

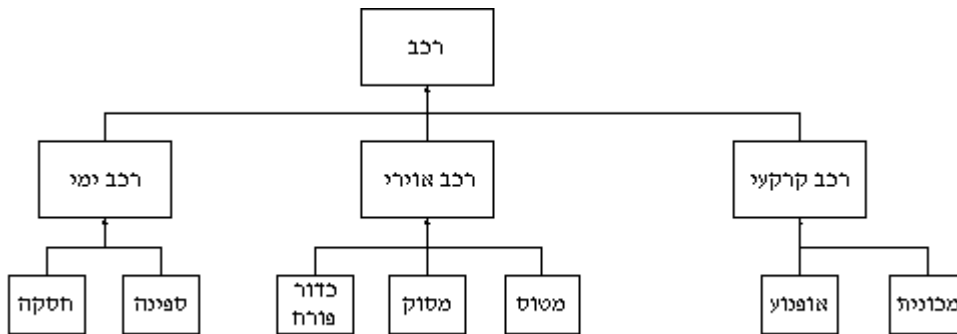
## Inheritance הורשה

### תרגיל

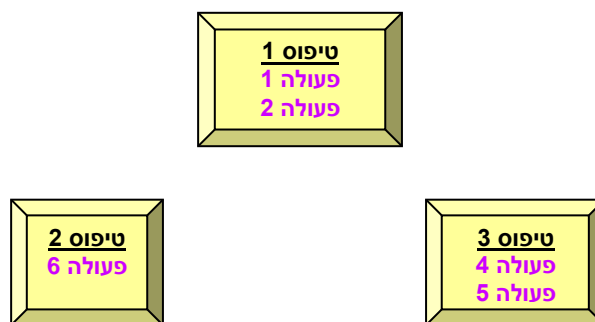
ניתן להגדיר מחלקה חדשה על בסיס מחלקה קיימת. למחלקה החדשה ישנן כל התכונות והפעולות שירשה מהמחלקה שעל-פיה הוגדרה, ובנוסף ניתן להגדיר פעולות נוספות במחלקה החדשה, או לשנות פעולות שירשה.

המחלקה המורשה קרויה מחלקת בסיס (base class), מחלקת אב (parent) או מחלקת-על (superclass).

המחלקה היורשת קרויה מחלקה נגזר (derived class), בן (child), או תת-מחלקה (subclass). לדוגמה,



אם נגדיר שלושה טיפוסים כך שטיפוס 3 וטיפוס 2 יורשים מטיפוס 1:



בצורה זו, לעצם שיוגדר מטיפוס 3, יש את מגוון הפעולות: 1,2,4,5, לעצם שיוגדר מטיפוס 2, יש את מגוון הפעולות: 1,2,6

**תרגיל**

במפעל שלוש דרגות תעסוקה :

עובד ייצור במחלקה,

מנהל/ת מחלקה

ומנהלת בחברה.

בכל הדרגות משכורת נטו מחושבת על פי גבוה ההכנסה :

גובה שכר	אחוזים
$\leq 4000$	0
$> 4000$	25%
$> 8000$	40%
$> 15000$	50%

נגדיר עובד/ת בחברה **Worker** ע"י התכונות הבאות :

ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובד/ת פנימי, מספר שעות עבודה בחודש ושכר בסיסי לשעה.

חישוב המשכורת מוגדר כמכפלת שעות העבודה בשכר לשעה.

נגדיר מנהל/ת במחלקה **Manager** ע"י התכונות הבאות :

ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובד/ת פנימי, שכר חודשי.

חישוב המשכורת מוגדר כשכר חודשי.

נגדיר מנהלת החברה **SeniorManager** ע"י התכונות הבאות :

ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובדת פנימי, שכר חודשי, מענק כספי.

חישוב משכורת : למשכורת החודשית יש להוסיף מענק כספי.

**(הגדרה)**

נגדיר עובד/ת בחברה **Worker** ע"י התכונות הבאות :

ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובד/ת פנימי, מספר שעות עבודה בחודש ושכר בסיסי לשעה.

חישוב המשכורת מוגדר כמכפלת שעות העבודה בשכר לשעה.

נגדיר מנהל/ת במחלקה **Manager** ע"י התכונות הבאות :

ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובד/ת פנימי, שכר חודשי.

חישוב המשכורת מוגדר כשכר חודשי.

נגדיר מנהלת החברה **SeniorManager** ע"י התכונות הבאות :

ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובדת פנימי, שכר חודשי, מענק כספי.

חישוב משכורת : למשכורת החודשית יש להוסיף מענק כספי.

בתחילת עבודה ו"פירוק" פרטי העובד למספר מחלקות נרכז את ארבעת התכונות הכלליות  
למחלקת Person (ת.ז., שם פרטי, שם משפחה, כתובת מגורים)

```
public class Person
{
    private int _id;
    private string _firstName;
    private string _lastName;
    private string _address;

    public Person(int id, string last, string first, string
        add) {
        _id = id;
        _firstName = first;
        _lastName = last;
        _address = add;
    }
    public Person(Person p) // copy constructor
    {
        _id = p._id;
        _firstName = p._firstName;
        _lastName = p._lastName;
        _address = p._address;
    }
    public string ToString()
    {
        string s = "ID"+_id + "Last&First name"+_lastName + " ";
        s = s + _firstName + " lives in " + _address;
        return s;
    }
    public int GetID() { return _id; }
    public string GetFirstName() { return _firstName; }
    public string GetLastName() { return _lastName; }
    public string GetAddress() { return _address; }
    public void SetFirstName(string first)
        { _firstName = first; }
    public void SetLastName(string last) { _lastName = last; }
    public void SetAddress(string add) { _address = add; }
}
```

ארבע התכונות הראשונות מהוות בסיס לכל תכונה של אדם או עובד.  
 בדוגמה שלנו נמשיך "בפרוק" פרטי העובדים ונגדיר תכונה נוספת המייצגת את כל העובדים  
 בארגון – מספר עובד.

לשם כך נגדיר מחלקה כללית **Employee** המכילה את התכונה הנוספת.  
 כיוון שמחלקת Person מכילה את שאר התכונות נזמן זאת באמצעות מנגנון הירושה:

**<מחלקת האב ממנה יורשים> : <שם המחלקה החדשה (היורשת)>** public class

מבנה:

## public class Employee : Person

ב Java במקום נקודתיים נגדיר את המילה extend שפירושה: האריך, התרחב; נמשך. ב-C#  
 נגדיר את מנגנון הירושה באמצעות **base**.

```
class Employee:Person
{
    private int empId;

    // בנאי
    public Employee(int id, string lastName, string
        firstName, string address, int empId)
        :base(id, lastName, firstName, address)
    {
        this.empId = empId;
    }

    public int GetEmpId() { return this.empId; }
    public virtual double Salary() { return 0; }
    public double NettoSalary()
    {
        double netsal = this.Salary();
        double percent = 0;
        if (netsal > 15000) percent = 0.5;
        else if (netsal > 8000) percent = 0.4;
        else if (netsal > 4000) percent = 0.25;
        return netsal * (1 - percent);
    }
}
```

```
public override string ToString()
{
    string st = "Employee id: " + this.empId;
    st = st + "\nearns nettoSalary: ";
    st = st + +NettoSalary() + " NIS";
    return base.ToString() + "\n" + st;
}
}
```

הגדרנו את הפעולה הבאה:

```
public virtual double Salary() { return 0; }
```

כך שבעתיד נוכל לרמוס אותה ובעת זימון מתבצעת הפעולה הקרובה ביותר להגדרה.

נזכר בדרישה הראשונית:

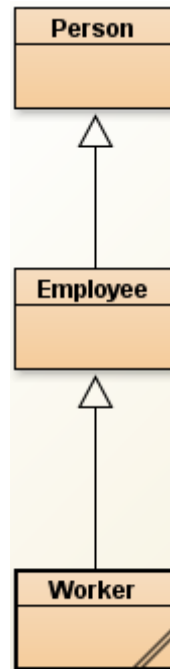
עובד/ת בחברה Worker מוגדר באמצעות התכונות הבאות:

ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובד/ת פנימי, מספר שעות עבודה בחודש ושכר בסיסי לשעה.

חישוב המשכורת מוגדר כמכפלת שעות העבודה בשכר לשעה.

```
public class Worker : Employee
{
    private int perHourSalary;
    private int amountHours;
    // בנאי
    public Worker ( int id, string lastName, string
        firstName, string address, int
        empId, int perHour, int
        amount) :
        base(id, lastName, firstName,
            address,empId)
        {
            perHourSalary = perHour;
            amountHours = amount;
        }
    public Worker (Worker w) //בנאי העתקה
        :base (w.GetID(), w.GetLastName(),
            w.GetFirstName(), w.GetAddress(),
            w.GetEmpId())
        {
            perHourSalary = w.perHourSalary;
            amountHours=w.amountHours;
        }
    public override string ToString()
    {
        return base.ToString() + " works "+
            this.amountHours + " hours for " +
            this.perHourSalary + " shekels per hour";
    }
}
```

```
public override double Salary()  
{  
    return this.perHourSalary*this.amountHours;  
}  
}
```





אם הגדרנו בתכנית הראשית את הפעולות הבאות :

```
Worker w = new Worker
    (111,"Zohar","Doron","Shpinoza",1234, 1500, 3);
Console.WriteLine(w);
Console.WriteLine();
Console.WriteLine("Salary -> " + w.Salary() );
Console.WriteLine();
Console.WriteLine("Net salary -> " + w.NettoSalary());
```

יתקבל הפלט :

```
ID 111 Last & First name Zohar Doron lives in Shpinoza
Employee id 1234 works at 1 department &
earns nettoSalary 3375 NIS works 3 hours for 1500 shekels per hour
Salary -> 4500
Net salary -> 3375
```

הפעם נגדיר מנהל/ת מחלקה בחברה Manager באמצעות התכונות הבאות :  
ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובד/ת פנימי, שכר חודשי.  
חישוב המשכורת מוגדר כשכרו החודשי.

```
public class Manager:Employee
{
    private int monthlySalary;

    public Manager (int id, string lastName,
                    string firstName, string address,
                    int empId, int sal)
        :base (id, lastName, firstName,
              address,empId)
    {
        monthlySalary = sal;
    }
    public Manager( Manager m)
        :base (m.GetID(), m.GetLastName(),
              m.GetFirstName(), m.GetAddress(),
              m.GetEmpId())
    {
        monthlySalary=m.monthlySalary;
    }
    public override string ToString()
    {
        return base.ToString()+" Monthly salary: "+
            monthlySalary;
    }
    public override double Salary()
    {
        return monthlySalary;
    }
}
```

עבור שורות הקוד הבאות:

```
Manager m = new Manager
    (222, "Zur", "Iris", "A-H", 2,7000);
Console.WriteLine(m);
Console.WriteLine();
Console.WriteLine("Salary -> " + m.Salary());
Console.WriteLine();
Console.WriteLine("Net salary -> "+m.NettoSalary());
```

יתקבל הפלט:

```
ID 222 Last & First name Zur Iris lives in A-H
Employee id 2 works at 2 department &
earns nettoSalary 5250 NIS Monthly salary: 7000
Salary -> 7000
```

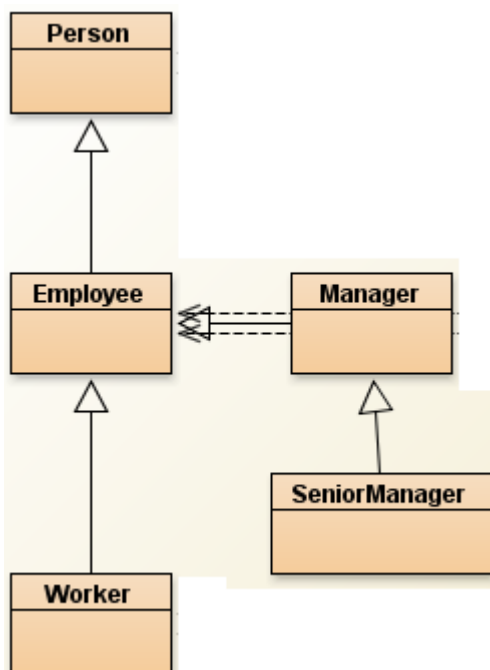
נגדיר מנהלת החברה SeniorManager באמצעות התכונות הבאות :  
 ת.ז., שם פרטי, שם משפחה, כתובת מגורים, מספר עובדת פנימי בחברה, שכר חודשי ומענק  
 כספי.

נגדיר חישוב משכורת : בנוסף למשכורת החודשית ניתן המענק הכספי.

כמו במחלקות הקודמות, גם כאן נשתמש במנגנון הירושה.

```
class SeniorManager : Manager
{
    private int bonus;

    public SeniorManager(int id, string last, string
        first, string address, int empId,
        int salary, int bonus)
        :base(id, last, first, address, empId,
            salary)
    {
        this.bonus = bonus;
    }
    public SeniorManager(SeniorManager s)
        : base(s.GetID(), s.GetLastName(),
            s.GetFirstName(), s.GetAddress(),
            s.GetEmpId(), s.Salary())
    {
        bonus = s.bonus;
    }
    public override string ToString()
    {
        return base.ToString() + " bonus " + bonus;
    }
    public override int salary()
    {
        return base.Salary() + bonus;
    }
}
```



עבור שורות הקוד הבאות:

```

SeniorManager s = new SeniorManager(333, "Levi",
    "Rachel", "A-H", 3, 30000, 2000);

Console.WriteLine(s);
Console.WriteLine();
Console.WriteLine("Salary -> " + s.Salary());
Console.WriteLine();
Console.WriteLine("Net salary -> " + s.NettoSalary());
  
```

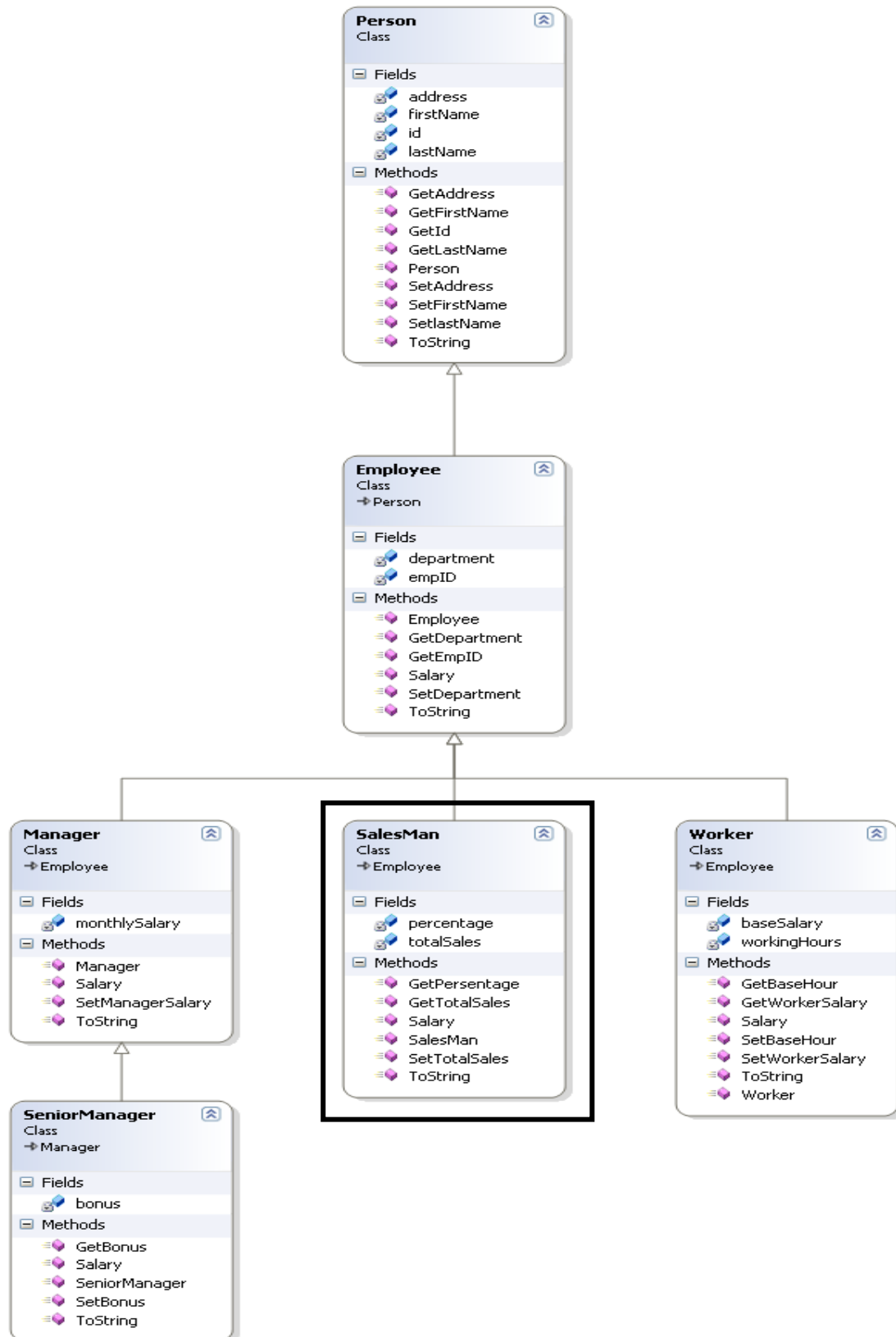
יתקבל הפלט:

```

ID 333 Last & First name Levi Rachel lives in A-H
Employee id 3 works at 3 department &
earns nettoSalary 10500 NIS Monthly salary: 30000 bonis 2000
Salary -> 30000
Net salary -> 10500
  
```

**תרגיל**

הגדירו מחלקה חדשה בשם "סוכן מכירות". לסוכן התכונות הבאות:  
 ת.ז., שם פרטי, שם משפחה, כתובת מגורים, שם המחלקה, מספר עובד/ת פנימי, משכורת  
 בסיסית, אחוז מהמכירות (הניתן כהכנסה) וסכום מכירות בערך ראשוני אפס.



## פולימורפיזם (רב-צורתיות)

פולימורפיזם הינה היכולת לתת מימוש שונה לאותה הפונקציה במחלקה היורשת כך שבזמן ריצה תופעל הפונקציה שמתאימה לטיפוס האובייקט.

נגדיר במחלקה הראשית מערך מטיפוס עובד בצורה הבאה:

```
static void Main(string[] args)
{
    Employee[] com = new Employee[3];
    com[0] = new Worker(111, "Zohar", "Doron",
        "Shpinoza", 222, 2, 100, 50);
    com[1] = new Manager(222, "Zur", "Iris", "RMG", 333,
        2, 5000);
    com[2] = new SeniorManager(333, "Cohen", "Oren",
        "BAT-YAM", 444,1, 50000,
        12000);
```

אומנם הגדרנו מערך מסוג `Employee` אך כל `Worker` או `Manager` או `SeniorManager` הוא בעצם טיפוס `Employee`, ולכן כל עצם מוכנס על פי הגדרתו.

מה מבצע הקטע הבא:

```
int total = 0, temp;
for (int i = 0; i < com.Length; i++)
{
    temp = com[i].Salary();
    total += temp;
    Console.WriteLine(temp);
}
Console.WriteLine(total);
```

כדי להבין טוב יותר את הרעיון הפולימורפיזם נגדיר את המחלקה הבאה:

```
class MyEmployees
{
    private int num; // מספר העובדים בפועל
    private Employee[] ep; // נתוני העובדים במערך
    private const int MAX=10; // קבוע להגדרת המערך
    public MyEmployees()
    {
        ep=new Employee [MAX];
        num=0;
    }
    public void addEmployee(Employee pr)
    {
        if (num < MAX)
        {
            if (pr is SeniorManager)
                ep[num] = new SeniorManager((SeniorManager)pr);
            else if (pr is Manager)
                ep[num] = new Manager((Manager)pr);
            else if (pr is Worker)
                ep[num] = new Worker((Worker)pr);
            else if (pr is Employee)
                ep[num] = new Employee((Employee)pr);
            else num--;
        }
        num++;
    }
    public override string ToString()
    {
        string s = "";
        for (int i=0; i<this.num; i++)
            s = s + "\n"+ep[i].ToString() + "\n";
        return s;
    }
}
```



```
public int SalaryExs()
{
    int sumExs = 0;
    for (int i=0; i<this.num; i++)
        sumExs+=ep[i].Salary();
    return sumExs;
}
```

נגדיר את הפעולות הבאות בתכנית הראשית:

```
MyEmployees mE = new MyEmployees();
```

```
SeniorManager s = new SeniorManager
```

```
(333, "Levi", "Rachel", "A-H", 3, 3, 30000, 2000);
```

```
Manager m = new Manager(222, "Zur", "Iris", "A-H", 2, 2,
                        7000);
```

```
Worker w = new Worker(111, "Zohar", "Doron", "Shpinoza",
                      1234, 1, 1500, 3);
```

```
mE.addEmployee(s);
```

```
mE.addEmployee(m);
```

```
mE.addEmployee(w);
```

```
Console.WriteLine(mE);
```

לפניכם פעולה פנימית אשר עברה הידור ואינה מבצעת את המבוקש.  
הסבירו את הבעיה?

```
public void addEmployee(Employee pr)
{
    if (num < MAX)
    {
        if (pr is Employee)
            ep[num] = new Employee((Employee)pr);
        else if (pr is Worker)
            ep[num] = new Worker((Worker)pr);
        else if (pr is Manager)
            ep[num] = new Manager((Manager)pr);
        else if (pr is SeniorManager)
            ep[num] = new SeniorManager((SeniorManager)pr);
        else num--;
    }
    num++;
}
```

בכל המחלקות, מחלקת האב והמחלקות היורשות יש פעולה בשם זהה המחשבת שכר.  
פעולת החישוב הראשונית נמצאת במחלקת **Employee** בה מוחזר הערך אפס.  
נגדירה כ `virtual` כדי שנוכל "לרמוס" אותה בהמשך.

פעולת חישוב שכר בתוך `Employee : Person`

```
public virtual double Salary()
{
    return 0;
}
```

פעולת חישוב שכר עובד (שכר בסיס כפול שעות עבודה) בתוך `Worker : Employee`  
מכיוון שהמחלקה יורשת ממחלקת `Employee` אנו מגדירים אותה ב-`override`.

```
public override double Salary()
{
    Return this.baseSalary*this.workingHours;
}
```

פעולת חישוב שכר איש מכירות (מקבל אחוז מן סכום כל המכירות שהוא מכר)  
בתוך `SalesMan : Employee`  
מכיוון שהמחלקה יורשת ממחלקת `Employee` אנו מגדירים אותה ב-`override`.

```
public override double Salary()
{
    return this.totalSales* (percentage/100);
}
```

פעולת חישוב שכר מנהל (שכר קבוע) בתוך `Manager : Employee`  
מכיוון שהמחלקה יורשת ממחלקת `Employee` אנו מגדירים אותה ב-`override`.

```
public override double Salary()
{
    return this.monthlySalary;
}
```

פעולת חישוב שכר מנהל בכיר (שכר קבוע של מנהל ועוד בונוס חודשי למנהל בכיר)  
בתוך `SeniorManager : Manager`  
מכיוון שהמחלקה יורשת ממחלקת `Manager` אנו מגדירים אותה ב-`override`.

```
public override double Salary()
{
    return base.Salary()+(this.bonus/12);
}
```