String[] args) {  
    String str1="Hello";  
    String str2="Good";  
}  
}
```

יתכן ש `str1` נמצא בכתובת 1000, והתוכן שלו הוא כתובת 2000, כך שכתובת 2000 מכילה את הערך Hello.

פעולות על הפניות

- בדוגמא הקודמת, ניתן להגדיר ששני משתני מסוג הפנייה יצביעו על אותה כתובת שמכילה תוכן כלשהו:
למשל: ■

```
public static void main(String[] args) {  
    String str1="Hello";  
    String str2="Good";  
    str2=str1;  
    System.out.println(str2);  
}
```

כאן גם str2 יצביע על כתובת 2000, ולכן יודפס Hello.
הערה: יש לאתחל הפנייה עם הערך null במידה ואין ערך התחלתי. זה יעזור לבדיקה בהמשך אם אותחל או לא.

שרשור מחרוזות

```
public class Program {  
  
    public static void main(String[] args) {  
  
        String str1="Hello";  
        String str2="Good";  
        String str3=null;  
        str3 = str1 + " " + str2;  
        System.out.println(str3);  
    }  
}
```

יציאה:
Hello Good

פניה לעצם המוצבע

- כשעובדים עם הפניות אנו בעצם עובדים על עצם מוצבע.
- כדי לפנות לעצם משתמשים ב '.' (נקודה) לשים לב שזה לא מצביע ולא משתמשים בחצים.
- בעזרת הגישה לעצם המוצבע אפשר לבקש בקשות, לשאול שאלות (וכמובן לקבל תשובות) ולגשת למאפיינים פנימיים.
- הבקשות השאלות והמאפיינים הפנימיים משתנים מעצם לעצם לפי טיפוסו (אם כי יש מספר קטן של בקשות שאפשר לבקש מכל עצם ב Java)

דוגמא

```
public class Program {
```

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

```
        String str = "Waww I like Java";
```

```
        int len = str.length();
```

```
        String upper = str.toUpperCase();
```

```
        String lower = str.toLowerCase();
```

```
        System.out.println("String length is "+ len);
```

```
        System.out.println("String in UPPERCASE is '"+ upper + "'");
```

```
        System.out.println("String in lowercase is '"+ lower + "'");
```

```
    }
```

```
}
```

לקבלת אורך מחרוזת

קבלת מחרוזת באותיות גדולות

קבלת מחרוזת באותיות קטנות

פונקציות שימושיות במחלקה String

- בהנחה שיש שני אובייקטים str1, str2 מסוג String

פונקציה	הסבר
str1.equals(str2);	יחזיר true אם המחרוזות שווים אחרת יחזיר false
str1.length();	יחזיר את אורך מחרוזת str1
str1.charAt(i);	יחזיר את התו שנמצא במקום i במחרוזת str1
str1.indexOf(ch);	יחזיר את האינדקס של תו במחרוזת str1
str1.contains(str2);	יחזיר אמת אם str2 נמצאת בתוך str1
str1.isEmpty();	יחזיר אמת אם str1 היא מחרוזת ריקה אחרת יחזיר false

מערכים

- אוסף של משתנים מאותו סוג
- אברי המערך יכולים להיות משתנים בין אם יסודיים או התייחסות – הפנייה
- משתנה מערך הוא מטיפוס הפנייה
- תאים במערך יושבים בדרך כלל ברצף בזיכרון (לזכור Java רצה על
- מכונה וירטואלית!) כך שגישה סידרתית אליהם עשויה להיות יעילה.

מערכים

- תזכור: במערכים של טיפוסים פרימיטיביים, הערכים הפרימיטיביים יושבים **במערך עצמו** (במקום שהוקצה לו בזיכרון), במערכים של טיפוס הפנייה, הערכים הנמצאים במערך הן **הפניות** לעצמים הנמצאים במקום אחר בזיכרון.
- לזכור: אסור לחרוג מגבולות המערך.
- כדי לגשת לאיבר מסוים במערך נשתמש באופרטור הסוגריים המרובעים
- חשוב: מכיוון שמערך הוא פנייה וניתן לפנות לעצם, בשונה משפת C אפשר לדעת את אורך המערך על ידי השירות `length`.

מתן ערכים לאברי המערך

- א. בשורת ההגדרה. לדוגמא:

```
public class Program {
```

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

```
        int[] arr1 ={5,6,7};
```

```
        String [] str={"good","yes","nice"};
```

```
        int i;
```

```
        for(i=0;i<arr1.length;i++)
```

```
            System.out.println(arr1[i]);
```

```
        for(i=0;i<str.length;i++)
```

```
            System.out.println(str[i]);
```

```
    }
```

```
}
```

מערך עם אברים בסיסיים

מערך עם אברים שהם
פניות

מתן ערכים לאברי המערך

- ב. בגוף התכנית/פונקציה. כאן יש להשתמש במילה השמורה: `new`.
- לדוגמא:

```
public class Program {
```

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

```
        int[] arr1;
```

```
        int i;
```

```
        arr1 = new int[3];
```

```
        arr1[0]=20;
```

```
        arr1[1]=30;
```

```
        arr1[2]=50;
```

```
        for(i=0;i<arr1.length;i++)
```

```
            System.out.println(arr1[i]);
```

```
    }
```

```
}
```

לציין גודל מערך

מתן ערכים לאברי המערך

ג. קליטה על ידי המשתמש:

• לדוגמא:

```
public class Program {  
  
    public static void main(String[] args) {  
        int[] arr1;  
        int i;  
        double avg=0;  
        Scanner input = new Scanner(System.in);  
        arr1 = new int[3];  
        for(i=0;i<3;i++)  
        {  
            arr1[i]=input.nextInt();  
        }  
        for(i=0;i<3;i++)  
            avg+=arr1[i];  
        avg=avg/3.0;  
        System.out.println("The average is:" + avg);  
    }  
}
```

קליטת מערך והדפסת
ממוצע

יצירת עצם מטיפוס מערך וגישה לאיבריו

- חשוב: אברי מערך שהוקצה ע"י `new` מאותחלים אוטומטית לפי טיפוסם (הוצגו בטבלה):
- הטיפוסים הפרימיטיביים השלמים מאותחלים ל- 0
- טיפוס הפנייה מאותחל ל- `null`
- הטיפוסים הפרימיטיביים הממשיים מאותחלים ל- 0.0
- הטיפוסים הפרימיטיבי `boolean` מאותחלים ל- `false`
- הטיפוסים הפרימיטיבי `char` מאותחל לתו שערך ה `Unicode` שלו הוא 0

שיתוף בין מערכים (sharing, aliasing)

- מכיוון שמערך הוא הפנייה, כאשר מערך אחר עם הפנייה לאותו מקום בזיכרון זה אומר שהעצם משותף לשניהם, ולכן, ניתן לשנות את המערך דרך שתי ההפניות. דוגמא:

```
public class Program {
```

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

```
        int[] arr1, arr2;
```

```
        int i;
```

```
        arr1 = new int[3];
```

```
        arr1[0]=20;
```

```
        arr1[1]=30;
```

```
        arr1[2]=50;
```

```
        arr2=arr1;
```

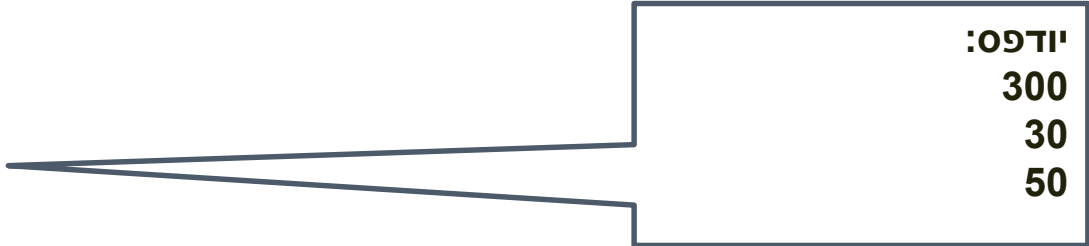
```
        arr2[0]=300;
```

```
        for(i=0;i<arr1.length;i++)
```

```
            System.out.println(arr1[i]);
```

```
    }
```

```
}
```



יודפס:
300
30
50

הפרוטקציה של מערכים ומחרוזות

- כדי לייצר עצם ב java יש להשתמש באופרטור new. מערכים ומחרוזות פטורים מזה. ניתן ליצור עצם מחרוזת ע"י שימוש בסימן המירכאות ("hello"), וניתן ליצור עצם מערך ע"י שימוש בסוגרים מסולסים ({1,2,3}).
- גישה לאברי המערך על ידי [] ולא על ידי נקודה.
- ניתן לבצע שרשור למחרוזות.

המערך כהפנייה

- ב-JAVA כל האובייקטים נוצרים על שטח הזיכרון הנקרא heap ומוקצים ע"י new, ומועברים לפונקציות אוטומטית by reference
- כאשר פונקציה או שיטה מקבלת כפרמטר אובייקט, הוא תמיד מועבר by reference (הפניה)
- מערך הוא גם אובייקט, ולכן מועבר לפונקציה by reference

```
public class Program {  
    public static void swap1(int a,int b)  
    {  
        int temp=a;  
        a=b;  
        b=temp;}  
    public static void swap2(int[] a,int[] b)  
    {  
        int temp=a[0];  
        a[0]=b[0];  
        b[0]=temp;}  
    public static void main(String[] args) {  
        int[] arr1={10}, arr2 = {20};  
        swap1(arr1[0],arr2[0]);  
        System.out.println("arr[0] = "+ arr1[0] + " arr2[0]= "+ arr2[0]);  
        swap2(arr1,arr2);  
        System.out.println("arr[0] = "+ arr1[0] + " arr2[0]= "+ arr2[0]);  
    }  
}
```

יודפס:

arr[0] = 10 arr2[0]= 20
arr[0] = 20 arr2[0]= 10

המחלקה Arrays

- `Arrays.copyOf` היא שיטה סטטית המקבלת כפרמטר מערך להעתקה ואת הגודל של המערך החדש. השיטה מבצעת השמה בין איבר לאיבר. אם גודל המערך החדש יותר גדול, שאר איבריו יאופסו.
- `Arrays.equals` היא שיטה סטטית המקבלת 2 מערכים ובודקת האם הם שווים: האם אורכם זהה והאם האיברים במקומות המתאימים זהים
- `Arrays.toString` היא שיטה סטטית המקבלת כפרמטר מערך ומחזירה מחרוזת כך שאיברי המערך מופרדים ע"י פסיק

```
import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
        int arr1[] = {3,4,5};
        int arr2[];
        int arr3[] = {3,4,5};
        for (int a:arr1)
        { System.out.println(a);}
        arr2 = Arrays.copyOf(arr1, 10);
        for (int a:arr2)
        {System.out.println(a);}
        System.out.println("Are arr1 and arr3 equals?" + Arrays.equals(arr1, arr3));
    }
}
```


חוזרים לשירות מחלקה

- **משתנים מקומיים** נקראים גם משתנים זמניים, משתני מחסנית או משתנים אוטומטיים והם נמצאים בתוך גוף השירות.
- משתנים מקומיים יכולים להופיע באותו שם אם בשתי פונקציות שונות, אין קשר בין ערכי המשתנים.
- הגדרת משתנה זמני צריכה להקדים את השימוש בו.
- תחום הקיום של המשתנה הוא גוף השירות בלבד.
- חייבים לאתחל או לשים ערך באופן מפורש במשתנה לפני השימוש בו.

שירות מחלקה שמחזיר ערך

- כאשר יש ניתוב (בעזרת תנאי) בתוך גוף הפונקציה, והפונקציה מחזירה ערך, יש לדאוג שבכל ענף יש החזרת ערך.
- לדוגמא:

```
public class Program {
```

```
    public static int func(int a,int b)
```

```
    {
```

```
        if(a>b)
```

```
            return a;
```

```
    }
```

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

```
        int x;
```

```
        x=func(5,7);
```

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

```
    }
```

```
}
```

קטע קוד שלא יעבור
קומפילציה

משתנים גלובליים – משתני מחלקה

- עד כה ראינו משתנים מקומיים – משתנים זמניים המוגדרים בתוך מתודה, בכל קריאה למתודה הם נוצרים וביציאה ממנה הם נהרסים.
- ב Java ניתן גם להגדיר **משתנים גלובליים** (global variables)
 - משתנים אלו נקראים גם:
 - משתני מחלקה (class variables)
 - אברי מחלקה (class members)
 - שדות מחלקה (class fields)
 - מאפייני מחלקה (class properties/attributes)
 - שדות סטטיים (static fields/members)
- משתנים אלו יוגדרו בתוך גוף המחלקה אך מחוץ לגוף של מתודה כלשהי, וסומנו ע"י המציון **static**

משתני מחלקה לעומת משתנים מקומיים

- משתנים אלו, שונים ממשתנים מקומיים בכמה מאפיינים:
 - **תחום הכרות:** מוכרים בכל הקוד (ולא רק מתוך פונקציה מסוימת)
 - **משך קיום:** אותו עותק של משתנה נוצר בזמן טעינת הקוד לזיכרון ונשאר קיים בזיכרון התוכנית כל עוד המחלקה בשימוש
 - **אתחול:** משתנים גלובליים מאותחלים בעת יצירתם. אם המתכנת לא הגדירה להם ערך אתחול - יאותחלו לערך ברירת המחדל לפי טיפוסם
(0, false, null)
 - **הקצאת זיכרון:** הזיכרון המוקצה להם נמצא באזור ה Heap (ולא באזור ה- Stack)
 - ניתן לקבע ערך של משתנה ע"י ציון המשתנה כ `final` כאשר למשתנה `final` ניתן לבצע השמה פעם אחת בדיוק. כל השמה נוספת לאותו משתנה תגרור שגיאת קומפילציה

דוגמא למשתנה מחלקה

- המשתנה סופר כמה פעמים קראנו לפונקציה מסוימת:

```
public class Program{  
    public static int count=0;  
    public static void func()  
    {  
        count++;  
        System.out.println("This is a call to function number:" + count);  
    }  
    public static void main(String[] args) {  
        func();  
        func();  
        func();  
    }  
}
```

שם מלא (qualified name)

- כאשר יש מחלקת שירות ונרצה לקרוא לשירות דרך מחלקה אחרת יש להשתמש בשם מלא, כולל שם מחלקה ושם שירות (מתודה):

במידה וקיימת המחלקה הבאה:

```
public class callingFromAnotherClass {  
  
    public static int sum(int a, int b)  
    {  
        return a+b;  
    }  
}
```

פלט:
res=50

קוד עם שגיאת קומפילציה:

```
public class First {  
    public static void main(String[] args)  
    {  
        int a=20,b=30,res;  
        res = sum(a,b);  
        System.out.println("res = " + res);  
    }  
}
```

קוד עובד:

```
public class First {  
    public static void main(String[] args) {  
        int a=20,b=30,res;  
        res = callingFromAnotherClass.sum(a,b);  
        System.out.println("res = " + res);  
    }  
}
```

שם מלא (qualified name)

- באותה דרך ניתן לפנות למשתנה גלובלי ממחלקה אחרת:

```
public class First {  
    public static void main(String[] args) {  
        int a=20,b=30,res;  
        res = callingFromAnotherClass.sum(a,b);  
        System.out.println("res = " + res);  
        a++;  
        b++;  
        res = callingFromAnotherClass.sum(a,b);  
        System.out.println("res = " + res);  
        System.out.println("Counter = " +  
            callingFromAnotherClass.counter);  
    }  
}
```

במידה וקיימת המחלקה הבאה:

```
public class  
callingFromAnotherClass {  
  
    public static int counter;  
    public static int sum(int a, int b)  
    {  
        counter++;  
        return a+b;  
    }  
}
```

פלט:

```
res = 50  
res = 52  
Counter = 2
```

תרגול

• בנה מחלקה ראשית שמכילה את הפונקציות (שירות מחלקה) הבאות:

1. פונקציה אשר מקבלת כפרמטרים מערך `arr1` ומספר שלם `n`. הפונקציה תבנה מערך חדש שמכיל את המערך `arr1` בצורה מסודרת מהקטן לגדול, בנוסף היא תשים את `n` בתוך המערך החדש במקום הנכון לפי הסדר. דוגמא: אם הגיע מערך `4,1,3` ומספר `n=2` המערך החדש יהיה `1,2,3,4` יש להדפיס את המערך החדש.
 2. פונקציה אשר מקבלת כפרמטרים שתי מחרוזות ומדפיסה את כל התווים המשותפים בין השתיים. להתעלם מכפילויות.
 3. פונקציה אשר מקבלת כפרמטר מחרוזת, ומחזירה `true` אם המילה הראשונה במחרוזת מופיעה שוב פעם בתוך המחרוזת, אחרת תחזיר `false`. יש לפתור מבלי שימוש בפונקציות עזר קיימות.
- התכנית הרשאית תקלוט ערכים ותשלח לפונקציות הנ"ל.