שאלות מראיונות עבודה בדוט נט – האם יש לך מה שדרוש?

בספטמבר 21 2008

שלום לכולם,

לאחרונה ישבתי עם מפתחת דוט נט מוכשרת מאוד. התוכניתנית הזו מחפשת משרה חלקית ובעלת שנתיים ניסיון בדוט נט בחברת אלביט. חלק מהשיחה התמקדה בשאלות טכניות על דוט נט שידוע לי שנוטים לשאול בראיונות עבודה. אחרי שהיא ענתה על השאלות הכי מכשילות שהיו לי והוכיחה רמת ידע מאוד גבוהה, החלטתי לכתוב מאמר מקיף המכיל אוסף רב של שאלות כאלו.

1. הפניות לרכיבים מסוג using או reffrences, באם לא נעשה בהן שימוש, יראו בקוד ה-IL?

לא. במהלך קומפילציה כל הפניה מסוג using או reffrences לא תירשם בקוד ה-IL אלא אם כן היא בשימוש כלשהו בתוך הקוד. קרי, הפניה לרכיב חיצוני או הפנייה ל-namespace שלא בשימוש לא יראו בקוד הסופי.

```
using System; // Will be included in the MSIL
using System.Data; // Will not even show up in the MSIL

class myClass
{
  public static void Main()
  {
    Console.Writeline("Hello world");
  }
}
```

2. מה ההבדל בין readonly ל-constant?

יותר קל לעמוד על הדימיון ביניהם: הם קבועים בזמן ריצת התוכנית. ההבדל המהותי הוא כזה: הערך שנותנים ל-constant נכתב בקוד ובזמן קימפול האפליקציה פשוט מועתק לכל מקום בו הוא בשימוש, כלומר שבזמן הריצה בכלל לא קיים שום משתנה מסוג constant. על הצד השני readonly אכן קיים בזמן הריצה כמשתנה וצריך לאתחל אותו או בשורת הצהרת המשתנה או בתוך ה-constructor של המחלקה. עוד הבדל חשוב הוא ש-constant לא יכול להיות אובייקט אלא חייב להיות משהו פשוט (int, string, bool וכיו"ב)

3. מה זה boxing ומה זה unboxing?

באחד מהמאמרים הקודמים שלי ("Struct ו-Class -ההבדלים והכוח הטמון בשוני ביניהם") דיברתי על ההבדלים בין Value types ל-reffrence type ומה זה Stack ו-Heap. בקצרה, כל מחלקה בדוט נט יורשת בהכרח מ-System.Object ובהקצאת זכרון תשב פיזית על ה-Heap כאשר ההפנייה למיקום על ה-Heap ישב ב-Stack. ישנה קבוצה קטנה של מחלקות שיורשות מ-System.ValueType (למשל int, bool, enums, structs וכיו"ב) שהם יושבים פיזית על ה-Stack. פעולת ה-boxing היא לקיחת ValueType והעתקו ל-Heap עם יצירת הפנייה ל-heap שיושבת על ה-stack. הפעולה unboxing היא ההפוכה והיא לקיחת ValueType שעשו לו boxing ויצירת אובייקט זהה ב-stack.

4. האם ניתן לתת הרשאות public/private/internal שונות ל-get של Property ול-Set של property?

לא. הגישה ל-get ול-set של property הם בעלי אותה רמת גישה בדיוק (שנלקחת מרמת הגישה כפי שנקבעה לכל ה-Property).

5. מהי הרשאת internal?

בדומה להרשאות גישה של public/private קיימת הרשאה בשם internal. הראשת internal קובעת כי המשאב הזה יהיה זמין רק בתוך ה-assembly שבה נכתבה המחלקה. למשל, אם יצרנו שכבה אחת במערכת שרצינו להגביל לה גישה למתודה\מחלקה\Property מסויימת לשימוש פנימי בתוך אותה assembly נשתמש בהרשאת הגישה internal.

6. מה ידפיס הקוד הבא?

```
public void myMethod()
{
  try
  {
    Console.Writeline("Hello");
    return;
  }

  catch
  {
    Console.Writeline(" my ");
  }

  finally
  {
    Console.Writeline(" world");
   }
}
```

הקוד הנ"ל ידפיס Hello World. סעיף ה-finally תמיד תמיד מתבצע. זה העקרון המנחה שכותבים בלוק finally. גם אם יש חריגה, גם אם יצאנו מהפונקציה, גם אם הקפצנו אירועים לא קשורים, תמיד יתבצע הקוד בתוך בלוק ה-finally.

7. מהם ה"דורות" של ה-Garbage collector?

ה-Garbage collector מבוסס על העקרון הבא: כל אובייקט חדש שנוצר מתווסף למה שנקרא "דור 0". מספר הדור אומר כמה "איסופים" של ה-Garbage collector הוא "שרד". "איסוף" מתייחס לכאשר ה-garbage collector מחליט שצריך לפנות זכרון ובודק אילו אובייקטים יש להשאיר ואיזה אפשר לזרוק. אם ה-garbage collector קובע שהאובייקט צריך להישאר הוא יעביר אותו ל"דור 1" ובאיסוף הבא ל"דור 2". אין דורות מעבר לדור 2.

8. האם ניתן להכריח את ה-Garbage collector לאסוף אובייקטים כדי לפנות מקום בזכרון? והאם כדי?

כן, אפשר. המתודה הסטטית GC.Collect מאפשרת להריץ את ה-garbage collector על "דור" מסויים או כל ה"דורות" בזכרון. לעומת זאת, העובדה שהפונקציה קיימת שם לא אומר שזה בהכרח רעיון טוב. מלבד מצבי קצה, עדיף להימנע מלקרוא ל-GC.Collect היות ורק עצם הזמן שה-Garbage collector משקיע בלברר איזה אובייקטים להשמיד ואיזה לא להשמיד (ולהעביר דור אחד הלאה) לוקח הרבה יותר משאבים וזמן מאשר אלו שאנו רוצים לחסוך. תנו ל-Garbage collector לנהל לעצמו את הזכרון.

9. האם אפשר לכתוב Desctructor למחלקות? והאם כדי? (Destructor היא פונקציה שמתבצעת בעת האיסוף של ה- garbage collector)

כן, אפשר.

```
class myClass
{
public myClass() {} // Constructor
public ~myClass() {} // Destructor
}
```

על הצד השני – לא עושים כזה דבר (אלמלא במצבי קצה). כאשר ה-Destructor רץ זה אומר שה-Garbage collector הגיע למסקנה ש- "טוב, אתה הולך למות כדי לפנות מקום! תמות! תמות! תמות!", אבל אם יש Destructor אז המחלקה אומרת "יש לי עוד פונקציה להריץ!". ומה שקורה בסוף הוא שבמקום שה-Garbage collector יהרוג את המחלקה, היא חוזרת לזכרון כדי להריץ את ה-Destructor שלה וה-Garbage collector יחכה לאיסוף הבא כדי להרוג אותה. במקום לחסוך זכרון או לעזור במשהו, רק דפקנו את ה-Garbage collection.

מתי כן נכתוב Destructor? כאשר כותבים "ישומיי תשתית", למשל אם כותבים מחדש מחלקה שמבצעת חיבור לקבצים ורוצים להיות בטוחים שהמשאב נסגר, אם כותבים מחלקה שמבצעת חיבור למסד נתונים ורוצים להיות בטוחים שהמשאב נסגר, אם כותבים Wrapper לקוד שנכתב ב-c++ וכך הלאה. אלא אם כן יש באמת צורך אמיתי וברור ל-Destructor, לא כותבים אותו.

10. איך משתמשים בפונקציה מ-DLL חיצוני?

משתמשים ב-namespace של System.Runtime.InteropServices, ב-Attribute שנקראת DLLImport המכילה את שם ה-DLL, כותבים פונקציה עם קידומת static extern לשם הפונקציה והחתימה שלה ב-DLL ומתפללים. למשל:

```
using System.Runtime.InteropServices;
class myImport
{
[DllImport("user32.dll")]

public static extern int MessageBoxA(int h, string m, string c, int type);

public static int Main()
{
  return MessageBoxA(0, "Hello World!", "myHello", 0);
}
}
```

11. כתבנו [DLLImport], הצהרנו על פונקציה וכאשר אנו קוראים לה, שום דבר לא קורה, מדוע?

חובה לציין לפני חתימת הפונקציה מה-DLL שהיא מסוג static extern.

12. מהי קידומת unsafe לבלוקי קוד \ מתודות?

קידומת זו מאפשרת לנו לכתוב בלוקי קוד עם פוינטרים (חברינו האבודים מ-c++). בנוסף לקידומת זו, צריך לשנות בתוך ה- Project properties, בתוך Configuration properties את התכונה Allow unsafe code blocks ל-true. נראה קוד לדוגמה:

```
unsafe
{
  int* myPointer; // Reffrence to new empty pointer
  int myIntValue = 9; // new int value

  // Set pointer to point to the new int memory location
  myPointer = &myIntValue;

  // print value of pointer: will print "myPointer is 9"
  Console.WriteLine("myPointer is: " + *myPointer);
}
```

13. האם חובה לכתוב על כל פונקציה extern שהיא מסוג Unsafe?

לא. ואלא אם כן זה באמת נדרש (עבודה מסוכנת עם pointerים) בתוך ה-DLL, עדיף להימנע מלעשות את זה.

14. כאשר אנו משתמשים ב-int בסביבת הפיתוח של #C, באיזה טיפוס נתונים אנו משתמשים באמת?

כאשר אנו משתמשים ב-int אנו משתמשים באמת ב-System.Int32.

15. אילו סוגים של intים קיימים?

Int16 – שהערך המינימלי שלו -32767 והערך המקסימלי שלו 32767.

Int32 – שהערך המינימלי שלו -2,147,483,648 והערך המקסימלי 2,147,483,648.

Int64 – שהערך המינימלי שלו -9,223,372,036,854,775,808 והערך המקסימלי שלו 9,223,372,036,854,775,808.

IntPtr – מחלקה זו היא תלויה בחומרת המחשב עליה היא רצה. במחשב bit32 היא Int32 ובמחשב bit64 היא Int64. המחלקה שימושית מאוד שצריך להעביר פוינטרים למספרים לכל מיני Reffrences שדורשות אותן (למשל המקרה הנפוץ הוא DLLImport למתודה שמקבלת פוינטר לספר).

16. מה זאת CLScompliant Attribute?

ה-Attribute הזאתי באה ומצהירה על המחלקה\אסמבלי\... עליה היא רשומה את הדבר הבא "אפשר לתרגם אותי לשפות אחרות בדוט נט". אם נוסיף את התכונה הזאת למשל לכל האסמבלי שלנו (דוגמה בהמשך) הקומפיילר יבדוק:
*** האם שמות המתודות שלנו שונות בלי קשר ל-casing (אחרת יכול להיווצר מצב של שתי מתודות עם אותו שם עם הבדל של casing ולמשל ב-VB.net זה בלתי אפשרי).**
*** כל מתודה\Property שמוצהרת כ-public אסור שתתחיל במקף תחתון.**
*** אין Opearator overloading, כלומר לא התחלנו להקצות התנהגות מיוחדת לכל האופרטורים שלנו.**
*** הוא בודק שאין חתימות זהות לפונקציות ומתעלם מפרמטרים מסוג ref/out.**

כדי להכיל על אסמבלי למשל שהיא CLScomplaint נוסיף ב-AssesmblyInfo.cs את השורה הבאה:

```
[assembly: CLSCompliant(true)]
```

17. איזה סוגי Intים לא הזכרתי בתשובה בסעיף 15? (רמז: שאלה 16)

UInt16, UInt32, UInt64, UIntPtr. הערכים המקסימליים של ה-UIntים כפולים מהערכים המקסימלים של ה-Intים הרגילים. אבל, הם לא CLScompliant. כלומר, הם קיימים ב-#C, אבל הם לא קיימים בפירוט השפה של דוט נט ולכן כנראה ולא יתרגמו טוב לשפות דוט נטיות אחרות. אם נרצה לכתוב על אסמבלי שהיא CLSComplaint בזמן שאנו עובדים עם Uintים נציין שהמתודה\... שעובדת עם ה-Uintים היא לא CLScomplaint.

```
// In AssesmblyInfo.cs
[assembly: CLSCompliant(true)]

// Somewhere in the assembly: (From MSDN)
[CLSCompliant(false)]
public int SetValue(UInt32 value);
```

18. מה המקביל (מבחינת ערכי מקסימום ומינימום) של Int128 (שאינו קיים) ושל Uint64 (שאמרנו שערכיו המקסימליים והמינימלים כפולים מאלו של Int64)?

הטיפוס decimal.

19. מדוע ומתי עדיף לעבוד עם StringBuilder מאשר עם String?
(אם הייתי מקבל שקל כל פעם ששאלו אותי את זה...)

String הוא מה שנקרא Imuteable, הווה אומר כל פעם שנשנה לו את הערך תיווצר מחרוזת חדשה לחלוטין עם הערך השונה. אם למשל נרצה לשרשר את כל המספרים בין אחד למיליון בתוך מחרוזת ונעשה את זה במחרוזת רגילה – כל פעם בתוך הלולאה תיווצר מחרוזת חדשה לחלוטין. מיליון מחרוזת זה קצת בזבזני. במקום זה למקרים כאלו עובדים עם StringBuilder ובעיקר עם מתודות Append ו-ToString. ככה ניתן להוסיף כל פעם ל-StringBuilder או לשנות אותו בהתאם לצרכינו וליצור מחרוזת אמיתית רק כאשר אנו צריכים אותה. כלל האצבע הוא שכאשר אנו עוברים 5 חיבורים של מחרוזות – עוברים לעבוד עם StringBuilder. חשוב לציין שעם עובדים עם פחות מ-5 השימוש ב-StringBuilder על פני String הופך להיות בזבזני מאוד במשאבי מערכת.

20. מה ההבדל בין System.Array.CopyTo ל-System.Array.Clone?

CopyTo מבצע Deep copy, כלומר יוצר מופע חדש לחלוטין ונותן לנו הפנייה אליו. Clone מבצע Shallow Copy, כלומר CopyTo יוצר הפנייה נוספת למופע הקיים. כלומר, ב-Clone אנו נעבוד על אותו מערך כל הזמן, כאשר ב-CopyTo נעבוד על מערכים שונים.

21. קיים מערך מסוג System.Array (או כל טיפוס שיורש ממנו) כיצד ניתן לסדר אותו בסדר יורד?

נשתמש במתודת Array.Sort() ואז במתודת Array.Reverse(). ככה נסדר את המערך בסדר עולה ונהפוך את הסדר כך שנקבל סדר יורד.

22. מהם שלושת סוגי ההערות בדוט נט ומה ההבדל העקרוני ביניהן?

סלאש כפול (\\), סלאש וכוכבית (*) עם סוגר של כוכבית וסלאש (*\) וסלאש משולש (\\\). שני הראשונים משמשים להערות בגוף הקוד, כאשר השלישי משמש לתיעוד הערות XML על מתודות\Properties\מחלקות\אסמבליס\....

23. כיצד ניתן לקבל את כל הערות ה-XML בגוף הקוד בתוך קובץ דוקומטציה מאורגן? ובפרט, מתוך ה-Command line?

בתוך Visual Studio בתפריט Tools קיים כפתור Generate XML documention, בלחיצה עליו כל הערות ה-XML של ה-Solution נאספות לקובץ HTML מאורגן ומעוצב היטב. אפשרות נוספת היא כאשר מבצעים קימפול מה-Command Line דרך פקודת csc.exe נוסיף doc/ וגם אז יווצר קובץ תיעוד לכל ה-XML מה-Solution.

24. קיים מסד מסוג SqlServer, כיצד ניתן להתחבר אליו מתוך דוט נט, מהן האפשרויות? מהן החסרונות והיתרונות של כל אפשרות?

למסד הנתונים סיקוול קיים DataProvider מיוחד ומותאם לו – System.Data.SqlClient. הוא המהיר מבין האפשרויות הבאות, אך דורש לרכוש רישיון שרת סיקוול מתאים ממיקרוסופט. בנוסף קיים OleDbClient שמאפשר גישה ג'נארית לכל מסד נתונים מתוך דוט נט. הוא חינם, אבל הוא איטי לפחות פי שתיים מה-DataProviderים היעודיים. בנוסף קיים Odbc.Net שגם הוא ג'נארי (ולמעשה חלק מ-OleDb) אבל מאפשר גישה יותר מיושנת והרבה יותר ג'נארית למסדי נתונים. הבעיה עם Odbc.Net היא שהוא איטי באופן ניכר מ-OleDbClient. היתרון שלו (כפי שנאמר), היא שהוא תומך באופן יותר ג'נארי ולכן יכול לתמוך גם במסדי נתונים מאוד מאוד ישנים.

25. מה ההבדל בין Overloading ל-Overriding?
(וברצינות, ראבאק, מי חשב על השאלה הזאת? איזה עובד מתוסכל מ-HR שחשב שהמושגים נשמעים ממש דומה וכולם מתבלבלים איתם? השאלה הכי מעצבנת ביקום.)

Overloading היא פיתוח מתודה בעלת שם מתודה ורמת גישה זהה למתודה קיימת אך עם פרמטרים שונים. כך ניתן לכתוב למשל 10 מתודות עם אותו שם ורמת גישה, אך שמקבלות פרמטרים שונים ומבצעות דברים שונים (ולרוב גם קיימים קשרים בין המתודות השונות). Overriding היא מושג מתחום הירושה המאפשר לדרוס מתודה מהמחלקה שנורשת המוסמנת ב-virtual ולכתוב

במחלקה היורשת שהמתודה היא overriden. בצורה זו, ניתן לדרוס פונקציות שנורשות, אך היתרון הכי חשוב הוא שגם בעת עבודה עם פולימורפיזם השימוש יהיו בפונקציה שדורסת ולא בפונקציה המקורית (כל עוד מתקיימת שרשרת הורשה).

```
class myInheritedClass
{
public myInheritedClass() {}

virtual public void myVirtualMethod()
  {
    Console.Writeline("I am virtual");
  }
}

class myInhertingClass :myInheritedClass

{
  public myInhertingClass() {}

  override public void myVirtualMethod()
  {
    Console.Writeline("I am override!");
  }
}
class myMain
{
  public static void Main()
  {
    myInheritedClass a = new myInheritedClass();
    a.myVirtualMethod(); // Will print out "I am virtual!"


   myInhertingClass b = new myInhertingClass();
    b.myVirtualMethod(); // Will print out "I am override!"

    a = b;
    a.myVirtualMethod(); // Will print out "I am override!"
  }
}
```

בדוגמה האחרונה למרות שאנו עובדים עם הפנייה לטיפוס שנורש עדיין מודפסת ההודעה מהטיפוס היורש. הסיבה לכך הוא שהפולימורפיזם בדוגמה זו דואג לכך שהמתודה myVirtualMethod עדיין נדרסת גם בתוך הפולימורפיזם.

26. האם ניתן לבצע הורשה מרובה בדוט נט?

לא. אי-אפשר. קפוט. זיפ. פיניטו. אין כזה דבר הורשה משתי מחלקות בו זמנית (כלומר שמחלקה אחת תירוש בהגדרה שלה שתי מחלקות שאינן קשורות דרך הורשה בעצמן), ותחליפים נורמליים קיימים פר מצב ב-Design Patterns.

27. מה ההבדל בין הצהרות switch ב-C++ ל-C#?
(שיהיה ברור דבר אחד, אני בקושי יודע איך לכתוב hello world ב-C++, אבל בשביל ראיונות עבודה למדתי לענות על השאלה הזאת.)

ב-C++ ניתן ליפול בין הצהרות שונות של case, וב-C# זה בלתי אפשרי.

```
switch(myInt)
{
case 0: // fall through to 2
  Console.Writeline("myInt Equels 0");
case 1: // fall through to 2
```

```
case 2:
   Console.Writeline("myInt Equels 2");
break;
}
```

הקוד למעלה ירוץ ב-++C, אך יזרוק שגיאת קומפיילר ב-#C. זאת היות וכל ענף (case) בהצהרת switch מחוייב להכיל break.
כלומר, לא ניתן להריץ קוד כלשהו בתוך case בלי לשים בסופו הצהרת break. אז מה כן אפשר לעשות? כאפשרות ראשונה קיים הפתרון של להשתמש ב-goto:

```
switch(myInt)
{
case 0: // fall through to 2
   Console.Writeline("myInt Equels 0");
   goto case 2;
   break;
case 1: // fall through to 2
   goto case 2;
   break;
case 2:
   Console.Writeline("myInt Equels 2");
break;
}
```

זה הפתרון הפשוט והבנאלי שכולם מחפשים. לעומת זאת, מנסיוני, אם תעבדו ככה במציאות עם מצבים קצת יותר מסובכים, כל ההירכיה הזאת של "תקפוץ לכאן! תקפוץ לשם! ותקפוץ מפה לפה!" תדרדר לג'יבריש לא ברור ובלתי קריא.

אז מה כן אפשר לעשות? אישית, אני יוצר delegate שמקבל את האובייקט עליו עושים switch, בתוך ההצהרות השונות מוסיף מתודות שרצות כחלק מ-multi cast delegate ואחרי ה-switch מריץ את ה-delegate.

```
class myClass
{
  // Delegate to hold functions to be executed (param curInt)
  private delegate void mySwitchDelegate(int curInt);

  // Static method for case1
  static private void DoCase1(int curInt)
  {
   Console.WriteLine("Case 1 execution!");
  }

  // Static method for case2
  static private void DoCase2(int curInt)
  {
   Console.WriteLine("Case 2 execution!");
  }
  static void Main(string[] args)
  {

   int myInt = 1;

   // Empty Delegate
   mySwitchDelegate myDelegate = null;

   switch(myInt)
   {
    case 0: // fall through to 2
     // Add Execution of DoCase2 to Delegate
     myDelegate +=  new mySwitchDelegate(DoCase2);
     break;
    case 1: // Do case 1
```

```
    // Add Execution of DoCase1 to Delegate
    myDelegate+=  new mySwitchDelegate(DoCase1);
    break;
   case 2: // Do case 2
    // Add Execution of DoCase2 to Delegate
    myDelegate +=  new mySwitchDelegate(DoCase2);
    break;
  }

  // Execute myDelegate with current myInt
  myDelegate(myInt);

  Console.ReadLine();
 }
}
```

אם נבחר myInt שהוא 1 יודפס (פעם אחת) Case 1 execution ואם נבחר myInt שהוא 2 יודפס (פעם אחת) Case 2 execution. בשיטה הזאתי הכל הרבה יותר נקי ומסודר ואפשר לפתח עץ לוגי שלם עם חזרות בתוך הצהרת ה-switch.

עוד שאלות נפוצות הן פשוטות ובסיסיות, למשל:

- מהו ממשק?
ממשק מאפשר לחייב כל מחלקה שיורשת אותו להכיל את המתודות אשר כתובות בו ובכך ניתן לאפשר פולימורפיזם וגם תאימות בין חלקים שונים של הפריימוורק למחלקה.

- האם אפשר לבצע הורשה מרובה מממשקים?
כן, אין סיבה שלא.

- כיצד יוצרים מערך מסוג זה או זה?

- מה הם Jagged arrays?
מערך של מערכים בגדלים שונים.

- מה ההבדל בין Struct ל-Class?
Struct ו-Class -ההבדלים והכוח הטמון בשוני ביניהם

- שאלות על מילת מפתח sealed, כגון: כיצד ניתן למנוע ירושה ממחלקה? כיצד ניתן למנוע דריסה של מתודה?

- איזה חתימות אפשריות יש לפונקצית Main?
public static void Main ()
public static int Main ()
(public static void Main(string[] args
(public static int Main(string[] args

- מה ההבדל בין Delegate ל-event?
Delegate אפשר לרוקן ממתודות, ו-event אי-אפשר לרוקן ממתודות. event הוא הדרך לחשוף delegateים ב-OOP.

- מהי אסמבלי?

- שאלות על Global assembly cache, למשל: מה ההבדל בין Shared assembly ל-private assmebly? מהו strongname, כיצד מייצרים אחד ומה הוא מכיל?
Shared assembly היא assembly שנמצאת ב-Global assembly cache (או GAC בקיצור), Private assembly בשימוש באפליקציה אחת בלבד. Strong name הוא שם שמזהה אפליקציה באופן ספציפי ומכיל גירסה, תרבות, שם האסמבלי ומפתח זיהוי יחודי. כל אסמבלי ב-Global assembly cache צריכה Strong name.

לסיכום

ניסיתי במאמר הזה לסקור שאלות טכניות אשר נפוצות במבחנים טכניים בראיונות עבודה. השאלות התמקדו באופן כללי בתחום הדוט נט, ובנוסף כמובן ניתן לשאול שאלות בתחומים יותר ספציפיים כמו עבודה עם מסדי נתונים, ASP.net, Winforms, אבטחה, פריסת ישומים, Webservices ועוד.

אם יש לכם שאלות נוספות, שאלות\הערות\תגובות על התשובות שלי, או כל דבר אחר – תרגישו חופשיים להשאיר תגובה.

עד כאן זה שאלות שנשאלו אנשים עם נסיון.

שאלות נוספות שנשאלו תלמידות מוולף: