

# תכנות מונחה עצמים

# Object Oriented Programming

# תכנות מונחה עצמים

# Object Oriented Programming

**גישה ישנה:**

– תוכנית ארוכה שמחולקת לאוסף של פונקציות.

**גישה חדשה:**

Object Oriented Programming – גישה שאמורה להתאים בדרך החשיבה  
שלנו כבני אדם (עבודה עם דברים יותר מוחשיים).

# תכנות מונחה עצמים

# Object Oriented Programming

**ארגוני היסודות:**

- 1) מחלקה
- 2) אובייקט

# מחלקה Class

מבנה לוגי שמאגד בתוכו משתנים ו/או פונקציות.

משתני המחלקה - נקראים מאפיינים members או תכונות attributes.

פונקציות המחלקה - נקראות מethods או שיטות operations.

# אובייקט Object

אובייקט = Object = עצם = מופע = Instance

אובייקט = משתנה מסווג המחלקה

int num;

Person p1;

Worker w1;

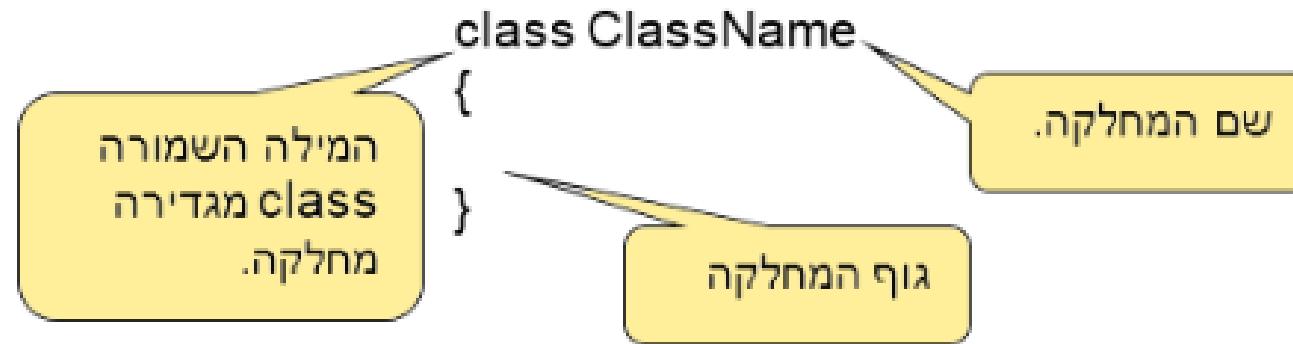
Student s1;

Dog d1;

# המחלקה Worker - תרשימים UML בסיסי

```
+-----+
|           Worker           |
+-----+
| + id : int
| + name : string
| + workingHours : int
| + salaryPerHour : double
+-----+
| + calculateSalary() : double
| + Print() : void
+-----+
```

# יצירת מחלקה



גוף המחלקה מכיל מאפיינים ו/או מתודות.

```
class Worker  
{  
...  
}
```

# יצירת אובייקט

Worker w1 = new Worker();

w1 = 2000

שם =  
ת.ז =  
שעות עבודה = 0  
תעריף שעה = 0

# השפת ערך למאפיין והפעלת מתודה

```
Worker w1 = new Worker();
```

```
w1.name = "Shaked";
```

```
w1.Print();
```

w1 = 2000

שם = "Shaked"  
ת.ז = ""  
שעות עבודה = 0  
תעריף שעה = 0

# המחלקה Worker - בואו נשדרג אותה

בואו נוסיף:

הרשאות גישה

Setters

Getters

בנאים

# הרשאות גישה למאפיינים וمتודות

## Access Modifiers

1. הרשאת גישה נועדה לצורך הסתרת מידע (בדרך כלל מאפיינים) ומתן גישה לחלקים שרוצים לחושף בתוכנית (בדרך כלל מתודות).
2. מי שקבע מי יכול לגשת למאפיין או מתודה זאת המחלקה ולא האובייקט.

# הרשאות גישה למאפיינים וمتודות

3. ישן חמוץ רמות הרשאה (נכיר בהתחלה את השנאים הראשונים):

א. **private** ( פרטי) – מותרת הגישה מתוך המחלקה הנוכחית בלבד (זאת גם הרשות בירית מחדל של המאפיינים והמתודות). בד"כ נגדיר מאפיינים כפרטיים.

ב. **public** ( ציבורי) – מותרת הגישה מכל מקום. בד"כ נגדיר מתודות שרצים לחשוף החוצה ציבוריות.

הרשאות גישה אחרות (נכיר בהמשך):

ג. **protected** ( מוגן)

ד. **internal** ( פנימי)

ה. **internal protected** ( פנימי מוגן)

# Methods ו- Get Set

Methods שנגידר כ- `public`.

ב"כ מקובל לא לחת גישה ישירה למאפייני המחלקה (נגדיר אותם כ- `private` למשל) והגישה אליהם תהיה ע"י Methods `Set` ו- `Get`.

Method `Set` – נקראת גם `Setter method` – תפקידה לחת אפשרות לעדכן המאפיין.

Method `Get` – נקראת גם `Getter method` – תפקידה להחזיר את המאפיין למי שביקש אותו.

בתוך Methods `Set` ו- `Get` נוכל להוסיף לוגיקה שבודקת מי זה ש牒קש לבצע את השינוי ולפי זה נחליט האם נרצה לבצע את השינוי או לא.

# מתודות ו - Get - Set דוגמא

```
class Worker
{
    private string name;

    public void SetName(string name)
    {
        this.name = name;
    }

    public string GetName()
    {
        return this.name;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Worker w1 = new Worker();
        w1.name = "Sean"; //is this ok?
        w1.SetName("Sean"); //is this ok?
        Console.WriteLine(w1.GetName());
    }
}
```

# בנאי – מетодת בנייה - constructor

תפקיד הבנאי הוא אתחול האובייקט בזמן הגדרתו (כדי שהאובייקט יקבל ערכים אמיתיים).

המבנה נקרא לאחר שהזכירן עברו האובייקט כבר הוקצה.

מבנה יכול לקבל ערכים אך אינם מוחזיר ערכים (אפילו לא void).

קיימים בניי' ברירת מחדל של השפה שתפקידו לשים ערכי' ברירת מחדל עברו המאפיינים.

אם בונים בניי' משלנו אנחנו דורסים override את בניי' ברירת המחדל של השפה.

ניתן לעשות העממת בניאים constructor overloading.

שם הבנאי כשם המחלקה.

# **סוגי בناאים**

**בנאי ברירת מחדל**

**בנאי שמקבל פרמטרים**

**בנאי מעתיק**

# בנייה ברירת מחדל

לא מקבל ערכים ולא מחזיר ערכים.

קיים בשפה בניין ברירת מחדל שמאתחל את הערכים בערכיו ברירת מחדל.

אם בונים בניין ברירת מחדל משלמו אז ידרס בניין ברירת המחדל של השפה.

דוגמא:

```
public Worker()
{
    this.name = "No Name";
    this.id = "000";
    this.workingHours = 100;
    this.salaryPerHour = 50;
}

class Program
{
    static void Main(string[] args)
    {
        Worker w1 = new Worker();
    }
}
```

# בנייה שמקבל פרמטרים

מקבל ערכים ולא מוחזיר ערכים.

אם בונים בניין משלנו אז ידרס בניין בירית המחדל של השפה.

חשבונים אובייקט נהיה חייבים לשלוח את הפרמטרים שציינו בזמן הגדרת האובייקט.

דוגמא:

```
public Worker(string name, string id, double workingHours, double salaryPerHour)
{
    this.name = name;
    this.id = id;
    this.workingHours = workingHours;
    this.salaryPerHour = salaryPerHour;
}

class Program
{
    static void Main(string[] args)
    {
        Worker w2 = new Worker("Sean", "042842154", 150, 60);
    }
}
```

# בנאי מעתיק copy constructor

מקבל אובייקט קיימ ולא מחזיר ערכאים.

דומה לבני עם פרמטרים אך מקבל אובייקט קיימ.

תפקידו להעתיק את תוכן האובייקט ששלחנו לאובייקט החדש.

אם בונים בני משלנו אז ידרס בני בניתה המחדל של השפה.

דוגמא:

```
public Worker(Worker obj)
{
    this.name = obj.name;
    this.id = obj.id;
    this.workingHours = obj.workingHours;
    this.salaryPerHour = obj.salaryPerHour;
}

class Program
{
    static void Main(string[] args)
    {
        Worker w2 = new Worker("Sean", "042842154", 150, 60);
        Worker w3 = new Worker(w2);
    }
}
```

# Constructor overloading

ניתן לעשות העמלה במבנה (בנאיים) שונים עם מספר פרמטרים שונה ו/או טיפוס פרמטרים שונה).

ופועל הבנאי המתאים לפי הפרמטרים שהבנו את האובייקט.

```
public Worker()
{
    this.name = "No Name";
    this.id = "000";
    this.workingHours = 100;
    this.salaryPerHour = 50;
}

public Worker(string name, string id, double workingHours, double salaryPerHour)
{
    this.name = name;
    this.id = id;
    this.workingHours = workingHours;
    this.salaryPerHour = salaryPerHour;
}

public Worker(Worker obj)
{
    this.name = obj.name;
    this.id = obj.id;
    this.workingHours = obj.workingHours;
    this.salaryPerHour = obj.salaryPerHour;
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Worker w1 = new Worker();

        Worker w2 = new Worker("Sean", "042842154", 150, 60);

        Worker w3 = new Worker(w2);
    }
}
```

# המחלקה UML - תרשימים מתקדם

```
+-----+
|           Worker           |
+-----+
| - id : int
| - name : string
| - workingHours : int
| - salaryPerHour : double
+-----+
| + Worker()
| + Worker(id:int, name:string,
|           workingHours:int, salaryPerHour:double)
| + Worker(other:Worker)
| + GetId() : int
| + SetId(id:int) : void
| + GetName() : string
| + SetName(name:string) : void
| + GetWorkingHours() : int
| + SetWorkingHours(h:int) : void
| + GetSalaryPerHour() : double
| + SetSalaryPerHour(s:double) : void
| + CalculateSalary() : double
| + Print() : void
+-----+
```

## **תרגיל בית - המחלקה Worker - תרשימים UML מתקדם**

1. עליך למש את המחלקה Worker לפי תרשימים UML המוצג.
2. בפונקציה Main עליך ליצור שלושה אובייקטים מסוג המחלקה Worker כאשר כל עובד עליך לבנות בעזרתו מבנה אחר.
3. קרא לפונקציות Get/Set המתאימות.
4. קרא לפונקציה Print להדפסת העובדים.