

**Agenda**

- Reference Types
- Overview of Arrays
- One Dimension Arrays
- Two Dimension Arrays
- Jagged Arrays
- Array Class Methods
- Arrays and Methods

**Reference Types**

משואה כתובות ולא שוכנת בזיכרון (==, !=, <, >)

- שמיישים פונקציות
- מוגדרת כפונקציית שפה
- לדוגמה שפה שטוחה לא מודרנית לא מודרנית, כמו קריין.
- Garbage Collection - ה-
- אובייקטים שנערכו
- Reference Type מגדירם
- מערכיהם

**Arrays**

Microsoft .NET

```

graph LR
    RT[Reference Type] --> I[Instance]
    I --> MH[Managed Heap]
    SS[Stack Segment] --- MH
  
```

**Reference Types**

משתנה המכיל "תו" ליותר

- המדייע עבורה מוקאה דואת זיכרון ומזאצא באאות זיכרון
- מושתת שפה שטוחה לא מודרנית לא מודרנית, כמו קריין.
- Garbage Collection - ה-
- אובייקטים שנערכו
- Reference Type מגדירם
- מערכיהם

הדרת "חו"ן: גיבוריים יקרים, על אליהם נמשכתה:

```

MyClass c1;
c1 = new MyClass();
: : :
MyClass c2 = c1;
c1 = null;
  
```

4

CORNER

# Overview of Arrays

## Array Notation

האצת מערך מתבצעת בשני שלבים:

- 1 הדריה יוט למחר.
- 2 היזאה דגמתה של המערך.

לדוגמא:

`type[] ArrayName = new type[5];`

מונוט הינה:

Stack Segment	Managed Heap
	ArrayName[4]
ArrayName[3]	
ArrayName[2]	
ArrayName[1]	
ArrayName[0]	

8

## Overview of Arrays

### Array Notation

הארת מערך מורכבה משלושה חלקים:

סימן האלמנטים של המערך, הגדלת תחומיים, שם המערך.

`type[] arr_name`

```

graph TD
    type["הטיפוס"]
    arr["הarry"]
    name["השם"]
    type --- arr
    arr --- name
    
```

6

CORNER

## Overview of Arrays

### Array Notation

המג' ב-**C#** מאריכים דומים ל-**C**, ומכילים דוחה בחרורה וויליאן של מערך (מערך משני).

- type[] arr\_name
- type[][] matrix\_name
- type[,,,] arr
- type[][] jagged\_arr

7

CORNER

# Overview of Arrays

## What Is an Array?

- מעריך אוקט שמל שמתווך ממאorts הטעפיים, המאוגנים במבנה האצפה בזיכרון.
- גישה לאביר המעריך מהבעתת מהברהות.
- גישה לאביר המעריך מהבעתת אמצעית אידאולוגית.
- .Address + Offset
- .Reference Type
- .Manageed Heap
- .מעריך מהקה באודר זיכרון

123	-23	456	5689	-654	9	273
[0]	[1]	[2]	[3]	[4]	[5]	[6]

## Overview of Arrays

### Array Notation

גדירה של מערך בתוכנית מורשת מורה מילולית. System.Array יירושת את התפקיד מהלילה זהה ומייצג מערך.

לדוגמה:

```
int[] ArrayName = new int[5];           // System.Int32]
ArrayBase = ArrayName.GetType();          // System.Array
```

## Overview of Arrays

### Array Notation

שורה מעוררת שיכrho ניכראם. מערך מושבב זיכrho ניכראם. בקההה בה אונ שור גבר מושבב לשחרור. השחרור און שחרור פא שיל מוקט לאו"ה. השחרור הפויי ייבצע כאשר ה- GC (Garbage Collection).

לדוגמה:

```
type[] ArrayName = new type[5];
...
ArrayName = null;
```

## One Dimension Arrays

### Declaration

int[] arr;

### Allocation

arr = new int[10];

הערכם מאופים בראקובה.

### Accessing

arr[3] = 13; Console.WriteLine(arr[3]);

לדוגמה:

### Freeing

arr = null;

## Agenda

- Reference Types
- Overview of Arrays
- One Dimension Arrays
- Two Dimension Arrays
- Jagged Arrays
- Array Class Methods
- Arrays and Methods

## Overview of Arrays

### Array Notation

שורה מעוררת שיכrho ניכראם. מערך מושבב זיכrho ניכראם. בקההה בה אונ שור גבר מושבב לשחרור. השחרור און שחרור פא שיל מוקט לאו"ה. השחרור הפויי ייבצע כאשר ה- GC (Garbage Collection).

לדוגמה:

```
type[] ArrayName = new type[5];
...
ArrayName = null;
```

One Dimension Arrays

Initializing

```
int[ ] arr = new int[5];  
int[ ] arr = new int[5] {0, 1, 2,  
int[ ] arr = new int[5] {0, 1, 2,  
OR  
int[ ] arr = {0, 1, 2, 3, 4};  
int[ ] arr = source;  
int[ ] arr = copy = source;
```

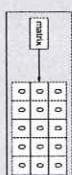
Copying

14

Two Dimension Arrays

### Declaration

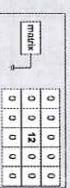
```
int[,] matrix;
```



הערכם מאוגוסט בהרצאה.

### Accessing

```
matrix[1,2] =  
Console.WriteLine(matrix[1,2]);
```



Freeing

One Dimension Arrays

Iterating

```
for(int i=0;i<arr.Length;i++)
    Console.WriteLine(arr[i]);
OR
foreach(int num in arr)
    Console.WriteLine(num);
- ליזיאת foreach נהוג לזרג את מירזע מערכם.

```

One Dimension Arrays

### Initializing of Objects Array

```

Person] arr = new Person[4]; // NULL כל האברים מכילים טרואן
בשעתה כוחה מוגבר שולץ על עמו הדרה אותלה מושך
כל האברים מכילים טרואן

```

```
Arr[2]=new Person();
Arr[2].Name="Sam";
Arr[2].Age=20;
Arr[2].Fruit="Banana";
```

```
Person[] arr = new Person[4]{new person(), "Sara", "John", "Mike", "Lily"};
```

15

## Two Dimension Arrays

### Initializing

```
int[,] arr = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
OR
int[,] arr = new int[3,4]{{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

```
int[,] source = new int[10,15];
int[,] copy = source;
```

### Copying

18

## Two Dimension Arrays

### Iterating

```
for(int x=0;x<matrix.GetLength(0);x++)
{
    for(int y=0;y<matrix.GetLength(1);y++)
        Console.WriteLine(matrix[x,y]);
}
OR
foreach(int num in matrix[x,y])
    Console.WriteLine(num);
```

17

## Jagged Arrays

### Freeing

```
arr = null;
```

### Iterating

```
for(int y=0;y < jagged.Length;y++)
{
    for(int x=0;x<jagged[y].Length;x++)
        Console.WriteLine(jagged[y][x]);
}
```

20

## Jagged Arrays

### Declaration

```
int[] arr;
```

### Allocation

```
arr = new int[3];
for(int i=0;i<arr.Length;i++)
    arr[i] = new int[5];
    arr[3][2] = 12;
    Console.WriteLine(arr[3][2]);
    Console.WriteLine(arr[3][2]);
```

19

## Array Class Methods

הארה שיר מאיל שואר בתוכנית מורה הדעת מהלכה הנתקאת

- `[array]`.
- מחלקה זו יושתת את המחלקה `System.Array`.
- פועלות עליה תמצית מוגבשת.
- פועלות אלו סמיון מין רני.
- המלהירה `System.Array` אוסף מוחדרות ואפ"י נים.

22

## Jagged Arrays

### Initializing

```
Random rd=new Random(10,100);
for(int y=0;y < jagged.Length;y++)
    for(int x=0;x<jagged[y].Length;x++)
        jagged[y][x] = rd.Next(100);
```

### Copying

```
int[,] source= new int[10][];
int[,] copy = source;
```

21

## Array Class Methods

### Array Methods

- `Array.BinarySearch(...)` חיפוש בarray על מספר אחד מודם מהו מס' שלילי, ומאז המספר אחריו בחדרה לא מודם מודם מהו מס' שלילי.
- `Array.Clear(...)` איפוא אובי המערך. ככל אם או בוטה מוגדר.
- `Array.GetLength(...)` קבלת מספר האברים במערך מסוים.
- `Array.IndexOf(...)` חיפוש אובי במס' של אלמנטים במערך.
- `Array.Reverse(...)` הפיכת סדר האלמנטים במערך

24

## Array Class Methods

### Array Properties

- `Array.Length` מציין את מספר האלמנטים המקיימים במערך.
- `Array.Rank` מציין את סוג המודם של המערך.
- `ArrayFixedSize` ערך בלאיי המודר האם המערך הוא בוחר קבוע, מערכם הם מודר קבוע.
- `Array.IsReadOnly` על מנת להגדיר אובי באוצר משנה גדרה `Collection`.
- `Array.ReadOnly` ערך בלאיי ראמ אולמנטים של מגנור כלים לרשנות, מערכם אינם ערך לעדכון לתקדים `Collection`.

23

## Arrays and Methods

### Returning Arrays from Methods

```
static int[] CreateArray(int size)
{
    int[] created = new int[size];
    return created;
}
```

26

## Array Class Methods

### Array Methods

- Array. SetValue (...)
- Array. Sort (...)

משה שיר אונדרו גוטמן.  
תין מעיר. מופיע את אלטמארט  
בתוך לאליה המולקה מכילה System.Array.

25

## Arrays and Methods

### Command-Line Argument

```
class Sum
{
    static void Main(string[] args)
    {
        int sum = 0;
        for (int i = 0; i < args.Length; i++)
        {
            sum += int.Parse(args[i]);
        }
        System.Console.WriteLine(sum);
    }
}
```

28

## Arrays and Methods

### Passing Arrays as Parameters

```
static int CalcAvg(int[] arr)
```

```
{
    int sum = 0;
    foreach(int num in arr)
        sum += num;
    return sum/arr.Length;
}
```

27

## Command Line Arguments

- לאחר שם הקובץ [סוליל כל הנתיב שו] ניתן לרשום פרמטרים עם רווחים ביןיהם.
- כל העריכים מתחביבים כפרמטרים לפונקציית ה- MAIN, כמפורט להלן.
- כדי לקבל מספרים יש להרמז אולם [בבטחה]

30

## Command Line Arguments

- ניתן לדבג תכנית שמקבלת פרמטרים ומורצת משורת הפקודה גם "ע'", attach to process
- שם מקובלים רשיימה של כל התוכניות שרצות כרגע במכשיר
- ואז ניתן לשם נקודות עצירה ולבדוק את התוכנית בתאי שקבץ הרטעה גנבה מוגresa זהה בדיק
- ליקובץ של הקוד.

31

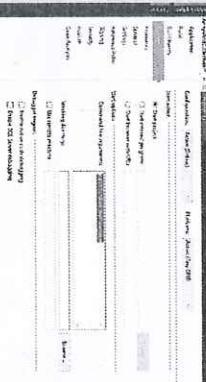
## Command Line Arguments

- ניתן להריץ התוכנית שתכתבו גם משורות הפקודה של command מרים את התכנית ע' כתיבת הנתיב המלא ושם קובץ ה- EXE.
- לא ח"בים לשום את הסיווייה EXE מצא בחר תקית debug של הפוך.

29

## Command Line Arguments

כדי להריץ את התוכנית עם פרמטרים מהווים VS properties ניתן לרשום רשותם בתלוין ה- settings של הפרוייקט בכרטיסיה debug:



32

- השוואה
- מערכים הם בغالל קבוע, אופקיים בהםיל משותה.
  - האלמנטים במערך אינם יסילים להיוות Read Only , באופנים כל.
  - מערכים הם מחרים יותר אללים פחת גמישים, אופייםם גמישים יותר אללים איטיים יותר.

## Arrays vs. Collections

# Collections

Microsoft  
.net

2

**Important Interfaces**

מחלקות רה-Collection הרשות ממשקי חשבונות:

### IEnumerable

אפשרת תחכיה במוגרי סדרית עלי אובי ראיוק, ליהיאת foreeach שעה בו שימוש.

### IDictionary

ממשק דטטי לכל המתקינות המייצאות אוסףם של Key and Value כטמל של Add, Clear, Contains, IndexOf, Insert, : מדריך מהדרות Remove, RemoveAt Indexer וди.

### IList

מייצג אוסף של אובייקטים שיש לנו לארון index(index) לשאת אליהם שיירית במאצועת .  
מדריך מהדרות Add, Clear, Contains, IndexOf, Insert, : מדריך מהדרות Remove, RemoveAt Indexer ודי.

**Collection**

מודר את מדריך אוסף, יזכרנו הסינכוא יכולת עבור פשת וסדרת על אברת המארא.

ממשק בודק למספר מהמשכנים המהמקבלים במאוסף.

3

4

CORNER

# ArrayList class

ניתן לרשף את הממלהה `ArrayList` עם מנת ליעזר מחלקה `Collection` מתמזהה.

לדוגמא:

```

public class EmployeesCollection extends ArrayList<Employee>
{
    public EmployeesCollection() { ... }

    public EmployeesCollection(int size) {base(size) { ... }

    public void Print()
    {
        for(int i=0; i<Count; i++)
        {
            ((Person)base[i]).Print();
        }
    }
}

```

8

CORNER

## ArrayList class

- מחלקה ה- Collection השיכחה הפעופריה מעיר ביזה.
- מיאוגת אօיך הרוגו, של אב"קיטם, מדירה פנימית של System.Object.
- יחוים מיטסיו מהולגה מדילה את עצמה אוטומטית בעת הצורך. כבירות מודול הראשנו הואה 16 מרכיבים אליהם נזקם 16 מהוות מספיקים, עם כל אחד נזקם לקבועה.
- מדרגת במרחבה השמות ממנהן על מנת ליצא מחלקות אויה מתחנה.
- מדרגת במרחבה [להרשת ממנהן] מונת ל"יצא מחלקות אויה מתחנה.

6

CORNER

# ArrayList class

ממשק:

```
class App
{
    static void Main(string[] args)
    {
        ArrayList arr = new ArrayList();
        arr.Add(39);
        arr.Add("39");
        arr.Add(new Person());
        arr.Add(Datetime.Now);
        arr.Add(new Worker("bitca", "shpitca", 12345, 12345, "QA", 17, 34f, 167, 5f));
        arr.Add(new Salesman("Zochaver", "Joshua", 4545, 4545, "Business", 14500));
        arr.Add(new Salesman("Zochaver", "Joshua", 4545, 4545, "Management", 25000, 10000));
        arr.Sort(); // המונט אוסף ומאפשר IComparable<Person>
    }
}
```

CORNER

# Agenda

- Arrays vs. Collections.
- Important Interfaces.
- ArrayList class.
- HashTable class.
- SortedList class.
- Queue Class.
- Stack Class.
- Custom Collection.

5

## Hashtable class

לדוגמא:

```

class Person
{
    private readonly int m_ID;
    private string m_Name;
    public Person(int id, string name) { ... }
    public int ID
    {
        ...
    }
    public string Name
    {
        ...
    }
}

```

10

## Hashtable class

- מחלקה שיכילה מסדרת, מייצגת אוסף התרוגן של אובייקטים.
- האובייקטים מארגנים באמצעות מפתח מפתח Key (Key) וערך (Value).
- הגישה לאובייקט רוא במאצעתה מפתח אש "ח"ב לירוח ייחוד.
- כל אובייקט המוחזק בתמולחת אוסף מבוקם מאובייקט DictionaryEntry Object Value -> Object Key
- הו המפתח והערך יכילים כל ספוא.
- המפתח איט נישל להיות טה.
- גס מחרה זו ניתן לזרור על מנת להזאתה לצרכים של התכנית.

9

## Hashtable class

המשר:

```

static void Main(string[] args)
{
    ...
    p = (Person)hash[11];
    Console.WriteLine(p.Name);
    p = (Person)hash[12];
    Console.WriteLine(p.Name);
    p = (Person)hash[13];
    Console.WriteLine(p.Name);

    foreach (int i in hash.Keys)
        Console.WriteLine(i);
    foreach (Object obj in hash.Values)
        Console.WriteLine(((Person)obj).Name);
}

```

12

## Hashtable class

המשר:

```

static void Main(string[] args)
{
    Hashtable hash = new Hashtable();
    Person p = new Person(11, "Shoshana");
    hash.Add(p.ID, p);
    p = new Person(12, "Verachmle");
    hash.Add(p.ID, p);
    p = new Person(13, "Zalman");
    hash.Add(p.ID, p);
    ...
}

```

11



## SortedList class

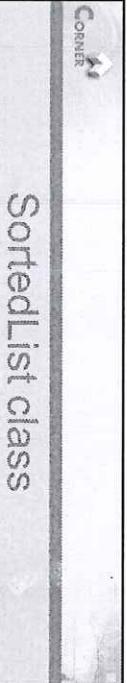
- מחריקת אוסף התרוגי של אובייקטים.
- גם במחלקה אוסף מארגנים באמצעות מפתח (Value) וערך (Key).
- במחלקה אוסף זה האובייקטים ממוקמים על פי ערך המפתח, המפתח ציריך למשת את סדרה Comparable. (Values must implement Comparable).
- היכולת לאובייקט הינה לאפשר לארון המפתח אשר חישוב,
- ואחסנת רמתם של האובייקטים באוסף.
- אויך הוא יכול לארש Dictionary שאל הרמיון, אלום גמיש.
- מהו שם שallow מאפשר גישה כמו על פ' האינדקס.

14

## Agenda

- Arrays vs. Collections.
- Important Interfaces.
- ArrayList class.
- Hashtable class.
- SortedList class.
- Queue Class.
- Stack Class.
- Custom Collection.

13



## SortedList class

### דגימה:

```
static void Main(string[] args)
{
    SortedList list = new SortedList();
    Person p = new Person(113, "Zalman");
    list.Add(p.ID, p);
    p = new Person(111, "Shoshana");
    list.Add(p.ID, p);
    p = new Person(112, "Yerachmier");
    list.Add(p.ID, p);
    ...
}
```

15



## SortedList class

### כל אובייקט המוצג במחלקה אוסף ומאוחזן באנדרט מוחבנה

- כל אובייקט המוצג במחלקה אוסף ומאוחזן באנדרט מוחבנה תמייל DictionaryEntry
- Object Value - ו- Object Key
- הערך והו המפתח והו הערך יכולים להיות כל טופו.
- המפתח אומנם יכול להיות שמה, הערך יכול להיות שמה.
- גם מחלקה זו ניתן לזרירה על מנת להציגם של התכנית.

16

- `list[112];`
- מחדיר את הערך שבו מפתח מסויים
- `list.GetKey(3)`  
(3) מוחזר את המפתח לפי אינדקס (3)
- `list.GetByIndex(3)`  
מחדר את הערך לפי אינדקס מסויים

18

Sorted List class

```
static void Main(string[] args)
{
    ...
    p = (Person) list[112];
    Console.WriteLine(p.ToString());
}

for (int i = 0; i < list.Count; i++)
{
    //Prints all the keys and values of the sorted list
    Console.WriteLine("KEY:{0} VALUE:{1}", list.GetKey(i),
        list.GetByIndex(i).ToString());
}
```

לדוגמה:

שלייפה  
אב"קיטס  
מראווי.

17

## Queue Class

מייצג אוסף אובייקטים המשמר באמצעות דוגמה:

```
.first-in, first-out collection
```

```
class App
{
    static void Main(string[] args)
    {
        ...
        int num;
        num = (int)q.Dequeue();
        Console.WriteLine(num);
        num = (int)q.Dequeue();
        num = (int)q.Enqueue();
        Console.WriteLine(num);
        num = (int)q.Dequeue();
        Console.WriteLine(num);
        ...
    }
}
```

דוגמה:

## Queue Class

מייצג אוסף אובייקטים המשמר באמצעות דוגמה:

```
.first-in, first-out collection
```

```
class App
{
    static void Main(string[] args)
    {
        ...
        Queue q = new Queue();
        q.Enqueue(12);
        q.Enqueue(13);
        q.Enqueue(14);
        q.Enqueue(15);
        q.Enqueue(16);
        q.Enqueue(17);
        q.Enqueue(17);
        ...
    }
}
```

דוגמה:

```
class App
{
    static void Main(string[] args)
    {
        ...
        int num;
        num = (int)q.Dequeue();
        Console.WriteLine(num);
        num = (int)q.Dequeue();
        num = (int)q.Enqueue();
        Console.WriteLine(num);
        num = (int)q.Dequeue();
        Console.WriteLine(num);
        ...
    }
}
```

20



## Stack Class

מיצ'ג אוק אובי'קטים רושמר במבנה דוגמה:

```
class App
{
    static void Main(string[] args)
    {
        ...
        int num;
        num = (int)s.Pop();
        Console.WriteLine(num);
        num = (int)s.Pop();
        Console.WriteLine(num);
        num = (int)s.Pop();
        Console.WriteLine(num);
        ...
    }
}
```



## Custom Collection

מיצ'ג אוק אובי'קטים רושמר במבנה דוגמה:

```
class App
{
    static void Main(string[] args)
    {
        Stack s = new Stack();
        s.Push(12);
        s.Push(13);
        s.Push(14);
        s.Push(15);
        s.Push(16);
        s.Push(17);
        ...
    }
}
```



## Custom Collection

ניתן לרשף את מחלקות ה- Collection על מנת לסייע מחלקות

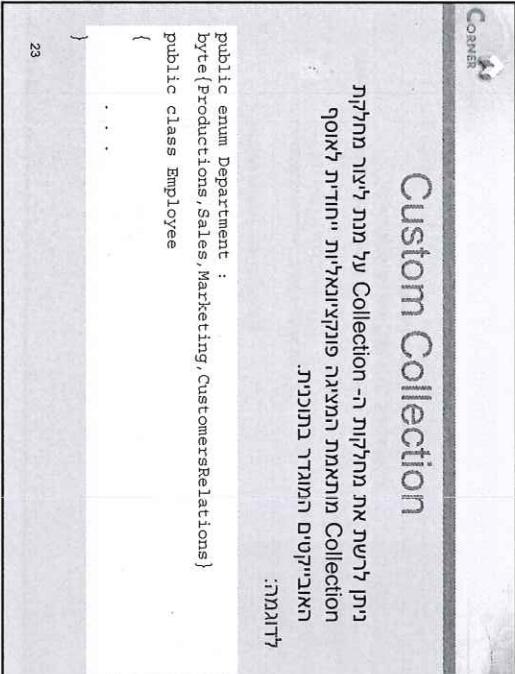
האובי'קטים המגדר בתכנית.

לדוגמה:

```
public class EmployeeArrayList : CustomCollection
{
    public class EmployeeArrayList() : ArrayList
    {
        public EmployeeArrayList() { }

        public decimal TotalSalary()
        {
            decimal sum = 0;
            foreach (Employee e in this)
            {
                sum += e.Salary;
            }
            return sum;
        }
    }
}

public SortedList GetEmptyDept(Department dept)
{
    SortedList list = new SortedList();
    foreach (Employee e in this)
    {
        if (e.Department == dept)
        {
            list.Add(e.ID, e);
        }
    }
    return list;
}
```



## Custom Collection

ניתן לרשף את מחלקות ה- Collection על מנת לסייע מחלקות

public enum Department : byte { Productions, Sales, Marketing, CustomersRelations }

public class Employee

```
{
    ...
    public string Name;
    public string ID;
    public decimal Salary;
    public Department Department;
}
```

23

## Custom Collection

CORNER

```
class App
{
    static void Main(string[] args)
    {
        EmployeeArr arr = new EmployeesArr();
        Employee e = new Employee();
        Employee e111, Sosolana, "Yerachmial", (decimal)1234.12, Department.Sales);
        arr.Add(e);
        e = new Employee(112, "Yerachmial", (decimal)2345.34, Department.Productions);
        arr.Add(e);
        Employee e113, "Zia Inan", (decimal)2314.34, Department.Sales);
        arr.Add(e);
        arr.Add(e);
        Console.WriteLine("Total sal : " + arr.TotalSal());
        SortDeptList = arr.GetEmpByDept(Department.Sales);
        Console.WriteLine(list.Count);
        foreach(DictionaryEntry emp in list)
        {
            e = (Employee) emp.Value;
            e.Print();
        }
    }
}
```

תמל"ש



# Polymorphism Basics and virtual Methods

## Virtual Methods

מודולו וירטואלית (Virtual Method) הינה מетодה אשר מודדת ומימושה במל Hitch הבסיס. אלם ניתן לסקף לה מתחום מסוים בלבד אחת מהמהדרות הנדרת.

ונואג, שמתחלפת המדרת נאלה לה להרשות (Override) את המתוח.

- הקיים במחולקת הפסudo.
- כאשר פועל מתרחדר יפעיל מתחוש המתאים לסטו האובייקט.
- תמקצעת אלה או יותר, במילאים הממשות על פה האובייקט המשותה.
- דוגמאות לול עלי רהיינו המהויק ככתבת הקאזה.
- המדרת מתרחדר מתקבצנית באמצעות המילה השמורה (keyword) override.
- המדרת מתרחדר מושך ונוסח מחריבת שמות המילה השמורה override.
- אונסן, מדרת מושך ונוסח מחריבת שמות המילה השמורה override.
- מודולו וירטואלית אינה יכולה להיות static וכומבו private.

<h1>Polymorphism Basics and virtual Methods</h1> <p>הבסיס ל-Polymorphism - המשפט הआ מושפע מהטיפוס הביא:</p> <p><b>"יוזו מטיפוס הבטייס יכול להתייחס לאובייקט מהתלה הנגרמת."</b></p>
<pre>class Base {     public void Func() {...} }  class Derived:Base {     public new void Func() {...} }</pre> <p>איך מודדה תופעת?</p>

CORNER

# Polymorphism Basics and virtual Methods

בעיה !

"יהו מטיפוס הבסיס יכול לחייב כתובות של אובייקט ממהלכה נגורת,อลס אותן י"חו"יכר רק את הattributs (and Members **Methods** מוגדרים (Methods אשר במחילה הבסיסי).

הפתרונות :

מחדן ורטואליות.

3

CORNER

# Implementing Polymorphism

Microsoft  
.net

<h2 style="text-align: center;">Polymorphism Basics and virtual Methods</h2> <p><b>Virtual Methods</b></p> <p>כלים ביחס למונט וירטואלי:</p> <ul style="list-style-type: none"> <li>התקינה של המונט הריאליות הבסיסי לחיות זהה בכל (שם המונט, פונקציית השם, הרשאות, ערך מוחלט, הרשות). מושם מושפע שליה במלואה.</li> <li>ניתן להציג ולשלב מונטת הגדלת גודלה וויאטיות. מונט בסיס מושפע מונטת הגדלת גודלה וויאטיות.</li> <li>לא ניתן לגדיר מונטת שאל static virtual. מושם שאל מונטת private וויאטיות.</li> <li>לא ניתן לגדיר מונטת private וויאטיות.</li> <li>לא ניתן לגדיר בואה פונקשיון virtual → override.</li> </ul> <p>6</p>	<h2 style="text-align: center;">Polymorphism Basics and virtual Methods</h2> <p><b>Virtual Methods</b></p> <p>לוגו:</p> <pre> class Base {     public virtual void Func() {...} }  class Derived:Base {     public override void Func() {...} }</pre> <p>5</p> <p>6 איז מונטת תפעל הפעם?</p>
<h2 style="text-align: center;">Polymorphism Basics and virtual Methods</h2> <p><b>New Virtual</b></p> <p>לוגו:</p> <pre> class Base {     public virtual void Print()     {         ...     } }  class Derived:Base {     public override void Print()     {         ...     } }</pre> <p>7</p>	<h2 style="text-align: center;">Polymorphism Basics and virtual Methods</h2> <p><b>New Virtual</b></p> <p>ניתן להגדיר מונטת וויאטיות כ-new, המשמע רוא שאותו יכיר יוניט לא ימיהר את המונט שמשלים רוחן. הורשת אשר הגדרו מונטת הגדוד, יחבא את כל המונטת בעלות הגדוד שפהגדרו קודם ל-<b>new</b> המונטת. מונטת הגדוד מודבר במתודה הושתת ל-<b>new</b> שמיירת לומטוד הינו רוחן וויאטיות המונט. קודם ל-<b>new</b> מונטת בועז מהוושתת.</p> <p><b>Syntax:</b></p> <pre> public new virtual void Func(...)</pre> <p>7</p>

CORNER

# Conversions Between Base and Derived

לדוגמה:

```

graph TD
    BS[BaseShape] --> DS[DerivedShape]
    subgraph BS
        R1[Rectangle]
        C1[Circle]
    end
    subgraph DS
        R2[Rectangle]
        C2[Circle]
    end
  
```

`BaseShape b = new Circle(...);`

`Rectangle r = (Rectangle) b;`

שורה ת'לון בעשיטות והן יגרמו להשתיקות התוכנית, מושם שויין  
ב מיל כתובת של אלט מומלחקה ואלט . Rectangle ו Circle

12

**Polymorphism Basics and virtual Methods**

### Virtual Methods

- המethodות חורטואליות מאפשרות להעתיקו אל המethodיות כל בירכיה בזרה ובכל זאת לא לוויה על ה"יחדויות של כל ממלכה וממלכה נדרגה ולהבאה את הרתאמאות ה"יחודית של לה ליר".
- בarraה זו מל מoad לעבד עם היררכיה וול לתרחץ אותה.
- מethodות ורטואליות ממאות את קזב בעשו התוכנית משם שמי'שא מתבצע בתום ריצה (Late Binding).

**Object-Oriented Programming Basics and Virtual Functions**

```
class Derived2:Derived1
{
    public new virtual void Print()
    {
        . . .
    }
}

class App
{
    static void Main(string[] args)
    {
        Base b1 = new Derived1();
        b1.Print();

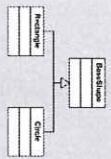
        Base b2 = new Derived2();
        b2.Print();
    }
}
```

**New Virtual**  
המשך:

## Conversions Between Base and Derived

- על מנת לבקש הרמה בפונקציה מסוימת יש לזרום ממנה:

```
class App
{
    static void Main()
    {
        BaseShape b = new Rectangle();
        if (b is Rectangle)
        {
            r = (Rectangle) b;
            r.PrintRectangle();
        }
        else
        {
            c = (Circle) b;
            c.PrintCircle();
        }
    }
}
```



## The Object class

המחלקה Object הנה המחלקה הבסיסית ביותר ב-.NET Framework

כל התייחסים המגדירים ב-C# (וכן C++) הם תחתיו של המחלקה Object. מופעשים מוחלט (Implicit) או מוחלט (Explicit).  
המחלקה Object מאפשרת שגבורת מוגבלת בין-

ורשות זו מבוטה שכל מחלקה צריכה לפחות איזוף מינימלי של פעולות.

## Conversions Between Base and Derived

- וצרר זה של טיסו בדמן ריצה אים ב-BI, תקון ניכן וגרירת מתחודת.
- ויטאלית אמרה תאריך התה כתמעת המשרת אל מס' אחד מהדים לירק שיאוות תכניתה האוצר לבקש המרת אל מס' אחד פולמי בתוכנית.
- לא ניתן להעתיק מתקולות בלאב התמזהה בו מוגלים צרכם דרישים ומשומות מוחלטות ורקי'ין.
- ליסטם גרובת התקומת מתקולות לא רק בשעתן את הטעמאניות הרהשות מוקן ומירות מוחלטות.
- ולעתם עצור מגר בהדריך האגדאחים שתוכננת עברת.
- על מנת להפעיל את אותן מוחמות גראיל שלהסמקת ולבצע המרות אלו.

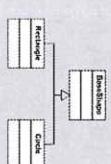
13

## Conversions Between Base and Derived

- או אמפרוס AS.

לדוגמא:

```
class App
{
    static void Main()
    {
        . . .
        b = new Circle();
        r = b as Rectangle;
        if(r == null)
            r.PrintRectangle();
        c = b as Circle;
        if(c != null)
            c.PrintCircle();
    }
}
```



16



**The Object class**

מתקודמות:

```
public Type GetType()
{
    //Reflections
}

public virtual string ToString()
{
    //Reflections
}
```

שיטות על מנת להציג את הסיסיות בפן ריעזה (Reflections). שיטות או כפ' שאר אמצעים מומשכים ב-CLR, דוגמת `Object`, `ToString()` ו-`GetType()`. שיטות או כפ' שאר אמצעים מומשכים ב-CLR, דוגמת `Object`, `ToString()` ו-`GetType()`.

**protected void Finalize()**

ונון לזרה באמצעות מזהה או `state` ב-CLR. בתוכלו פורט האובייקט מתחם גירויים. מתחם גירויים עליון מתחם שאבבים. מתחם גירויים עליון מתחם שאבבים.

18

**The Object class**

מתקודמות:

```
public virtual bool Equals(object obj)
{
    //Reflections
}

public virtual int GetHashCode()
{
    //Reflections
}
```

שיטות או כפ' שאר אמצעים מומשכים ב-CLR, דוגמת `Object`, `ToString()` ו-`GetType()`. שיטות או כפ' שאר אמצעים מומשכים ב-CLR, דוגמת `Object`, `ToString()` ו-`GetType()`.

**public void PrintPerson()**

שיטות או כפ' שאר אמצעים מומשכים ב-CLR, דוגמת `Object`, `ToString()` ו-`GetType()`.

17

**The Object class**

המשר:

```
public override bool Equals(object obj)
{
    if(obj is Person)
    {
        Person p;
        p = (Person) obj;
        if(this.m_FirstName == p.m_FirstName &&
           this.m_LastName == p.m_LastName &&
           this.m_Age == p.m_Age &&
           this.m_ID == p.m_ID)
            return true;
        return false;
    }
    public override int GetHashCode()
    {
        return this.m_ID;
    }
    public override string ToString()
    {
        return m_LastName + "," + m_FirstName + "," + m_Age;
    }
}
```

19

## Abstract Classes and Abstract Methods

**Abstract Classes**

- מחלקה אבtract (א'ב'קראקט) הנה מחלקה אשר לא יכולה לבצע פעולה מסוימת (לע"מ מטרת המחלקה).
- מחלקה מופשטת (abstract) מוגדרת באמצעות עמלת מחלקה גילה ומחלקה נוספת.
- מחלקה מופשטת מגדירים בערות מיליה המתוודה במאפייניה.

**Syntax:**

```
abstract class BaseClass
{
    ...
}
```

## Abstract Classes and Abstract Methods

**Abstract Classes**

המשר:

```
class App
{
    static void Main(string[] args)
    {
        Person p1 = new Person("Grisha", "Abrosimov", 12, 2245);
        Person p2 = new Person("Muhammad", "Yanklevitch", 34, 678);
        Person p3 = new Person("Muhammad", "Yanklevitch", 34, 678);

        Console.WriteLine("Objects are equals");
        if(p1.Equals(p2))
            Console.WriteLine("Objects are the equals");
        else
            Console.WriteLine("Objects are not equals");
    }
}
```

## The Object class

**דגם:**

```
class App
{
    static void Main(string[] args)
    {
        Base b = new Derived();
        Base d = new Derived();

        public override void Print()
        {
            Console.WriteLine("Base class");
        }
    }
}
```

## Abstract Classes and Abstract Methods

**Abstract Classes**

המשר:

```
abstract class Base
{
    public virtual void Print()
    {
        Console.WriteLine("Base class");
    }
}

class Derived:Base
{
    public override void Print()
    {
        Console.WriteLine("Derived class");
    }
}
```

## The Object class

**דגם:**

```
class App
{
    static void Main(string[] args)
    {
        Person p1 = new Person("Grisha", "Abrosimov", 12, 2245);
        Person p2 = new Person("Muhammad", "Yanklevitch", 34, 678);
        Person p3 = new Person("Muhammad", "Yanklevitch", 34, 678);

        Console.WriteLine("Objects are equals");
        if(p1.Equals(p2))
            Console.WriteLine("Objects are the equals");
        else
            Console.WriteLine("Objects are not equals");
    }
}
```

## Abstract Classes and Abstract Methods

### Abstract Methods

```
Syntax:  
abstract class Sample  
{  
    public abstract void FuncName();  
    public abstract int Num  
    {  
        get;  
        set;  
    }  
}
```

## Abstract Classes and Abstract Methods

### Abstract Methods

```
class Derived:Base  
{  
    private string m_Str;  
    public Derived()  
    {  
        m_Str = "Empty String";  
    }  
    public Derived(int num,string str):base(num)  
    {  
        this.m_Str = str;  
    }  
}
```

## Abstract Classes and Abstract Methods

### Abstract Methods

- (Abstract Methods) לתמוך מתחדשת נוון למתדר ורטיאלי אשר אין לה ממש במחלקה בה היא מוגדרת.
- מוגדרת, אלים ייבם לממשה במחלקה המוגדרת.
- אפשרויות להגדיר גם מאפיינים (Abstract Properties) של C# למות שרים מופשטת או רטאלית, לא נוון לוגרמות.
- כיטואלים בוגורה מופשטת או מאפיין מופשטת.
- המדרמת כוותאים ריאי מוגדרת מוגדרת מוחבנית אוטומטית על ידי הסביבה.

25

## Abstract Classes and Abstract Methods

### Abstract Methods

המשל:

```
abstract class Base  
{  
    protected int m_Num;  
    public Base() { ... }  
    public abstract void printMsg();  
    public abstract void print();  
    public abstract int Num  
    {  
        get;  
        set;  
    }  
}
```

26

Abstract Classes and Abstract

## Methods

רמאל:

class Derived : Derived

```
public override void PrintMsg()
{
    Console.WriteLine("Stam Message 2");
}
```

המחלקה Derived2 אינה מוסיפה מימוש מושל עצמה לאפ"י וענף אם רואים את הטענה כט甂יפליטה.

Abstract Classes and Abstract

## Methods

רשות:

```
public override void PrintMsg() { ... }
public override void Print() { ... }
public override int Num {
```

```
    }  
    get { ... }  
    set { ... }  
}  
public String St
```

Design Issues

0 - Generic code במתלהה מושפעת מהיראותם נגדיר וממש בשורה מודול ויטאלית.

ה'מכוּם

- 4 – מאושפץ להתייחס לאובייקטים מהIMALות השונות היוצרים את אותן מהילות מייצ' קיד אוחד לכל ההוררכו.
- 5 – ביצורו זהה באמצעות הגדלת הבסיס.

**Declaring Interfaces**

Syntax:

```
interface ISample
{
    void Method1();
    float Method2();
    object Method3();
    ...
    string MethodN();
    int Property1
    {
        set;
        get;
    }
    event DelegateName EventName;
}
```

מקובל ששם משמש מוחרי פאות, ל'

**Agenda**

- Declaring Interfaces
- Implementing Multiple Interfaces
- Virtual Implementing Interface Methods
- .NET Build-IN Interfaces

**Declaring Interfaces**

בשפת C# מחלקה יכולה יכליה לרשותה אותה בלבד, ואולם בתוסס היא יחול על השם הפונקצייתו (Interface) המהארה על מethodים (Methods) או מאפיינים (Properties). Data Members או מethodות עם יימושים נכני לתכלי מודולות טריטוריה (Abstract class) בודה שלא ממש דומה למחלקה אבסטרקטית. ניתן לא"ז א"מ מופעם.

- אולם הוא שונה ממחלקה אבסטרקטית משום שהוא רק תבריר מופעם.
- בממיהק לנתן לתמיך הרשאות גישת המ"מ זיהוי ריבוטאליסטי.
- המימושים במלחוקות הנדרות לא ריהוי ריבוטים מ.פ.

**Implementing Interfaces**

**Microsoft .net**

## Declaring Interfaces

**CORNER**

**לדוגמא:**

- מילההו הינה שמשת למתוך את המילים ותפקידים שתוכנן מדריך, הממשיכל לירוח ורטאל, אולם זה לא חי'ב לירוח כזו.
- מקבל לומר שטולריה'ה'ם מושך.

```
class Sample : IPrint
{
    private byte m_Num;
    public Sample() { ... }
    public void Print()
    {
        this.m_Num = num;
    }
    public byte Num
    {
        get
        {
            return m_Num;
        }
    }
}
```

## Declaring Interfaces

**CORNER**

**לדוגמא:**

מודע מושיכים - "הוושה מורה"  
 (Multiple Inheritance) אם תמכה בורשות מורה:

```
class Vehicle
{
    void Print();
    byte Num
    {
        get;
    }
}
```

## Declaring Interfaces

**CORNER**

**לדוגמא:**

```
class App
{
    static void Main(string[] args)
    {
        Sample s = new Sample(123);
        s.Print();
        IPrint p = new Sample(45);
        p.Print();
    }
}
```

## Declaring Interfaces

**CORNER**

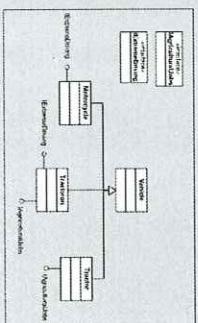
**לדוגמא:**

מודע מושיכים - "הוושה מורה"  
 (Multiple Inheritance) אם תמכה בורשות מורה:

```
class Vehicle
{
    void Print();
    byte Num
    {
        get;
    }
}
```

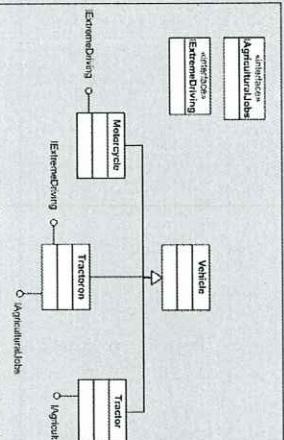
## Declaring Interfaces

**מדוע ממשיכם - "הורשה מרובה" :**  
התחל ש C# מיענה רוא ממשיכם.



Declaring Interfaces

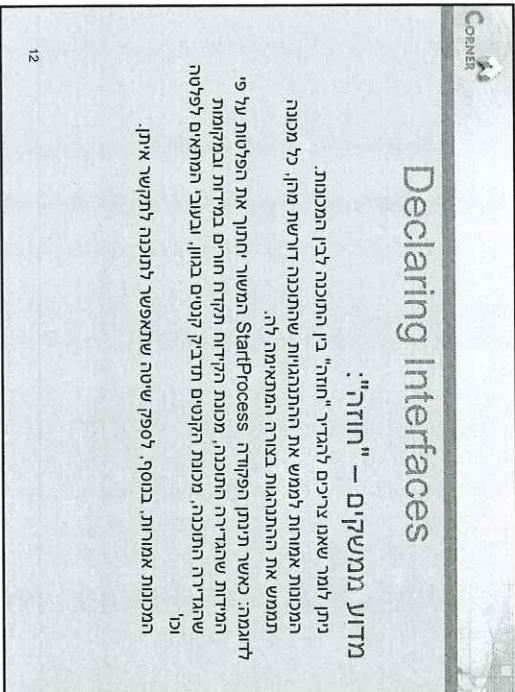
**מגדע משיקם** - הורשה מרובה :  
ההולך C# שמעה הא ממשיקם.  
לדגמה:



## Declaring Interfaces

Declaring Interfaces

**מדוע גמישקרים – “זהות”**  
נוירו שאמ פתרות תונכה לעילא פועל לעזר רודיטם, המונגה מוגלה להילא  
פרישת המומן רוט, ליהיל הראות (פלטילון), וליחסו של הרשות (CNC).  
“וואו – מכונת הרינו איה אוטומטית מהותה (פלטילון).”  
בתהילן “וואו” משנתנווט מסך מתוונת של כל מוכנה של הפוך מד ג’, לדאגמה:  
מכונית ריתוך פטלסט, מכונית קיטם, מכונית קיטם, ו... אין מתגעיגים  
מענינו לא במכונית הדאנס. אלמן, אולם, אינן מתגעיגים  
בדרכן בהרבות המכוניות הדאנס את המהו און שולחן:  
למשל נויר למדראש שטמי המכונית דוא דריש און הרהגואות של להילא:  
כל כוונת האות אמותה לשלב הראות פועל לעזר רודיטם.





## Declaring Interfaces

מודע ממשקים – "חוזה":

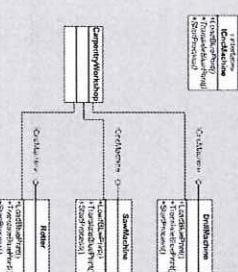
```
public interface ICncMachine
{
    void LoadBlueprint();
    void TranslateBlueprint();
    void StartProcess();
}
```

14



## Declaring Interfaces

מודע ממשקים – "חוזה":  
לאור כר C# מדרה מושך:



13



## Declaring Interfaces

מודע ממשקים – "חוזה":

```
public class Rotter : ICncMachine
{
    public void LoadBlueprint()
    {
        . .
    }
    public void TranslateBlueprint()
    {
        . .
    }
    public void StartProcess()
    {
        . .
    }
}
```

15

16



## Declaring Interfaces

מודע ממשקים – "חוזה":

```
public class SawMachine : ICncMachine
{
    public void LoadBlueprint()
    {
        . .
    }
    public void TranslateBlueprint()
    {
        . .
    }
    public void StartProcess()
    {
        . .
    }
}
```

## Declaring Interfaces

מודיעת מושגים – מושג "מזהב"

```

public class CarpentryWorkshop
{
    private ICncMachine[] machines = new ICncMachine[100];
    private int counter;
    public void AddMachine(ICncMachine new_machine)
    {
        machines[counter++] = new_machine();
    }
    public void CreateFurniture()
    {
        for(int i=0;i<counter;i++)
        {
            machines[i].LoadBlueprint();
            machines[i].TranslateBluePrint();
            machines[i].StartProcess();
        }
    }
}

```

## Declaring Interfaces

מודיעת מושגים – מושג "

```

public class DrillMachine : ICncMachine
{
    public void LoadBlueprint()
    {
        ...
    }
    public void TranslateBluePrint()
    {
        ...
    }
    public void StartProcess()
    {
        ...
    }
}

```

## Declaring Interfaces

מודיעת מושגים – מושג "בעל המנגנון"

בעל המנגנון להזין מכות אירוח דואשה:

```

public class PackingMachine : ICncMachine
{
    public void LoadBlueprint() { . . . }
    public void TranslateBluePrint() { . . . }
    public void StartProcess() { . . . }
}

```

## Declaring Interfaces

מודיעת מושגים – מושג "Main"

```

public class App
{
    public static void Main()
    {
        CarpentryWorkshop carpentry = new CarpentryWorkshop();
        carpentry.AddMachine(new SawMachine());
        carpentry.AddMachine(new DrillMachine());
        carpentry.AddMachine(new Rotor());
        carpentry.CreateFurniture();
    }
}

```



## Implementing Multiple Interfaces

מחלקה יכולה לרשת רק מחלקה אחת בלבד, אולם בנוסח (או במקומו) רוא כל פעולה של ממשקים אחדים ריבטם. המחלקה הינה בתייה כמודול למשתמשים כל המתוחזק ומאופיינם של הממשקים אותם היא יושתת.

לדוגמא:

```
public interface IPrint
{
    void Print();
}

public interface IMath
{
    int Sum();
    float Avg();
}
```

22



## Declaring Interfaces

מודול משיקים אחד או יותר. גל

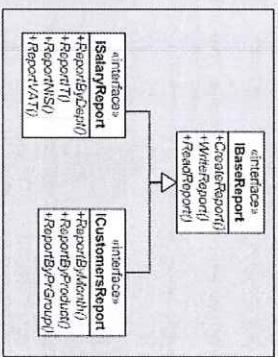
```
С:\Windows\Temp\Documents\TopPackage\bin\Debug\TopPackage.exe
С:\Windows\Temp\Documents\TopPackage\bin\Debug\TopPackage.exe
SawMachine - SawMachineBluePrint
SawMachine - SawMachineBluePrint
DrillMachine - DrillMachineBluePrint
DrillMachine - DrillMachineBluePrint
Router - RouterBluePrint
Press any key to continue...
```

21



## Implementing Multiple Interfaces

משהו יכול לרשת ממשק אחד או יותר.  
לדוגמא:



24



## Implementing Multiple Interfaces

המשר:

```
class Sample : IPrint, IMath
{
    ...
    public Sample() { ... }
    public Sample(byte n1, byte n2, byte n3) { ... }
    ...
}
```

```

    public int Sum()
    {
        return m_Num1+m_Num2+m_Num3;
    }

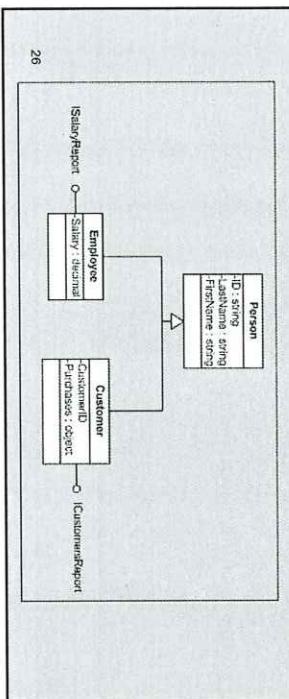
    public float Avg()
    {
        return (float)Sum()/3;
    }
}
```

2

## Implementing Multiple Interfaces

מתקנה הממשת משיק נגן, ח"י בית למשת את כל מה שמתאר ה[ במשק הבסיס ווּה במשק הנגן.

לדוגמה:



לדוגמא: משק בעשו יכול לרשת ממשק או יוּת.

## Implementing Multiple Interfaces

משק בעשו יכול לרשת ממשק או יוּת.

לדוגמא:

```

public interface IBaseReport
{
    void CreateReport();
    void WriteReport();
    void ReadReport();
}

public interface ICustomersReport : IBaseReport
{
    void ReportByMonth();
    void ReportByProduct();
    void ReportByPGroup();
}
  
```

## Implementing Multiple Interfaces

### Explicitly Implementing Interface Methods

לעתם קורא שטולקה ירושת שבי מושקים שונים, אלם שנות מילים גדרה זהה של מודוליה? ח"י ביט למשת את כל מה במרק שירה נהיה ח"יבם למשת של הממשקים באופן מושך (Explicit).

המגבלות הקיימות במרק זה הנה:

- אל ווּת לגדיר היטאת גשה באופן מושתמע,
- תחד סוכנות באטן מרווח.
- עמוש מושך און כל להיות ווּטואיל.

לדוגמה:

## Implementing Multiple Interfaces

מתקנה הממשת משיק נגן, ח"י בית למשת את כל מה שמתאר ה[ במשק הבסיס ווּה במשק הנגן.

לדוגמא:

```

public class Customer : Person, ICustomersReport
{
    private int CustomerID;
    private object Purchases;
    public void ReportByMonth() { . . . }
    public void ReportByProduct() { . . . }
    public void ReportByPGroup() { . . . }
    public void CreateReport() { . . . }
    public void WriteReport() { . . . }
    public void ReadReport() { . . . }
}
  
```



## Implementing Multiple Interfaces

### Explicitly Implementing Interface Methods

ק"א להעת מתוחמת מיהי-בהת שמשות

לדוגמה:

```

class Application
{
    static void Main(string[] args)
    {
        Sample s = new Sample(33);
        ((IPrint1)s).Print();
        ((IPrint2)s).Print();
        IPrint1 i1 = new Sample(12);
        i1.Print();
        IPrint2 i2 = new Sample(45);
        i2.Print();
    }
}

```

```

public interface IPrint1
{
    void Print();
}

public interface IPrint2
{
    void Print();
}

class Sample:IPrint1,IPrint2
{
    public Sample() { ... }

    public Sample(int num) { ... }

    void IPrint1.Print() { ... }

    void IPrint2.Print() { ... }
}

```



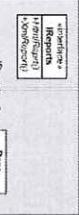
## Virtual Implementing

### Explicitly Implementing Interface Methods

מחלקה המימוש משתק נינה - ח'יבת למש את המethodים  
הוירטואליות המגדרות בה.

הימוש יכול להיות מימוש ורטואלי, קר שמהלה גזרת תכל לדרו

את התמוש של מתקן ה-*host*.



32



## Implementing Multiple Interfaces

### Explicitly Implementing Interface Methods

סביר ליתריה שבעמישיקים אשר מתוכננים על דינ לא יוציא מאכט לא בר"א

שכחה.

אם, מגב'ה, של שן מושרים או יותר המגדרים מודה באלת  
היהם זה אפשרי כאשר נעדור בשירותים ח'זנים, או כאשר נשלחים  
בתוכית רכיבים (Components) אשר כתבו על ידי גורם אחרם (Third Party).

31

## Virtual Implementing

Virtual Implementing  
המשל:

```

public class Base : IReports
{
    public virtual void HtmlReport()
    {
        Console.WriteLine("Base class : Generate Html
                           Report");
    }

    public virtual void XmlReport()
    {
        Console.WriteLine("Base class : Generate XML
                           Report");
    }
}

```

34

## .NET Build-IN Interfaces

וביבר.NET Framework מגדירה אוסף גדול שלழרים, תחילה ובה  
המזההות של IReports. מימוש את אותן ממשקים, או  
מעפה מעתם למתוך ממשקים אלו על מנת שיכל ליהנות משלימות  
שונם שאו על מנת שיכל ליהנות מעיצמות.

- IComparable
- IFormattable
- ICloneable
- IConvertible
- IWin32Window
- ISerializable
- IDisposable

35

## Virtual Implementing

Virtual Implementing  
המשל:  
לוגמה:

```

public interface IReports
{
    void HtmlReport();
    void XmlReport();
}

```

33

## Virtual Implementing

Virtual Implementing  
המשל:

```

public class Derived : Base, IReports
{
    public override void HtmlReport()
    {
        Console.WriteLine("Derived class : Generate
                           Html Report");
    }

    public override void XmlReport()
    {
        Console.WriteLine("Derived class : Generate
                           XML Report");
    }
}

```

35

**IComparable**

לדוגמה : System.Int32 הטיפוס המימוש ב-  
**public struct Int32 : IComparable, IFormattable, IConvertible**  
**CompareTo(Object obj)** . הינה מומשת את המethode השוואת בין שני אובייקטים מסוימים באמצעות  
**byte, short, ushort, float, long, ulong** כמותם היפשיום המודדים ב-

המהו כל תיפסיטם המודדים ב-  
**Array.Sort(Array)** (או **QSort**) אשר האלגוריתם זו רק מוחש את האלגוריתם זה רק אם קיימת Array.Sort(Array)  
**CompareTo** של מודד אחד.

38

**IComparable**

המזהה מדריך מדריך **IComparable** :

```
int CompareTo(Object);
```

שורה של המזהה לאucz השוואת בין שני אובייקטים מסוימים מאותה טיפוסים שונים  
 בעלי מלהלך בסיס משפטית ? מה נאץ מלהלך זה ?  
 אשר נעה ליהן אוקטם עלי אובייקטים פ. ג'ירניון.NET Framework.  
 פ. רישים את המזהה מהללה בשם כל System.Array. System.ArrayStatic. ArraySort ו-  
 מלהירה ומירה מודה שארם יSORT שארם י-  
 מומשת את אמורמתם של **IComparable.CompareTo(object)**. מודה.

37

**IComparable**

לדוגמה :  
**public class Circle : IComparable**  
**{**  
**. . .**  
**// IComparable implementation**  
**{**  
**public int CompareTo(object obj)**  
**{**  
**Circle c = null;**  
**if (obj is Circle)**  
**c = (Circle) obj;**  
**else**  
**return 1;**  
**return this.m\_Radius - c.m\_Radius;**  
**}**  
**}**

40

39

.NET Build-In Interfaces

1BDisposable

המחלקה `Disposable` מגדירה את Method `Dispose()`:

הא יירה. Streams, Files, Handles: כמו מונחים אלו מושגים אבטחים מילויים אבל לא מושגים מילויים אבטחים. Disposible המושג הזה לה במשמעותו מושג אחד אשר לא יכול לארוך זמן רב.

CORNER

IDisposable

הפעלה המתודה Dispose מtbodyutz בצוות ישירה או עוקפה.

```
static public void Main()
```

```
    }  
  
    static void Main() {  
        Sample t = new Sample();  
        t.DoSomething();  
        t.Dispose();  
    }  
}
```

44

## .NET Build-In Interfaces

### Comparable

```
Random rnd = new Random();
Circle[] arr = new Circle[10];
for(int i=0;i<10;i++)
    arr[i] = new Circle(rnd.Next(200));
```

CORNER

## Disposable

לדעתה:

לידות

```
        Console.WriteLine("Calling the Dispose Method...");  
        Console.WriteLine("Erasing Resources ...");  
        GC.SuppressFinalize(this);  
    }  
}
```

## NET Build-IN Interfaces

### IDisposable

פונקציית Dispose מובנית בברא "שירה או עיקפה".  
פונקציה זו מוחזקת בפונקציית Dispose.

```
public class SampleMain
{
    static public void Main()
    {
        using (Sample t = new Sample())
        {
            t. DoSomething();
        }
    }
}
```

משפט מסוג מדריך בלבד. בואו מהר לתקן תחולת המהווה

## מוסכמות בנתינת שמות. (כללים למשזהים)

### Pascal case

האות הראשונה במילה הראשונה-גדולה, וכל מילה נוספת נוספת מתחילה באות גדולה. השתמשי במשזה מסוג זה רק למילוט 3 אותיות או יותר.

לדוגמה: BackColor

### Camel case

האות הראשונה במילה הראשונה-קטנה, וכל מילה נוספת נוספת מתחילה באות גדולה. השתמשי במשזה מסוג זה רק למילוט 2 אותיות.

לדוגמה: backColor

### Uppercase

כל האותיות – גדולות

השתמשי במשזה מסוג זה רק למילוט 2 אותיות.

לדוגמה: System.IO System.Web.UI

תכן שימוש במשזהים עם אותיות גדולות לצורך תאימות עם קודים קיימים (ישנים) לא מנוהלים. (symbol schemes unmanaged

באופן כללי שימוש כאלו לא אמרו להיות חשוף מחוץ לאסמבלי.

### תמצית הכללים המקבילים

באופן כללי, כל שם שהוא ציבורי – יכתב לפי – pascal case – ויתחיל באות לטינית גדולה,

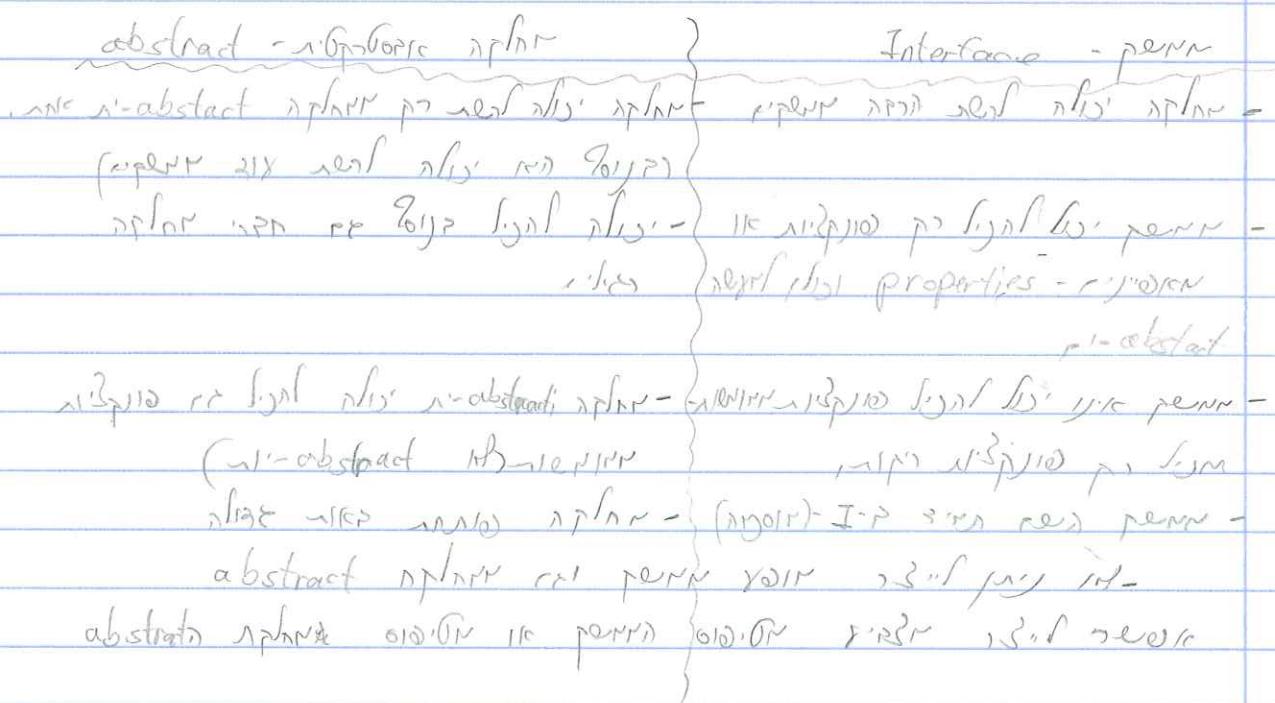
וכל שם פרטי יתחליל יכתב לפי camel case ויתחליל באות קטנה.

### פירוט הכללים המקבילים

מזהה עבורך:	סוג השם	לדוגמא:
Class	Pascal	<b>AppDomain</b>
Enum type	Pascal	<b>ErrorLevel</b>
Enum values	Pascal	<b>FatalError</b>
Event	Pascal	<b>ValueChange</b>
Exception class	Pascal	<b>WebException</b> <b>Note</b> Always ends with the suffix <b>Exception</b> .
Read-only Static field	Pascal	<b>RedValue</b>
Interface	Pascal	<b>IDisposable</b> <b>Note</b> Always begins with the prefix <b>I</b> .
Method	Pascal	<b>ToString</b>
Namespace	Pascal	<b>System.Drawing</b>
Parameter	Camel	<b>typeName</b>
Property	Pascal	<b>BackColor</b>
Protected instance field	Camel	<b>redValue</b> <b>Note</b> Rarely used. A property is preferable to using a protected instance field.
Public instance field	Pascal	<b>RedValue</b> <b>Note</b> Rarely used. A property is preferable to using a public instance field.



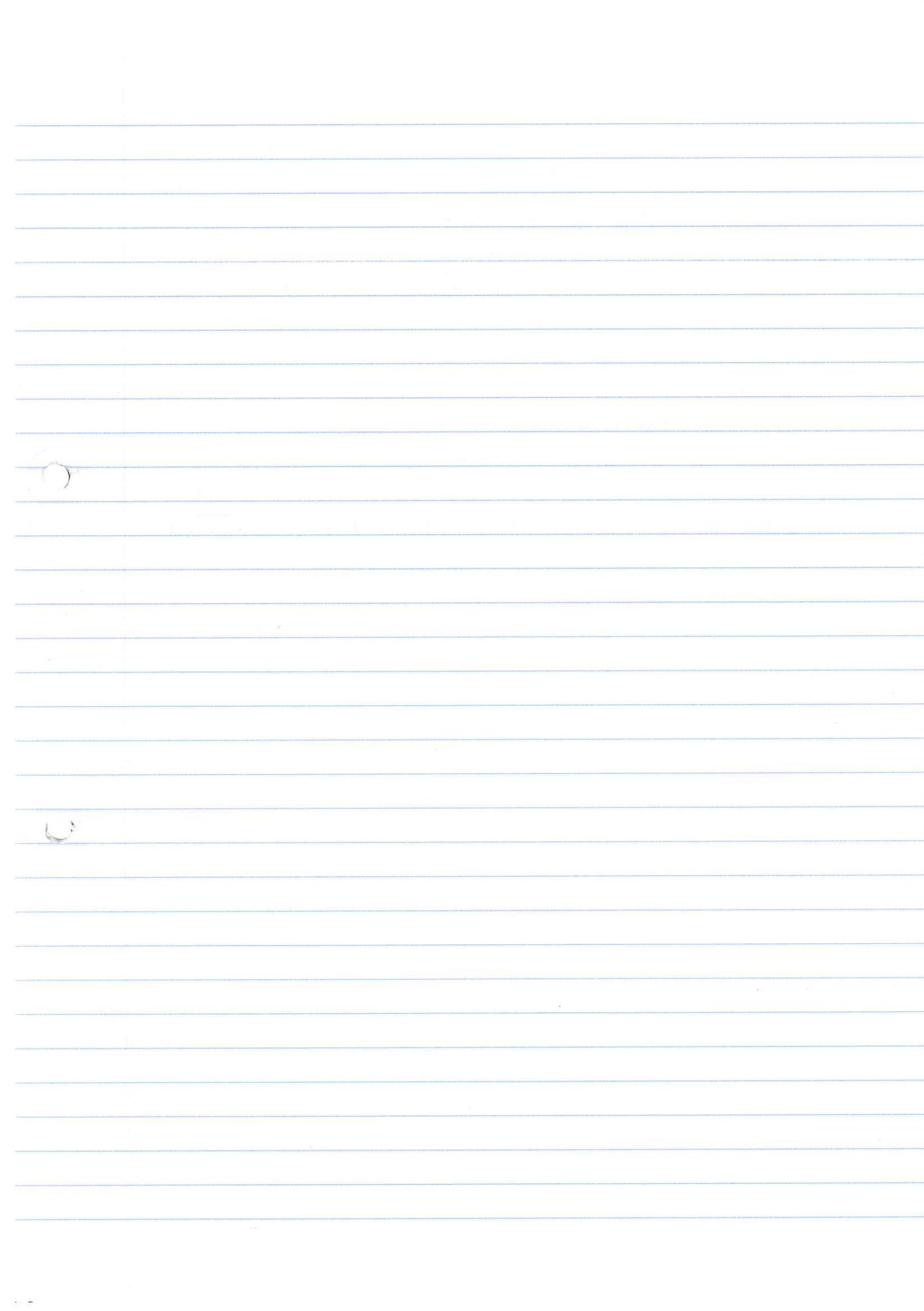
202



multiple inheritance  
parent implements  
multiple interfaces

ArrayList  
Stack  
Queue  
SortedList  
HashMap

using System.Collections  
List< >  
Stack< >  
Queue< >  
SortedList< >  
Dictionary< >



## 1. שאלות ממבחן בגד

ענין על כל 10 השאלות הבאות! . (4 \* 10 = 40 )

לפניך קוד מחלקת בגד קראי אותו ענין על השאלות שאחריו.

```
class clsChomer
{
    public string shem { get; set; }
    public int achuz { get; set; }

    האחזוֹד של החומר בגבגָד// }

abstract class clsClothing
{
    public int shem { get; set; }
    public int shnatYizur { get; set; }
    public List<clsChomer> chomarim { get; set; }
    public int price { get; set; }
    public int hanacha()

    {
        if (shnatYizur < DateTime.Today.Year)
            return (DateTime.Today.Year - shnatYizur) * 5;
    }
}
```

מחלקת בגד ניתן לרשף מחלקות רבות למשל: בגדי חורף , בגדי קיץ , בגדי שבת וכו' אנו נבצע ( חלקית ) את מחלקת בגדי חורף

שיימי לבן במקורה והתשובה דורשת שינוי/תוספה במחלקת הבסיס, הדגישי זאת בתשובהך.

### שאלות:

1. במחלקת בגדי חורף הנחפה נקבעת לפי הכלל הבא : אם העונה הנוכחית היא חורף אין הנחה נוספת , בכל עונה אחרת ישנו 10 אחוזי הנחה נוספים מעבר לאחוזי הנחה שמחושבים במחלקות בגד . כתבי את פונקציית חישוב הנחפה כאשר חורף משמעו חודשים ( 3 – 10 ) קיץ ( 9 – 4 ) לזכור חישוב הנחפה השטמי בתאריך הנוכחי .
2. כתבי פונקציית הנחפה נוספת שהפעם קיבל בטור פרמטר מסוג enum של העונה הנוכחית . enum מוגדר ב namespace הבא public enum eOna{winter, summer}
3. בדיקת שעתנד היא פונקציה בוליאנית שחייבת להיכיל במחלקת בגדי חורף מה יש לבצע לשם כך ? איך ? ( כתבי קוד )
4. רשמי את הפונקציה הבוליאנית של בדיקת שעתנד – פונקציה זו בודקת באוסף החומרים האם צמר ופשתן כלולים בה – כתבי ביעילות ובקצרה .
5. כתבי מתודה המחזירה את הבגד הzel ביותר בתור אוסף מסוג בגדים .
6. ערך בירית מחדל של שנת יוצר של בגד הוא השנה הנוכחית . לבגדי חורף יש להגדיר בנוסף גם ערך בירית מחדל למשתנה הבוליאני האם אוטום לגורם ? כן . היכן תכתבו את הקוד ? כתבי את ההגדרות הנ"ל ללא כפל קוד .
7. כתבי פונקציית Equals הבודקת אם שני בגדים הינם שווים , הבדאים שווים אם יש להם אותו שם ? אותה שנת יוצר .
8. הגדרי ב Main שלושה בגדים מסוגים שונים ואתחלי את משתני המחלקה שלהם . נתונה הגדרת אוסף בגדים – ההגדרה כתובה בחו"ג main; Is > List<clsClothing
- הניח כי האוסף הת מלא בפרטי לבוש בהתאם לבחירת המשמש .
9. בדק אם בגד השלישי באוסף יש שעטנד . אם יש בו שעטנד מחקי אותו מהאוסף .
10. ברצוני למיין את אוסף הבדים לפי מחיר . המינון יבוצע באמצעות הפקודה () sort; Is . – מה עלי לבצע לשם כך ? ( כתבי קוד )



## שיעור 6# – ממשקים - Interfaces

### ראשי פרקיים:

- ★ ממשק – Interface
- ★ ירשה מרובה ממשקים לעומת יחידה ממחלקות,
- ★ היררכית ממשקים
- ★ימוש הממשק – שימוש ממשי ושימוש וירטואלי
- ★ ירושת שני ממשקים, בהם פונקציה באותו שם, שימוש מפורש
- ★ שימוש במשקיים מובנים

IEnumerator, IEnumerator ↗

IDisposable ↗

IComparable ↗

★ מצביע מטיפוס המשק

★ פרמטרים לפונקיה מטיפוס המשק

★ השואה בין ממשק למחלקה אבסטרקטית:

מחלקה אבסטרקטית	ממשק	ממשק
מכמה מהם מחלוקת יכולה לרשות	תיזען	תיזען
יכול להכיל members-data	תיזען	תיזען
אם כל החברים אבסטרקטיים	תיזען	תיזען
יכול להכיל שימוש לפונקציות	תיזען	תיזען
האם ניתן ליצור ממנו מופע	תיזען	תיזען
אפשרות צוין מצינית גישה לחברים	תיזען	תיזען
אפשרות לכתיבת בנאים	תיזען	תיזען

כגון

תיזען

תיזען (בזאת שפונקצייתו מוגדרת)



## תרגיל – מחלקות אבסטרקטיות

עבור מחשבון קיימים מספר סוגים של תרגילים: חיבור, חיסור, כפל, חילוק, חזקה, שורש.  
לכל אחד מהתרגילים יש שני ארגומנטים, ותאור פעולה.  
כל תרגיל יודע להציג תרגיל, וגם להחזיר פתרון.  
הצגת התרגיל היא איחודית לכל התרגילים, אולם אופן הפתרון מתבצע באופן אחר עבור כל תרגיל.  
עליך לכתוב:

1. רשיימה [numen] שתכילה את כל סוגי הפעולות. [שם באנגלית ולא הסימן של הפעולה]

2. מחלקה אבסטרקטית בשם Exercise אשר תכיל :

- שני מאפיינים מטיפוס double shirao Argument1-Argument2, argument1, argument2, בהתחמה.
- מאפיין אבסטרקטי מטיפוס מחוזצת שיחזר את סימן הפעולה של התרגיל. (למשל <sup>2</sup> - זה תלוי גם בארגומנטים של התרגיל המסוים ולא אחד לכל המהלך, וכך const)
- משתנה מסווג הרשיימה שיצין את סוג התרגיל. [חיבור/חיסור/חילוק וכו']
- פונקציה רגילה [לא אבסטרקטית] שתחזיר את התרגיל כולו כמחוזצת.
- פונקציה וירטואלית שתבדוק האם התרגיל הוא בר-פתרון, או לא. [isValid] המחלקות הנגזרות יכולן לדרכו את הפונקציה אם ירצו, ויכולו שלא לדרכו. במחלקה הבסיס הפונקציה תחזיר true , במחלקות הנגזרות כל מחלקה תוכל למשב באופן אחר את הפונקציה או לא למשב. [למשל בשורש אם הארגומנט הראשון שלילי והשני זוגי- אין פתרון]
- פונקציה אבסטרקטית המחשבת ומוחזירה את הפתרון לתרגיל.

3. מחלקות נגזרות:

- a. חיבור
- b. חיסור
- c. כפל
- d. חילוק
- e. שורש
- f. חזקה

4. כל אחת מהמחלקות הנגזרות תוכל למשב את הפונקציה isValid , האם התרגיל הוא בר-פתרון , או לא.

5. כל מחלקה נגזרת תהיה חיבת למשב את המאפיין "סימן פעולה" ואת פונקציית הפתרון.

בתכנית הראשית עלייך ליצור רשיימה של 10 תרגילים.

הארגומנטים יהיו אקרים.

עברית בלולאה על כל הרשיימה, ובדקי האם ניתן לפתור את התרגיל.

אם כן - התכנית תציג בחולון הקונסול את התרגיל ואת פתרונו, כמחוזצת.

אם לא – התכנית תציג ואת הפתרון: NaN. [Not a number] לדוגמה:



## תרגילים

### \* תרגיל 1

1. כתבי 2 מחלקות: מחלקת עצהו ומחלקתILD
2. כתבי ממשק המחייב את המחלקות הממשות אותו להחזיר שם מלא של הפריט.
3. במחלקה ILD קיימים החברים הבאים: שם פרטי שם משפחה ותאריך לידה.
4. במחלקה עצהו קיימים מאפיינים: שם, שם החברה, התאמה לגיל
5. כתבי פונקציה במחלקה program המקבלת פריט מסווג המשתק ומדפיסה את השם המלא שלו.

### \* תרגיל 2

הגדרי ממשק בשם RememberTime אשר מגדיר 2 דברים:

פונקציה המקבלת תאריך-שעה ומעדכנת עבור האובייקט את התאריך.

מאפיין המחזיר את זמן יצירת האובייקט. ממילא כל מחלקה יורשת תכיל נתון פרטי המציין את תאריך היצירה שלו.

חשבוי היכן לאותה נתון זה.

כתבו מחלקת ל��וח ומחלקת מוצר, (המציאי 3 מאפיינים לכל אחד מהם) בנוסף, שניים יורשים מהמשתק הנ"ל.

הוסיפו מחלקת ל��וח קבוע היורשת מלקוח, ומוסיפה עבורה קוד כרטיס ות.ז.

האם המחלקה זו יורשת מהממשתק? \_\_\_\_\_ האם צריך לכתוב זאת? \_\_\_\_\_ אם ניתן להמיר אובייקט מחלוקת?

כתבו מחלקה סטטית בשם CheckClass ובתוכה פונקציה סטטית המקבלת אובייקט כלשהו מסווג המשתק כפרמטר, ומחזירה true אם הוא נוצר היום ו- false אם לא.

בutor הפונקציה עליך לבצע את הבדיקה.

### \* תרגיל 3:

1. צרי מחלקת Person שם, תאריך לידה ותעודת זהות.
2. המחלקה תשתמש את המשתק Compareable.

ממשי את הפונקציה Compare כך שתשווה בין האנשים על פי שם המשפחה שלהם.

a. במחלקה הראשית:

צרי לפחות 3 אובייקטים של אנשים.

קלטי מהמשתמש נתונים בולולאה עבור 2 אנשים.

הדף יא את האדם "הגדל" יותר.

b. צרי מערכ או רשימה של 6 אנשים.

c. מייני את המערך (על ידי פונקציה מוכנה) בסדר עולה.

d. הדפיס את המערך הממוין. (בולולאה, הדפסי שם+משפחה משורשרים.)

3. כתבי מחלוקת מפעל המכילה מערכ/רשימה פרטית של אנשים.

a. ממשי במחלוקת את המשקרים - IDisposable ienumerator כדי לאפשר לעבורה על

המערך בולולאת foreach



## מחלקות גנרייט

לעתים נרצה לבנות מחלקות אשר ממד דומות בתבניתם, ושותות רק בסוג הנתונים שהם מחייבות.

- נרצה מחלוקת Point המיצגת נקודה במשור בעלות יכולות, כמו למשל להציג את הנקודה, כאשר לעתים נרצה שהאורדינאות (y,x) של הנקודה יהיו מסוג זה ולעתים נרצה שהיא מסוג double (או מכל סוג אחר – long, byte, decimal ...)
  - מחלוקת לניהול רישימה של נתונים (Collection) עם יכולת להוסיף ולמחוק איברים כאשר בכל פעם נרצה לשים שם נתונים אחרים
- בעזרת הידע שיש לנו עד עתה נצטרך לבנות את המחלקות מספר פעמים, פעם אחת עבור כל טיפוס. אפשרות אחרת היא להשתמש בטיפוס Object אך אפשרות זו לא מספיק טוביה כי היא מחייבת אותנו לבצע casting בכל פעם שנרצה לעבוד עם המשתנה ובנוסף אנו עלולים לטעות ולהכניס למשתנה משתנה מטיפוס שלא התכוונו אליו והקומפיאילר לא יתריע על כך, אלא נקבל את השגיאה רק בזמן ריצה ב- casting הלא חוקי.
- Generics נותן פתרון לבעה זו בכך שהוא מאפשר להגדיר גם את סוג המשתנה כפרמטר, וכך יוכל ביצירת האובייקט לקבוע מה יהיה הטיפוס עבור האובייקט המסופים ובאובייקט אחר נקבע טיפוס אחר. בשיטה זו לא נדרש לבצע casting מפני שהוא מגדירים את הטיפוס ואם ננסה להכניס ערך מטיפוס לא נכון בטעות נקבל שגיאת קומפיאילציה.
- ב כדי להשתמש ב- Generics יש להגדיר את הטיפוס כפרמטר באמצעות סוגים מושלמים <> מיד לאחר שם המחלוקת לדוגמה:

```
class Point <T>
{
    ...
}
```

לאחר מכן יוכל להשתמש בפרמטר זה בתוך המחלוקת:

```
class Point<T>
{
    private T x;
    private T y;

    public T X
    {
        get { return x; }
        set { x = value; }
    }

    public T Y
    {
        get { return y; }
        set { y = value; }
    }

    public Point(T x, T y)
    {
```

```

        X = x;
        Y = y;
    }

    public override string ToString()
    {
        return string.Format("X = {0}, Y = {1}", X, Y);
    }
}

```

כasher ביצור את האובייקט נספיק לו את הפרמטר של הטיפוס גם כן בסוגרים משולשים. לדוגמה Point כאשר ה-T הוא מסוג int:

```

Point<int> pInt = new Point<int>(10, 5);

```

שימוש לב שבדוגמא הבאה שתי השורות האחרונות יגררו שגיאת קומpile'יה מפני שתיפוס הנתונים מוגדר מראש ולא ניתן להכניס ערך מסוג אחר:

```

static void Main(string[] args)
{
    Point<int> pInt = new Point<int>(10, 5);
    Point<string> pString = new Point<string>("a", "b");
    pInt.X = "a";
    pString.X = 10;
}

```

עד נקודות חשובות לגבי Generics:

- ניתן לקבל יותר פרמטר אחד עבור מספר סוגים של משתנים.

- ניתן לבנות כמעט כל אלמנט שאנו מכירים כ- generics כגון: class, struct, method, delegate, interface

- ניתן לרשת מחלוקת generics ולהגדיר לה את הטיפוסים קבועים generics להורשת מחלוקת generics:

```

class PointDouble : Point<double>
{
}

```

בספריות של .NET ישנו הרבה טיפוסים מוכנים המממשים את הרעיון של Generics ומייניטם לשימושינו, לדוגמה:

- **IComparable** - למימוש **T>IComparable>**

**CompareTo**

- דומה ל- **ArrayList** אך עם הגדרת הטיפוס של הנתונים ביצירת האובייקט

- דומה ל- **Dictionary** - עם הגדלתו הסוג של ה- key ועל ה-

**value** ביצירת האובייקט

### **מחלקות גנריות - תרגילים:**

1. צרי מחלקה גנרית אשר מכילה 3 ערכים מסווג כלשהו. הערכים יתקבלו במבנה:  
המחלקה תכיל פונקציה אשר מחז'י רה מהרזהת המשרשת את כל הערכים באוסף זה לזה.
2. צרי מחלוקת טווח (range) גנרית אשר תוכל לקבל שני ערכים מאותו סוג ולחשב את ההפרש ביניהם.

לדוגמא: הפרש בין תאריכים, בין מספרים(שלמים או שברים), בין 2 נקודות (קוואורדינטות) ועוד.

- המחלוקת תכיל שני מאפיינים לערכים,
- בניין שיקבל את הערכים,
- פונקציה subtract אשר תחשב ותחזיר את ההפרש ביניהם.
- צרי ממשק אשר יחייב להגדיר פונקציה המשרשת שני אובייקטים אחד מהמשני במחלוקת ומהזירה INT כהפרש. (פונקציית מופע אשר תחסר את הפעמטר מה- this)
- הגביל את המשתמש במחלוקת ליצור אובייקטים מהחלוקת בלבד רק עבור סוגים המשמשים את המשך יצירתה.
- צרי מספר סוגים כאלה ונסוי את המחלוקת.

העתיקי את הודעת השגיאה המופיעה כאשר את משתמש במשוך ממשך עבור סוג שאינו מימוש את המשך.

---



## מחלקות גנריות - כ"ח בתשרי תשעט

```
class GenericAdd<T>    where T : IMinus
{
    public T Value1 { get; set; }
    public T Value2 { get; set; }

    public GenericAdd(T val1,T val2)
    {
        Value1 = val1;
        Value2 = val2;
    }
    public GenericAdd()
    {

    }
    public object Sub()
    {
        return Value1 .Subtract( Value2);

        ////if ( Value1.GetType()==typeof (int ))
        //if (typeof (T) == typeof( int? ) )
        //{
        //    return (Value1 as int?) - (Value2 as int?);
        //}
        //if (typeof(T) == typeof(float?))
        //{
        //    return (Value1 as float ?) - (Value2 as float ?);
        //}

        //return 0;
    }
}

class Program
{
    static void Main(string[] args)
    {
        //GenericAdd<int?> genInt = new GenericAdd<int?>();
        //genInt.Value1 = new int?( 7);
        //genInt.Value2 = new int?(3);

        //GenericAdd<float?> genFloat = new
        GenericAdd<float?>(9.33F,5.66F);

        //Console.WriteLine(    genInt.Sum() );
        //Console.WriteLine(    genFloat .Sum());

        //GenericAdd<Person> genPersons = new GenericAdd<Person>();
        //genPersons.Sum();

        GenericAdd<MyDouble > genDouble = new GenericAdd<MyDouble >();
        genDouble.Value1 = new MyDouble() { DoubleValue = 4.2 };
        genDouble.Value2 = new MyDouble() { DoubleValue = 5.9 };

        GenericAdd<MyFloat> genFloat = new GenericAdd<MyFloat>()
        {Value1=new MyFloat(){FloatValue = 5.3F }, Value2 = new MyFloat() { FloatValue
        = 5.9F } } ;
        //שגיאה
        // GenericAdd<string> genFloat
```



```
Console.WriteLine(genDouble.Sub ());
Console.WriteLine(genFloat.Sub ());

//האיגש.
//GenericAdd<Person> genPersons = new GenericAdd<Person>();
//genPersons.Sum();

Console.Read ();

}

}

class MyInt : IMinus
{
    public int val { get; set; }
    public int Subtract(object value)
    {
        //doto: sub the dateimes
        if (value is MyInt)
            return Math.Abs(this.val - (value as MyInt).val);
        else
            return 0;
    }

}

interface IMinus
{
    int Subtract(object value);
}
```



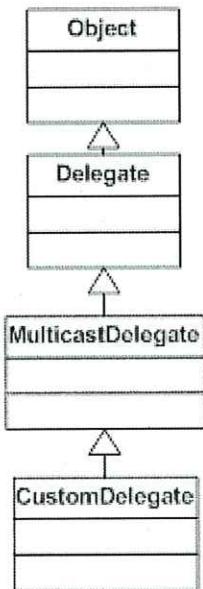
## Delegate (ડેલેગેટ)

એક પ્રોગ્રામ લાભ કરતું હોય તો એનુભવ  
અને વિશ્વાસ કરી જો એ કોઈ નાચાય છે  
એનુભવ પ્રાપ્ત કરી જો એ કોઈ નાચાય છે (૧)  
જુદી વિશ્વાસ કરી જો એ કોઈ નાચાય છે (૨)  
બૃદ્ધિ કરી જો એ કોઈ નાચાય છે (૩)  
બૃદ્ધિ કરી જો એ કોઈ નાચાય છે (૪)  
બૃદ્ધિ કરી જો એ કોઈ નાચાય છે (૫)



**נציגים (delegates)** ו**אירועים (events)** הינם נושאים מאד מרכזיים בעבודה ב .NET - הם מאפשרים למפתח לבנות רכיב בלבד אשר מי שמשתמש בו לפעולות מסוימות.

### מהם נציגים?



- נציג הוא למעשה משתנה הכליל פונקציה [כתבת של פונקציה] בנגדו למשתנה רגיל שמכיל נתונים – או הפניה, מצביע, כתובת, נתונים).
- תפקידו של ה delegate - הוא ליצור אובייקט המצביע על פונקציה. במקום להפעיל את הפונקציה בצורה הרגילה, נפעיל אותה דרך ה delegate. היתרון של עבודה עם delegate בהצבעה על פונקציה שאנו לא מכירים בזמן כתיבת רכיב (מחלקה / פונקציה ועוד) מסוים,ומי שיקבע איזו פונקציה תופעל הוא מי שמשתמש ברכיב.
- בכל פונקציה, שם הפונקציה הנה למעשה מצביע (Pointer) המכיל את הכתובת של הפונקציה בזיכרון, קרייה לפונקציה הנה למעשה קפיצה בזיכרון המחשב.
- Delegate, הנה למעשה מצביע לפונקציה, דהיינו, טיפול אשר ייעודו להכיל את הכתובת של הפונקציה בזיכרון, בדיק כמוה שיחוס[הפניה לאובייקט] מכיל כתובת של אובייקט.
- באמצעות השימוש ב- Delegate נוכל לגשת ולהפעיל פונקציה, בדיק כמוה שבאמצעות הפניה לאובייקט ניתן לבצע פעולה על אובייקט.
- Delegates הם למעשה שיטות מושב (Callback Function).

### לשם מה צריך נציגים?

ל- Delegate מספר שימושים מרכזיים :

- בנייה קשר דו סיטרי בין מחלקות מכלולות.
- GUI (Graphic User Interface) .
- תכנות אסינכרוני (Asynchronous Programming).
- שליחת פונקציה כפרמטר לפונקעה אחרת.
- שימוש אירועים (Event), עליהם נרחב בהמשך.

ניתן לראות את הגדרת Delegate, יצירת מחלקה חדשה. אובייקטים שיוציאו מחלוקת זו יוכל להצביע על פונקציה אחת או יותר , בתנאי שהחтиימה של הפונקציה תואמת למבנה מסוים. שם המחלוקת שנבחר יהיה שם ה – Delegate. אמנם יצירת מחלוקת זו נכתבת בתחריר מקוצר. המחלוקת נוצרת בצורה מרמזת (Implicit). מחלוקת זו ירושת מחלוקת בשם MulticastDelegate אשר היא ירושת את המחלוקת Delegate אשר יירושת את המחלוקת Object.

```

namespace WhatIsDelegate
{
    delegate void MyDelegate();
    ...
}
  
```

### תחים :

#### 1- הגדרת הנציג:

מצהירים על Delegate באמצעות המילה השמורה Delegate. בשורת הגדרת ה - Delegate מצינים גם איך תראה חתימת הפונקציות אשר אובייקטים מה- Delegate יוכל להצביע عليهم. זו פקודה שלמעשה מגדרה מחלוקת-נציג בשם MyDelegate:



# C#

```

delegate void MyDelegate();
class App

{
    static void Func()
    {
        Console.WriteLine("Delegate");
    }

    static void Main(string[] args)
    {
        MyDelegate del = new MyDelegate(Func);
    }
}

```

בוגר שגיאת קומפילציה.  
בדוגמא זו הפונקציה תהיה חייבת להציג void ואין היא יכולה לקבל אף Parameter, כל ניסיון להעביר ל- Delegate כתובת של פונקציה בעל חתימה שונה.

## 2 - יצירת מופע של Delegate :

שלב שני ביצירת נציג הוא ליצור מופע מטיפוס הנציג שיצרנו. בדוגמה: del הנה "חומר מהמחלקה MyDelegate".

כאשר נקצתה אובייקט מהמחלקה MyDelegate נשלח ל- Ctor כפרמטר את שם המתודה[הראשונה] שהוא מפנה אליה.

חתימת המתודה חייבת להיות זהה להגדרת ה- Delegate .

## 3 - כתיבת פונקציה/פונקציות המתאימה/ות בחיתומה להגדרת הנציג.

## 4 - הוספת הפונקציות לנציג.

נציג יכול להציג על כמה פונקציות בו זמינות. אין למתכונת שליטה על סדר הביצוע של הפונקציות בנציג.

Del+=func2

Del-=func2

אם מנסים להסיר פונקציה שאינה קיימת בנציג – לא תתרחש שגיאה.

## 5 – הפעלת הנציג

del();

הקריאה הנ"ל תפעיל את המתודה Func משום שה- Delegate מחדיק את הכתובת שלה. אם מנסים להפעיל נציג אבל אין לו שיור לשום פונקציה – תתרחש שגיאת זמן ריצה . (exception)

כדי למנוע בעיה זו אנחנו שואלים לפני הפעלת הנציג האם הוא

מכיל ערך ריק:

(null!=del) If

תנאי זה יחזיר true אם משויות לנציג פונקציה אחת או יותר.

## דוגמאות לשימושים בנציג:

מетодות של מחלקת אב (מחלקה המכללה) יכולות להפעיל מethodות של מחלקת בן (מחלקה המוכלת).

מетодות של מחלקת בן אין מכירות את המethodות של מחלקת האב, لكن אין יכולות להפעיל אותן.

לעתים קרובות רצחה לייסד קשר דו סיטרי בין מחלקות מיכולות.

לדוגמא:

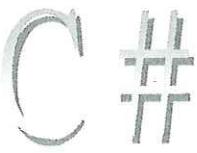
המתודה () Point.Print() מפעילה את המתודה () Circle.Print()

המתודה () Point.Print() אינה יכולה להפעיל את המתודה () Circle.Print() או כל מתודה אחרת של המחלקה Circle, הן פשוט מחוץ לטווח ההכרה שלהן.

לעתים רצחה שמחלקת האב תבצע פעילות מסוימת התלויה בפעולות המתבצעת במחלקה הבן, לעתים רצחה שמחלקת האב תקבל משוב (Feedback) על הפעולות המתבצעת במחלקה הבן.

במילים אחרות, רצחה לייסד תקשורת דו כיוונית, בה מתודה תדוחה לשות אחרת, שאינה מכירה, על פעילות המתבצעת בה או ערכי נתונים וכו'.





בדוגמה:

- נרצה ש- Circle ידע ליצור את עצמו מחדש על המסך בכל פעם ש- Point משתנה.
- נרצה ש- Circle יקבל הודעה על ערכיהם לא חוקיים שמתאפשרים ב- Point.
- איך הנסייג יפתר לנו את הבעיה?
- שלב א' – הוצאה על Delegate :

```
public delegate void PointChangedDelegate(Point p);
```

• שלב ב' – הגדרת ייחוס מטיפוס ה- Delegate לחבר מחלקה :

```
public class Point
```

```
{
```

```
    private int m_X;
```

```
    private int m_Y;
```

```
    public PointChangedDelegate m_Callback;
```

```
    ...}
```

• שלב ה' – הפעלת ה- :Callback Function

```
public class Point
```

```
{
```

```
    ...
```

```
    public int X
```

```
{
```

```
    get {...}
```

```
    set
```

```
{
```

```
    ...
```

```
    m_Callback(this);
```

```
}
```

```
}
```

```
}
```

• שלב ד' – הגדרת מетодות המשוב של ה- Delegate :

```
class App
```

```
{
```

```
    private static void PointChanged(Point p) {...}
```

```
    static void Main(string[] args)
```

```
{
```

```
        Point p = new Point();
```

```
        p.m_Callback = new
```

```
            PointChangedDelegate(App.PointChanged);
```

```
        ...
```

```
}
```

```
}
```

## Multicast Delegate

• Delegates ב- .NET הינם למעשה Multicast Delegates.

• המשמעות היא ש- Delegate ייחד יכול להתייחס ולהפעיל מספר מетодות משוב ולא רק מетодה יחידה.

•Delegate הינו מערך של מצביעים לMETHODS.

• delegateMulticastDI SHA – יכול להציג גם על מספר פונקציות, ניתן לעשות זאת באמצעות האופרטור += לדוגמה:



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Delegates
{
    //1
    public delegate void DelStudentLeave(int studentId);
    class Program
    {
        //1 ==example3
        public delegate bool delFilter(int i);
        public static List<int> Filter(delFilter filterFunc, List<int> list)
        {
            List<int> l = new List<int>();
            foreach (int item in list)
            {

                if ( /*//5 ==example3*/ filterFunc(item))
                    l.Add(item);
            }
            return l;
        }

        static void Main()
        {
            // C# 1.0 style
            Console.WriteLine("=====GetPositives=====");
            List<int> list = new List<int> { 1, 43, -6, 12, 5, -23, -7, 86 };
            int x;
            List<int> filteredList = Filter( /*//2+4 ==example3*/ GetPositives,
list);
            foreach (var item in filteredList)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine("=====negetives=====");
            List<int> negetives = Filter(GetNegetive, list);
            foreach (var item in negetives)
            {
                Console.WriteLine(item);
            }

            //=====
            Console.WriteLine("=====onlyEvenNumbers=====");
            List<int> onlyEvenNumbers = Filter(GetEven, list);

            foreach (var item in onlyEvenNumbers)
            {
                Console.WriteLine(item);
            }
            Console.ReadLine();
            // C# 2.0 style
            //anonimiyut ponkzia shel shilohah
        }
    }
}

```

```

int xValue = 200;
List<int> GraeterThenX = Filter(delegate (int i) { return i > xValue; }
, list);

List<int> GraeterThen10 = Filter(delegate (int number) { return (number > 10); }, list);

List<int> GraeterThen10a = Filter(    number => number > 10    , list);
//    // C# 3.0 style
List<int> GraeterThenXValue = Filter(number => number > xValue , list);

}

private static bool GetNegetive(int i)
{
    return i < 0;
}

//3 ==example3
private static bool GetPositives(int i )
{
    return i >=0 ;
}

private static bool GetEven(int i)//להזיר רק זוגיים
{
    return (i % 2 == 0);
}
public static bool GreaterThanTen(int num)
{
    return num > 10;
}

static void Main2 (string[] args)
{
    Calculator myCalculator = new Calculator();
    Console.WriteLine("please type action");
    string op= Console.ReadLine();

    Console.WriteLine("please type first argument");
    string snum1 = Console.ReadLine();
    double num1;

    bool b = double.TryParse (snum1, out num1);
    while (!b)
    {
        Console.WriteLine("please type first argument again");
        snum1 = Console.ReadLine();
        b = double.TryParse(snum1, out num1);
    }
    Console.WriteLine("please type second argument");
    string snum2 = Console.ReadLine();
    double num2;
    b = double.TryParse(snum2, out num2);
}

```

```

while (!b)
{
    Console.WriteLine("please type second argument again");
    snum2= Console.ReadLine();
    b = double.TryParse(snum2, out num2);
}

double result;
switch (op)
{
    case "+":
    {
        result = myCalculator.GetResult( /*//4 ==>example2 */ Add,
num1, num2);
        break;
    }
    case "-":
    {
        delCalc delC = new delCalc(Sub);
        delC += Add;
        delC -= Add;
        result = myCalculator.GetResult(delC, num1, num2);
        break;
    }
    case "*":
    {
        result = myCalculator.GetResult(new delCalc( Double), num1,
num2);
        break;
    }
    case "/":
    {
        result = myCalculator.GetResult(Div, num1, num2);
        break;
    }
    default:
    {
        result = 0;
        break;
    }
}

Console.WriteLine("the result is {0}", result.ToString () );
Console.ReadLine();
}
//3 ==>example2
#region Operations for calc
public static double Add(double n1, double n2)
{
    return n1 + n2;
}
public static double Sub(double n1, double n2)
{
    return n1 - n2;
}

```

```

}
public static double Div(double n1, double n2)
{
    return n1 / n2;
}
public static double Double(double n1, double n2)
{
    return n1 * n2;
}
#endregion

static void Main1(string[] args)
{
    Student student1 = new Student()
{StudentId=1,FirstName="aaa",LastName="2222", Level = 8 };
    //4
    //student1.DelStudentLeave_obj = new
DelStudentLeave(TellUsersStudentLeave);
    student1.DelStudentLeave_obj += TellUsersStudentLeave;

    School school = new School(new List<Student>())
    {
        new Student(){StudentId=2,FirstName="sara",LastName="levi",Level=8 },
        new Student() { StudentId = 3, FirstName = "eee", LastName = "levi"
,Level=8},
        new Student() { StudentId = 4, FirstName = "ttt", LastName = "levi"
,Level=8},
        student1
    };
}

InsuranceCompany ic = new InsuranceCompany();

ic.AddCustomer ( school.StudentList[1]);
ic.AddCustomer( school.StudentList[2]);

foreach (var item in school.StudentList)
{
    item.Upgrade();
}

//func1(funcOkToTemplate, 10);

Console.ReadLine();
}

//private static void funcOkToTemplate(int studentId)
//{
//    Console.WriteLine("func as parameter");
//}

//3
public static void TellUsersStudentLeave(int sId)
{
    Console.WriteLine("the student with id {0} leave his school;",sId );
}

```

```

    }

    static string func1(DelStudentLeave leave, int i)
    {
        //5
        leave(i);
        return "yes";
    }

}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Delegates
{
    class School
    {
        public List<Student> StudentList { get; set; }
        //.....
        public School(List<Student> studentList)
        {
            StudentList = studentList;
            foreach (Student stItem in studentList )
            {
                stItem.DelStudentLeave_obj += AnyStudentLeavesScholl;
            }
        }
        //3
        public void AnyStudentLeavesScholl(int id)
        {
            Console.WriteLine("send mail to... to tell about {0}",id);
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Delegates
{
    class Student
    {
        public int StudentId { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public int Level { get; set; }
    }
}

```

```

//2
public DelStudentLeave DelStudentLeave_obj;

public void Upgrade()
{
    if (Level == 8)
        //5
        if (DelStudentLeave_obj!=null)
            DelStudentLeave_obj(StudentId );
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Delegates
{
    class InsuranceCompany
    {
        //לכונאי הקראה לפני מתבצע המהלך משתני אתחול/
        List<Student> listOfCustomers = new List<Student>()
        {
            new Student(){ StudentId=2,FirstName="sara",LastName="levi",Level=8 },
            new Student() { StudentId = 3, FirstName = "eee", LastName = "levi"
,Level=8},
            new Student() { StudentId = 4, FirstName = "tttt", LastName = "levi"
,Level=8}
        };

        public InsuranceCompany()
        {
            //
            foreach (var item in listOfCustomers )
            {
                item.DelStudentLeave_obj += InsuranceCompanyLostCustomer;
            }
        }

        public void AddCustomer(Student newSt)
        {
            listOfCustomers.Add(newSt );
            newSt.DelStudentLeave_obj += InsuranceCompanyLostCustomer;
        }

        private void InsuranceCompanyLostCustomer(int studentId)
        {
            Console.WriteLine("customer left:{0} from insurance company",studentId );
        }
    }
}

```

1. בית הספר רשימת תלמידים.
2. הגדרי מחלקת בית ספר ומחלקת תלמיד.

הגדרי [טיפוס] נציג מתאים להודעה של התלמיד כאשר הוא נעדר.

מחלקה תלמיד מכילה אובייקט-נציג, המודיע על העדרות.

בית הספר מכיל:

1. מערכת של תלמידים
2. פונקציה לביצוע בעת העדרות. [הודעה על רישום ביום]
3. פונקציה לשירות תלמיד בבית הספר .
4. פונקציה לשירות תלמידים בבית הספר .
5. בתכנית הראשית צרי מערכת המכיל 10 תלמידים.

במחלקה הראשית צרי אובייקט בית ספר.

הווסף את כל התלמידים לבית הספר.

הפעלי את הودעת ההעדרות של 2 תלמידים

.2

בבית תוכנה קיימים מספר אגפים. בכל אגף רשימת עובדים.

כאשר עובד נעדר מהעבודה, בכל אגף יש פעילות מסוימת שיש לבצע. באגף פיתוח נרשם חיסור ביום העובדים, באגף הניקיון יש להזמין מחברת הנקיון מנקה מלא-מקום, באגף הנהלה – יש לשלוח הודעה מייל לכל העובדים על המנהל שחרור והראות התנהלות בלבדיו. באגף קשרי חוץ יש לשלוח הודעה SMS לכל הלוקחות.

הגדרי [טיפוס] נציג מתאים

כתב מחלקת תלמיד המכילה אובייקט-נציג, המודיע על העדרות.

כתב מחלקת אגף. האגף מכיל:

מערכת של עובדים

fonktsiya libitzu' be'at ha'udrot.

fonktsiya leshior ubod la'agaf.

fonktsiya leshior murek ubodim la'agaf.

בתכנית הראשית צרי מערכת המכיל 10 עובדים.

צרי שלושה אובייקטים מסווג אגף. לכל אחד מהם שייכי חלק מהעובדים. יהיו עובדים המשווים לשני אגפים.



207

- extension:  $\rightarrow \text{non}$

$\rightarrow$  non static by default ch. if we use  $\text{static}$   
 $\text{, static} \rightarrow$  non static

non static need to be null for  $\text{prop}$   $\rightarrow$   $\text{Q3116015}$

-  $\text{public}$   $\text{prop}$  your type must be  $\text{as}$  and  
cannot declare  $\text{as}$

more  $\text{as}$  is error like  $\text{C#}$  -  $\text{.NET}$

-  $\text{public}$   $\text{prop}$  extensions,  $\text{Get...}$  is inaccessible due to  
 $\text{private}$   $\text{as}$   $\text{for} \rightarrow$   $\text{non}$

- implicitly typed variables must be initialized  
 $\text{initial value}$

,  $\text{priv}$  is public  $\text{etc}$ ,  $\text{as}$   $\text{prop}$   $\text{as}$   $\text{value}$  or  $\text{as}$

If you want  $\text{priv}$   $\text{value}$   $\text{line} \rightarrow$  be assigned to  $\text{as}$



## הMANDRICK זמינה ל - LINQ.

### Language integrated query

בסיומו של מאמר זה אתם תדעו לכתוב קוד QINQ בשני התחבירים שלו ותבינו למה כדאי לעבוד עם LINQ ומה זה נותן לנו בתור מפתחים.

#### חומר ענייניים:

- מה זה LINQ ומה נותן לנו השימוש בו.
- לפנ LINQ.

#### Predicates & Accounts

var keyword

Anonymous Types

Extension Methods

Lambda Expressions

כתיבת קוד שمدמה QINQ

LINQ תחביר

Extension Methods

Syntactic sugar

PLINQ

הסדר שנראה לי:

1. נציג.
2. נציגים מובנים

#### מה זה QINQ ומה נותן לנו השימוש בו.

הweeney המרכז מתחורי השימוש ב - LINQ הוא להפסיק לכתוב "איך" ולהתחיל לכתוב "מה", כלומר כשאנו כותבים קוד אנחנו רגילים לכתוב קוד שהמשמעות שלו הם הראות למחשב איך לבצע את מה שאנחנו רוצים לעשות, למשל אם אנחנו רוצים למצוא קוד שמוצא את כל המספרים הזוגיים במערך נכתב מן הסתם קוד זהה:

```
IEnumerable<int> arr = Enumerable.Range(1, 100).ToArray();
```

## Summary

```
List<int> list = new List<int>();
foreach (int item in arr)
{
    if (item % 2 == 0)
    {
        list.Add(item);
    }
}
```

למעשה כתבנו לולאה שרצה על כל המספרים ובודקת האם המספר מתחלק ב - 2, במידה וכן גוסיף אותו לרשימה החדשה – וכך שאפשר לראותו אנחנו מפרטים בדיק צעד אחר צעד איך אנחנו רוצים לבצע את הקוד שלנו, לעומת זאת כשכתבו LINQ נרצה לכתוב "מה" אנחנו רוצים לעשות ולא מעוניין אותנו שלבי הביצוע ונכתב קוד מהסוג זהה:

```
List<int> list = arr.Where(item => item % 2 == 0).ToList();
```

마וחר יותר נבין את המשמעות של כל-tag ופסיק מהמשפט, אבל גם בלי להבין את LINQ אפשר לראות שאפשר לנו מפעלים שאלתית חיתוך (Where) על המערך וכותבים שאנחנו רוצים רק את המספרים שמתחלקים בשניים  $item \% 2 == 0$ .

## לפנינו LINQ

### Predicates & Accounts

מה זה פדריקט?

פרדיקטים הם פשוט Delegates ג'נריים שהוגדרו ב- .NET 2.0, בצורה הבאה:

Public delegate bool Predicate (T obj)

הרעיון הוא שבפרדיקטים מקבלים אובייקט מסוים ומחזירים true אם הוא עונה על>Kriterion מסוים. זה לא נשמע מרגש יותר מדי, אבל תשתיית 2.0.NET מאפשרת לעשות שימוש בפרדיקטים האלה כשהיא מתייחסת להגדר את הפונקציה הבאה:

```
Private bool IsSmall (int num) { return (num < 50) ; }
```

ואז, אפשר לעשות חיפוש על המערך, או יותר מזה – להציג מערך שמכיל את האובייקטים שעונים על הקритריון שבחרנו. האפשרות השימוש בMETHODS הסטטיות FindAll או Array.FindAll. למשל, אם המערך שלנו נקרא array, אנחנו יכולים למצוא את כל המספרים הקטנים בו בצורה זו:

```
Int [ ] results = Array.FindAll (array, IsSmall) ;
```

הטריק הזה עובד גם על ה-List הגנרי. למשל, אם הגדרנו רשימות ג'נריות של מספרים:

```
List list = new List();
```

אנו יכולים להחזיר את כל המספרים הקטנים ב-list לערך רשימה חדשה בצורה זו:

```
List result = list.FindAll(IsSmall);
```

שימוש לב Ci המתודות שמשתמשות בפרדיקטים על מערך הן מתודות סטטיות של המחלקה Array, בעוד שהמתודות שמשתמשות בפרדיקטים על רשימות מופעלות מטור Instance ספציפי של רשימה. זה לא באמת חשוב, אבל בכל זאת – שתוודו.

מה עוד אפשר לעשות עם פרדייקטים?

מתודות נוספות של מערך ורשימה שמשתמשות בפרדיקטים הן: - Exists, Find, FindAll, FindLast, TrueForAll. עבור רשימה, קיימת מתודה נוספת נסفة שמשתמשת בפרדיקטים בשם All. עבור מערך, ניתן להשתמש בפרדיקטים גם בשבייל להציג אינדקסים של איברים עם המתודות `FindIndex` ו-`FindLastIndex`.

*Converter - Action*  
שתי מתודות她们 וריאציה קלה על כל נושא הפרדייקטים, והן מזוכרות כאן רק כי הן ממש, אבל ממש שימושית. הראשונה היא המתודה `ForEach`. היא לא משתמשת ב-delegate מסווג פרדייקט, אלא ב-delegate מסווג `Action`. ההגדרה שלו היא:

Public delegate void Action (T obj)

המתודה הזו מבצעת פעולה כלשהי על כל האובייקטים בתוך מערך או רשימה: למשל, מוציאה אותם לפלט כלשהו.

דוגמא:

```
List<String> names = new List<String>();
names.Add("Bruce");
names.Add("Alfred");
names.Add("Tim");
names.Add("Richard");

// Display the contents of the list using the Print method.
names.ForEach(Print);

// The following demonstrates the anonymous method feature of C#
// to display the contents of the list to the console.
names.ForEach(delegate(String name)
{
    Console.WriteLine(name);
});
```

```

        }

    private static void Print(string s)
    {
        Console.WriteLine(s);
    }
}

```

אבל המתודה הנפלהה באמת היא All Convert – מתודה שתיהה שימושית במיוחד לכל המאבחןים אי – שם בחוץ שרצו להמיר, בצורה קלה, בין מערכיהם של char-ים למערכיהם של byte-ים. אם תבצעו את המתודה הזאת על רשיימה ג'נרט מסוג מסוים, תקבלו בחזרה רשיימה ג'נרט מסוג אחר. לשם כך, בסך הכל תצטרכו לספק למתודה .Converter מסוג ConvertAll מתודה מסוג Converter היא כזו:

Public delegate TOutput Converter (TInput input)

כלומר, אם יש לנו רשיימה של int-ים, אנחנו רוצים אותה לרשיימה של מחרוזות, אנחנו יכולים ליצור את המתודה הבא:

Public double IntToString (int num) {return num.ToString(); }

ואז לקחת רשיימה של int-ים (גניך שקוראים לה list) ולבצע:

List result = list.ConvertAll (IntToString);

אנחנו יכולים להשתמש במתודה הזאת גם עבור מערכים, אבל אנחנו צריכים, במקרה זה, לציין גם את סוג מערכת הקלט וגם את סוג הפלט, מכיוון שאנחנו מפעילים את המתודה ממחלקת סטטית – ולכן אין לה אף מידע על מקורו שלנו. למשל, אם יש לנו מערך של int-ים בשם arr:

String[] result = Array.ConvertAll (arr, IntToString);

כמו כן, אתם לא חייבים להסתפק ב-casting פשוט. אתם יכולים להשתמש במתודת המרה, למשל, בשיביל להחזיר מערך שמכיל את השורש הריבועי של כל האיברים במערך המקורי. בסך הכל, כמו ForEach, גם All ConvertAll שימושי על מנת לעשות פעולות על מערך גדול במיוחד.

var keyword

אפשר להשתמש ב – var במקום לכתוב את ה – type המלא, זה יכול לחסוך כתיבה לדוגמא:

var list = new List<string>;

אין צורך לכתוב הצד שמאל את טיפוס המשתנה היה שכתבנו הצד ימין, חשוב לשים לב שלא מדובר בטיפוס מסווג חדש אלא זה בסך הכל טרייק של הקומפיילר שמעתיק אותו לצד ימין לצד שמאל, וכך:

var list = null;

```
static void f(var num)
```

מכיוון שהקומpileר לא יודע באיזה טיפוס מדובר.

לכוארה זה נראה כמו יותר אבל למעשה צריך את ה - var לשימוש ה - Anonymous Types כדי שנראה בהמשך ובנוסף כשאתם מגדירים טיפוס מורכב (כמו Dictionary עם כל מיני טיפוסים) ה - var יכול לפחות את קריאות הקוד.

### Anonymous Types

יש לנו את היכולת לייצר בזמן כתיבת הקוד טיפוסים חדשים מבלתי להגדיר אותם במפורש, לדוגמה:

```
var pi = new
{
    Name = "Shlomo",
    Age = 25
};
Console.WriteLine("Name: {0}, Age: {1}", pi.Name, pi.Age);
```

למעשה מאחורי הקלעים מוגדר בזמן קומpileציה מחלוקת חדשה עם שני המאפיינים שהוגדרו (הם לקריאה בלבד).

למעשה הסיבה שהמציאו את var היא מכיוון שבולדיה اي אפשר היה להגדיר ולהשתמש ב - Anonymous Types, כמובן שאפשר היה להשתמש ב - object (מה שנחנו צריכים לעשות במידה ונרצה לשולח את המשתנה למוגדר אחר) אבל במקרה זהה הגישה למאפיינים היא רק באמצעות reflection שזה לא עיל כמושב. בנוסף אפשר לכתוב הקוד הזה:

```
class MyClass
{
    public int Age { get; set; }
    public string Name { get; set; }
}

class YourClass
{
    public float Salary { get; set; }
}

MyClass mc1 = new MyClass();
YourClass yc1 = new YourClass();
```

```
var pil = new
{
    mc1.Age,
    mc1.Name,
    yc1.Salary
};
```

כמו שאפשר לראות אפשר להגדיר את מאפייני הטיפוס האונימי מבלתי לתת להם שם במפורש והם יקבלו את השם מהמאפיין שהם לוקחים את הערך שלו.



לכורה כל הסיפור זהה נראה מיותר – בהמשך כשןלמד LINQ נראה שלפעמים אין ברירה וחיבים להשתמש בתחביר זהה.

### Extension Methods

נניח שיש מחלוקת שלא אנחנו כתבנו (כמו string) ואנחנו רוצים להרחיב אותה (כלומר להוסיף לה פונקציות) וכך נקבע לרשות ממנה (במקרה של string אי אפשר – זהו דוגמא) אבל במקרה הזה בכל מקום באפליקציה שלנו נדרש להשתמש בטיפוס שלנו וקומו מה קורה אם באמצעות רוצים להוסיף פונקציה ל – string שכאמר או אפשר לרשות ממנה, בשביל זה יש את ה – Extension Methods, לדוגמא:

```
static class StringExtension
{
    public static intToInt(this string s)
    {
        return int.Parse(s);
    }
}

int num = Console.ReadLine().ToInt();
```

וכעת יוכל לכתוב

מכיון ש – מחזיר אובייקט מסוג string וכל אובייקט מסוג string יוכל מתודה חדשה שנគראת `.ToInt`.

זה מאד נוח ונוטן לנו את יכולת להוסיף פונקציונליות לאובייקטים שלא אנחנו כתבנו ומשתמשים בה ברחבי המערכת.

כדי שפונקציה תתווסף לטיפוס מסוים צריך לעשות את הדברים הבאים:

- להגדיר מחלוקת סטטית.
- להגדיר מתודה סטטית שהפרמטר הראשון שלה מוגדר עם `this` [כל המופעים של הטיפוס שלו (הפרמטר הראשון) יקבלו את המתודה החדשה)]
- namespace שבו הוגדרה המתודה החדשה צריכה להיות מוגדרת בקובץ שבו רוצים להשתמש במתודה החדשה, ככלمر אם המחלוקת הסטטית תוגדר ב – `a` namespace בשם `a` ובקובץ אחר בफונקציה לא יהיה `using` ל – `a` המתודה החדשה לא תופיע, כדי לגרום לכך שלא יצטרכו באמצעות `using` שונים אפשר להגדיר namespace בשם `MyExtension` ותחתיו להגדיר את כל המחלקות החדשות, אופציה נוספת היא לתת את ה – `namespace` של המערכת למשל – אם מדובר בתוספות ל – `string` לעטוף את המחלוקת ב – `string` של namespace וכן הלאה.

### Lambda Expressions

בגרסאות הראשונות של השפה במידה והשתמשנו ב – `delegate` היינו צריכים לכתוב מתודה מיוחדת לשילוח, לדוגמא

```
static void Print(int number)
```

```

    {
        Console.WriteLine(number);
    }

Action<int> action = new Action<int>(Print);
action(10);

```

(כמובן שלא היה קיים generic אבל לצורך המחתת נשא ה – delegate נניח שהוא קיים.)  
במידה ורצינו להשתמש ב – delegate של Action היינו צריכים להגיד מתיידר מתודה שעומדת בחთימה ולשלוח אותה כפרמטר ל – ctor של ה – delegate.

ברגשה 2.0 של השפה קיבלנו את המושג של anonymous methods ויכלנו לכתוב קוד זהה:

```

Action<int> action = delegate(int num1)
{
    Console.WriteLine(num1);
};
action(10);

```

הסיבה המרכזית לתמייה ב – anonymous methods היא היכולת להשתמש במשתנים מקומיים של הפונקציה שקוראת ל – delegate לדוגמא:

```

int userNum = int.Parse(Console.ReadLine());
Action<int> action = delegate(int num1)
{
    if (userNum == 10)
        num1++;
    Console.WriteLine(num1);
};
action(10);

```

בגרסאות הקודמות של השפה שימוש במשתנה userNum היה מחיב להגדיר אותו כמשתנה גלובלי של המחלקה לעומת זאת החל מרגשה 2.0 ניתן להשתמש בכל המשתנים המקומיים של הפונקציה שפעילה את אותו delegate

החל מהגרסאות המתקדמות של השפה ניתן לכתוב Lambda Expressions שהו תחביר אחר לו – anonymous methods (אחרי קומפיילציה קיים רק anonymous methods לדוגמא):

```

Action<int> action1 = num1 => Console.WriteLine(num1);
Action<int> action2 = num1 =>
{
    if (userNum == 10)
        num1++;
    Console.WriteLine(num1);
}

```

נפרק את השורה.

```
1. Action<int> action1=
2. num1 =>
3. Console.WriteLine(num1);
```

השורה הראשונה זה כמובן המופיע של ה – delegate,

השורה השנייה מכילה את רשימת הפרמטרים, במידה ויש יותר מפרמטר אחד צריך לעטוף אותם בסוגרים, לאחר רשימת הפרמטרים יש הסימון המוזר של <=,

השורה השלישית מכילה את הקוד עצמו, במידה ויש רק שורה אחת לא צריך סוגרים מסולסלים ואם ה – delegate אמרו להחזיר ערך לא צריך לכתוב את המילה return, למשל:

```
Func<int, int> mul = num1 => num1 * 10;
int res = mul(50);
```

הגדרנו מופיע של delegate שאמור לקבל מספר ויכפיל אותו ב – 10 ולאחר מכן הפעלנו אותו, כמו שאפשר לשים לב אין צורך לכתוב את המילה return.

### עוד על ביטוי למבدا – lambda expression

שיטות אונימיות שהכרנו כבר אפשרו לנו לכתוב קטעים קצריים של קוד בעת יצירה או הוספה של נציגים.

ביטוי הלמבדא חוסך מהמתכנת הקלדתו קוד מיותר, לכארה, שהמהדר כבר יכול להסיק בעצמו.

למעשה הביטוי היה יכול להחליף את השיטות האונימיות לגמרי.

תזכורת לגבי השיטה האונימית:

```
Public delegate void myDel(Int a)

myDel del=new delegate(function);
...
private int function(int i)
{
    Return x+1;
}
```

והנה השיטה האונימית שבמקום זה :

```
myDel del= new delegate(int x){return x+1;};
```

תהליך ההמרה בין השיטה האונימית לביטוי הלמבדא:

מתבצע למשעה על ידי השמטה כל המידע שההדר יכול להסיק בעצמו:

```
myDel del= delegate(int x){return x+1;};
```

בהצבה לנציג זה תמיד יהיה רק נציג, שכן אפשר להשתמש את המילה delegate.

```
myDel del= (int x)=> {return x+1;};
```

סוגי הפרמטרים ידועים כבר על פי סוג הנציג. لكن אפשר לוותר גם עליהם:

```
myDel del= (x)=>{return x+1;};
```

הפרמטר שנשלח מקבל מובן של 'holeך אל' .  
כשיש רק פרמטר אחד ניתן לוותר גם על הסוגרים.

```
myDel del= x =>{return x+1;};
```

```
myDel del= x => { x+1};
```

```
myDel del= x => x+1 ;
```

## כתבת קוד שמודה LINQ

נניח שאנו רוצים לכתוב מתודה חדשה ש偶像עת לפילטר List של int לפי תנאי שהוא מקבלת פרמטר, יוכל לכתוב מתודה כזאת:

```
static List<int> Filter(List<int> source, Predicate<int> prediacte)
{
    var newList = new List<int>();
    foreach (var item in source)
    {
        if (prediacte(item))
            newList.Add(item);
    }
    return newList;
}
```

והשימוש בה יהיה כך:

```
List<int> list = Enumerable.Range(0, 50).ToList();
var evenList = Filter(list, x => x % 2 == 0);
```

כמו שאפשר לראות השתמשנו במתודה ושלחנו אליה את האובייקט המקורי ומימוש ל Lambda Expressions לא – Prediacte (אמורה לקבל פונקציה שתקבל T ותחזר bool)

אם נרצה להיות יותר חכמים נגידיר את Filter כ – Extension Methods ובקשה זהה נכתב כך:

```
static List<int> Filter(this List<int> source, Predicate<int> prediacte)
{
//.....
}
var evenList = list.Filter(x => x % 2 == 0);
```

זה בעצם מה שקרה – הוגדרו מספר רב של Extension Methods לאובייקט:



`IEnumerable<T>`

ולכן כל האוסףים יכולים להשתמש במגוון רחב של מתודות חדשות. לדוגמה

```
if (list.Any(x => x == 10)) { }
```

כדי לבדוק האם יש ברשימה את הערך 10. דוגמא נוספת:

```
var evenIdsAsString = list.Where(x => x % 2 == 0)
    .Select<int, string>(x => x.ToString());
```

מוציא מהרשימה את כל המספרים הזוגיים וממיר אותם למחרוזת, מה שב證ו של דבר יחזיר לנו רשימה חדשה של מחרוזות עם מספרים זוגיים.

ובעזרת רישימת המתודות ניתן לכתוב קוד נקי ופשטוט יותר.

## LINQ

### Extension Methods

את Extension Methods ראיינו כבר בחלק הקודם, נראה עוד דוגמה אחת יחסית מורכבת, נניח שאנו רוצים לחפש את כל התיקיות שיש להם מעל 10 קבצים ולפחות קובץ אחד גדול יותר מ – 20 KB, בנוסף נרצה ליצור מחלקה משלהן ולא להחזיר אובייקט של `DirectoryInfo` של `Info`.  
נגיד מחלקה שנקראת כך:

```
class My DirectoryInfo
{
    public int FileNumbers { get; set; }
    public stringDirectoryName { get; set; }
}

var x1=Directory.GetDirectories
("path", "", SearchOption.AllDirectories)
    .Select(x => new DirectoryInfo(x))
    .Where(x =>
    {
        var files = x.GetFiles();
        return files.Length >= 10 && files.Any(fi => fi.Length > 10);
    })
    .Select(x => new My DirectoryInfo()
    {
        DirectoryName = x.Name,
        FileNumbers = x.GetFiles().Length
    });

```

במידה ולא היה לנו מחלקה שנקראת `My DirectoryInfo` יכולנו להחזיר `Anonymous Types`

כדי להקל על הכתיבה (לפעמים) אפשר לכתוב (חלק) שאלות LINQ בתחביר אחר. נתחיל עם משהו פשוט ויחסית, מציאת כל העובדים שהמשכורת שלהם קטנה מ – 2000

```
class Employee
{
    public string Name { get; set; }
    public double Salary { get; set; }
}
```

בתחביר הקודם הימנו כתובים קוד צזה:

```
List<Employee> listEmp = GetList();
var lowSalary = listEmp.Where(x => x.Salary < 2000);
```

בתחביר החדש ניתן לכתוב כך

```
var lowSalary = from n in listEmp
where n.Salary < 2000
select n;
```

את השאלות המורכבת (של הקבצים) ניתן לתרגם זהה

```
var directories = from n in Directory.GetDirectories("path", "", 
SearchOption.AllDirectories)
let di = new DirectoryInfo(n)
let files = di.GetFiles()
where files.Length >= 10 && files.Any(x => x.Length > 10)
select new MyDirectoryInfo()
{
   DirectoryName = di.Name,
    FileNumbers = files.Length
};
```

### Linq – בתחביר שפת שאלות :

לפני הבנת תחביר השאלות צריך לדעת:

טיפוס אונונימי

מאפיינים אוטומטיים

### PLINQ

PLINQ הוא `parallel linq` (רק מגרסה 4.0 ומעלה) מאפשר לנו בקלות לבצע את שאלות ה – LINQ בצורה פרללית (כלומר כמה threads שיבצעו את העבודה) אני לא ארכחיב בנושא מכיוון שהוא נושא בפני עצמו, אני רק אציג דוגמא קטנה

```
List<int> list = Enumerable.Range(1, 500 00000).ToList();
for (int i = 0; i < 4; i++)
{
    Stopwatch sw = Stopwatch.StartNew();
    var count = list.Count(x => IsPrime(x));
```

```

        sw.Stop();
        Console.WriteLine(sw.ElapsedMilliseconds);
        sw = Stopwatch.StartNew();
        count = list.AsParallel().Count(x => IsPrime(x));
        sw.Stop();
        Console.WriteLine(sw.ElapsedMilliseconds);
        Console.WriteLine("=====");
    }

static bool IsPrime(int num)
{
    for (int i = num - 1; i > 1; i++)
    {
        if (num % i == 0)
            return false;
    }
    return true;
}

```

בדוגמא הזאת אנחנו מפעילים על כל איבר את הפונקציה IsPrime (שMethodImpl בזורה שייקח לה זמן להתבצע) ונוחנו סוכמים כמה מספרים ראשוניים יש במערך.

הדוגמה הראשונה משתמשת ב - LINQ רגיל לעומת הדוגמא השנייה שהיא משתמשת ב - PLINQ בעזרתו קרייה ל - AsParallel.

הקוד הפרללי רץ במהירות כפולה (אצלו במחשב) מהקוד הרגיל.

#### סיכום

LINQ הוא כלי מיוחד שבעזרתו אפשר לשנות את הרגלי הכתיבה שלנו ולכתוב קוד קרייא יותר ומהיר יותר גם מבחינת הביצועים וגם מבחינת מהירות הכתיבה.

### Person.cs

```
namespace UseMyExtention
{
    class Person
    {
        public int Code { get; set; }
        public string LastName { get; set; }= "defaltName";
        public string Address { get; set; }

        int[] Marks = new int[100];

        public Person()
        {
        }
    }
}
```



### Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ExtentionMethodsAndLinq;

namespace UseMyExtention
{
    //1 - הגדרת מבנה הנציג
    //2 - יצירת מופע מסוג הנציג
    //3 - כתיבת פונקציה התואמת למציג
    //4 - הצבת הפונקציה באובייקט הנציג
    //5 - הפעלת הנציג

    class Program
    {
        static void Main(string[] args)
        {

            //use the extention
            string str1 = "dddd";

            //var keyword
            string s1 = "aaaa";
            var s2 = "aaa";
            var len = s2.Length;
            var person = new Person();
            // var a;//implicitly-type variable must be initialized;

            var x = str1.GetFirstAndLast();
            //anonymouse type:
            var anonymouseValueTyped = new { Key = 1, Name = "Michal", LastName = "Levi" };
            // anonymouseValueTyped.LastName = "fff";//
            // all Uri components must be strings
            // var anonymousePerson = new { person.LastName, person.Address, 900 };

            //tipos האנונימי צריך להיות שם - מפורש או מרמז
            List<int> myInts = new List<int> { 3000, 7, 6, 800, 10, 90, 100 };
            myInts.Where(new Func<int, bool>(newFunc));

            myInts.MyWhere(new Func<int, bool>(newFunc));
            //או בקיצור באמצעות למברא
            List<int> results = myInts.MyWhere(x1 => x1 > 100).ToList();
            IEnumerable<int> results1 = myInts.MyWhere(x1 => x1 > 100);
            //List<int> results1 = myInts.MyWhere(x1 => x1 > 100); //
            //mywhere שאלתה הפונקציה mywhere
            //רשות להזאת
            var res = myInts.MyWhere(x1 => x1 > 100).ToList();

            foreach (var item in results1)
            {
                Console.WriteLine(item);
            }
            //=====
            myInts.Where(i => i > 10);
```



## Program.cs

```
        Console.Read();  
    }
```

שלב 1 - נكتب על ידי מיקרוסופט//

```
//public delegate TResult Func<in T, out TResult>(T arg);
```

```
private static bool newFunc(int arg)  
{  
    return arg > 100;  
}
```

```
public delegate int del1(string s);  
//1
```

כערך מוחזר T2 -ו T1 הגדרת נציג גברי שמקבל/T1 value;//

וכפרמטר

שלב 2 + שלב //4

```
delG<bool, string> delObj = new delG<bool, string>(nameOffunc);
```

ניתן להוריד את המובן מלאיו - תמיד הטיפוס שנוצר יהיה;//

בדיקות מאותו מבנה

שלב //3

```
private static bool nameOffunc(bool value)  
{  
    throw new NotImplementedException();  
}
```

```
}
```



### Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExtentionMethodsAndLinq
{

    class Program
    {
        static void Main(string[] args)
        {
            string s1 = "Father";
            var v=s1.GetFirstAndLast();
        }
    }

    public static class Extentions
    {
        public static char[] GetFirstAndLast (this string str)
        {
            char[] arr = new char[2];

            if (str==null || str.Length < 2)
                return null;
            {
                arr[1] = str.Last();
                arr[0] = str.First();
            }
            return arr;
        }

        /// <summary>
        /// של מיקרוסופט where -פונקציה לסינון המחקה את ה
        /// </summary>
        /// <typeparam name="TSource"></typeparam>
        /// <param name="source">the list to enumerate in thre func - הרשימה שאזורה אני רוצח
        /// </param>
        /// <param name="myPredicate">הנציג או ביטוי הלמבדא שמכיל בתוכו את התנאי לסינון</param>
        /// <returns>שמציביע על השאלה [IEnumerable] משתנה מטיפוס אינטראפייס של רשימה</returns>
        public static IEnumerable<TSource> MyWhere<TSource>(this IEnumerable<TSource> source,
Func<TSource, bool> myPredicate)
        {
            List<TSource> list1 = new List<TSource>();
            foreach (TSource item in source )
            {
                if (myPredicate(item))//באמצעות הפרדיקיט
                    list1.Add(item);
            }
            return list1;
        }

        //public static bool
    }
}
```



## שיעור 3 – פונקציות הרחבה וLINQ - דוגמאות מהשיעור – 2

Soltion : ExtentionMethodsAndLinq

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExtentionMethodsAndLinq
{
    class Program
    {
        static void Main(string[] args)
        {
            string s1 = "Father";
            var v=s1.GetFirstAndLast();
        }
    }

    public static class Extentions
    {
        public static char[] GetFirstAndLast (this string str)
        {
            char[] arr = new char[2];

            if (str==null || str.Length < 2)
                return null;
            {
                arr[1] = str.Last();
                arr[0] = str.First();
            }
            return arr;
        }

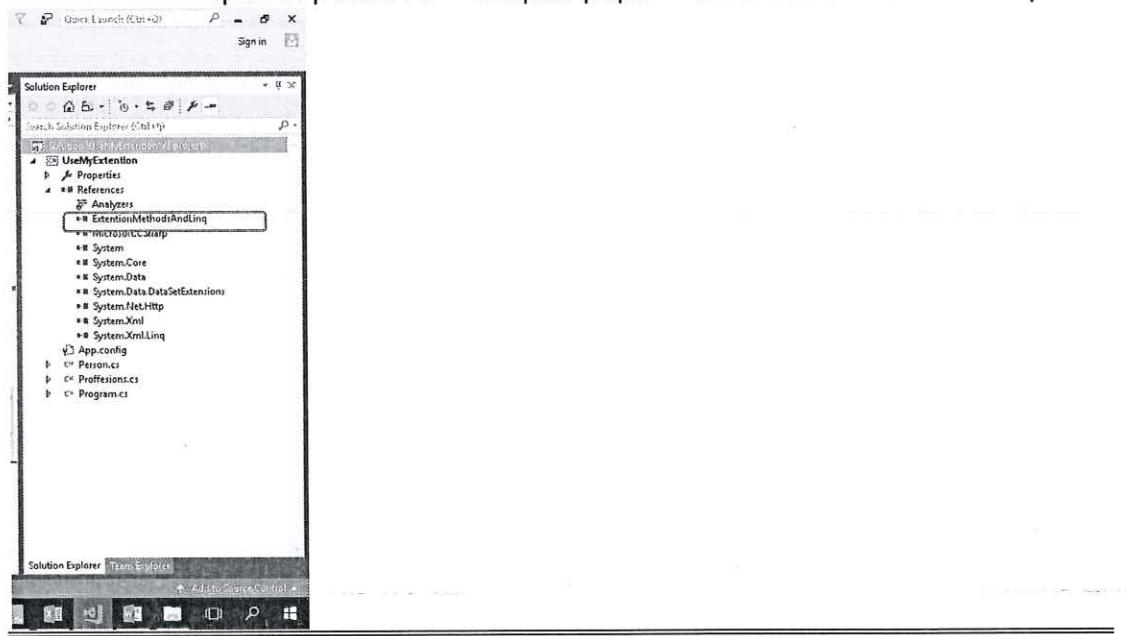
        /// <summary>
        /// מיקורוסופט של ה- את המהקה לסנון פונקציה
        /// </summary>
        /// <typeparam name="TSource"></typeparam>
        /// <param name="source">the list to enumerate in thre func -
        הרשימה - אני שואותה
        התנאי את בתוכו شامل הלמבדא ביטוי או הנציג></param>
        /// <param name="myPredicate">bool</param>
        על שמאזביע [ I Enumerable ] רשימה של אינטראיס מטיפוס משתנה></returns>
        והשאלתה
        public static IEnumerable<TSource> MyWhere<TSource>(this
        IEnumerable<TSource> source, Func<TSource, bool> myPredicate)
        {
            List<TSource> list1 = new List<TSource>();
            foreach (TSource item in source )
            {
                if (myPredicate(item))
                    list1.Add(item);
            }
            return list1;
        }

        //public static bool
    }
}
```

}

}

בפתרון השני - הוספנו הפניה לאסמבלי – הקובץ המקבול – של הפתרון הראשון.



Solution: UseMyExtention

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ExtensionMethodsAndLinq;

namespace UseMyExtention
{
    //הנציג מבנה הגדרת -
    //1 - הנציג מסוג מופיע יצרה
    //2 - למציג התואמת פונקציה כתיבה
    //3 - הנציג באובייקט הפונקציה האצתה
    //4 - הנציג הפעלה
    //5 - 

    class Program
    {
        static List<Person> ListPersons = new List<Person>();
        public List<Person> Ex_1()
        {
            return ListPersons.Where(p => p.TelephonNumbers.Count() > 1).ToList();
        }

        static void Main(string[] args)
        {
            String nameProfessional = "יהודה";

            var result_a = ListPersons.Where(p => p.LastName == nameProfessional)
                .Select(s => new { s.LastName,
s.ProffesionList }) ;
            var result_a_querySyntax = from p in ListPersons
                where p.LastName == nameProfessional

```

```

                select new { p.LastName,
p.ProffesionList };

        var result2 = (from pers in ListPersons
                      where pers.Address.EndsWith("נ")
                      select pers.Code ).ToList();

        var result_1 = ListPersons.Where(p => p.TelephonNumbers.Count() >
1)
            .Select(a => new { a.LastName, Tel1 = a.TelephonNumbers[0] })
            .ToList();
        result_1.First(a => a.Tel1.StartsWith("08"));

        //use the extention
        string str1 = "dddd";

        //var keyword
        string s1 = "aaaa";
        var s2 = "aaa";
        var len = s2.Length;
        var person = new Person();
        //    var a;//implicitly-type variable must be initialized;

        var x = str1.GetFirstAndLast();
        //anonymouse type:
        var anonymouseValueTyped = new { Key = 1, Name = "Michal", LastName
= "Levi" };
        // anonymouseValueTyped.LastName = "fff";//טיפוס
        // var anonymousePerson = new { person.LastName, person.Address ,
        מrownז או מפורש - שם להיות ציריך האונימי לטיפוס שמכנים עםן לכל
        900 };

        List<int> myInts = new List<int> { 3000, 7, 6, 800, 10, 90, 100 };
        myInts.Where(new Func<int, bool>(newFunc));

        myInts.MyWhere(new Func<int, bool>(newFunc));
        //例外 באמצעות בקיצור או
        List<int> results = myInts.MyWhere(x1 => x1 > 100).ToList();
        IEnumerable<int> results1 = myInts.MyWhere(x1 => x1 > 100);
        //List<int> results1 = myInts.MyWhere(x1 => x1 > 100) ; //פונקציה
mywhere רשיימה ולא שאלת מהזירה
        var res = myInts.MyWhere(x1 => x1 > 100).ToList(); ;

        foreach (var item in results1)
        {
            Console.WriteLine(item);
        }
        //=====
        myInts.Where(i => i > 10);

        Console.Read();
    }

    מיקורסופט ידי על נכתב - 1 שלב //
    //public delegate TResult Func<in T, out TResult>(T arg);

private static bool newFunc(int arg)
{
    return arg > 100;
}

```

```

public delegate int del1(string s);
// שלב 1
public delegate T1 delG<T1,T2>(T1 value); // T2 -נו T1 שמקבל נרי'ג נציג הגדרת/
וכפראטר מוחזר כערך
// שלב 2 + 4
delG<bool, string> delObj = new delG<bool, string>(nameOffunc);
delG<bool, string> delObj1 = nameOffunc ;// ניתן
תמיד - מאליו המובן את להוריד ניתן
מבנה מאותו בדיקת היה שנוצר הטיפוס
// שלב 3
private static bool nameOffunc(bool value)
{
    throw new NotImplementedException();
}

}

```

---

```

using System.Collections.Generic;

namespace UseMyExtention
{
    class Person
    {
        public int Code { get; set; }
        public string LastName { get; set; }= "defaltName";
        public string Address { get; set; }
        public List<Proffesion> ProffessionList { get; set; }
        public string [] TelephonNumbers { get; set; }

        int[] Marks = new int[100];

        public Person()
        {
        }
    }
}

```

---

```

namespace UseMyExtention
{
    public class Proffesion
    {
    }
}

```

---

# C# EX

e r c

i s e

c

## Linq to objects

1. בני שלוש מחלקות:

a. בעלי מקצוע – קוד, שם פרטי, משפחה, טלפון, וטלפון נייד

d. מקצועות. – קוד מקצוע, שם מקצוע

c. מקצועות לבועל מקצוע – קוד בעל מקצוע, קוד מקצוע

2. הגדרי בתכנית הראשית רשיימה של כל אחת מהמחלקות, ואתה לאי אותה בנתונים.

3. כתבי שאלתה:

a. שתקבל קוד בעל מקצוע ותחזיר את שמו עם רשיימת המקצועות שלו

b. שתחזר את כל בעלי המקצוע המתאימים לקוד מקצוע שהתקבל כפרמטר

c. שתחזר את כל המקצועות שיש להם בעל מקצוע

d. שתחזר את כל המקצועות שיש להם בעל מקצוע

e. שתחזר את כל בעלי המקצוע שעובדים ביותר מממקצוע אחד.

f. שתחזר כמה בעלי מקצוע עובדים בכל מקצוע.

בצלחה !



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace LinqToObjects
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13
14            Subject doctor = new Subject() { Code = 3, NameSubject =
15                "doctor" };
16            Subject plumber = new Subject() { Code = 2, NameSubject =
17                "plumber" };
18            Subject teacher = new Subject() { Code = 1, NameSubject =
19                "teacher" };
20            Subject principal = new Subject() { Code = 6, NameSubject =
21                "principal" };
22            List<Subject> Subjects = new List<Subject>();
23            List<Profession> Professions = new List<Profession>()
24            {
25                new Profession()
26                    {Code=3,FirstName="Aharon",LastName="Levi",Phone="065879828",SubjectsList=new List<Subject>(){teacher,principal,plumber } },
27                new Profession()
28                    {Code=2,FirstName="Moshe",LastName="Choen",Phone="0678225830",SubjectsList=new List<Subject>(){plumber,principal } }
29            };
30            int CodeProfessional = 3;
31            var result_a = Professions.Where(p => p.Code == CodeProfessional)
32                .Select(s => new { s.FirstName, s.LastName,
33                    s.SubjectsList }).First();
34            //var result_a =from p in Professions
35            // where p.Code==CodeProfessional
36            // select p.Code ).ToList();
37
38            Console.WriteLine("a");
39            Console.WriteLine(result_a.FirstName + " " + result_a.LastName);
40            foreach (var item in result_a.SubjectsList)
41            {
42                Console.Write(item.NameSubject + " ");
43            }
44            Console.WriteLine();
45            int CodeSubject = 2;
46
47            Console.WriteLine("\n b");
48            var result_b = Professions.Where(p => p.SubjectsList.Any(s =>
49                s.Code == CodeSubject)).ToList();
50            foreach (var item in result_b)
51            {
52                Console.Write(item.FirstName + " " + item.LastName + " ");
53            }
54            Console.WriteLine("\n c");
55            var result_c = Professions.SelectMany(p => p.SubjectsList).Distinct();
```



```
    ().ToList();
47        foreach (var item in result_c)
48        {
49            Console.Write(item.NameSubject + " ");
50        }
51
52
53        var resualt_e = Professions.Where(p => p.SubjectsList.Count >
54            1).ToList();
54        Console.WriteLine("\n e");
55        foreach (var item in resualt_e)
56        {
57            Console.Write(item.FirstName + " " + item.LastName + " ");
58        }
59
60
61        Console.ReadLine();
62    }
63 }
64 }
65 }
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace LinqToObjects
8 {
9     class Profession
10    {
11        public int Code { get; set; }
12        public string FirstName { get; set; }
13        public string LastName { get; set; }
14        public string Phone { get; set; }
15        public List<Subject> SubjectsList = new List<Subject>();
16
17    }
18 }
19
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace LinqToObjects
8 {
9     class Subject
10    {
11        public int Code { get; set; }
12        public string NameSubject { get; set; }
13    }
14 }
15
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Group2_Overview
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            Factory f = new Factory();
14            f.Address = new Address() { City = "Jerusalem" };
15            f.Header.Address.City = "Jerusalem";
16            Console.WriteLine(f.Header.Address.City);
17            f.Header.Address.City = "sadsad";
18            Console.WriteLine(f.Header.Address.City);
19            Console.ReadLine();
20
21        }
22    }
23 }
24
```



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Group2_Overview
8 {
9     public class Person
10    {
11         public int Id { get; set; }
12         public string Name { get; set; }
13         public Address Address { get; set; }
14         public DateTime BornDate { get; set; }
15         public int NumChildren { get; set; }
16
17         //סעיף 3 - כתיבת פונקציה במבנה המתאים
18         public static bool BigPerson(Person p)
19         {
20             return DateTime.Today.Year - p.BornDate.Year > 12;
21         }
22         public static bool BigFamily(Person p)
23         {
24             return p.NumChildren > 10;
25         }
26 }
```



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Group2_Overview
8  {
9      //1. הצהרה על דלגייט פודשי
10     public delegate bool dCityChanged(string oldValue, string newValue);
11
12     public class Address
13     {
14         //2. הגדרת משתנה מסוגו
15         public dCityChanged CityChanged;
16         private string city;
17
18         public string City
19         {
20             get { return city; }
21             set {
22                 //NULL מונה אם הדלגייט שוניה
23                 if (CityChanged!=null)//אם הדלגייט מלא בכתובת פונקצייה
24                     //מסויימת
25                     //5 שלב
26                     if(CityChanged(city, value))
27                         city = value;
28             }
29         }
30
31     }
32 }
33 }
```



## שלבים לייצור דלגייט

1. הגדרה ( ניתן לשימוש בדלאגיט'ס מובנים)
2. הגדרת משתנה מסווג
3. יצירת פונקציה במבנה המתאים
4. שייר פונקציה למשתנה שהוגדר בשלב 2
5. הפעלת הדלאגיט

## שימושי הדלאגיט:

1. כאשר מחלקה מוכלת אמורה להפעיל פונקציה הנמצאת במחלקה המכילה נוצר בדלאגיט – אם למשל רצים שמנהל יגור בעיר של המפעל ניצור דלאגיט בressAddress ונפעיל אותו בעריה Factory.
2. כאשר פונקציה מקבלת כפרמטר פונקציה נשימוש בדלאגיט.  
במחלקה myList היכלה פונקציה MyWhere מקבלת כפרמטר דלאגיט. הפונקציות כתובות בקורס Person ומופעלות בחויב Main.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Group2_Overview
8 {
9     class Factory
10    {
11        public int Id { get; set; }
12        public string Name { get; set; }
13        public Person Header { get; set; }
14        public Address Address { get; set; }
15        //שלב 3 - כתיבת פונקציה במבנה המתאים
16        private bool CheckAddress(string old, string newC)
17        {
18            return Address.City == newC;
19        }
20        public Factory()
21        {
22            //בכל ייצירה מופע של מפעל יוצרים מופע למנהל המפעל ויוצרים
23            //מוופע לכנתובת שלו
24            // כדי שנוכל לשใช את הפונקציה לאירוע
25            //יתכן דרכי נספנות שכרגט לא ינוסו
26            //שלב 4
27            Header = new Person() { Address = new Address() { CityChanged =
28                CheckAddress } };
29        }
30    }
31 }
```

בכל ייצירה מופע של מפעל יוצרים מופע למנהל המפעל ויוצרים  
מוופע לכנתובת שלו  
כדי שנוכל לשใช את הפונקציה לאירוע  
יתכן דרכי נספנות שכרגט לא ינוסו  
שלב 4



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Group2_Overview
8 {
9
10    //1 שלב
11    public delegate bool MyPredicate(Person p);
12    class MyList
13    {
14
15        //2 - הפקודזיה מגדירה פרמטר מסווג הדלגיט/
16        public static List<Person> MyWhere(List<Person> lp, MyPredicate pre )
17        {
18            List<Person> ln = new List<Person>();
19            foreach (var item in lp)
20            {
21                //3 שלב 5 - הפעלת הדלגיט/
22                if (pre(item))
23                    ln.Add(item);
24            }
25            return ln;
26        }
27    }
28 }
29 }
```



שאלות - נכון או לא נכון?

### על לינק

פונקציית הרחבה ניתן לכתוב לכל טיפוס שהוא מסוג reference type פונקציות הLINK – נכתבו באמצעות פון' הרחבה. מאיזה טיפוס יהיה המשתנים ..queryLondonCustomers123.. לאחר הפעלת כל אחת מהשירות הבאות:

```
Class Customer
{
    Public int Code;
    Public string FullName;
    Public String City;
}

var queryLondonCustomers1 = from cust in customers
                            where cust.City == "London"
                            select cust;

var queryLondonCustomers2 = from cust in customers
                            where cust.City == "London"
                            select cust.City;

var queryLondonCustomers3 = from cust in customers
                            where cust.City == "London"
                            select new {cust.FullName, cust.City} ;

var queryLondonCustomers4 = (from cust in customers
                            where cust.City == "London"
                            select cust).ToString();
```

### על גנרייק

- (1) בניית מחלקה גנרטיב חוסכת שימוש בהמרות
- (2) בעת ייצירת מחלקה גנרטיב חובה להציג על הטיפוס אותו היא מקבלת.
- (3) מחלקה גנרטיב יכולה לקבל טיפוס אחד בלבד.
- (4) פונקציה גנרטיב חייבת להיות מוכלת במחלקה גנרטיב.
- (5) מחלקה/פונקציה גנרטיב תפקד בעזרת המרות כלפי מטה וככלפי מעלה.
- (6) בעת ייצירת אובייקט מטיפוס גנרטיב יש הכרח להציג על הטיפוס רק אם הבנייה חייב לקבל אותו.
- (7) בעת שימוש במחלקות גנרטיב מובנות [ע"י Microsoft] אין צורך להציג על הטיפוס הגנרטיב בעת הגדרת אובייקט.

8) מה עושה הפונקציה הבאה?

```
static void Swap<T>(ref T lhs, ref T rhs)
{
    T temp;
    temp = lhs;
    lhs = rhs;
    rhs = temp;
}
```

כתבו קוד לשילוחה לפונקציה!

### על נציגים – delegates

- (1) פונקציה דומה יותר ל- REF TYPE מאשר ל- VAL TYPE
- (2) נציג הוא TYPE REFERENCE
- (3) נציג יכול לקבל כל כתובות של פונקציה.
- (4) מחלוקת מכילה מכירה את האובייקט המוכל שלו.
- (5) מחלוקת מוכלת מכירה את המחלוקת המכילה שלה.
- (6) באמצעות נציג ניתן להעיבר מידע בין משק המשמש לתווים.
- (7) בכל אובייקט מסווג נציג ניתן להציב עד 10 פונ'
- (8) תמיד ניתן להפעיל את הנציג מכל מקום בהזד.
- (9) בהפעלת הנציג חובה לשלוח לו את הפרמטרים הנדרשים.
- (10) ביצוע הפונקציות המוצבעות על ידי הנציג מתקיים לפי סדר ההצבה של ההפניה לפו' בנציג.
- (11) כל יצירה של טיפוס נציג חדש חייב להיות CPUBLIC
- (12) בהפעלת נציג שמחזיר ערך - חוזר הערך שהזר מהפעלת הפונקציה הראשונה שהופעלה.
- (13) ניתן להגדיר טיפוס-נציג בתוך מחלוקת או מחוץ למחלוקת.
- (14) לא ניתן להסיר פונקציה שהוגדרה לנציג.
- (15) ניתן לדחוס פונקציה שהוגדרה לנציג ע"י האופרטור +=
- (16) לא ניתן לשלוח נציג כפרמטר לפונקציה.
- (17) פונקציה יכולה לקבל נציג כפרמטר רק אם הוא און מכיל TNU

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace Linq_Rev
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             int i=5;
15             int j=7;
16             int t=i.GetValue(j);
17
18             //Linq functions:
19
20             int[] arr = new int[] { 7, 5, 7, 09, 2, 90, 10, 1, 0, -4, 6 };
21             PrintArr(arr); //פה יש המרזה כלפי מעלה, מרווחה
22
23             var greaterthan10 = arr.Where(new Func<int, bool>(myFunc));
24
25             I Enumerable<int> greaterthan11 = arr.Where(item=>item>11).ToList
26             ();
27             int c = arr.Count(i1 => i1 < 0);
28
29         }
30
31         private static bool myFunc(int arg)
32         {
33             return arg > 10;
34         }
35
36         static void PrintArr(Array arr)
37         {
38             foreach (var item in arr)
39             {
40                 Console.WriteLine(item );
41             }
42         }
43
44         static class Ext
45         {
46             //דוגמה לפונקציית הרחבה לכל מי שמחמ את
47             public static int GetValue(this IComparable obj, IComparable obj2 )
48             {
49                 if (obj2.GetType()==obj.GetType())
50                     return obj.CompareTo(obj2);
51                 return 0;
52             }
53         }
54
55         class Library : IEnumerable<Book>
```



```
56     {
57         public IEnumerator<Book> GetEnumerator()
58         {
59             throw new NotImplementedException();
60         }
61
62         IEnumerable IEnumerable.GetEnumerator()
63         {
64             throw new NotImplementedException();
65         }
66     }
67
68
69
70
71 }
```



## מבחן ב C#

## OOP

 חלק א' (34%) עניין או לא (17 מתוך 21)

1. האם `C#` כל הפונקציות חייבות להיות כתובות בתוך מחלקה? \_\_\_\_\_
2. האם `C#` כל האובייקטים מוקצים דינמיות? \_\_\_\_\_
3. האם לכל מחלוקת חייבים לכתוב `ctor`? \_\_\_\_\_
4. האם משתנה `readonly` ניתן לעדכן בתוך `Property`? \_\_\_\_\_
5. האם `const` תופס מקום בזיכרון כמו משתנה? \_\_\_\_\_
6. האם `Property` תופס מקום בזיכרון כמו משתנה? \_\_\_\_\_
7. האם לכל `Property` חיבב להיות `get` ו `set`? \_\_\_\_\_
8. האם ניתן להגדיר "יחסים" (-מצבי) מחלוקת אבסטרקטית? \_\_\_\_\_
9. האם חובה למשם במחלוקת הירושת מחלוקת אבסטרקטית את כל המתודות של המחלוקת האבסטרקטית? \_\_\_\_\_
10. האם חובה למשם במחלוקת הירושת ממשך את כל המתודות של המשך? \_\_\_\_\_
11. האם בעת הקצת אובייקט מחלוקת `Employee` (`Employee` מופעלים לפחות 2 `tors`) מופעלים לפחות 2 `tors`? \_\_\_\_\_
12. האם ניתן לגשת למשנה סטטי מתחור מתודה לא סטטיתית? \_\_\_\_\_
13. האם מספר המופעלים של משתנה סטטי בזיכרון מקביל למספר האובייקטים שהוקצו מהמחלוקת שבה הוא מוגדר? \_\_\_\_\_
14. האם לפי עקרון `Capsulation` נגדיר את משתני המחלוקת בהרשאת `Private`? \_\_\_\_\_
15. האם בבני סטטי מופעל בעת ייצרת אובייקט מהמחלוקת שלו? \_\_\_\_\_
16. האם ב `Flags Enum` ניתן לשמר במשנה אחד כמה ערכיהם? \_\_\_\_\_
17. האם ממשך יכול לרשף ממשך אחר? \_\_\_\_\_
18. האם כאשר נשנה ערך של פרמטר שנשלח כ `ref`, יופיע השינוי גם במשנה של הפונקציה הקוראת? \_\_\_\_\_
19. האם אפשרי לגרום שיוכלו לעבור בולולאת `foreach` על אובייקט מחלוקת שאנו כתבנו? \_\_\_\_\_
20. האם ניתן לנתח מספר מתודות בעלות שם זהה באוטה מחלוקת? \_\_\_\_\_
21. האם כאשר הסתרנו מתודה במחלוקת ירושת עזרת מילת המפתח `New`, היא תופעל רק במקרה האובייקט וגם המצביע "הוא" מהחלוקת זו (ירושת)? \_\_\_\_\_
22. כאשר כותבים שאלתת לינק בתחריר שיטות- מתודות, הביצוע הינו תמיד מיידי.
23. מחלוקת דליגיט מאפשרת להעמס על נציג-(משתנה מטיפוס הדליגיט) מספר מתודות. \_\_\_\_\_
24. מתודות הרחבה מאפשרת למשם מתודה שמקבלת כפרמטר - ייחוס מסווג המשך. \_\_\_\_\_



## חלק ב' - 16%

### סמנן את התשובה הנכונה (8 מתוך 10)

1. מה יוצרת הגדירה `? Shape[] arr = new Shape[5]` ?  
 א. הגדירה אינה חוקית, כי מחלקת shape הינה אבסטרקטית  
 ב. 5 אובייקטים של shape + מצביע לערך  
 ג. 5 מצביעים מסוג shape + מצביע לערך  
 ד. 5 מצביעים מסוג shape
2. איזו מההגדרות הבאות היא לא הגדרה שמאפשרת דרישת במחלקה היורשת?  
 א. Virtual  
 ב. New  
 ג. Override  
 ד. Abstract
3. לאובייקטים מיילו מחלקות קיימת המתודה `?ToString` ?  
 א. Object  
 ב. מחלקות שבן דרסנו פונקציה זו  
 ג. כל המחלקות  
 ד. תשובות 1 ו 2 נכונות
4. איזו הרשאה ניתן למשנה salary במחלקת Employee, אם נרצה לגשת למשנה זה מתוך מחלקת person ?  
 א. public  
 ב. private  
 ג. protected  
 ד. בלתי אפשרי
5. איך נגדיר את המתודה CalcPrice במחלקת Motor, אם נרצה לאפשר שימוש בה מתוך מחלקת מתכוון ?  
 א. public  
 ב. protected  
 ג. static



ד. בלתי אפשרי

6. מאייזה טיפוס ניתן להקנות אובייקט?

- א. מחלקה אבסטרקטית
- ב. מחלקה סטטית
- ג. מחלקה Sealed
- ד. ממשך

7. איך נגדיר את המשתנה פאי במחלקה עיגול?

- א. static
- ב. ref
- ג. readonly
- ד. const

8. הגדרנו Employee e = new Manager(). לאייזה חבר מחלקה לא יוכל לגשת בלי לבצע המרה?

- א. Salary
- ב. Bonus
- ג. Equals(...)
- ד. LastName

9. אובייקטים מאייזה סוג יוכל להכנס למערך מסוג Employee?

- א. object | Manager, Employee
- ב. Employee בלבד
- ג. Employee | Person
- ד. Manager | Employee

10. כאשר דורסים מתודה, מה לא חייב להתאים לשורת הקריאה של מחלקת הבסיס

- א. טיפוס הערך המוחזר
- ב. כמות הפרטרים
- ג. טיפוס הפרטרים
- ד. שמות הפרטרים



## חלק ג' – 50%

### כתבת קוד

ברצוננו למחשב נתונים סקרים שנערכים ע"י חברת סקרים.

לצורך כך נבנתה המחלקה הראשית, מחלקה שאלון: questionnaire

```
class Questionnaire//שאalon/
{
    DateTime QuestionnaireDate { get; set; }
    string Subject { get; set; }
    Question [] Questions { get; set; }

    public Questionnaire()
    {
        QuestionnaireDate = DateTime.Today;
    }
}
```

מבנה של כל שאלה (אובייקט Question) בשאלות הסקר מורכב מ 2 נתונים: הישות הנסקרת ודרוג בין 1 ל 5.

קיימים 3 סוגי של ישות שניתנות לסקרה: 1. אדם (לדוגמה: ראש ממשלה, מנהל כלשהו) 2. מוסד (לדוגמה: קופ"ח, ב"ס), 3. שירות (לדוגמה: קביעת תור לרופא, ביצוע תאום מס מחושב).

בעבור אדם נשמר את שמו ואת תפקידו, בעבור מוסד נשמר את שמו ואת תחומו, בעבור שירות נשמר את שם השירות ואת שם האדם או המוסד שנותן שירות זה.

1. בני את המחלקות הנדרשות לשימרת המידע בעבור השאלות והישויות שלהן. (cotracts של מחלקות + משתנים ומאפיינים ללא בדיקות תקינות).
2. שם של אדם יכול להיות מורכב רק מאותיות ורווחים (לא מספרים ולא תווים מיוחדים). דאגן לאכוף זאת.
3. תחום של מוסד יכול לבחור מתוך רשימה מסוימת של תחומים: חינוך, בריאות, כלכלה, וכו'. הגדרי את הנתון תחום בהתאם.
4. כתבי פונקציה המחשבת ציון ממוצע של שאלון. הממוצע יחושב ע"פ הדרוג שניתן לכל אחת מהשאלות בשאלון, ויתעלם משאלות שלא ניתנה עלייה תשובה (הדרוג שלהן שווה 0). הפונקציה תחזיר את הממוצע וכן תחזיר את מספר השאלות שנענו בפועל.
5. כתבי פונקציה נוספת המחשבת ציון ממוצע. הפונקציה מקבל מספר שאלונים וישות לבדיקה, ותחשב את הציון הממוצע לשירות זו מתוך השאלות אודוטיה בכל השאלונים.
6. כתבי בנאי שמקבל שאלון קיים, ומאתחל בשאלון החדש שאלות חדשות לאלו שיש בשאלון הקיום שנשלחו כפרמטר. נושא השאלה, ותוכן השאלה יועתק, אך לא הדרוג שהן קיבלו.
7. הוסיף למחלקה שאלון פונקציה הממיינת את השאלות לפי השם של הישות שלהן.

רוצים לבדוק את משרד' הממשלה השונים ולתת להם ציון על אופן תפקודם. כדי לתת ציון למשרד ישוקללו הנתונים הבאים:

- ציונים של 500 שאלונים שיכילו שאלות הנוגעות למשרד זה.
- התנהלות סופית תקינה
- קידום חוקים



לצורך כך יש לבנות מחלקות השומרות נתונים אודוות הנהול הכספי וקידום החוקים.

במחלקה ניהול כספי ישמרו הנתונים תקציב מתוכן בש"ח והוצאות בפועל, וכן תהיה פעולה שתחשיב ציון להתנהלות הכספית, כאשר ככל שההוצאות בפועל קרובות יותר למיניהן, הציון גבוה יותר. הציון יחוسب כך: ערך מוחלט של מתוכן פחות בפועל לחלק למתוכן כפול 5.

מחלקת קידום חוקים תכיל מידע על החוקים שנחקקו או קודמו ע"י המשרד, וגם לה תהיה מתודה שתעניק ציון להתנהלות בנושא זה, אך לא נ ממש אותה במבחן. שם המחלקה הוא LowTreatment

8. ממשי את מחלקת התנהלות פיננסית

9. הגדרי ב Main מערכ שיכיל את הנדרש כדי לקבוע ציון למשרד הבריאות. (מספיק כמובן לאותחל בו שני שאלוני בלבד), יש להציג את 2 הבנאים של מחלקת שאלון, ולמלא לפחות אחד מהם את כל נתונים האובייקט.

10. המשרד יקבל ציון מצטיין אם ממוצע הציונים שלו גדול או שווה ל 4 והציון של התנהלות הכספית שווה ל 5. בדק אם המשרד מצטיין, והדפיס זאת.

## בהצלחה רבה!



שם: טל נס

## חלק ד' – בניית מחלקות

ענין על כל 10 השאלות הבאות! (4 \* 10 = 40)

לפניך קוד מחלוקת חדר קראי אותו ענין על השאלות שאחריו.

```
class Window
{
    public int Width { get; set; }
    public int Height { get; set; }
}

abstract class Room
{
    public int Length { get; set; }
    public int Width { get; set; }
    public DateTime BuildingDate { get; set; }
    public Window[] Windows { get; set; }
    public int CalcArea()
    {
        return Length * Width
    }
}
```

מחלוקת חדר ניתן לרשות מחלוקות רבות למשל: חדר שינה, חדר רחצה, סלון וכו' אנו נבצע (חלוקת) את מחלוקת מטבח

שםי לב! במקרה שתשובה דורשת שינוי/תוספה בחלוקת הבסיס, הדגישי זאת בתשובהך.

### שאלות:

1. בחלוקת מטבח חישוב השטח נקבע לפי הכלל הבא: אם המטבח נבנה בחמש השנים האחרונות

הוא מטבח מסווג מטבח פתוח, ולכן יש להפחית 5% משטחו שימושו למעבר, אם המטבח ישן יותר

חישוב השטח זהה לחישוב הבסיס.

כתבו את פונקציית חישוב שטח המתאימה למטבח.

2. כתבו פונקציית חישוב שטח נוספת שהפעם תקבל את סוג המטבח בתור פרמטר מטיפוס enum של

סוג המטבח. enum מוגדר בspace namespace בצדקה הבאה:

```
public enum eKitchenType{open, close}
```

3. חיפוי היא פונקציה שמחשבת את שטח המיועד לחיפוי בקרמיקה, פונקציה זו חייבת להיכلل

בחלוקת מטבח ובמחלקות נוספות למשל חדר רחצה וכדומה - מה יש לבצע לשם כך?

איך?(כתבו קוד)

4. רשמי את הפונקציה המחשבת את השטח המיועד לחיפוי לפי התקף החדר כפול גובה חיפוי של 2

מטרים, יש לזכור להפחית את שטח החלונות מהשטח הכללי.

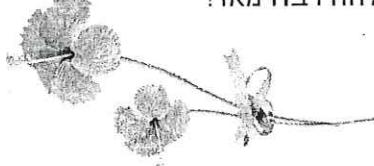


5. כתבי מתודה המחשבת שטח דירה, דירה היא אוסף של חדרים. המתודה מקבלת את החדרים ותחזיר את שטחם הכלול ואת השטח הממוצע לחדר.
6. אורך של צל חיב להיות גדול מ 2 מטרים וקטן מ 4, אך אורך של מטבח יכול להיות עד 6 מטרים. כתבי קוד האוכף את תקינות הערכים המוזנים לשדה אורך.
7. כתבי פונקציה Equals בודקת אם שני חדרים שוויים, חדרים שוים הם חדרים שיש להם אותו שם ואוטו שטח. זכרו כי הפונקציה מוגדרת במחלקה object.
8. ב Main נתון המערך הבא: Room[] arr = new Room[2]
- הकצי אובייקטים מתאימים, אתחל את כל המאפיינים שלהם והציבו אותם במערך. לצורך אתחול המאפיינים השתמשי גם בקלט מהמשתמש.
9. כתבי לולאה שבודקת במערך החדרים איזה צל משללים לחפות, צל שמשתלים לחפות אותו הוא צל שטח החיפוי שלו קטן מ-10 מטרים.
10. ברצוני למיין את מערך החדרים לפי שטחם. המיוון יבוצע באמצעות הפקודה Array.Sort(arr) – מה עלי לבצע לשם כך? (כתבו קוד)
11. כתבי פונקציית הרחבת לכל מחלקה אוסף המכילה חדרים, שתתקבל כפרמטר סוג צל ותחזיר את מספר החדרים הקיימים בראשימה המתאיםים לסוג צל זה שהתקבל כפרמטר.
12. לפניך הגדרת פונקציה גנרטית אשר אמורה להדפיס ריבועים של כוכיות לכל צל שבאוסף, לפי גודל הצל.

השלими את הפונקציה.

```
public static void Print<T>() where T: IEnumerable<Room>
{
    . . .
}
```

בהצלחה רבה מאד!



הו סוף

## Object Oriented / C# - מבחן

### תרגיל א' - מנוע הדפסה היררכית

#### הסבר כללי

מנוע זה מיועד להדפסת אובייקט היררכי כל סוג שהוא ללא הגבלת רמת ההיררכיה. הדפסה תבוצע לוחון Console.

אובייקט היררכי הינו אובייקט המכיל לפחות מאפיין אחד מסווג רשיימה כאשר כל אחד מן האובייקטים בראשימה יכולם להכיל גם הם לפחות מסווג רשיימה.

#### דוגמא א': הדפסת מבנה ארגוני

מחלקות הקיימות:

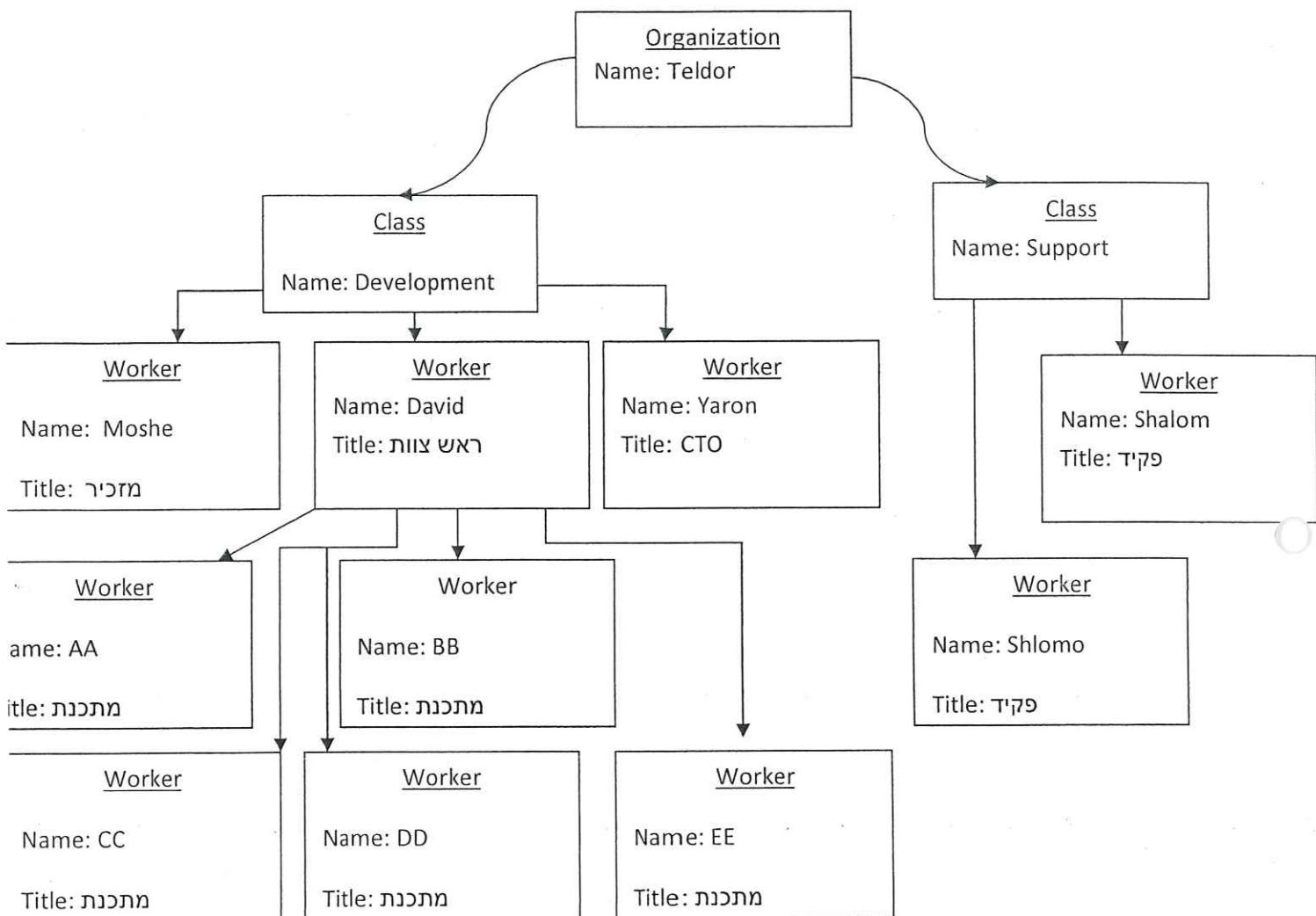
```
public class Organization
{
    public string Name { get; set; }
    public List<Class> Classes { get; set; }
}

public class Class
{
    public string Name { get; set; }
    public List<Worker> Workers { get; set; }
}

public class Worker
{
    public string Name { get; set; }
    public List<Worker> SubWorkers { get; set; }
}
```



דוגמא לארגון קיימן:



מנוע הרדפסה ייצר את הפלט הבא:

&lt;- (ארגון) Teldor

&lt;- (מחלקה) Support

(פקיד) Shalom

(פקיד) Shlomo

&lt;- (מחלקה) Development

(מזכיר) Moshe

(CTO) Yaron

&lt;- (ראש צוות) David

(מתכנת) AA

(מתכנת) BB



(מתכנת) CC

(מתכנת) DD

(מתכנת) EE

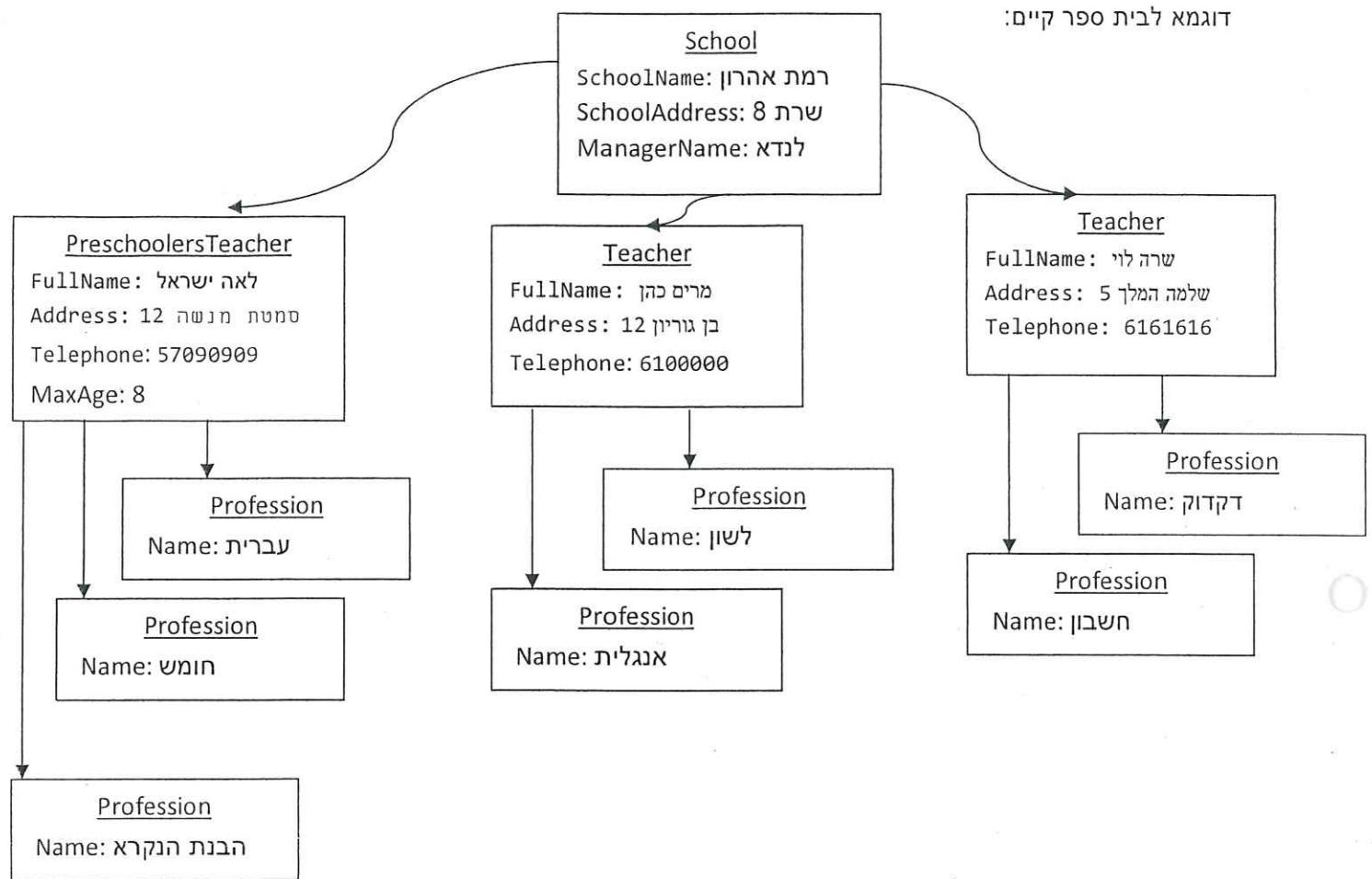
דוגמא ב' – הדפסה עברו בית ספר

המחלקות הקיימות:

```
public class School
{
    public string SchoolName { get; set; }
    public string SchoolAddress { get; set; }
    public string ManagerName { get; set; }
    public List<Teacher> Teachers{get;set;}
}

public class Teacher
{
    public string FullName { get; set; }
    public string Address { get; set; }
    public string Telephone { get; set; }
    public List<Profession> Professions { get; set; }
}
public class PreschoolersTeacher : Teacher
{
    public int MaxAge { get; set; }
}
public class Profession
{
    public string Name { get; set; }
}
```





בית ספר רמת אהרון שרת 8 המנהלת לנדא ->

המורה שרה לוי, כתובות: שלמה המלך, טלפון: 5 6161616 ->

דקדוק

חשבון

המורה מרים כהן, כתובות: בן גוריון 12, טלפון: 6100000 ->

לשון

אנגלית

המורה לגיל הרק לאה ישראלי, כתובות: סמטת מנשה 12, טלפון: 57090909, גיל מקסימלי: 8

עברית

חומש

הבנייה הנקרה



**דרישות פונקציונליות:**

1. אובייקט שיש לו בניים יודפס עם ח' בסופו ובניו יודפסו מתחתיו
2. עבור כל אובייקט תבוצע הזחה עפ"י מס' הרמה אליה הוא משתייך. לדוגמה: הארגון Teldor נמצוא ברמה ראשונה וכאן לא יכול הזחה, המחלקה `Supporth`. נמצאת ברמה שנייה וכאן יכול הזחה של Tab אחד, המתכנת AA נמצא ברמה רביעית וכאן יכול הזחה של 3 Tab-ים.

**דרישות טכניות**

1. יש לבנות מחלקה המהווה את מנוע הדפסה.
2. על המנוע להתאים הן להדפסת מבנה ארגוני, הן להדפסה עבור בית ספר והן להדפסת כל מבנה אחר המהווה מבנה היררכי.
3. האפליקציה הראשית תכיל קריאה למנוע שבנית פעם עבור המבנה הארגוני ופעם עבור בית הספר.

**תרגיל ב'**

שכללי את הפתרון מסעיף א' כך שהמנוע יתמוך בפורמטי הדפסה נוספים:

1. הדפסה ל-`console`: כפי שנדרש בסעיף א', מספר הזרחות על פי הרמה, בסוף שורה של אובייקט שיש לו בניים יודפס ח'.
2. הדפסה לקובץ: במקום הזרחה יופיעו מספרים מדורגים, לדוגמה: 1.1, 1.1.2, 1.1.2. על פי הרמה, בסוף שורה של אובייקט שיש לו בניים יודפס ח', התוצאה כתוב לקובץ (אין צורך לבצע את הכתיבה עצמה לקובץ).
3. הדפסה למדפסת: במקום הזרחה יופיעו כוכיות עפ"י מספר הרמה, בסוף שורה של אובייקט שיש לו בניים יודפס: "להלן נתן רשיימה:" (במקום ח'), התוצאה תשלוח למדפסת (אין צורך למש את הדפסה עצמה)

בהצלחה



for EntityDataSource  
zip into DataBase - DB-first  
(reverse) Model First  
code first

use child (char) primary key  
constraint - like unique - use int identity  
- for DataBase - 131

create database BabyHouse

go

use BabyHouse

go

create table childs

(  
childId int primarykey identity (1,1),  
childIdentity char(6) unique,  
char - not null after id

RoomId int references Rooms(RoomId)

create table Rooms

- Rooms id0

(  
RoomId int identity (1,1) primary key,

( - primary key added at end of table )

1- SQL Visual Studio pic

3rd button re -> Add -> New item -> Data -> ADO.NET Entity Data Model

- connection string -> DB-first (use EntityDataSource)

- here (1) set new connection -> select Database as dropdown at bottom of pic

provider  
datasource  
data base & separte

-> continue and in database copy  
Server name: SQL SERVER  
Authentication - [ ]

and then update when run - config and modify help  
, etc.

Generate Script Tasks you can database to for  
Types of - Advance, next - one or many, next  
data to script  
schema and data

Q)

Event Handler - right click  
resource - file  
component - C#

## WinForm

20P

InitializeComponent() - constructor

Tools → option → projects and solution →  Track Active ...  
Solution Explorer

suspend layout - suspend & pause

TabIndex - index for tabs in TabControl

new window will open at depth 1st depth → TopMost  
2nd press 3rd press next depth

Controls - form to manage controls

.resx - exe → convert into windows application

- if you have inner class / file delegate to another class event //  
- file = p, = very nice

delegate - 1 event for subscriber

when you pass file delegate to another class then pass p to event name  
, public, private, protected, internal

and then you can use another class for subscriber and event -  
, public, private

and then use another class for subscriber and event -  
, public, private

Debug → windows → exception settings → common - throw unhandled exception  
catch / try & catch in ac

Sender - ac

EventArgs

Down - left mouse key

Press → left mouse

keyPreview - before click adapt right click for your need

handle - when be clicked -- handle

UC - user control

if user want to use some feature  
then there are two ways

partial and handle up down

APP reference.

CC - custom control -

the main of this is, use it for the app

add user control

## תרג'il סיכום - WinForms

### המשחק:

**מטרת המשחק:** מציאת האמרה המסתתרת מהלך המשחק: המשתמש יקיש על לחצני הא'ב', בלחיצה על אחת הקיימת באמרה, האות מתגלגת בכל המקומות בהם היא קיימת, במקרה שהאות לא קיימת מתגלגה חלק אחד של התמונה.

לחילופין: הקשה במקלדת (שני את המאפיין המתאים בטופס)

**סיום המשחק:** שלושה אופנים לסיום המשחק:

א. השחקן גילה את האמרה

ב. התמונה התגלתה

ג. הזמן שהוקצב למשחק הסתיים

### הוראות

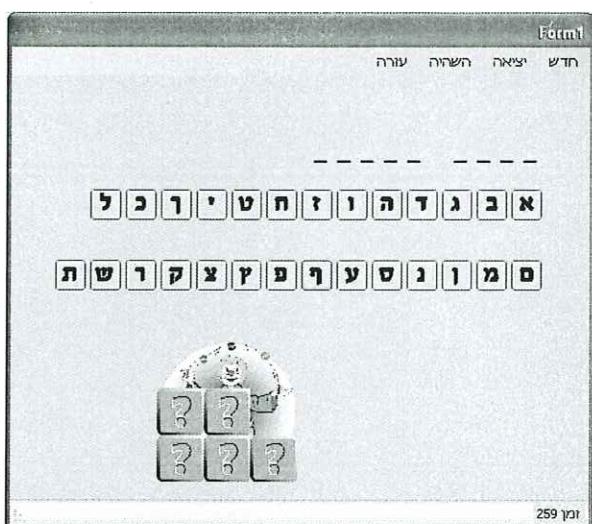
1. הכנסת משפטים לקובץ טקסט
2. קריית המשפטים מהקובץ לתוך מערך מחוזות
3. הגרלת משפט בעזרת הפונקציה `Rnd()`, לדו: `myArray(Rnd(), 6)`
4. שחרור LABELS כמספר התווים במשפט תור שימת לב לרוחחים (באופן דינמי ע"ג `GroupBox`, שימי לב שהוספה הפקדים היא לאווסף הפקדים של הפקד המכיל)
5. שחרור לחצני א' ב' (באופן דינמי ע"ג `GroupBox`-כנ"ל או לחילופין יוצרה בזמן עיצוב אך לדאוג שהקוד ייכתב פעמי אחית)
6. שחרור TIMER

### הערות

1. עלייר לאפשר למשתמש לבחור מהירות משחק.
2. עלייר לאפשר למשתמש לשחק פעמי נוספת נספפת
3. עלייר לדאוג שלחצן א' ב' שכבר נלחץ יהפוך לבליי מאאפשר
4. קובץ עזרה, תפריטים, אפשרי שמירת משחק לפני סיום – ופתיחת המשחק מנוקודת ההפסקה.

5. טיפול בחיריגים

6. נצל את יכולות של המחלקה STRING
7. שימי לב לקריטריונים להערכתה





## הערות לתרגילים:

- ☒ בשלב זה אין צורך לאפשר הקשה במקלדת.
- ☒ בשלב זה את מכניסה את המשפטים למערך מחזרות ולא לקובץ.
- ☒ את המחרוזות תקלטי מהמשתמש בטופס נפרד המכיל `textbox + לחץ "שמור"`.
- ☒ מומלץ להכניס כמה מחרוזות למערך כבר בקוד, כדי שלא תצטרכי בכל ריצה מחדש להכניס מחרוזות כדי לבדוק אם התכנית רצה נכון.
- ☒ הגרלת משפט באמצעות אובייקט `Random` ולא כמו כתוב בדף הקודם, אשר מותאם לשפת VB
- ☒ סעיפים 4 ו-5 בדף הקודם – לא לביצוע בשלב זה.
- ☒ פקד `timer` הוא פקד שיכל למדוד משך זמן ולהפעיל פונקציה כאשרם הזמן הקצוב.
- ☒ כשגוררים אותו על המסך הוא מופיע מתחתית הדף. הוא רכיב ואיןו פקד כיוון שאיןו מוצג על הטופס.

`timer1.Tick += Timer1_Tick; //הגדרת פונקציה שתתבצע כסיום הזמן`  
`timer1.Interval = 60000; //הגדרת משך הזמן לששים שניות`  
`timer1.Start(); //הפעלת הטימר – מתחילה מהרגע`

- ☒ כשגוררים או מיצרים דינמיות קבוצת פקדים דומים, מומלץ לאסוף אותם בתוך פקד `GroupBox`.
- זהו פקד הנועד להכיל פקדים אחרים. שימושיפים לו פקדים מסוימים אותם אליו במקומות לטופס:

לדוגמא, כאן תציג מטריצה של תוויות המכילות טקסט \* בתוך פקד קבוצה:

```
GroupBox grp1 = new GroupBox();
grp1.Location = new Point(10, 10);
grp1.AutoSize = true;
for (int i = 0; i < 20; i++)
{
    for (int j = 0; j < 20; j++)
    {
        Label lb = new Label();
        lb.AutoSize = false;
        lb.Text = "*";
        lb.BackColor = Color.Pink;
        lb.Size = new Size(20, 20);
        lb.Location = new Point(i * 20, j * 20);
        grp1.Controls.Add(lb);
    }
}
this.Controls.Add(grp1);  
המכלול לפקד יחסוי המיקום //; 20 * 20;
```



## Entity Framework

### עבודה

כתבו תוכנה העורכת מבחנים בחשבון לתלמידים.

ה מבחנים מתבססים על מאגר של שאלות. במאגר שמור תוכן השאלה, מס' האחזים המוקצב לה, וה תשובה הנכונה, לשאלות מסווג שאלות אמריקאיות שמורות בנוסף גם 4 תשבות.

מבחן יכול להבצע רק ע"י תלמיד הקיים במאגר התלמידים לאחר הגדהות באמצעות שם ות.ז.

בסוף המבחן התוכנה תבדוק את התשובות שענה התלמיד ותשמר את ציון התלמיד ואת התאריך בו השיג את הציון.

**א.** בני Entity Data Model מתאים שיואר את הishiות הנדרשת לכתיבה התוכנה ואת הקשרים ביניהם.

חוללי בסיס נתונים מקביל לשירות שיצרת.

**ב.** בני את ממשק המשתמש

1. מסך login

- המסך משמש לקליטת שם תלמיד ות.ז. בלחיצה על "כניסה" יבדק אם התלמיד קיים במאגר, אם כן יעבור למסך המבחן

2. מסך מבחן

- למסך זה תישלפנה השאלות המרכיבות את המבחן.  
הבחן אמרור להיבנות בצורה רנדומלית מ 5 שאלות רגילות השווות 6 אחזים כל אחת, מ 4 שאלות רגילות השווות 10 אחזים כל אחת, ומ 6 שאלות אמריקאיות השווות 5 אחזים כל אחת.

- בני שני user Control להציג השאלות, האחד מכיל label של שאלה ו NumericTextBox של תשובה עבור שאלה רגילה, והשני מכיל Label של שאלה, ו Radio Button 4 של תשבות עבור שאלה אמריקאית.

- המסך יוכל לחוץ "הבא", בלחיצה עליו תיבדק נכונות התשובה, ויחסב כמה נקודות נצברו בשאלת זו, ולאחר מכן תוצג השאלה הבאה.

3. מסך סיום

- המסך מציג את מספר הנקודות שהתלמיד צבר במבחן.
- המסך יוכל לחוץ "הציג כל הציונים" המציג ב DataGridView את נתוני הציונים הקודמיים של התלמיד, ומתחתיו ממוצע הציונים.

- ביציאה מסך זה ישמר הציון של המבחן הנוכחי.

**ג.** בני מסך המשמש לניהול רשימת התלמידים.



## וילף ע"ט - נושאים מתקדמים ב C #

- המסר יציג DataGridView ובו רשימת התלמידים.
- לחיצה על "חדש" תעביר למסר שבו שדות ריקים בעבר נתוני התלמיד: שם, ת.ז.
- וכיתה.
- לחיצה על "ערוך" תעביר למסר שבו שדות עם נתונים התלמיד הנבחר.
- לחיצה על "מחק" תסיר את התלמיד ואת ציונו מהמאגר.
- לחיצה על "שמור" תעדכן את נתונים התלמידים כפי שנערכו בבסיס הנתונים.
- פעולות העדכון תשתמשנה בפרוצדורות המאוחסנות בסיס הנתונים.



