קורס JAVA

מרצה: ד"ר עסאקלה שאדי

הרצאה בנושא מחלקות

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 המתכנתת 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();
}
}
```

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.

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:

במידה וקיימת המחלקה הבאה:

```
public class
callingFromAnotherClass {

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

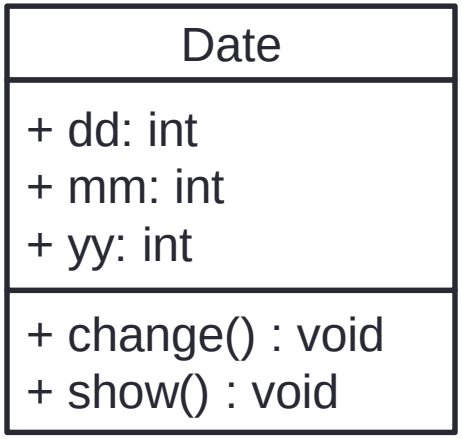
```
public class First {
public static void main(String[] args) {
int a=20,b=30,res;
res = callingFromAnotherClass.sum(a,b);
System.out.println("res = " + res);
a++;
b++;
res = callingFromAnotherClass.sum(a,b);
System.out.println("res = " + res);
System.out.println("Counter = " +
callingFromAnotherClass.counter);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

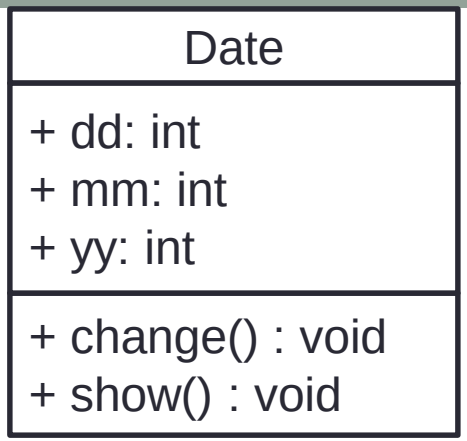
פלט:

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

מחלקה – לתיאור עצם

- מחלקה היא טיפוס נתונים בעלת תכונות ופעולות
  - תכונות הן משתנים
  - פעולות הן פונקציות (מתודות) שניתן להפעיל על טיפוס הנתונים
- לדוגמא:
  - תאריך מאופיין ע"י ימים, חודשים ושנה
  - על תאריך ניתן לבצע פעולה של "שינוי" שבו משנים ימים, חודשים או שנים
  - על תאריך ניתן לבצע פעולה של הצגה אשר מציגה את התאריך, או חלק ממנו.





מימוש המחלקה Date

```
public class Date{
        public int dd;
        public int mm;
         public int yy;

        public void change() {
                dd++;
                  mm++;
                }
        public void show(){

        System.out.print(dd+"/"+mm);
                System.out.print("/"+yy);

                        }
}
```

יצירת אובייקט למחלקה

- כדי ליצור במחלקה הראשית אובייקט של המחלקה נרשום את הפקודה:
- className objName= new className();

- new פקודה מיוחדת אשר בונה אובייקט.
- ערכי התכונות של האובייקט מאותחלים לאחר הקצאה בערכי ברירת מחדל לפי טיפוס הנתונים שלהם.

שימוש באובייקט

- כדי לפנות לתכונות משתמשים בסימן . (נקודה). למשל פעולת השמה:
- objName.var1 = value;
- כדי להפעיל פעולה (פונקציה):
- objName.funcName();

הפעלת עיקרון הכימוס

- הכימוס בגדול הוא הצורך להפוך את המחלקה לקופסא שחורה. הגישה לתכונות המחלקה צריכה להיות מבוקרת.
- לצורך גישה לתכונות המחלקה יש שלוש הרשאות:
- public, ניתן לגישה מכל מחלקה.
- private, גישה מהפונקציות של אותה מחלקה בלבד
- protected, יתנים לגישה מפונקציות של אותה המחלקה בלבד ומהמחלקות היורשות ממנה. (על ההורשה יורחב בהמשך).

הפעלת עיקרון הכימוס – סגירת גישה ושימוש ב set/get

```
public class Main {
        public static void
main(String[] args) {
        Date d1=new Date();
        d1.setDay(10);
        d1.setMonth(10);
        d1.setYear(2018);
        d1.show();
        }
}
```

```
public class Date {
private int dd;
private int mm;
private int yy;
public void setDay(int day)
{
        dd=day;
}
public void setMonth(int month)
{
        mm=month;
}
public void setYear(int year)
{
        yy=year;
}
public void show()

{
System.out.println(dd+"/"+mm+"/"+yy);
}}
```

הפעלת עיקרון הכימוס – סגירת גישה ושימוש ב set/get

```
public class Main {
        public static void
main(String[] args) {
        Date d1=new Date();
        d1.setDay(10);
        d1.setMonth(10);
        d1.setYear(2018);
   System.out.println(d1.getDay());
        d1.show();
        }
}
```

```
public int getDay()
}
return dd;
{
public int getMonth()
}
return mm;
{
public int getYear()
}
return yy;
```

10 ידפיס

המצביע this

- כאשר משתמשים בשמות משמעותיים בדרך כלל יוצרים משתנה מקומי בעל שם זהה לתכונה במחלקה (משתנה מחלקה).
- משתמשים במצביע this עבור משתנה המחלקה, כדי להבדיל בין משתנה מקומי למשתנה מחלקה.
- this הינו מצביע המייצג התייחסות לאובייקט הנוכחי שעליו עובדים.
- כדי לבצע השמה של פרמטר במשתנה מחלקה עם שמות זהים הסינטקס הוא:

```
this.תכונה = שם_פרמטר;
```

תיקון למחלקה Date

```
public class Main {
		public static void
main(String[] args) {
		Date d1=new Date();
		d1.setDay(10);
		d1.setMonth(10);
		d1.setYear(2018);
		d1.show();
		}
}
```

```
public class Date {
private int dd;
private int mm;
private int yy;
public void setDay(int dd)
{
		this.dd=dd;
}
public void setMonth(int mm)
{
		 this.mm=mm;
}
public void setYear(int yy)
{
		this.yy=yy;
}
public void show()

{
System.out.println(dd+"/"+mm+"/"+yy);
}}
```

תרגיל

- בנה מערך מסוג Date, הכנס ערכים.
- הדפס את המערך על המסך בעזרת show()

```
public class Program {
	public static void main(String[] args) {
		Date[] dateArray;
		dateArray = new Date[5];
		int i;
		for(i=0;i<5;i++)
			dateArray[i] = new Date();
		for(i=0;i<5;i++)
		{
			dateArray[i].setDay(20+i);
			dateArray[i].setMonth(4+i);
			dateArray[i].setYear(2014+i);
		}
		for(i=0;i<5;i++)
			dateArray[i].show();
	}
}
```