

PeTel Tester C#

ממשק לבדיקת קוד בסי שארפ
גלעד מרקמן



כללי

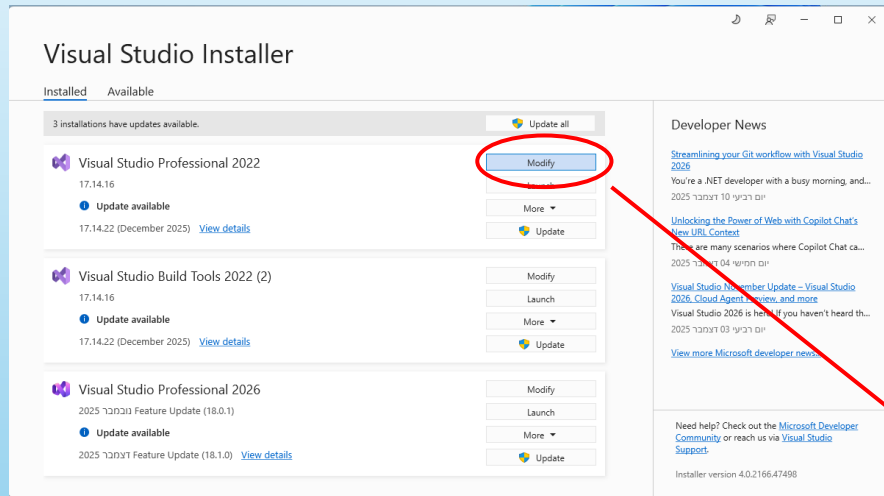
- ממשק בדיקת הקוד כולל שלושה חלקים:
- **Case Tester** – בדיקה של תוצאות הרצת קוד התלמיד לעומת פתרון המורה.
- **Code Tester** – בדיקה של תחביר התשובה של התלמיד (האם יש קריאה רקורסיבית, האם הפונקציה סטטית וכד').
- **AI Tester** – בדיקת הקוד באמצעות מודל בינה מלאכותית (בפיתוח)
- פיתוח בדיקת הקוד מתבצע בסביבת העבודה Visual Studio
- הרצת בדיקות על פתרונות אפשריים של תלמידים.
- לאחר סיום הפיתוח העלאה של הקבצים לסביבת פטל.



התקנות



הכנה של ויזואל סטודיו



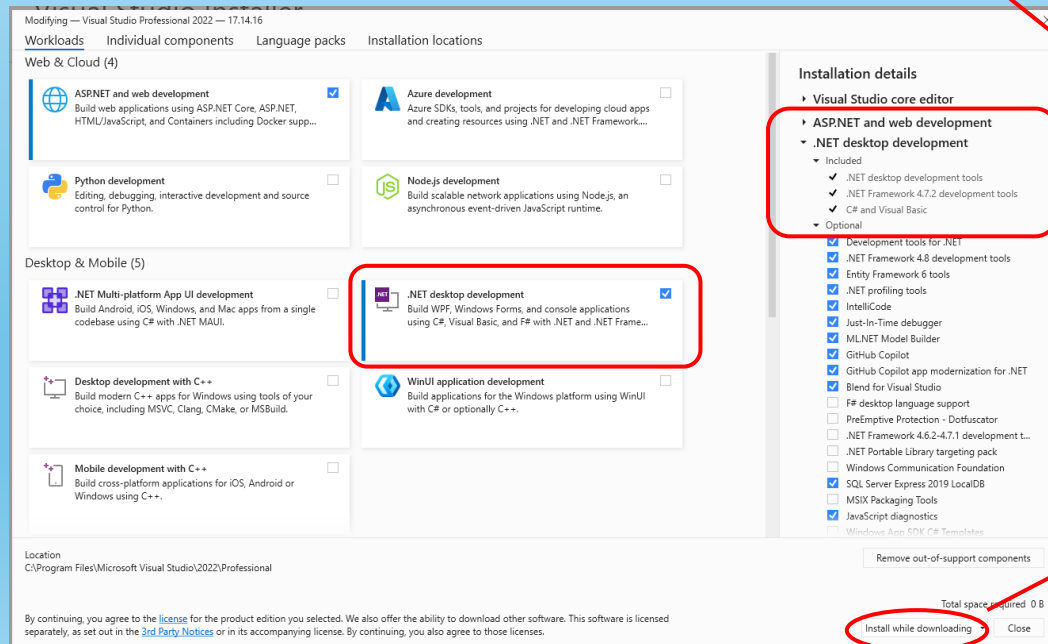
• סביבת הפיתוח שלנו היא Visual Studio Community. התקנה חינמית: Visual Studio

• על מנת לוודא כי הרכיבים הנדרשים מותקנים יש להריץ את Visual Studio Installer -> ללחוץ על Modify.

• לוודא ש .Net desktop development מותקן.

• לוודא שכולל את .Net Framework 4.7.2.

• אם חסר – להתקין.

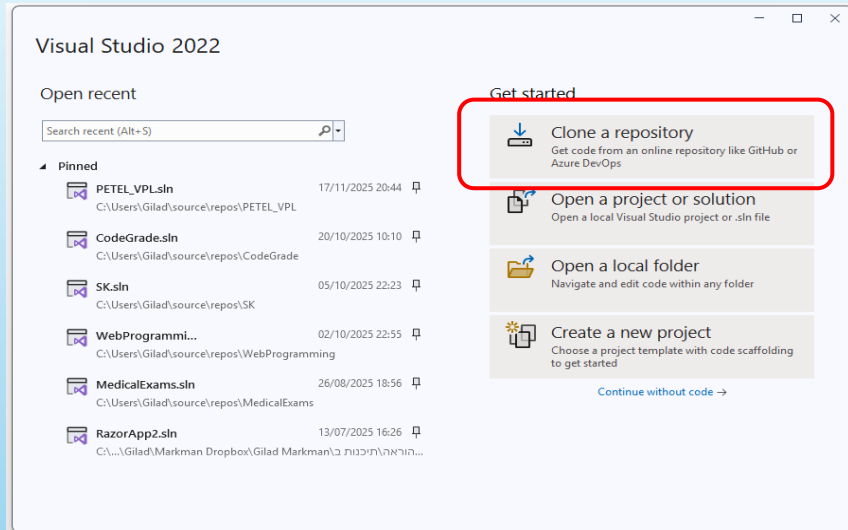


התקנה של הממשק

- הממשק נמצא GitHub פומבי לפי הכתובת הבאה:
https://github.com/MarkmanGilad/PETEL_Tester
- ניתן להוריד את הממשק בשתי דרכים חלופיות:
 - לבצע cloning לפי הכתובת לעיל (נדרשת התקנה של Git במחשב).
 - הורדת הפרויקט כקובץ zip, שמירת הקובץ על המחשב האישי, פתיחת הקובץ, והרצתו.
- נדגים את שתי האפשרויות.



Cloning

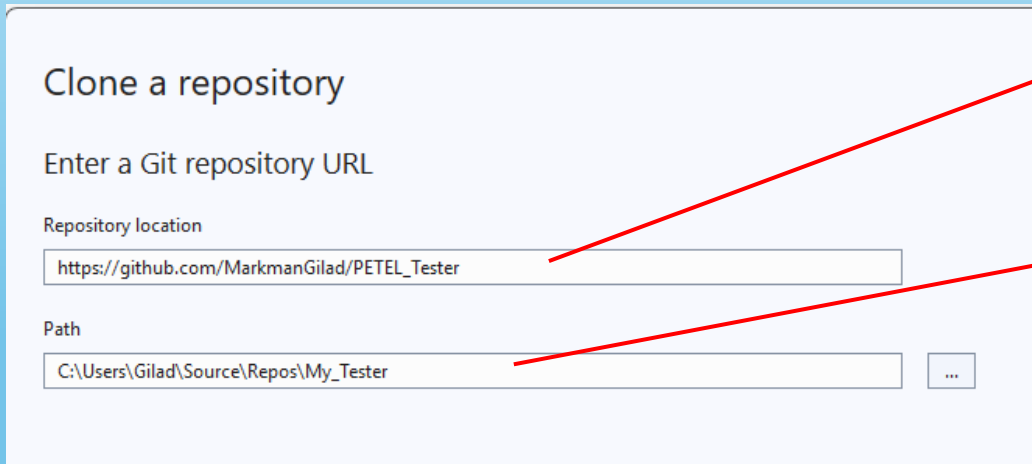


- במסך הפתיחה של ויזואל סטודיו 2022 יש לבחור clone a repository.

- בתיבת המיקום יש להכניס את כתובת הפרויקט בגיטהב:

https://github.com/MarkmanGilad/PETEL_Tester

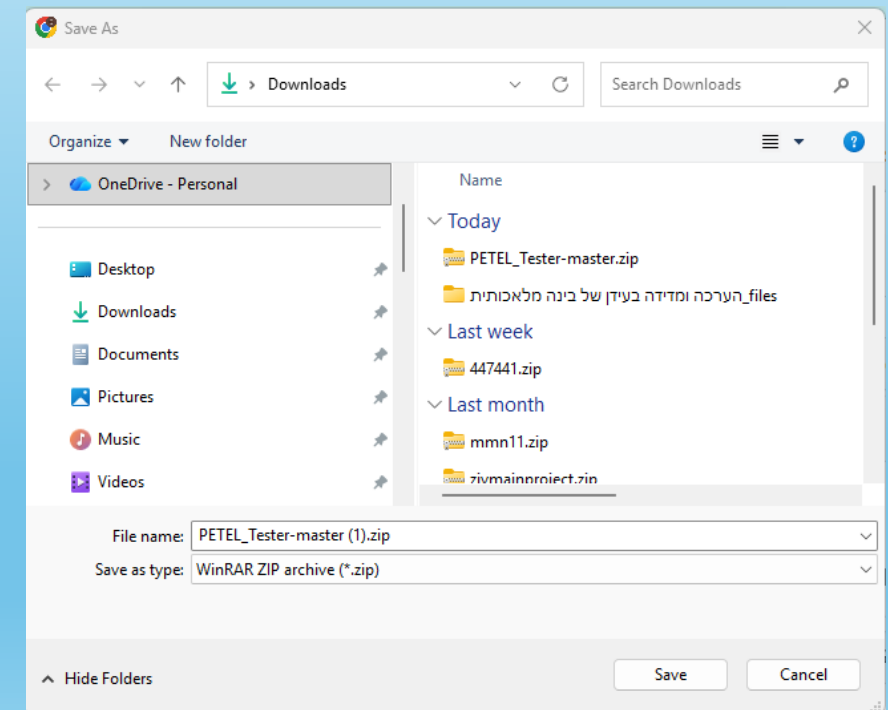
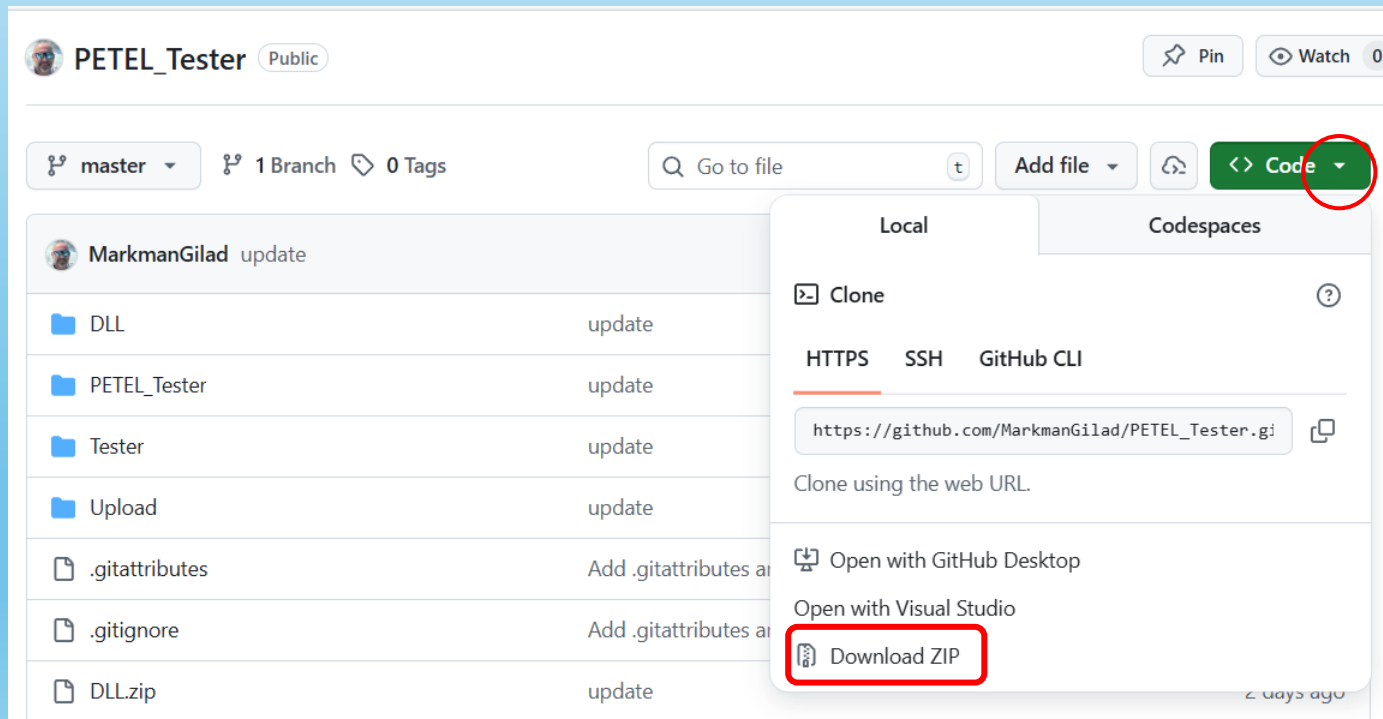
- ניתן להשאיר את הנתיב לשמירת הפרויקט במחשב שלנו ללא שינוי או לבחור מיקום אחר.



- ללחוץ על כפתור clone.

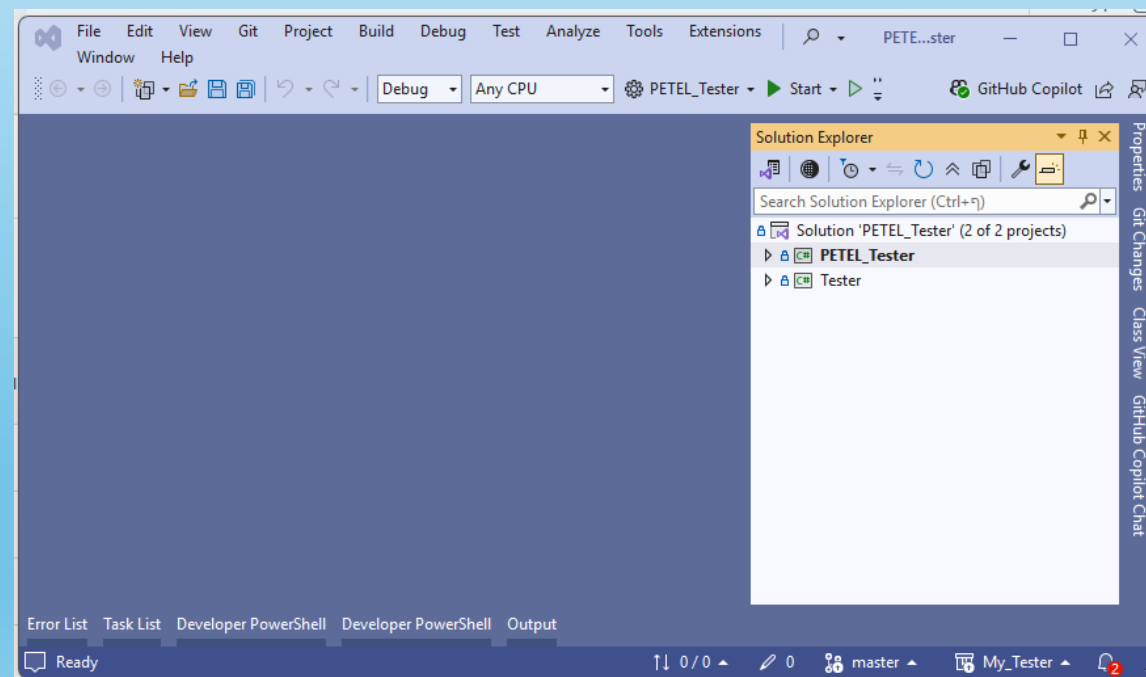
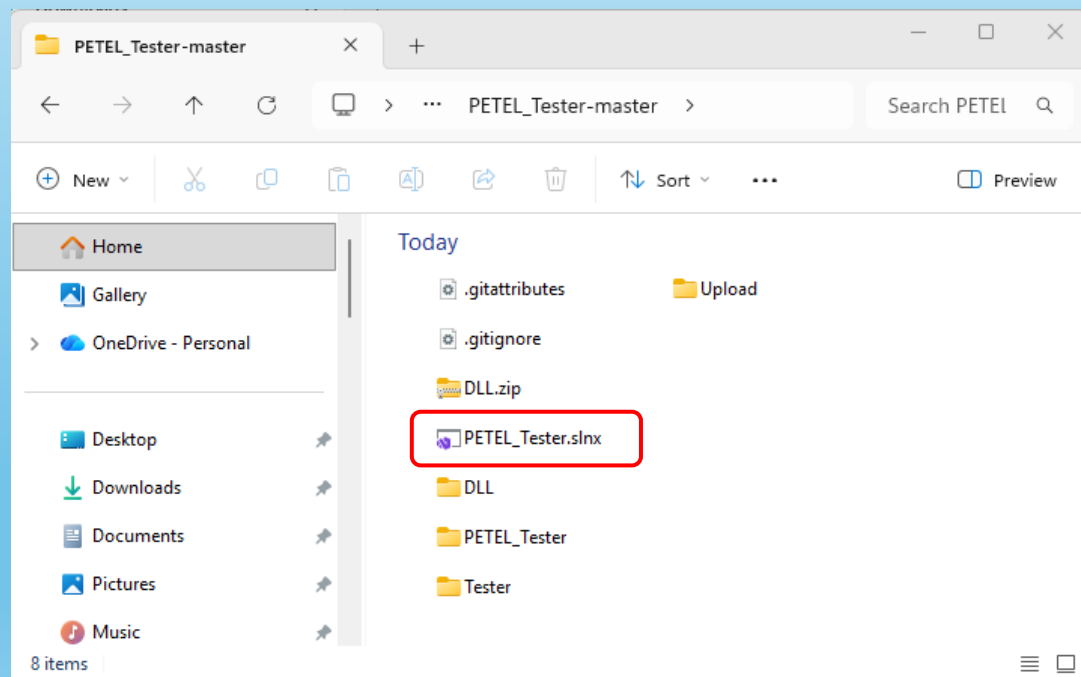
הורדת הפרויקט למחשב (חלופי)

- באתר גיטהב: https://github.com/MarkmanGilad/PETEL_Tester יש ללחוץ על החץ ליד Code, ולבחור Download Zip.
- לשמור את הקובץ במחשב שלנו.



המשך הורדת הפרויקט

- לאחר הורדת קובץ Zip למחשב יש לפתוח (Extract).
- להכנס לספרייה PETEL_Tester-master ולחיצה כפולה על PETEL_Tester.slnx תפעיל את הפרויקט.



הכנת סביבת העבודה למשימה חדשה



יצירת פרויקט חדש

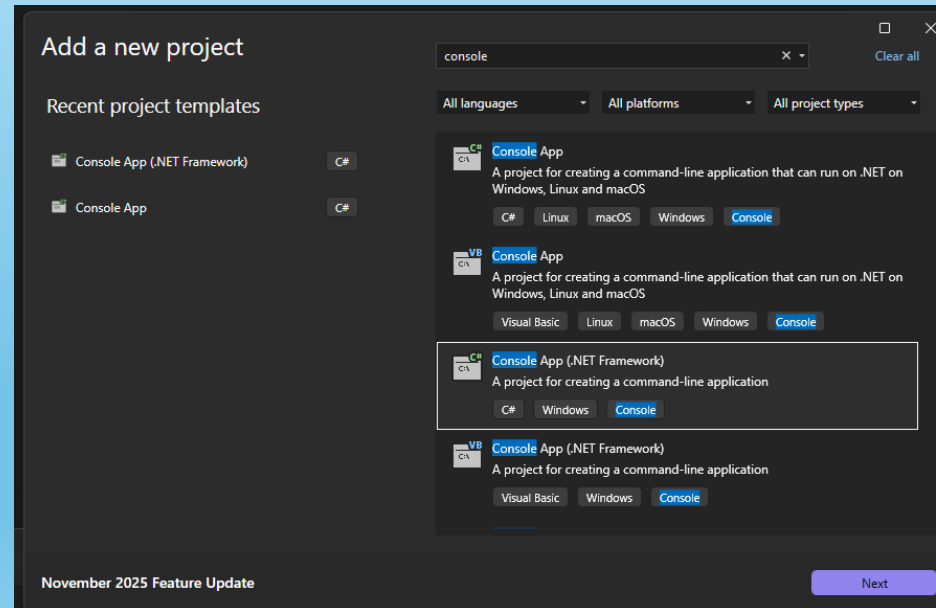
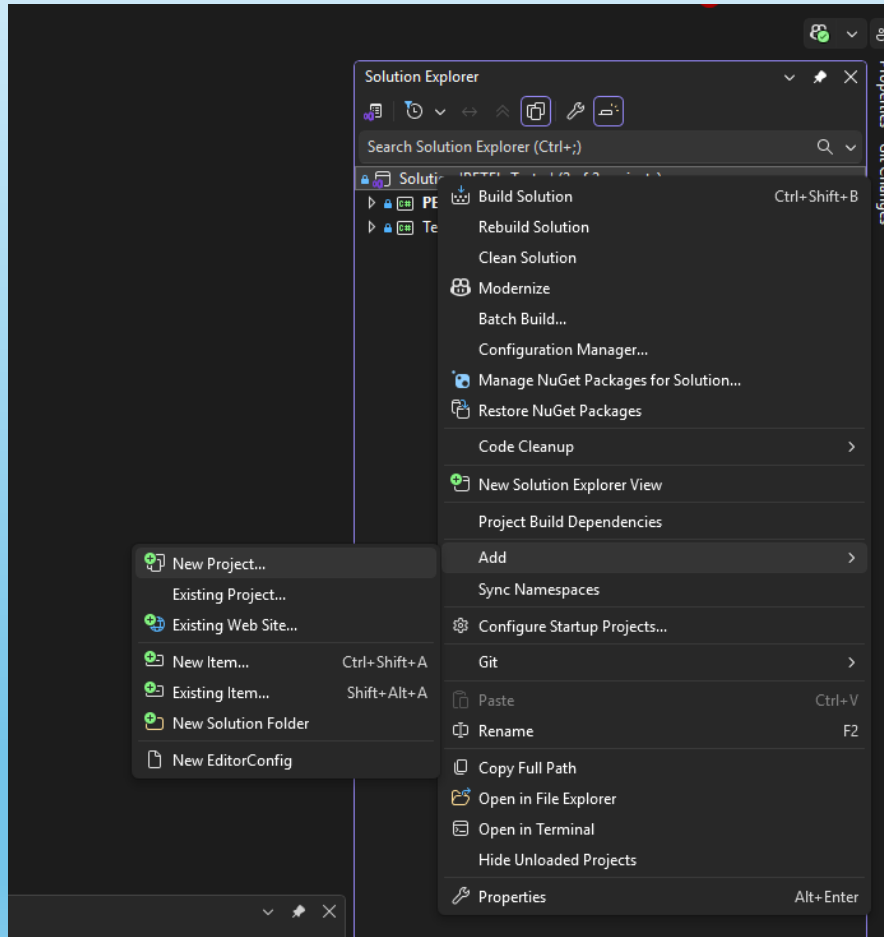
• לחיצה ימנית על ה Solution

• Add -> New Project ...

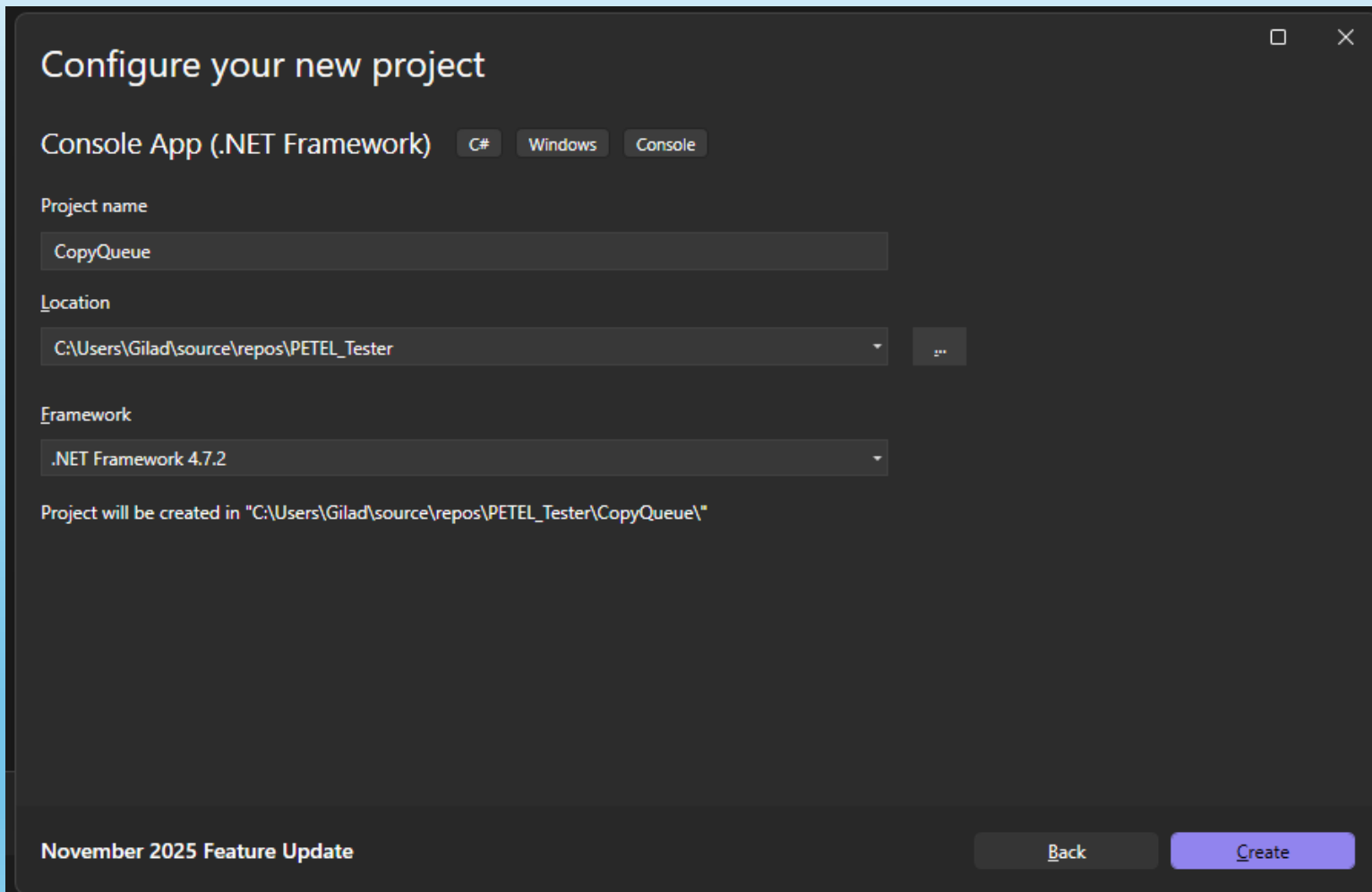
• לבחור פרויקט מסוג:

Console App (.Net Framework) C#

• ללחוץ Next



יצירת פרויקט חדש - המשך



Configure your new project

Console App (.NET Framework) C# Windows Console

Project name
CopyQueue

Location
C:\Users\Gilad\source\repos\PETEL_Tester

Framework
.NET Framework 4.7.2

Project will be created in "C:\Users\Gilad\source\repos\PETEL_Tester\CopyQueue\"

November 2025 Feature Update

Back Create

• לבחור שם לפרויקט.

• ללחוץ Create.

התקנת ספריות נחוצות

- על מנת להתקין ספריות נחוצות נלחץ קליק ימני על ה Solution ונבחר:

- Manage NuGet Packages for Solution ...

- בחלון שיפתח:

- נבחר Installed.

- נבחר Microsoft.CodeAnalysis.Csharp

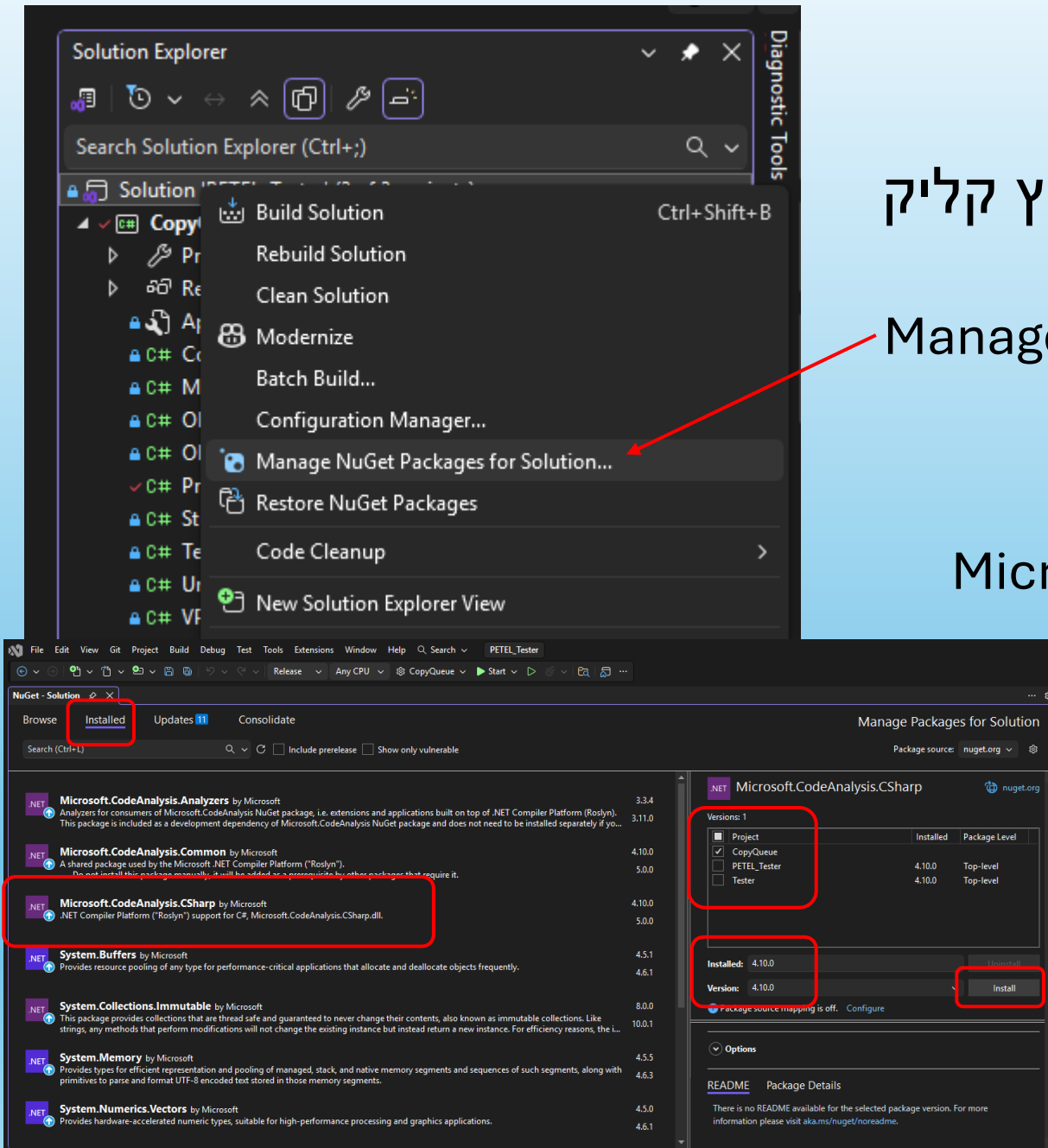
- נסמן V בשם הפרויקט שלנו.

- נוודא שהגרסה היא 4.10.0

- ***** אסור לעדכן גירסה *****

- נלחץ Install.

- נאשר Accept I



העתקת קבצים לפרויקט החדש

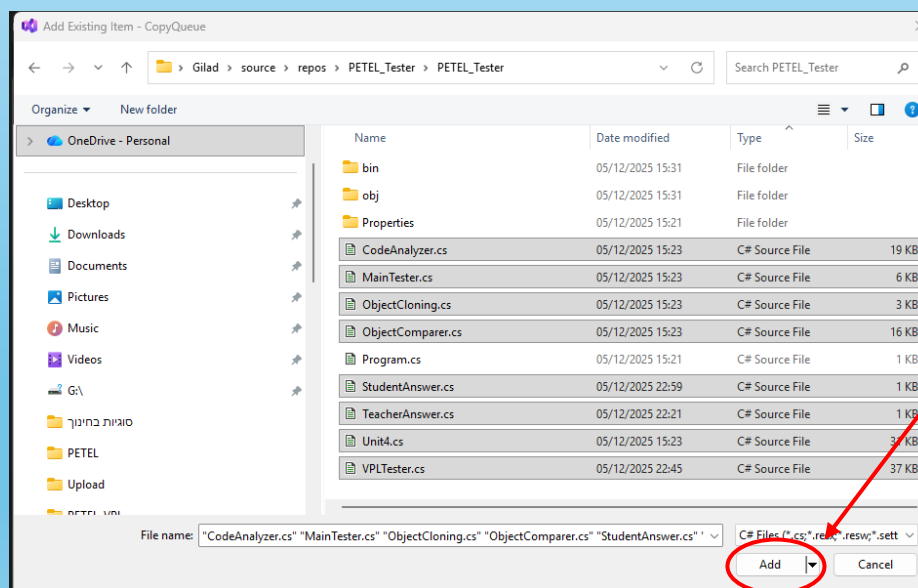
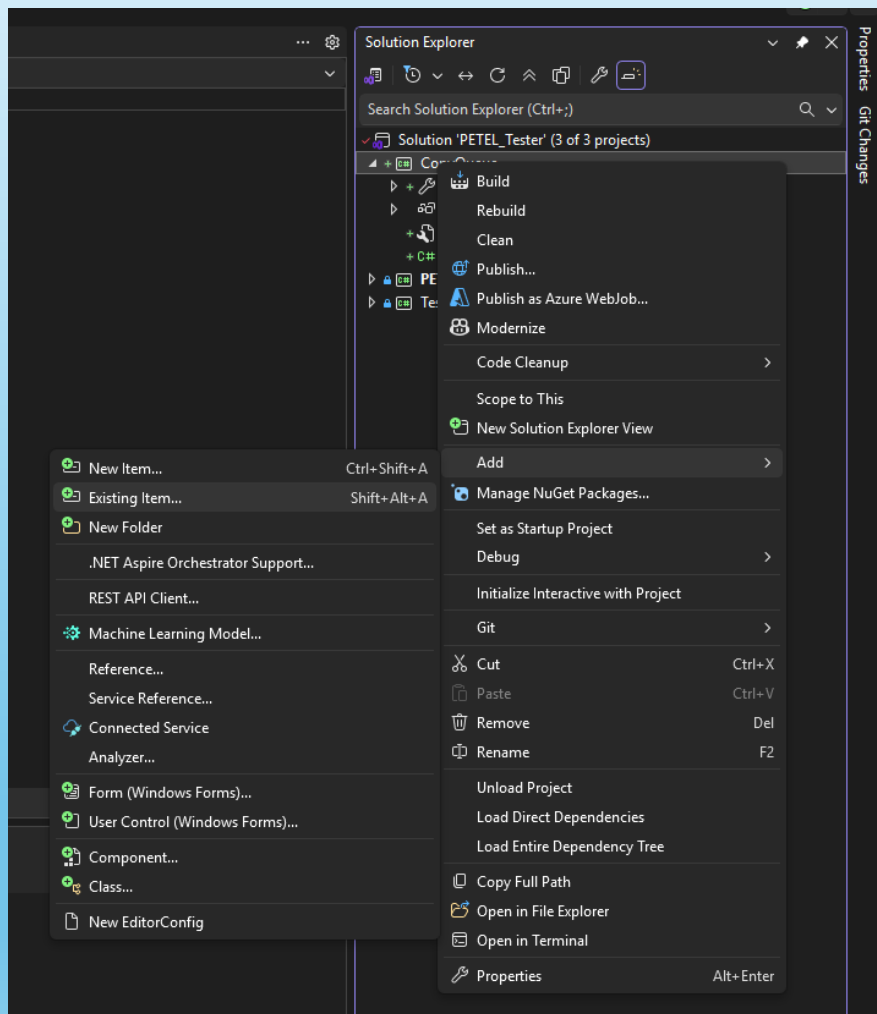
• לחיצה ימנית על הפרויקט ולבחור:

• Add -> Existing Item ...

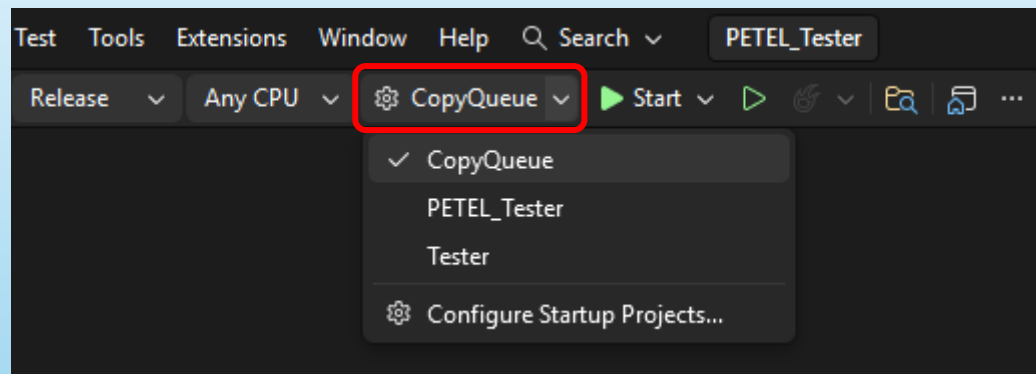
• להגיע לספריה PETEL_Tester/PETEL_Tester.

• לסמן ולהוסיף את כל הקבצים למעט Program.cs

• ללחוץ Add



קביעת פרויקט ונקודת כניסה

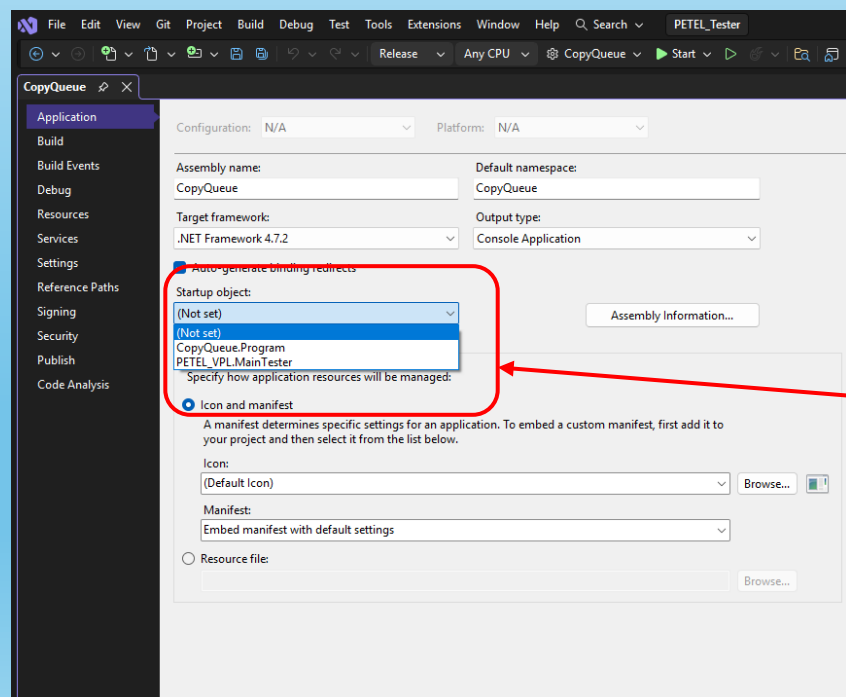


- נבחר את הפרויקט שאנחנו מריצים.

- הפרויקט כולל שתי פעולות Main:

- במחלקה Program.cs

- במחלקה MainTetser.cs



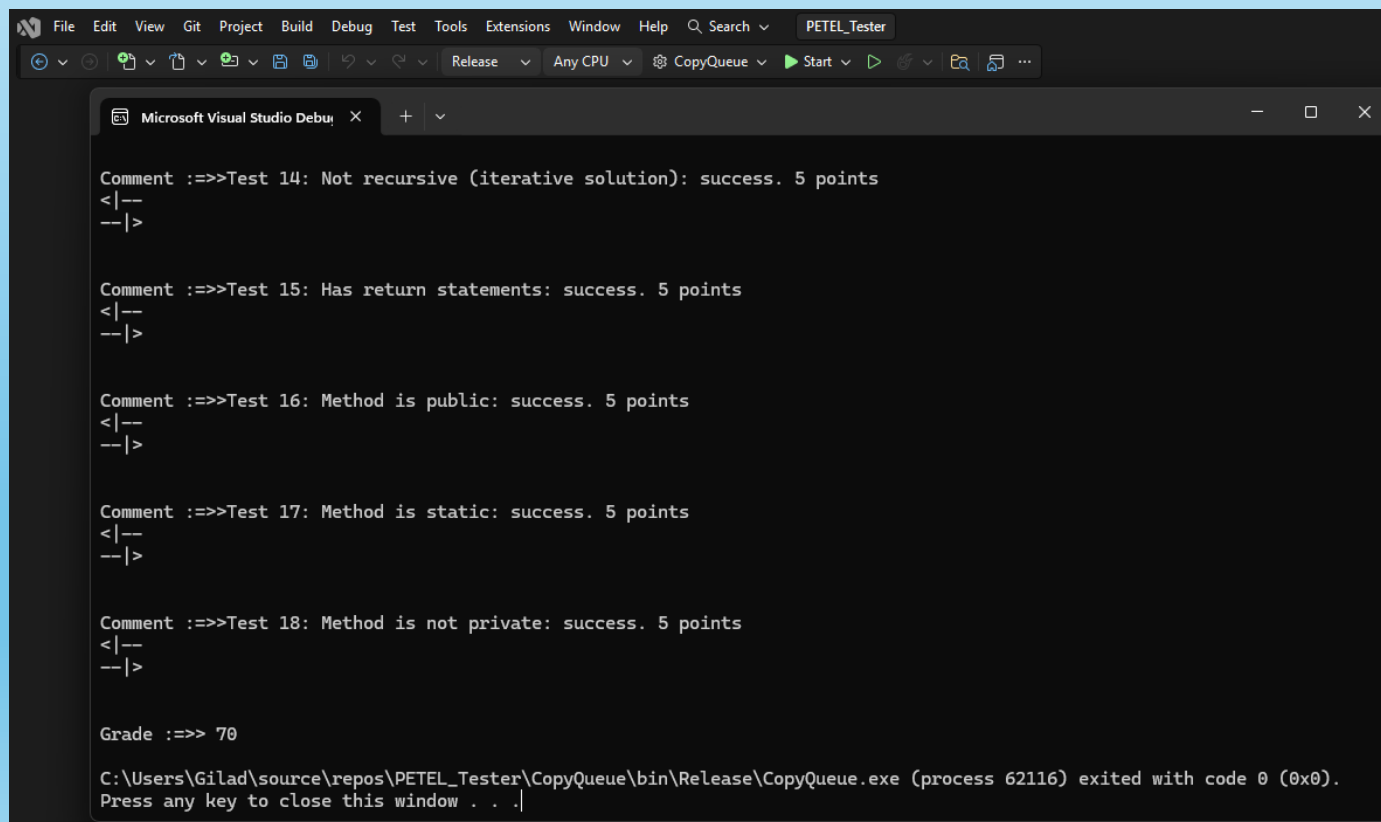
- קליק ימני על שם הפרויקט ולחיצה על Properties.

- בחירת Main_Tetser כנקודת הכניסה שלנו.

- נשמור את ההגדרות ctrl-s

הפרויקט מוכן לפיתוח

- הפרויקט כולל את כל הקבצים הנדרשים להכנת משימה חדשה.
- ניתן להריץ את הפרויקט ctrl-F5 ולראות את תוצאות ההרצה של תרגיל לדוגמה: `countRemoveItem()`.



```
Microsoft Visual Studio Debug Console
PETEL_Tester

Comment :>>>Test 14: Not recursive (iterative solution): success. 5 points
<|--
--|>

Comment :>>>Test 15: Has return statements: success. 5 points
<|--
--|>

Comment :>>>Test 16: Method is public: success. 5 points
<|--
--|>

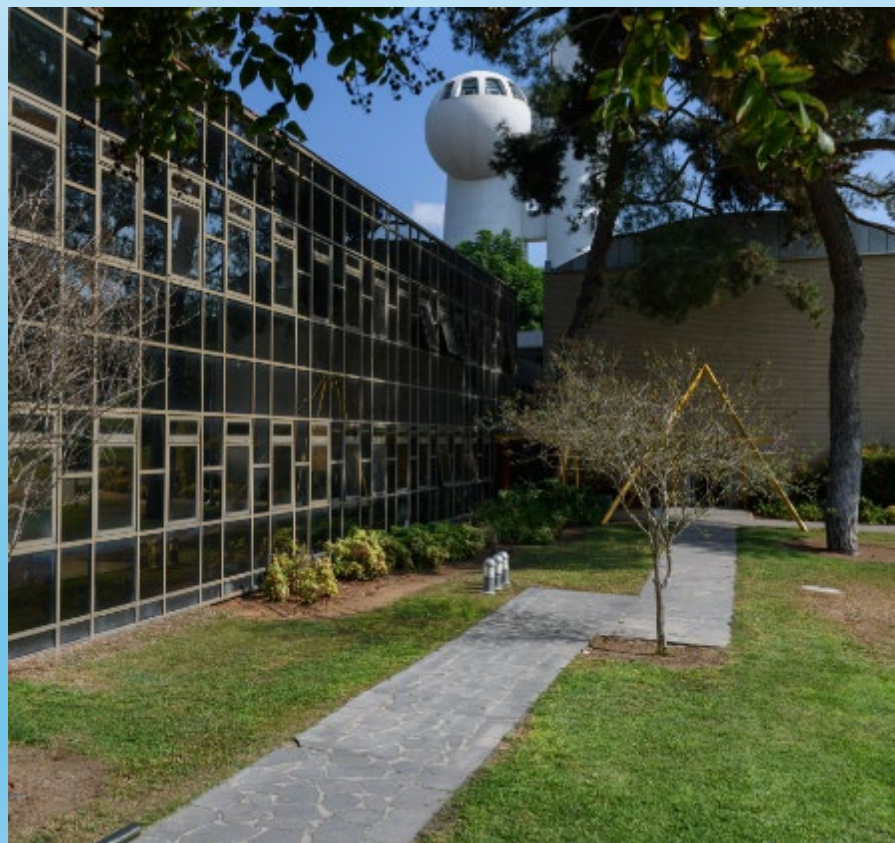
Comment :>>>Test 17: Method is static: success. 5 points
<|--
--|>

Comment :>>>Test 18: Method is not private: success. 5 points
<|--
--|>

Grade :>>> 70

C:\Users\Gilad\source\repos\PETEL_Tester\CopyQueue\bin\Release\CopyQueue.exe (process 62116) exited with code 0 (0x0).
Press any key to close this window . . .|
```

בניית Tester



כללי

- בניית Tester מחייבת עדכון שלושה קבצים:
 - MainTeter.cs - קובץ הבדיקות.
 - TeacherAnswer.cs – קובץ הפתרון המשמש בסיס להשוואה לקוד התלמיד.
 - StudentAnswer.cs – דוגמה לפתרון תלמיד לבדיקת גילוי שגיאות שונות.
- אם מבקשים לבדוק את תוצאות הרצת הפתרונות של המורה או של התלמיד, ניתן לזמן אותם מהקובץ Program.cs ולהגדיר אותו כנקודת כניסה.
- אם רוצים לבדוק כיצד הטסטר בודק את פתרון התרגיל, מגדירים את MainTester כנקודת כניסה ומריצים את הקוד.
- המחלקה Unit4 כוללת את כל המחלקות הנדרשות ליחידה מבנה נתונים ובנוסף כוללת Helper ליצירת והדפסת מבני נתונים (ראו בהמשך).

בדיקת התשובה של המורה

Unit4Helper



פתרון המורה והתלמיד

- נבנה את פתרון המורה ואת פתרון התלמיד לתרגיל של העתקת תור.
- פתרון התלמיד כולל שגיאה בכך שלא שומר על התור המקורי.

```
StudentAnswer.cs TeacherAnswer.cs
CopyQueue
1 using System;
2 using Unit4;
3
4 namespace PETEL_VPL
5 {
6     class TeacherAnswer
7     {
8         public static Queue<int> Copy(Queue<int> q)
9         {
10             Queue<int> q1 = new Queue<int>();
11             Queue<int> temp = new Queue<int>();
12             while (!q.IsEmpty())
13             {
14                 int item = q.Remove();
15                 temp.Insert(item);
16                 q1.Insert(item);
17             }
18             while (!temp.IsEmpty())
19             {
20                 q.Insert(temp.Remove());
21             }
22             return q1;
23         }
24     }
25 }
```

```
StudentAnswer.cs TeacherAnswer.cs
CopyQueue
1 using Unit4;
2
3 class StudentAnswer
4 {
5     public static Queue<int> Copy(Queue<int> q)
6     {
7         Queue<int> q1 = new Queue<int>();
8         while (!q.IsEmpty())
9         {
10             int item = q.Remove();
11             q1.Insert(item);
12         }
13         return q1;
14     }
15 }
16
17
```

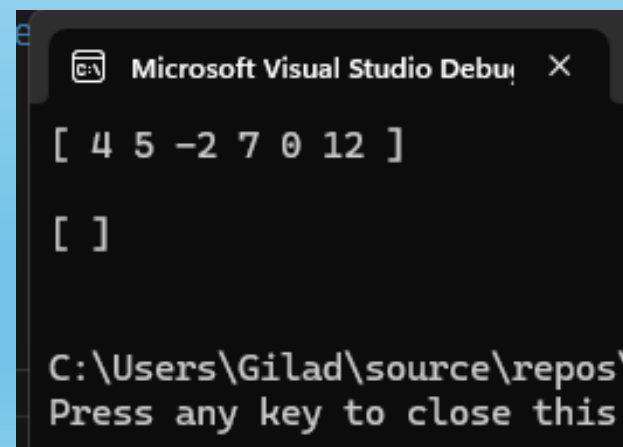
הרצת הקוד לבדיקה – Unit4Helper

- נגדיר את Program.cs כנקודת הכניסה באמצעות המאפיינים של הפרויקט.
- נעדכן את using בהתאם לתמונה.
- ניצור תור לדוגמה.
- נקרא לפעולה של התלמיד.
- נדפיס תוצאות

```
using System;
using Unit4;

namespace CopyQueue
{
    internal class Program
    {
        static void Main(string[] args)
        {
            RunStudentCode();
        }

        public static void RunStudentCode()
        {
            Queue<int> q1 = Unit4Helper.BuildQueue(
                new int[] { 4, 5, -2, 7, 0, 12 });
            Queue<int> q2 = StudentAnswer.Copy(q1);
            Console.WriteLine(q2);
            Console.WriteLine(q1);
        }
    }
}
```



Microsoft Visual Studio Debug Console

[4 5 -2 7 0 12]

[]

C:\Users\Gilad\source\repos\

Press any key to close this

פעולות ב Unit4Helper

```
public static void NodeList()
{
    Node<int> lst = Unit4Helper.BuildNodeList(new int[] { 4, -2, 7, 0, -1, 0, 0 });
    Unit4Helper.PrintList(lst);
    int[] arr = Unit4Helper.NodeListToArray(lst);
    string str = Unit4Helper.NodeListToString(lst);
    Console.WriteLine(str);
}
```

• רשימת חוליות

```
public static void QueueMethods()
{
    Queue<int> q = Unit4Helper.BuildQueue(new int[] { 4, -2, 7, 0, -1, 0, 0 });
    Console.WriteLine(q);
    int[] arr = Unit4Helper.QueueToArray(q);
}
```

• תור

```
public static void StackMethods()
{
    Stack<int> s = Unit4Helper.BuildStack(new int[] { 4, -2, 7, 0, -1, 0, 0 });
    Console.WriteLine(s);
    int[] arr = Unit4Helper.StackToArray(s);
}
```

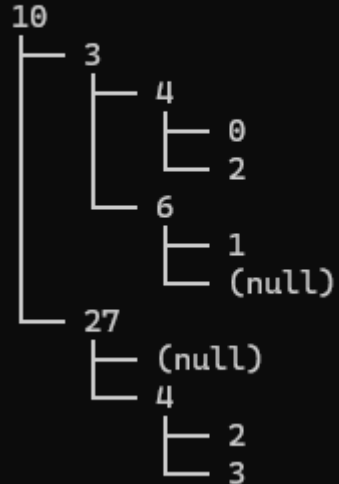
• מחסנית

פעולות עבור עצים

- הוספנו פעולה המאפשרת לבנות עצים מקובץ טקסט.
- אנחנו בונים קובץ טקסט בצורה הבאה. יש לעשות שימוש בטאב לירידה ברמה.
- הפעולה קוראת את העץ מהקובץ ויוצרת אובייקט של עץ בינארי.
- הוספנו הדפסה של העצים.

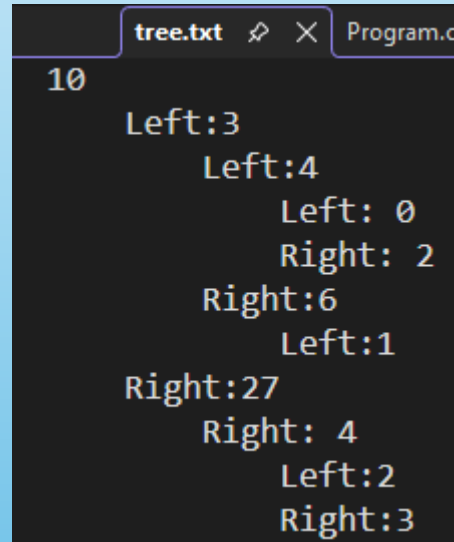
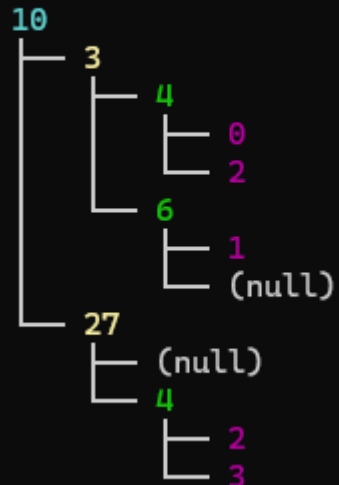
Binary Tree Structure:

=====



Binary Tree Structure:

=====



```
public static void BinTreeMethods()
{
    string path = Unit4Helper.GetTreeFilePath("tree.txt");
    BinNode<int> tree = Unit4Helper.BuildBinaryTree<int>(path);
    Unit4Helper.PrintBinaryTree(tree);
    Unit4Helper.PrintBinaryTreeColored(tree);
    string str = Unit4Helper.BinaryTreeToString(tree);
    Console.WriteLine(str);
}
```

בניית Case Test



MainTester

- לאחר שבנינו את תשובת המורה ותשובת התלמיד נוכל לעבור לבניית הבדיקה עצמה.
- יש להגדיר את MainTester כנקודת הכניסה של הפרויקט (במקום Program).
- המחלקה בנויה משלוש פעולות סטטיות:

```
class MainTester
{
    public static void Main(string[] args)...
    private static void CaseTester(VPLTester tester)...
    private static void CodeTester(VPLTester tester)...
}
```

- Main – הפעולה הראשית
- CaseTester – בדיקת הרצת הקוד.
- CodeTester - בדיקה תחביר התשובה של התלמיד.

Main

```
public static void Main(string[] args)
{
    // Initialize the tester
    var tester = new VPLTester(
        studentFile: "StudentAnswer.cs",
        studentNamespace: "",
        studentClassName: "StudentAnswer",
        studentMethodName: "Copy",
        teacherNamespace: "PETEL_VPL",
        teacherClassName: "TeacherAnswer",
        teacherMethodName: "Copy",
        showDetails: true
    );

    // Run all test suites
    CaseTester(tester);
    CodeTester(tester);

    // Display results (VPL parses this output)
    Console.WriteLine("\n" + tester.FormatResponse());
    Console.WriteLine($"Grade :=>> {tester.GetGrade()}");
}
```

- בשלב ראשון עלינו להגדיר אובייקט של הטסטר:
 - StudentFile – שם הקובץ ב VPL
 - StudentNamespace – אם לא הוגדר יש להשאיר ריק.
 - StudentMethodName – שם הפעולה הנבדקת.
 - teacherNamespace – בהתאם לפתרון המורה (מומלץ לא לשנות).
 - teacherMethodNake – שם הפעולה הנבדקת בקוד המורה.
 - showDetails – האם לתת תשובה מפורטת לתלמיד.
- פרט לכך, אין צורך לעשות שינויים.

CaseTester

- דוגמה לבדיקה של הפעולה העתקת קוד. הבדיקה נכשלה כיוון שהתלמיד שינה את התור המקורי.

```
private static void CaseTester(VPLTester tester)
{
    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return",
        points: 10,
        parameters: new object[] { q1 }
    );
}
```

```
Comment :=>>Test 1: check the correct return | params: p0=[ 3 5 -9 3 5 5 2 1 2 ]: failure. 0 points
<|--
Return value check:
Expected: Queue<[3, 5, -9, 3, 5, 5, 2, 1, 2]>
Actual:   Queue<[3, 5, -9, 3, 5, 5, 2, 1, 2]>
Explanation: Returned value matches the expected result.

Input parameter state after call does not match the requirement:
p0: expected=Queue<[3, 5, -9, 3, 5, 5, 2, 1, 2]> | actual=Queue<[]>
--|>

Grade :=>> 0
```

השלבים לבניית בדיקה

- בניית הפרמטרים (הדוגמאות), כגון רשימה, תור וכד'.
- קריאה לפעולת הבדיקה TestMethod:
- testName – שם הבדיקה ניתן לכלול פירוט לתלמיד.
- Points – מספר הנקודות שהבדיקה שווה.
- Parameters - רשימה של אובייקטים הכולל את הפרמטרים המועברים לפעולה.
- ניתן לבנות מספר בדיקות בזו אחר זו.

```
private static void CaseTester(VPLTester tester)
{
    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return",
        points: 10,
        parameters: new object[] { q1 }
    );
}
```

הפרמטרים של TestMethod

פרמטר	סוג	הסבר
testName	String	* שם הבדיקה והסבר קצר לתלמיד
Points	Int	* מספר הנקודות שיקבל התלמיד אם יעבור את הבדיקה
Parameters	Object []	מערך אובייקטים הכולל את הפרמטרים שיש להעביר לפעולה.
consoleInput	String , string [], List<string>	הקלט במקרה והפעולה דורשת read. הקלט יכול להיות כל אחד מהסוגים המצויינים. במקרה של לולאת קלט ניתן לתת מערך של מחרוזות או רשימה של מחרוזות.
captureConsoleOutput	Bool	האם להשוות את הפלט לקונסול במקרה והפעולה כוללת write ?
compareParams	Bool (default: true)	האם הפעולה שינתה את הפרמטרים (משווה את הפרמטרים לאחר הפעולה) ?
exceptionComments	Dictionary<Type , string>	אופציונאלי: אפשרות לשנות את הודעת השגיאה במקרה של Exception אצל התלמיד.
compareReturn	Bool (default: true)	אופציונאלי: האם לבדוק את הערך המוחזר מהפעולה?
* פרמטר חובה. השאר אופציונאלי		

דוגמאות לבדיקות

```
private static void CaseTester(VPLTester tester)
{
    tester.TestMethod(
        testName: "Test 1: 3 numbers. capture Console Output",
        points: 10,
        parameters: new object[] { 3 },
        compareParams: false,
        consoleInput: new string[] { "5", "3", "7" },
        captureConsoleOutput: true
    );

    tester.TestMethod(
        testName: "Test 2: 3 numbers. Only Compare Return",
        points: 10,
        parameters: new object[] { 3 },
        compareParams: false,
        consoleInput: new string[] { "5", "3", "4" },
        captureConsoleOutput: false
    );

    tester.TestMethod(
        testName: "Test 3: 1 numbers",
        points: 10,
        parameters: new object[] { 1 },
        compareParams: false,
        consoleInput: "3",
        captureConsoleOutput: true,
        compareReturn: false
    );
}
```

- תרגיל: כתוב פעולה המקבלת מספר שלם num. הפעולה תקלוט מחירים של מוצרים כמספר ה num. הפעולה תדפיס את סכום המוצרים ותחזיר את הממוצע שלהם.

```
Comment :=>>Test 1: 3 numbers. capture Console Output | params: p0=3 | input: "5\n3\n7\n": success. 10 points
<|--
output: "Enter price\nEnter price\nEnter price\nThe Sum is: 15\n"
return: 5
--|>

Comment :=>>Test 2: 3 numbers. Only Compare Return | params: p0=3 | input: "5\n3\n4\n": success. 10 points
<|--
return: 4
--|>

Comment :=>>Test 3: 1 numbers | params: p0=1 | input: "3\n": success. 10 points
<|--
output: "Enter price\nThe Sum is: 3\n"
--|>

Grade :=>> 30
```

דוגמאות לבדיקה

```
private static void CaseTester(VPLTester tester)
{
    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return. Don't check the original Queue ",
        points: 10,
        parameters: new object[] { q1 },
        compareParams: false
    );

    Queue<int> q2 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 2: check only the original Queue if it changed",
        points: 10,
        parameters: new object[] { q2 },
        compareParams: true,
        compareReturn: false
    );

    q2 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 2: check both return and original",
        points: 10,
        parameters: new object[] { q2 },
        compareParams: true,
        compareReturn: true
    );
}
```

- כתבו פעולה המעתיקה תור מבלי לפגוע בתור המקורי.
- בדיקה ראשונה – בודק רק את התור המוחזר ולא בודק אם התור המקורי השתנה.
- בדיקה שניה – בודק רק אם התור המקורי השתנה.
- בדיקה שלישית – בודק את שניהם

הוספת הודעת שגיאה מותאמת אישית

- באפשרותנו להוסיף להודעות השגיאה של המערכת הודעות שגיאה מותאמת אישית.
- לדוגמה הודעת השגיאה:

```
Comment :=>>Test 1: check the correct return. Don't check the original Queue | params: p0=[ 3 5 -9 3 5 5 2 1 2 ]: failure. 0 points
<|--
Error during test execution: Object reference not set to an instance of an object.
--|>
```

- לאחר השינוי הוספה ההודעה הבאה:

```
Comment :=>>Test 1: check the correct return. Don't check the original Queue | params: p0=[ 3 5 -9 3 5 5 2 1 2 ]: failure. 0 points
<|--
Error during test execution: Object reference not set to an instance of an object.
Teacher note: You tried to remove from empty queue
--|>
```

הוספת הודעת שגיאה מותאמת אישית

- עלינו להגדיר מילון בו המפתח הוא סוג השגיאה והערך הודעת השגיאה.
- בפרמטרים של הבדיקה נוסיף את המילון ב exceptionComments.

```
private static void CaseTester(VPLTester tester)
{
    // Optional: Custom exception message
    var commonExceptionComments = new C.Dictionary<Type, string>
    {
        { typeof(NullReferenceException), "You tried to remove from empty queue" },
    };

    Queue<int> q1 = Unit4Helper.BuildQueue(new int[] { 3, 5, -9, 3, 5, 5, 2, 1, 2 });
    tester.TestMethod(
        testName: "Test 1: check the correct return. Don't check the original Queue ",
        points: 10,
        parameters: new object[] { q1 },
        compareParams: false,
        exceptionComments: commonExceptionComments
    );
}
```


בניית Code Test



בדיקת תחביר הקוד

- במערכת הבדיקות ניתן לבצע בנוסף בדיקה תחבירית של הקוד שכתב התלמיד כגון: האם הפעולה רקורסיבית, האם הפעולה סטטית, האם הפרמטרים של הפעולה תקינים, האם הערך המוחזר תקין ועוד.
- דוגמה – בדיקה האם התלמיד השתמש בלולאות מכוונות:

```
private static void CodeTester(VPLTester tester)
{
    // Initialize code analyzer (handles errors internally)
    tester.InitializeCodeAnalyzer();

    // Test 11: Verify no nested loops (ensures O(n) not O(n²))
    tester.TestCodeStructure(
        testName: "Test 11: No nested loops (O(n) complexity)",
        points: 10,
        checkType: CodeStructureCheck.HasNestedLoops,
        shouldPass: false, // We want HasNestedLoops to return FALSE
        failureMessage: "Method must not have nested loops to maintain O(n) complexity"
    );
}
```

בניית בדיקת תחביר

- בתחילת הפעולה CodeTester עלינו לאתחל את CodeAnalyzer.

```
private static void CodeTester(VPLTester tester)
{
    // Initialize code analyzer (handles errors internally)
    tester.InitializeCodeAnalyzer();

    // Test 11: Verify no nested loops (ensures O(n) not O(n^2))
    tester.TestCodeStructure(
        testName: "Test 11: No nested loops (O(n) complexity)",
        points: 10,
        checkType: CodeStructureCheck.HasNestedLoops,
        shouldPass: false, // We want HasNestedLoops to return
        failureMessage: "Method must not have nested loops to ma
    );
```

- נוסיף את שם הבדיקה והסבר.
- מספר הנקודות.
- סוג הבדיקה לפי רשימה קבועה מראש (ראו במהשך).
- פרמטרים נוספים אופציונאליים: כגון הודעת שגיאה.

פרמטרים של TestCodeStructure

פרמטר	סוג	פירוט
testName	String	שם הבדיקה והסבר קצר שלה – הטקסט יופיע לתלמיד.
points	Int	מספר הנקודות שיקבל התלמיד אם יעבור את הדיקה.
checkType	CodeStructureCheck	סוג הבדיקה לפי רשימה קבועה מראש (ראו בהמשך).
shouldPass	Bool (default=true)	אופציונאלי: האם תוצאות הבדיקה המצופה היא חיובית או שלישית.
expectedCount	Int ? (default=null)	אופציונאלי: בבדיקות כמותיות המספר המצופה (לדוגמה: מספר הלולאות).
failureMessage	String (default= null)	אופציונאלי: הודעת שגיאה לתלמיד אם הבדיקה נכשלת.

סוגי הבדיקות – המחלקה CodeStructureCheck

שם הבדיקה	ערך מוחזר	תיאור
IsRecursive	bool	האם קיימת קריאה רקורסיבית
CountForLoop	Int	מספר לולאות for בהם השתמש התלמיד (לאו דווקא מכוננות).
CountWhileLoop	Int	מספר לולאות while בהם השתמש התלמיד (לאו דווקא מכוננות).
CountForEachLoop	Int	מספר לולאות foreach בהם השתמש התלמיד (לאו דווקא מכוננות).
CountAnyLoop	Int	מספר לולאות מכל סוג בהם השתמש התלמיד (לאו דווקא מכוננות).
CountIfStatements	Int	מספר התנאים בהם השתמש התלמיד
CountRecursiveCalls	Int	מספר הקריאות הרקורסיביות בפעולה
CountReturnStatements	Int	מספר פקודות ה return בפעולה.
CountNewNodes	Int	מספר הפקודות new Node<T> בפעולה
CountNewQueue	Int	מספר פקודות new Queue<T> בפעולה
CountNewStack	Int	מספר פקודות new Stack<T> בפעולה
CountNewBinNode	Int	מספר פקודות new BinNode<T> בפעולה

סוגי הבדיקות – המשך

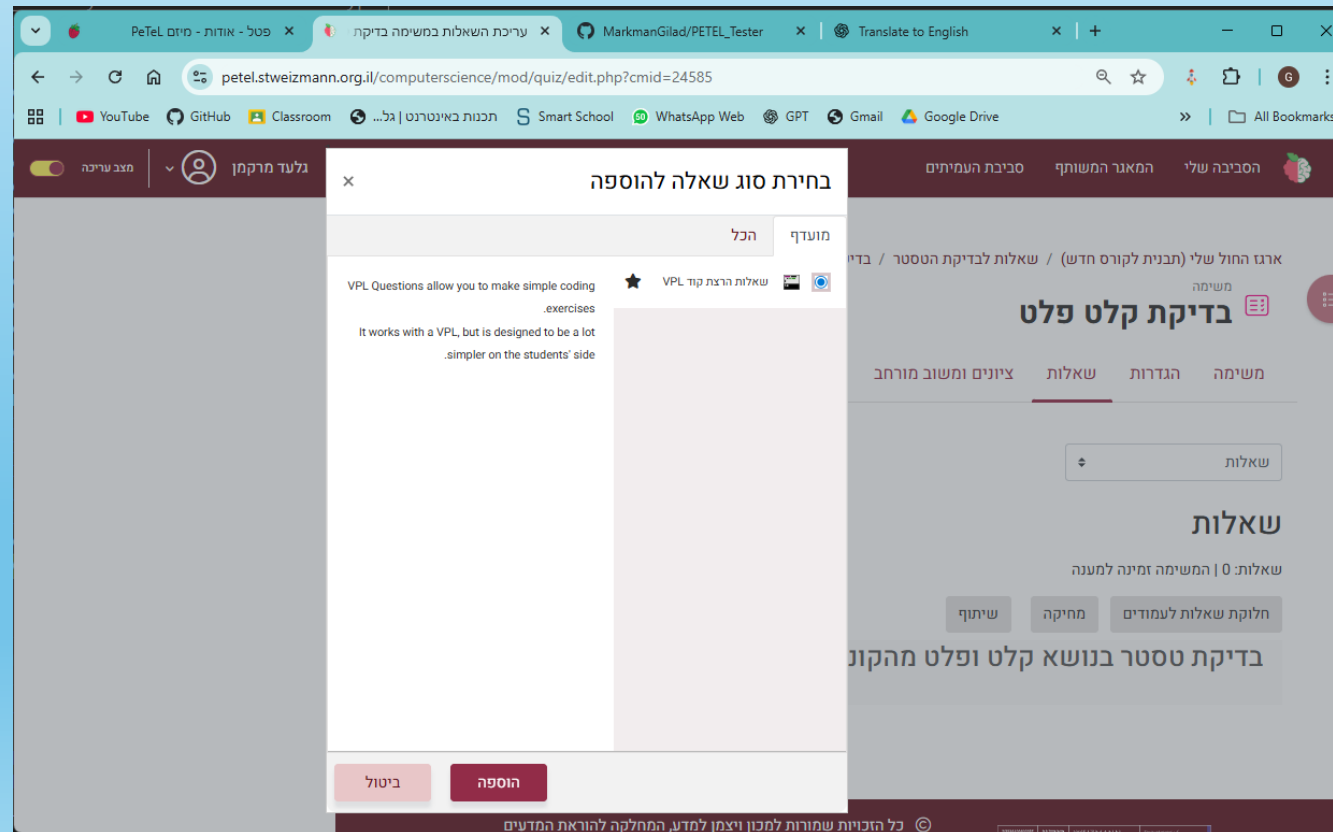
שם הבדיקה	ערך מוחזר	תיאור
CountSetNext	Int	מספר פקודות SetNext של המחלקה Node<T>
CountGetNext	Int	מספר פקודות GetNext של המחלקה Node<T>
HasNestedLoops	Bool	האם התלמיד השתמש בלולאות מכוננות מכל סוג (לצורך סיבוכיות).
IsStatic	Bool	בדיקת חתימת הפעולה
IsPublic	Bool	בדיקת חתימת הפעולה
IsPrivate	Bool	בדיקת חתימת הפעולה
IsProtected	Bool	בדיקת חתימת הפעולה
IsInternal	Bool	בדיקת חתימת הפעולה
CheckParams	Bool	בדיקת חתימת הפעולה – השוואת הפרמטרים לחתימת פעולת המורה.
CheckReturnType	Bool	בדיקת חתימת הפעולה – השוואת הערך המחוזרת לחתימת פעולת המורה.

העלאת בדיקה ל PETEL



העלאת בדיקה למערכת Moodle PeTel

- לאחר שהרצנו את הבדיקה בסביבת העבודה וניסינו תשובות שונות של התלמידים הגיע הזמן להעלות למערכת PETEL.
- ניצור שאלת הרצת קוד VPL.



בניית השאלה

תבנית תשובה

```
1 using System;
2 using Unit4;
3
4 class StudentAnswer
5 {
6     _____ InputOutput(_____ )
7     {
8         // Your code here
9     }
10 }
11
12
```

תבנית תשובה ?

דוגמת קוד נכונה של המורה

```
1 using System;
2
3 class TeacherAnswer
4 {
5     public static double InputOutput(int num)
6     {
7         double sum = 0;
8         for (int i = 0; i < num; i++)
9         {
10             Console.WriteLine("Enter price");
11             double price = double.Parse(Console.ReadLine());
12             sum += price;
13         }
14         Console.WriteLine("The Sum is: " + sum);
15         return sum / num;
16     }
17 }
18
```

דוגמת קוד
נכונה של
המורה ?

• לבנות שאלה כרגיל

• שם קובץ: מומלץ StudentAnswer.cs
אחרת יש לשנות את הקובץ ב
MainTester.

StudentAnswer.cs



שם קובץ

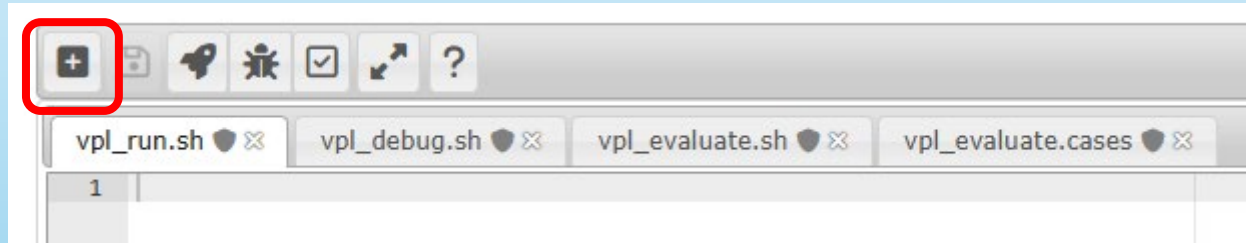
• בתבנית התשובה מומלץ להעתיק את
תשובת התלמיד מסביבת העבודה
ולמחוק את הפרטים אותם יתבקש
התלמיד לכתוב.

• חשוב – לא לשכוח using.

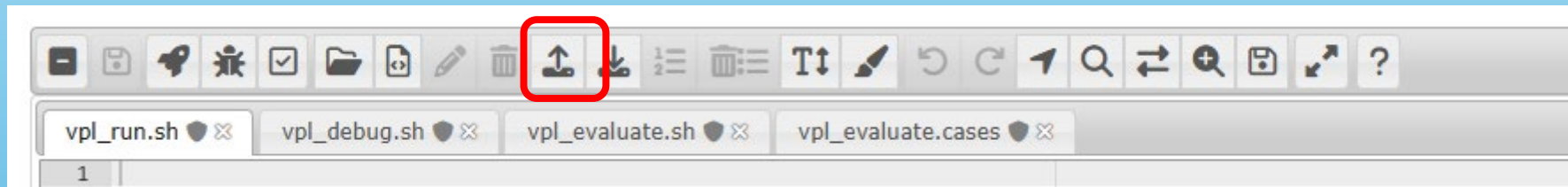
• מומלץ - לא להוסיף namespace
(אחרת יש לעדכן בטסטר)

העלאת קבצי הבדיקה

- בחלון קבצי הרצה ובדיקת קוד יש ללחוץ על +



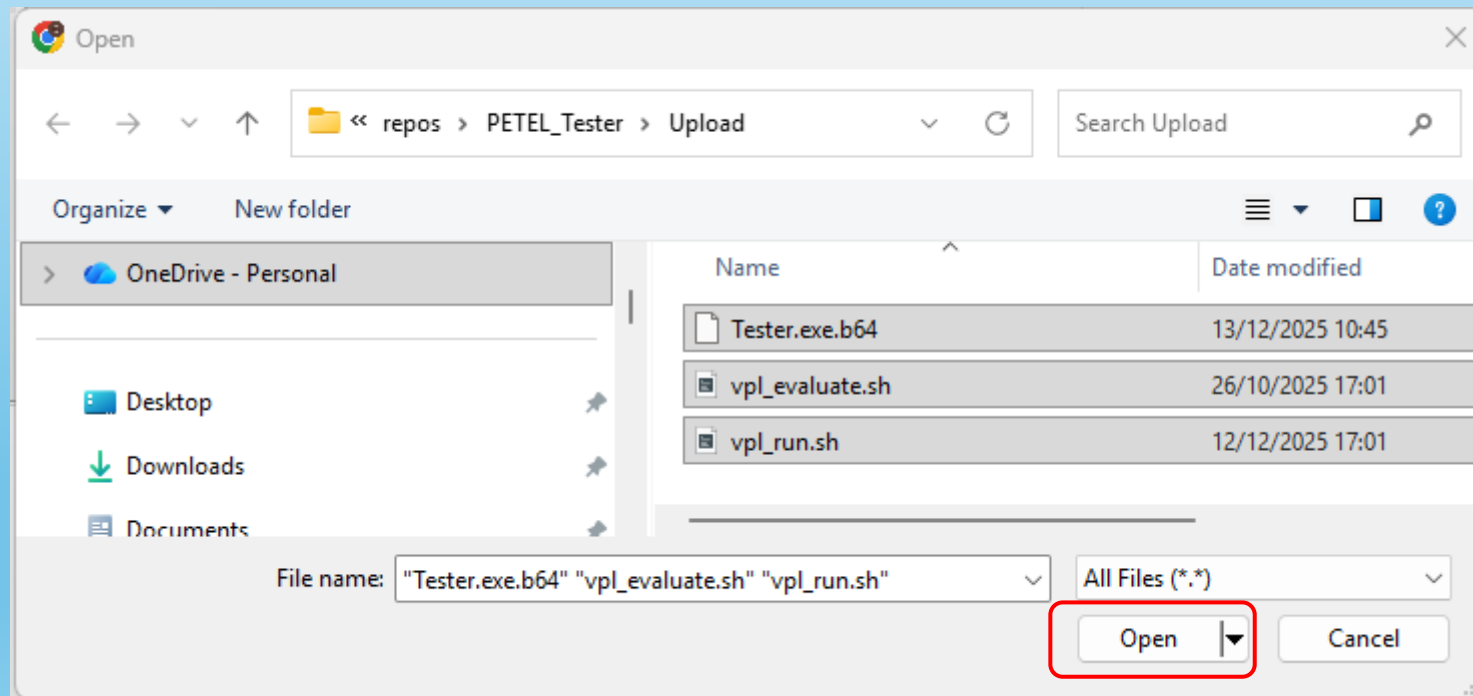
- ולאחר מכן, ללחוץ על העלאה (חץ למעלה).



- לאחר הלחיצה יפתח חלון במחשב שלכם.

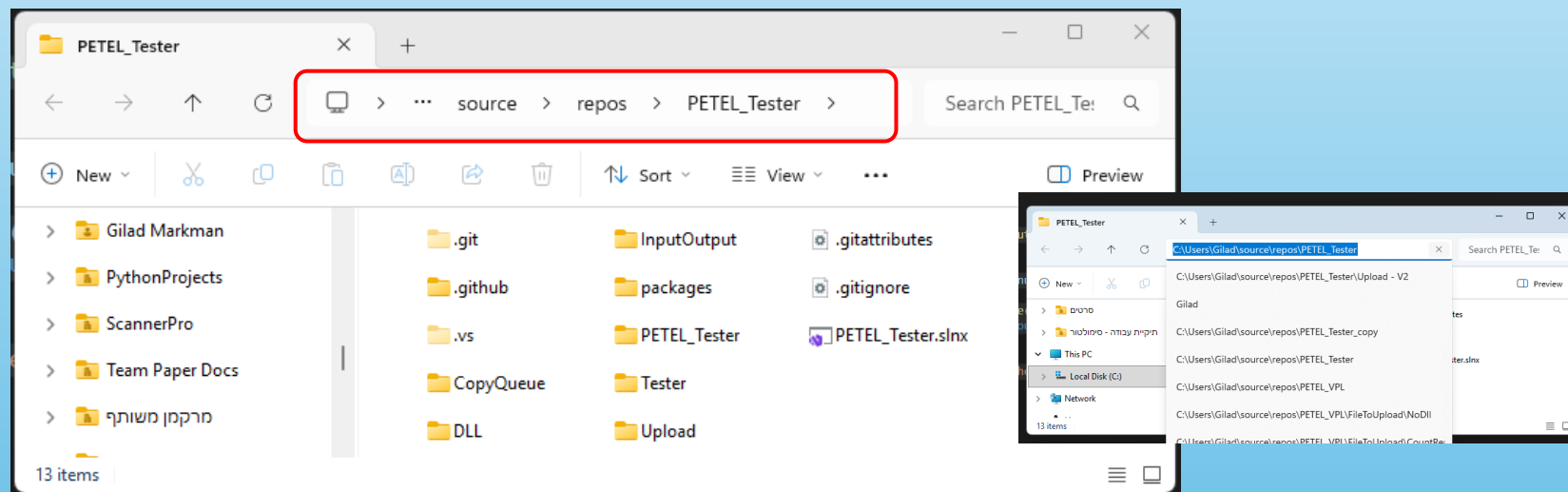
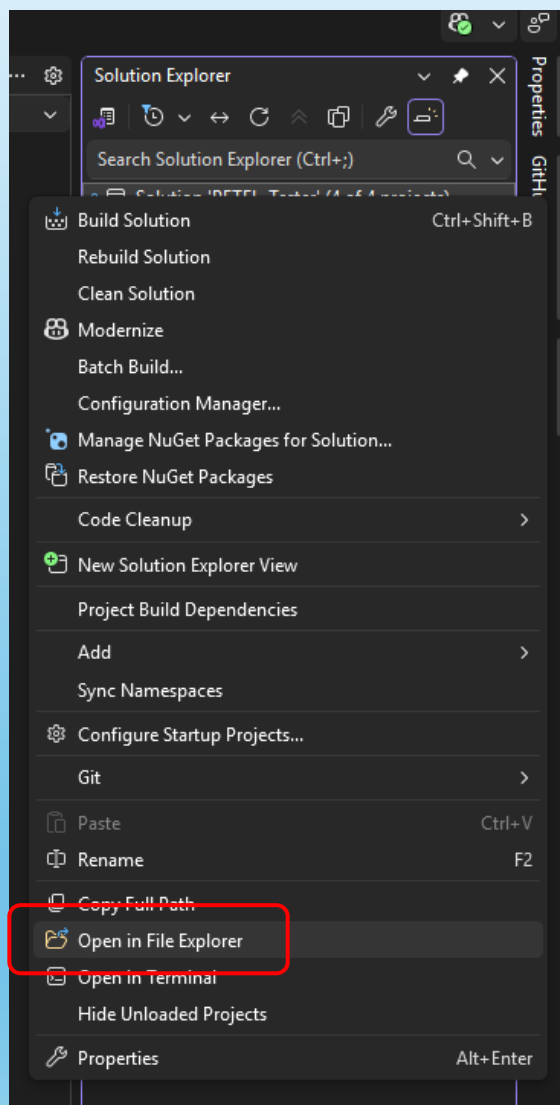
העלאת קבצי הבדיקה הכללים

- יש לאתר את הספרייה Upload בתוך הפרויקט שלכם, לסמן את שלושת הקבצים בספרייה, ולהעלות אותם (עזרה באיתור הספרייה בשקופית הבאה).
- קבצים אילו קבועים בכל המשימות.



איתור ספריית העבודה במחשב שלנו

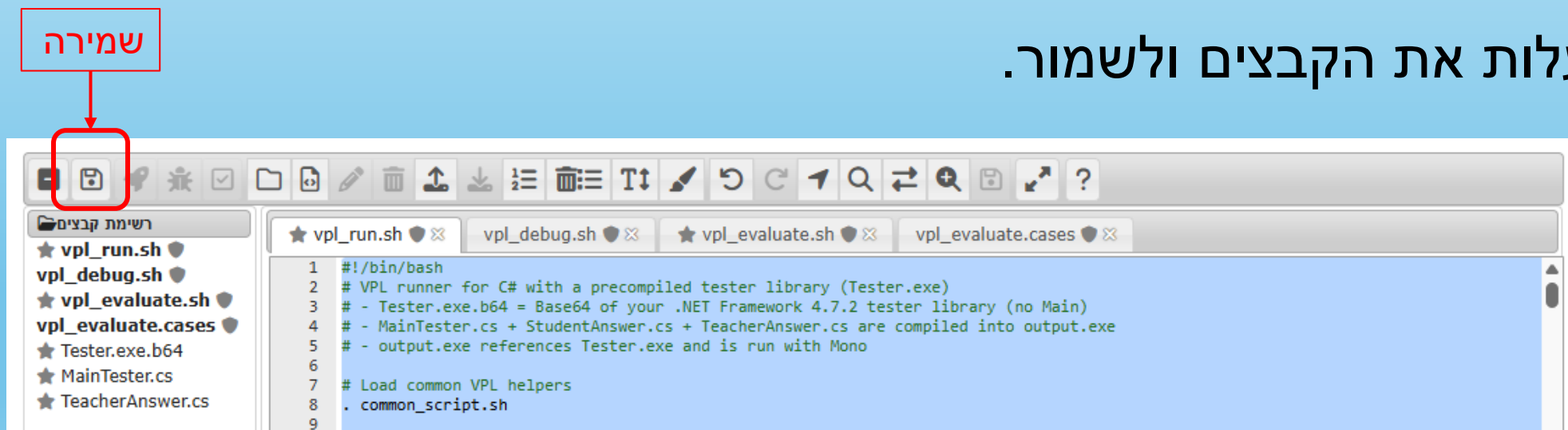
- על מנת לאתר את ספריית העבודה במחשב שלנו נלחץ קליק ימני על ה solution שלנו.
- בחירה ב Open in File Explorer.



- ניתן להעתיק את הנתוב של הפרויקט והתמש בו בחלון העלאה של ה PETEL.

העלאת הקבצים של המשימה

- בשלב הבא נעלה את הקבצים של המשימה הספציפית.
- ללחוץ שוב על החץ למעלה, לאתר את הספרייה של הפרויקט, לסמן רק את שני הקבצים:
 - MainTester.cs
 - TeacherAnswer.cs
- לעלות את הקבצים ולשמור.



בדיקה

- בחלק התחתון של הטופס נלחץ על שמירה והמשך עריכה ולאחר מכן על תצוגה מקדימה.

שמירת שינויים והמשך עריכה 🔍 תצוגה מקדימה

- נעתיק את תשובת התלמיד מסביבת העבודה ל PETEL ונלחץ "בדיקת קוד מקדימה".

- נוודא שקיבלנו תוצאה מצופה.

Google Chrome - הסביבה שלי | קלט פלט בלולאה | תצוגה מקדימה של שאלה

petel.stweizmann.org.il/computerscience/question/bank/previewquestion/preview.php?id=453967&restartversion=0&...

קלט פלט בלולאה גרסה 1 (latest)

שאלה 1

עליכם לבנות פעולה המקבלת מספר שלם n. הפעולה תדפיס למשתמש "Enter price" תקלוט n מחירים של מוצרים (המחיר יכול לכלול אגורות).

הפעולה תדפיס את סכום הרכישה הכולל: "The Sum is: 24.5" ותחזיר את הממוצע של הפריטים שנרכשו.

תשובה נכונה | איפוס

```
4 class StudentAnswer
5 {
6     public static double InputOutput(int num)
7     {
8         double sum = 0;
9         for (int i = 0; i < num; i++)
10         {
11             Console.WriteLine("Enter price");
12             double price = double.Parse(Console.ReadLine());
13             sum += price;
14         }
15         Console.WriteLine("The Sum is: " + sum);
16         return sum / num;
17     }
18 }
19
```

בדיקת קוד מקדימה

בדיקת קוד מקדימה

Evaluation:

Test 1: 3 numbers. capture Console Output | params: p0=3 | input: "5\n3\n7\n": success. 10 points
output: "Enter price\nEnter price\nEnter price\nThe Sum is: 15\n"
return: 5

Test 2: 3 numbers. Only Compare Return | params: p0=3 | input: "5\n3\n4\n": success. 10 points
return: 4

Test 3: 1 numbers | params: p0=1 | input: "3\n": success. 10 points
output: "Enter price\nThe Sum is: 3\n"

עבודה נעימה

