תירגול 7 דוט נט מכון טל XML

XML (Extensible Markup Language) - קבוצה של חוקים המגדירים תגים סמנטיים. אותם תגים סמנטיים, מחלקים את המסמך לחלקים שונים, המוגדרים היטב.
זה תקן לייצוג נתונים בקובץ בצורה שמזכירה קצת טבלה.
כל ערך במסמך חסום בין שני תגיות:

- תגית פותחת <*שם התגית*>
- תגית סוגרת <*שם התגית*>

לדוגמה:

```
<name> oshri </name>
```

שם התגית הפותחת ושם התגית הסוגרת חייב להיות זהה (הסוגרת בתוספת של '/')
ניתן להכיל תת תגיות בתוך תגיות והעיקר שלכל תגית יש סוגר מתאים.

לדוגמה:

```
<students>

  <student>
    <id> 123456 </id>
    <name>
      <firstName>oshri</firstName>
      <lastName>choen</lastName>
    </name>
  </student>

  <student>
    <id> 654321 </id>
    <name>
      <firstName>yehuda</firstName>
      <lastName>borisyuk</lastName>
    </name>
  </student>

</students>
```

הסבר:
התגית "סטודנטים":
מכילה בתוכה תת תגיות של סטודנט
התגית "סטודנט" מכילה בתוכה תת תגית של תעודת זהות ושם.
התגית "שם" מכילה בתוכה תת תגית של שם פרטי ושם משפחה

שימו לב שלכל מבנה XML חייב להיות שורש יחיד שמכיל את כל התגיות.
שורש זה נקרא Root Element.
(בדגמה לעיל השורש זה התגית `<students>` שמכילה בתוכה את כל ההיררכיה)

לדוגמה לא ניתן ליצור קובץ XML כזה:

```
<pepole>
  <name>yehuda</name>
</pepole>

<student>
  <name>oshri</name>
</student>
```

שגיאה !!!

אלא הקובץ חייב להיות כך:

```
<someRoot>
  <pepole>
    <name>yehuda</name>
  </pepole>

  <student>
    <name>oshri</name>
  </student>
</someRoot>
```

צריך להגדיר שורש התחלתי!

בנוסף יש לכלול בראש כל קובץ XML את השורה:

```
<?xml version="1.0" encoding="utf-8"?>
```

שאומרת שמדובר בקובץ XML וגרסת השפה והקידוד שלו.

XElement

החל מגירסת c#3 יש לנו אלמנט שנקרא XElement (תחת הספרייה System.Xml.Linq)
אלמנט זה עוזר לנו להכיל קבצי XML בצורה פשוטה.

נראה מה הוא יכול לקבל בפונקציה הבונה:

Name	Description
XElement(XElement)	Initializes a new instance of the XElement class from another XElement object.
XElement(XName)	Initializes a new instance of the XElement class with the specified name.
XElement(XStreamingElement)	Initializes a new instance of the XElement class from an XStreamingElement object.
XElement(XName, Object)	Initializes a new instance of the XElement class with the specified name and content.
XElement(XName, Object[])	Initializes a new instance of the XElement class with the specified name and content.

(מתוך http://msdn.microsoft.com/en-us/library/system.xml.linq.xelement.aspx)

הסבר:

XName – זה אלמנט שיש לו המרה מרומזת ל string (implicit operator) ולכן נוכל לרשום פשוט מחרוזת במקום זה.
XStreamingElement - לא נסביר זאת כעת.
Object – כמובן יכול להכיל כל דבר שנרצה.
Object[] – יכול להכיל כמה ערכים של object עם פסיק בניהם (ע"י שימוש ב params)
(`public XElement(XName name, params object[] content)`)

יש למחלקה XElement פונקציות שימושיות לעבודה על מבנה xml

העיקריות שבהן:

`public void Save(string fileName);`	פונקציה ששומרת את התוכן של ה XElement לתוך קובץ xml שנמצא בנתיב שנשלח לפונקציה
`public static XElement Load(string uri);`	פונקציה **סטטית** שמחזירה XElement בעל מבנה זהה לקובץ xml שנמצא בנתיב שנשלח לפונקציה
`public void SetValue(object value);`	פונקציה זו מעדכנת את הערך בתוך ה XElement למה שנשלח לפונקציה
`public string Value { get; set; }`	מאפיין גישה ל value שהוא כבר מתואר כמחרוזת

בנוסף XElement יורש מ XContainer (abstract class)
ולכן יש לו גם את הפונקציות הבאות:

`public IEnumerable<XElement> Elements();`	פונקציה זו מחזירה IEnumerable<XElement> של כל ה XElement שנמצאים תחתיו
`public void Add(object content);`	מוסיפה אובייקט חדש (יכול להיות גם XElement) תחת אותו XElement (בסופו)
`public void Add(params object[] content);`	מוסיפה אובייקטים חדשים (יכול להיות גם XElement) תחת אותו XElement (בסופו)
`public XElement Element(XName name);`	מחזיר את ה XElement שה XName שלו זהה ל XName שנשלח בפונקציה. (מתוך ערכי ה XElement שנמצאים תחת ה XElement שהפעיל את הפונקציה)
`public void AddFirst(object content);`	מוסיפה אובייקט חדש (יכול להיות גם XElement) תחת אותו XElement (בתחילתו)

דוגמה:

נניח שיש לנו את המחלקה הבאה:

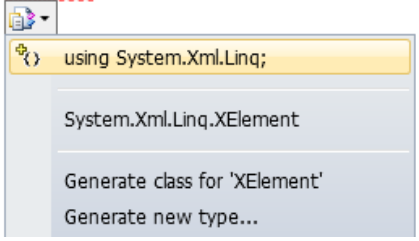
```
public class Student
{
    public double Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

ואנו רוצים לשמור את הנתונים בקובץ XML בצורה הבאה:

```
<students>

  <student>
    <id> 123456 </id>
    <name>
      <firstName>oshri</firstName>
      <firstName>choen</firstName>
    </name>
  </student>

  <student>
    <id> 654321 </id>
    <name>
      <firstName>yehuda</firstName>
      <firstName>borisyuk</firstName>
    </name>
  </student>

</students>
```

לשם כך ניצור מחלקה חדשה שתטפל בשמירת הסטודנטים לקובץ XML
נקרא לה לצורך הנוחות XmlSample וניצור בה משתנה פרטי מסוג XElement
צריך להוסיף using מתאים (ראו תמונה)

```
class XmlSample
{
    XElement studentRoot;
```



כעת נוסיף גם משתנה עבור נתיב השמירה של קובץ ה XML שלנו:

```
class XmlSample
{
   XElement studentRoot;
   string FPath = @"StudentXml.xml";
}
```

נגדיר פונקציה שמקבלת רשימה של סטודנטים ושומרת את כולה בתוך הקובץ:
נבצע זאת בדרך ארוכה ואחרי זה ניראה כיצד ניתן לבצע זאת בפשטות ע"י linq
תחילה נגדיר את השורש של ה XElement ונדאג בסוף גם לשמור אותו בנתיב המתאים (StudentXml.xml)

```
public void SaveStudentList(List<Student> studentList)
{
            studentRoot = new XElement("students");

            studentRoot.Save(FPath);
}
```

הסבר: יצרנו XElement ע"י שימוש בפונקציה הבונה:

```
public XElement(XName name);
```

בשורה האחרונה מתבצעת שמירה, כמובן לא שמרנו עדיין את הרשימה אלא רק:

```
<students> </students>
```

כיצד נכניס את כל ערכי הרשימה ל XElement?

ראשית ברור שצריך לעבור בלולאה עבור כל הרשימה ולייצר כל פעם את ה XElement של הסטודנט:

```
public void SaveStudentList(List<Student> studentList)
        {
            foreach (Student item in studentList)
            {
                XElement id = new XElement("id", item.Id);
                XElement firstName = new XElement("firstName", item.FirstName);
                XElement lastName = new XElement("lastName", item.LastName);
            }
```

הסבר: יצרנו 3 XElement ע"י הפונקציה הבונה:

```
public XElement(XName name, object content);
```

(כפי שצוין לעיל יש המרה מרומזת בין string ל XName)

כלומר יצרנו כעת:

```
<id> ?????? </id>
<firstName>????</firstName>
<lastName>????</lastName>
```

לפי הפרמטרים שברשימת הסטודנטים.
(כמובן בכל הלולאה זה נמחק ומאותחל מחדש אבל בהמשך נעשה בזה שימוש)

כעת ניצור XElement נוסף שיכיל את השם שמורכב מ XElement שם פרטי ו XElement שם משפחה
וניצור עוד XElement שיכיל את כל ה XElement של הסטודנט:

```
foreach (Student item in studentList)
            {
                XElement id = new XElement("id", item.Id);
                XElement firstName = new XElement("firstName", item.FirstName);
                XElement lastName = new XElement("lastName", item.LastName);

                XElement name = new XElement("name",firstName,lastName);

                XElement student = new XElement("student", id, name);
            }
```

הסבר: יצרנו עוד 2 XElement שניהם ע"י הפונקציה הבונה:

```
public XElement(XName name, params object[] content);
```

כעת הערך student מסוג XElement
יראה כך:

```
<student>
  <id> ?????? </id>
  <name>
    <firstName>????</firstName>
    <lastName>????</lastName>
  </name>
</student>
```

כל פעם בהתאם לערכים של item

כעת בהתאם לפונקציות שראינו שקיימות ב XElement
נוכל לרשום את הקוד הבא:

```
public void SaveStudentList(List<Student> studentList)
        {
            studentRoot = new XElement("students");
            foreach (Student item in studentList)
            {
                XElement id = new XElement("id", item.Id);
                XElement firstName = new XElement("firstName", item.FirstName);
                XElement lastName = new XElement("lastName", item.LastName);

                XElement name = new XElement("name",firstName,lastName);

                XElement student = new XElement("student", id, name);

                studentRoot.Add(student);
            }
            studentRoot.Save(FPath);

        }
```

הסבר: כל פעם השתמשנו בפונקציה add על מנת להוסיף את הסטודנט למבנה של ה XElemen
מבנה הקובץ כעת יראה בערך כך (תלוי במה שקיים ברשימה ששלחנו)

```
<students>

  <student>
    <id> 123456 </id>
    <name>
      <firstName>oshri</firstName>
      <lastName>????</lastName>
    </name>
  </student>

  <student>
    <id> 654321 </id>
    <name>
      <firstName>yehuda</firstName>
       <lastName borisyu</lastName>
    </name>
  </student>

</students>
```

נוכל לנצל את העובדה שבשימוש ב LINQ בניגוד לשימוש רגיל בלולאה אנו גם מקבלים ערכים חזרה
ולכן נוכל לרשום את כל מה שכתבנו בצורה כזו:

```
public void SaveStudentListLinq(List<Student> studentList)
        {
            studentRoot = new XElement("students",
                                    from p in studentList
                                    select new XElement("student",
                                        new XElement("id", p.Id),
                                       new XElement("name",
                                            new XElement("firstName", p.FirstName),
                                            new XElement("lastName", p.LastName)
                                            )
                                        )
                                    );
            studentRoot.Save(FPath);
        }
```

הסבר: יש פה שימוש בפונקציה הבונה:

```
public XElement(XName name, params object[] content);
```

טעינת קובץ XML לתוך XElement:

נוכל להגדיר פונקציה לטעינת הקובץ כך:

```
private void LoadData()
        {
            try
            {
                studentRoot = XElement.Load(FPath);
            }
            catch
            {
                Console.WriteLine("File upload problem");
            }
        }
```

שימו לב שהפונקציה Load היא סטטית. (לכן מפעילים אותה ע"י אופרטור נקודה על שם הטיפוס)

על מנת להחזיר את רשימת כל הסטודנטים מהקובץ נגדיר את הפונקציה הבאה:

```
public List<Student> GetStudentList()
        {
            LoadData();
            List<Student> students;
            try
            {
                students = (from p in studentRoot.Elements()
                            select new Student()
                            {
                                Id = Convert.ToInt32(p.Element("id").Value),
                                FirstName = p.Element("name").Element("firstName").Value,
                                LastName = p.Element("name").Element("lastName").Value
                            }).ToList();
            }
            catch
            {
                students = null;
            }
            return students;
        }
```

הסבר:

- בתחילה הגדרנו את הרשימה שנרצה להחזיר
- קיבלנו את כל ערכי ה XElement שלנו ע"י הפונקציה `studentRoot.Elements()`
- עברנו על כולם באמצעות LINQ
- יש לשים לב שעבור טיפוסים ששונים מ string יש לבצע המרה מתאימה מ string לאותו טיפוס
- שימו לב שהערך firstName מקונן בתוך אלמנט אחר
- בשאילתת LINQ אנו מקבלים `IEnumerable` ולכן יש צורך להמירו לרשימה (`ToList`)

דוגמה נוספת:

פונקציה לקבלת סטודנט ע"פ תעודת הזהות שלו:

```
public Student GetStudent(int id)
        {
            LoadData();
            Student student;
            try
            {
                student = (from p in studentRoot.Elements()
                           where Convert.ToInt32(p.Element("id").Value)==id
                            select new Student()
                            {
                                Id = Convert.ToInt32(p.Element("id").Value),
                                FirstName = p.Element("name").Element("firstName").Value,
                                LastName = p.Element("name").Element("lastName").Value
                            }).FirstOrDefault();
            }
            catch
            {
                student = null;
            }
            return student;
        }
```

הסבר:

הפונקציה הזו דומה מאוד לפונקציה להחזרת כל הסטודנטים עם כמה שינויים

- מחזירים הפעם Student ולא רשימה של סטודנטים
- מקבלים id שמציין את מספר הסטודנט שנרצה להחזיר
- יש תנאי (where) על מנת להחזיר רק את הסטודנט הרצוי
- בסוף אנו ממירים את השאילתה לאובייקט ע"י הפונקציה FirstOrDefault שמחזירה את הערך הראשון ברשימת הערכים שחזרו מהפעלת LINQ במידה ואין ערך כזה יוחזר ערך ברירת מחדל (עבור int יוחזר 0 עבור אובייקט מורכב יוחזר null וכו...)

דוגמה לקבלת שם הסטודנט בלבד:

```
public string GetStudentName(int id)
        {
            LoadData();
            string studentName;
            try
            {
                studentName = (from p in studentRoot.Elements()
                               where Convert.ToInt32(p.Element("id").Value) == id
                               select p.Element("name").Element("firstName").Value
                               + " " +
                                  p.Element("name").Element("lastName").Value
                                  ).FirstOrDefault();
            }
            catch
            {
                studentName = null;
            }
            return studentName;
        }
```

שימו לב לשרשור של השם פרטי עם שם המשפחה.

לאחר שראינו דוגמאות עבור פעולות שונות על XElement נראה עכשיו כיצד נגדיר מחלקה שתבצע את פעולות ההוספה, עידכון, מחיקה ושליפה עבור נתונים ששמורים ב XML

נתחיל בכך שנגדיר את המשתנים הבאים:

```
class XmlSample
{
     XElement studentRoot;
     string studentPath = @"StudentXml.xml";
}
```

על מנת שיהיה לנו קובץ שאליו נוכל להוסיף, לעדכן, וכו...
בפונקציה הבונה נאתחל את הקובץ במקרה שהוא לא קיים:

```
class XmlSample
    {
        XElement studentRoot;
        string studentPath = @"StudentXml.xml";

        public XmlSample()
        {
            if (!File.Exists(studentPath))
                CreateFiles();
            else
                LoadData();
        }

        private void CreateFiles()
        {
            studentRoot = new XElement("students");
            studentRoot.Save(studentPath);
        }

        private void LoadData()
        {
            try
            {
                studentRoot = XElement.Load(studentPath);
            }
            catch
            {
                Console.WriteLine("File upload problem");
            }
        }
```

הסבר:

- הפונקציה CreateFiles יוצרת אלמנט XElement שיהווה את השורש של הקובץ בצורה `<students> </students>` ושומרת את התוכן שלו לקובץ XML שמוגדר בנתיב.
- הפונקציה LoadData זה בדיוק כפי שראינו לעיל, במקרה שיש כבר קובץ מוכן, לא נירצה ליצור חדש אלא לטעון אותו לאלמנט XElement שלנו
- בפונקציה הבונה המתודה File.Exists מקבלת נתיב ומחזירה אמת אם הקובץ (הנתיב המלא) קיים ושקר אחרת.

על מנת להוסיף סטודנט נגדיר את הפונקציה הבאה:

```
public void AddStudent(Student student)
        {
            XElement id = new XElement("id", student.Id);
            XElement firstName = new XElement("firstName", student.FirstName);
            XElement lastName = new XElement("lastName", student.LastName);
            XElement name = new XElement("name", firstName, lastName);

            studentRoot.Add(new XElement("student", id, name));
            studentRoot.Save(studentPath);
        }
```

הסבר:

- כפי שהראנו בתחילה, גם כאן אנו מגדירים אלמנט אלמנט ולבסוף מחברים אותם יחד לאלמנט XElement חדש שיתאר את הסטודנט ויצטרף לתוך השורש שלנו
- היה ניתן להגדיר את הכול כבר בתוך השורה studentRoot.Add(..)
- בסוף אנו שומרים את השינויים של ה XElement בקובץ המתאים

על מנת למחוק סטודנט נגדיר את הפונקציה הבאה:

```
public bool RemoveStudent(int id)
        {
            XElement studentElement;
            try
            {
                studentElement = (from p in studentRoot.Elements()
                                  where Convert.ToInt32(p.Element("id").Value) == id
                                  select p).FirstOrDefault();
                studentElement.Remove();
                studentRoot.Save(studentPath);
                return true;
            }
            catch
            {
                return false;
            }
        }
```

הסבר:

- בתחילה אנו מוצאים את ה XElement המתאים באמצעות שאילתת LINQ
- המתודה FirstOrDefault() מחזירה את האלמנט הראשון באוסף (כזכור שאילתת LINQ מחזירה בדרך כלל IEnumerable) במידה ואין אלמנטים באוסף יוחזר ערך ברירת מחדל (תלוי באוסף, לדוגמה עבור int יוחזר 0 עבור float יוחזר 0.0 ועבור עצם יוחזר NULL)
- המתודה studentElement.Remove() מסירה את ה XElement שעליו מצביע studentElement מתוך השורש שלנו
- בסוף אנו שומרים את השינויים של ה XElement בקובץ המתאים

על מנת לבצע עידכון נגדיר את הקוד הבא:

```
public void UpdateStudent(Student student)
        {
            XElement studentElement = (from p in studentRoot.Elements()
                                       where Convert.ToInt32(p.Element("id").Value) == student.Id
                                       select p).FirstOrDefault();

            studentElement.Element("name").Element("firstName").Value = student.FirstName;
            studentElement.Element("name").Element("lastName").Value = student.LastName;

            studentRoot.Save(studentPath);
        }
```

הסבר:

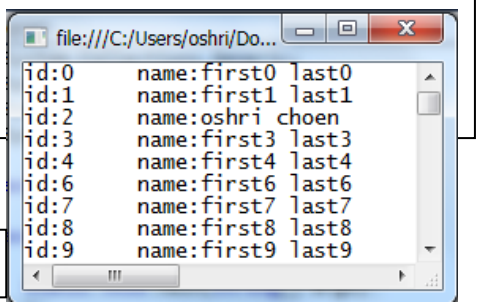
- הפונקציה עובדת בצורה כזו:
  - מקבלת סטודנט.
  - מחפשת סטודנט מתוך הנתונים בעל תעודת זהות זהה לסטודנט שקיבלה בפונקציה.
  - מעדכנת כל ערך (פרט לתעודת זהות) בסטודנט שמצאה שיהיה זהה לערך בסטודנט שקיבלה.
- בתחילה אנו מוצאים את ה XElement המתאים באמצעות שאילתת LINQ (בדיוק כפי שביצענו עבור המחיקה)
- לאחר מכן אנו מעדכנים את הערכים בהתאם למה שקיבלנו בפונקציה.

הערה כללית:

- בשלושת הפונקציות של הוספה הסרה ועידכון אנו מניחים שה LoadData האחרון שבוצע משקף בדיוק את נתוני הקובץ ,אחרת אם יש חשש שהנתונים שבקובץ לא מעודכנים ב XElement שלנו (לדוגמה כמה משתמשים שיכולים ע"פ תור מסוים לשנות משהו בקובץ) יהיה עלינו לבצע זאת בתחילת כל פונקציה.

דוגמת הרצה:

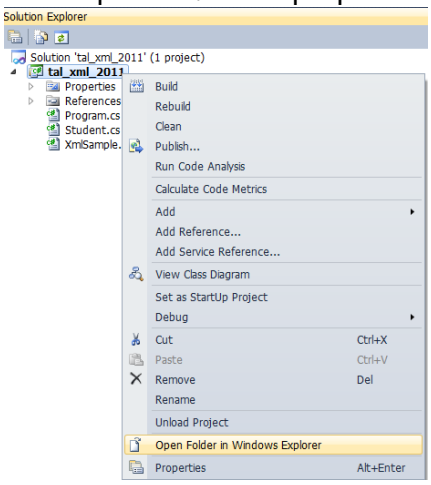
```
static void Main(string[] args)
        {
            XmlSample xml = new XmlSample();

            for (int i = 0; i < 10; i++)
            {
                Student s = new Student { Id = i, FirstName = "first" + i, LastName = "last" + i };
                xml.AddStudent(s);
            }
            xml.RemoveStudent(5);
            xml.UpdateStudent(new Student { Id = 2, FirstName = "oshri", LastName = "choen" });
            List<Student> list = xml.GetStudentList();

            foreach (var item in list)
            {
                Console.WriteLine("id:{0,-5} name:{1} {2}", item.Id, item.FirstName, item.LastName);
            }

            Console.ReadLine();
        }
```



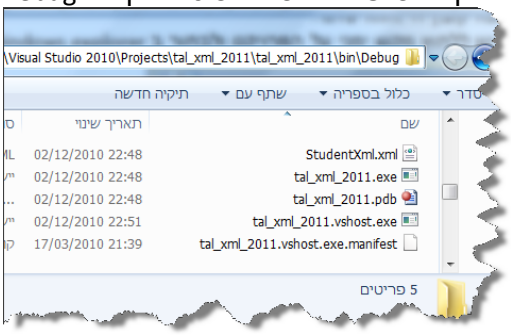
(כפי שניתן לראות הסטודנט עם ה ID 5 נמחק, והסטודנט עם ה ID 2 עודכן)

איפה נמצא קובץ ה XML שלנו?

- יש ללחוץ מקש ימני על הפרויקט ולבחור ב open folder in windows explorer

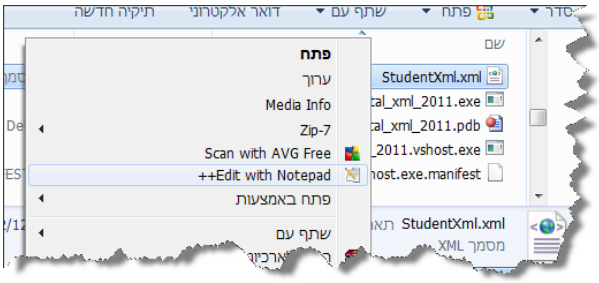


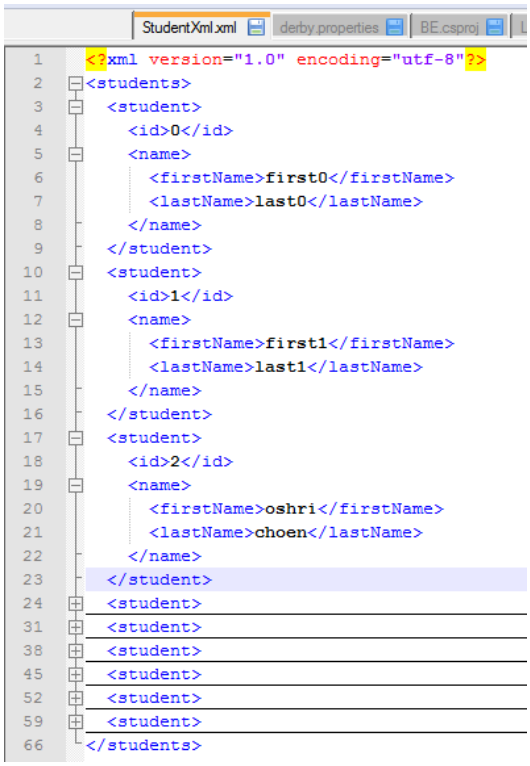
- בתיקייה שנפתחה יש להיכנס לתיקיות bin\Debug\



ניתן לראות את הקובץ שהגדרנו " StudentXml.xml "

כדאי לפתוח את הקובץ בעזרת notepad++ (ניתן ללחוץ כאן ולהוריד) על מנת לראות את תוכן הקובץ:





(לצורך הצילום מסך כפי שניתן לראות סגרתי את האלמנטים שמתארים את הסטודנטים מ 3 עד 9)