אוסף מחלקות גרפיות

הקובץ *Games.dll* מכיל אוסף מחלקות גרפיות, המאורגנות במרחבי שמות *(namespace)*.

הטבלה הבאה מסכמת את כל מרחבי השמות והמחלקות שקיימות בהם:

שם המרחב	תיאור	המחלקות
using *Games.TurtleLib;*	מרחב זה מכיל את המחלקה הגרפית לציור בעזרת 'צב'.	*Turtle*
using *Games.BucketLib;*	מרחב זה מכיל את המחלקה המגדירה 'דלי גרפי'.	*Bucket*
using *Games.DieLib;*	מרחב זה מכיל את המחלקה הגרפית 'קובייה', בעלת *6* פאות, כאשר על הפאות מופיעים המספרים *1* עד *6*. המחלקה הנחוצה לסימולציה של "הטלת קובייה".	*Die*
using *DreidleLib;*	מרחב זה מכיל את המחלקה הגרפית 'סביבון', בעלת *4* פאות, כאשר על הפאות מופיעות האותיות **נ ג ה ש**. המחלקה הנחוצה לסימולציה של " הגרלת סביבון".	*Dreidle*
using *Games.FrogLib;*	מחלקת הבסיס המופשטת לשתי הצפרדעים: 'צפרדע חום' ו'צפרדע ירוקה'. מרחב זה מכיל את שני המחלקות הגרפיות 'צפרדע ירוקה' ו-'צפרדע חום', הנחוצות לסימולציה של פיתרון "בעיית הצפרדעים".	*FrogsBase* *BrownFrog* *GreenFrog*
using *Games.TowerLib;*	מרחב זה מכיל את המחלקה הגרפית 'מוט' הנחוצה לסימולציה של העברת הדסקיות ממוט למוט של המשחק "מגדלי הנוי".	*Tower*

המחלקה *Turtle*

המחלקה *Turtle* מגדירה צב גרפי, שבאמצעותו אפשר לצייר קווים.

Turtle( )	הפעולה הבונה עצם מסוג צב הממוקם במרכז, פניו צפונה וזנבו למעלה.
void *MoveForward(**double** x)*	הפעולה מזיזה את הצב x צעדים קדימה . הערה: אם זנבו של הצב למטה הצב ישאיר אחריו קו.
void *MoveBackward(**double** x)*	הפעולה מזיזה את הצב x צעדים אחורה . הערה: אם זנבו של הצב למטה הצב ישאיר אחריו קו.
void *TailDown()*	הפעולה מורידה את זנב הצב.
void *TailUp()*	הפעולה מרימה את זנב הצב.
void *TurnLeft(**double** x)*	הפעולה מפנה את פני הצב x מעלות נגד כיוון שעון.
void *TurnRight(**double** x)*	הפעולה מפנה את פני הצב x מעלות עם כיוון שעון.
void *SetVisible(**double** status)*	הפעולה מציגה או מסתירה את הצב על-פי ערך הפרמטר שמועבר אליה. אם הערך הוא 'אמת' הפעולה תציג את הצב, אחרת תסתיר את הצב .
void *SetTailColor(**Color** color)*	הפעולה קובעת את הצבע בו מצייר הצב על-פי הפרמטר המועבר *color* .
void *Clear( )*	הפעולה מוחקת את כל הקווים שהצב השאיר אחריו.
void *SetDelay(**int** milliseconds)*	הפעולה קובעת את קצב הזזת הצב על-פי הפרמטר המועבר *milliseconds* : מספר מילי שניות *(גדול מאפס)* להמתנה בהזזת הצב .

כאשר יוצרים צב, הוא ממוקם במרכז המשטח הגרפי כשפניו צפונה. כעת, ניתן לתת לצב "פקודות" על מנת שיצייר על המשטח הגרפי - ניתן לשנות את מיקומו וכיוונו של הצב במטרה לגרום לו לנוע על המשטח הגרפי. לצב יש זנב, אותו ניתן להרים ולהוריד ובאמצעותו ניתן לצייר על המשטח הגרפי. ציור על המשטח הגרפי יתאפשר רק אם זנבו של הצב למטה.

לדוגמה, התוכנית הבאה מציירת בעזרת הצב הגרפי ריבוע שצלעותיו בגודל *100* :

```
class TurtleDrawRectangle
{
    static void Main(string[] args)
    {
        Turtle t1 = new Turtle();
        t1.TailDown();
        t1.MoveForward(100);
        t1.TurnRight(90);
        t1.MoveForward(100);
        t1.TurnRight(90);
        t1.MoveForward(100);
        t1.TurnRight(90);
        t1.MoveForward(100);
        t1.TailUp();
        t1.MoveForward(50);
    }
}
```

המחלקה *Tower*

המחלקה *Tower* מגדירה 'מוט' הנחוץ לסימולציה של העברת הדסקיות ממוט למוט של המשחק "מגדלי הנוי".

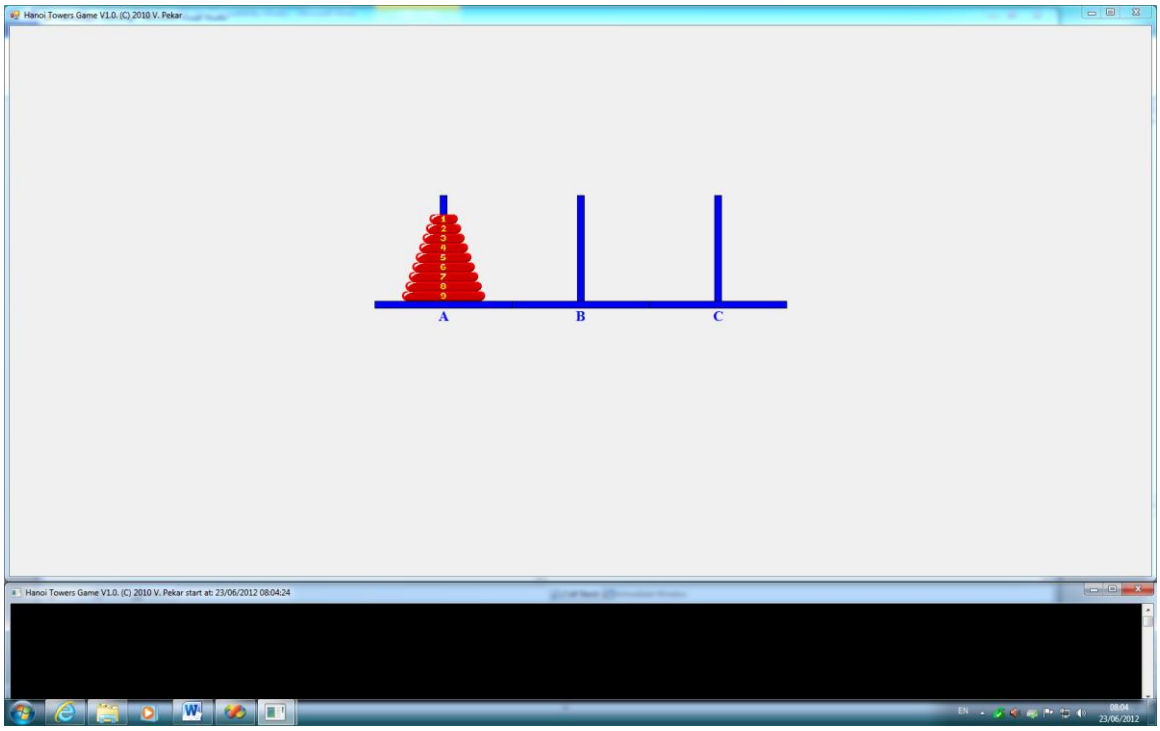
<u>הערה:</u>

ניתן ליצור רק *5* מוטות בוזמנית.

לפניכם ממשק המחלקה:

*Tower(**string** caption)*	הפעולה בונה מוט ריק שכותרתו *caption* מתקבל כפרמטר.
*Tower(**int** n, **string** caption)*	הפעולה בונה מוט. על מוט מושחלות *n* דסקיות בסדר יורד ביחס להיקפן, ושכותרתו של המוט *caption*. מספר מקסימלי של הדסקיות *n* המושחלות על המוט הוא *9*.
***string** GetCaption( )*	הפעולה מחזירה כותרתה של המוט הנוכחי.
***void** SetCaption(**string** caption)*	הפעולה משנה את הכותרת של המוט הנוכחי לפי הערך המתקבל כפרמטר.
***int** GetNumOfDiscs()*	הפעולה מחזירה את מספר הדסקיות המושחלות על המוט הנוכחי.
***int** Top()*	הפעולה מחזירה את גודל הדסקית שנמצאת בראש המוט הנוכחי. אם המוט הנוכחי ריק, יוחזר הערך *0*.
***void** MoveDisc(Tower toTower)*	הפעולה מעבירה את הדסקית שנמצאת בראש המוט הנוכחי למוט *toTower* שהתקבל כפרמטר. הפעולה מעבירה את הדסקית רק בצורה תקינה (דסקית גדולה לא יהיה על דסקית קטנה, והמוט הנוכחי לא ריק).

דוגמה: יצירת שלושה מגדלים שמייצגים המשחק "מגדלי הנוי"




```
Tower[] towers = new Tower[3];
 towers[0] = new Tower(9, "A");
towers[1] = new Tower("B");
towers[2] = new Tower("C");
```

דוגמה :

```
using System;
using Games.TowerLib;
namespace HanoiTowersTest
{
    class Program
    {
        public static void Main()
        {
            Tower[] towers = new Tower[3];
            towers[0] = new Tower(5, "A");
            towers[1] = new Tower("B");
            towers[2] = new Tower("C");
            HanoiTowersSolution(towers, 5, 0, 2);
        }
        private static void HanoiTowersSolution(Tower[] towers, int nRings,
                            int fromPole, int toPole)
        {
            int extraPole;
            if (nRings > 0)
            {
                extraPole = 3 - (fromPole + toPole);
                HanoiTowersSolution(towers, nRings - 1, fromPole, extraPole);
                towers[fromPole].MoveDisc(towers[toPole]);
                HanoiTowersSolution(towers, nRings - 1, extraPole, toPole);
            }
        }
    }
```

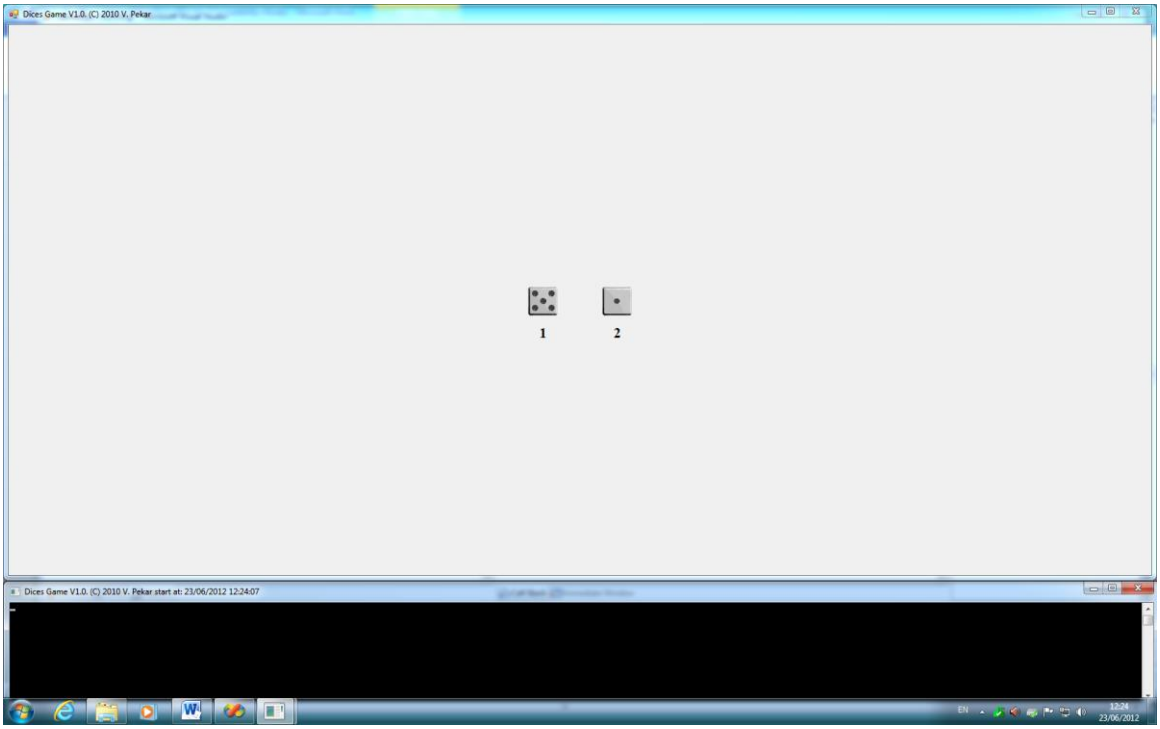
המחלקה *Die*

המחלקה *Die* מגדירה 'קובייה' הנחוצה לסימולציה של "הטלת קובייה".

<u>הערה:</u> ניתן ליצור רק *5* קוביות בוזמנית.

*Die(**string** caption)*	הפעולה בונה קובייה שכותרתה *caption*, מתקבל כפרמטר. הקובייה שנוצרה מראה מספר אקראי בין *1* ל-*6*.
***string** GetCaption( )*	הפעולה מחזירה כותרתה של הקובייה הנוכחית.
***void** SetCaption(**string** caption)*	הפעולה משנה את הכותרת של הקובייה הנוכחית לפי הערך המתקבל כפרמטר.
***void** Roll()*	הפעולה מדמה "הטלת הקובייה" הנוכחית. בתום הפעולה מתעדכן המספר שהקובייה הנוכחית מראה לאחר הטלה.
***int** GetNum()*	הפעולה מחזירה מספר שמראה הקובייה הנוכחית.
***string** GetCaption()*	הפעולה מחזירה כותרתה של הקובייה הנוכחית.

דוגמה: יצירת שני קוביות




```
Die d1 = new Die("1");
Die d2 = new Die("2");
```

```
using System;
using Games.DieLib;
namespace DieTest
{
    class Program
    {
        static void Main(string[] args)
        {
            Die d1 = new Die("1");
            Die d2 = new Die("2");
            Console.ReadLine();
            int count = 0, n1, n2;
            d1.Roll();
            n1 = d1.GetNum();
            d2.Roll();
            n2 = d2.GetNum();
            while (n1 != 6 || n2 != 6)
            {
                d1.Roll();
                d2.Roll();
                n1 = d1.GetNum();
                n2 = d2.GetNum();
                count++;
            }
            Console.WriteLine(count);
        }
    }
}
```

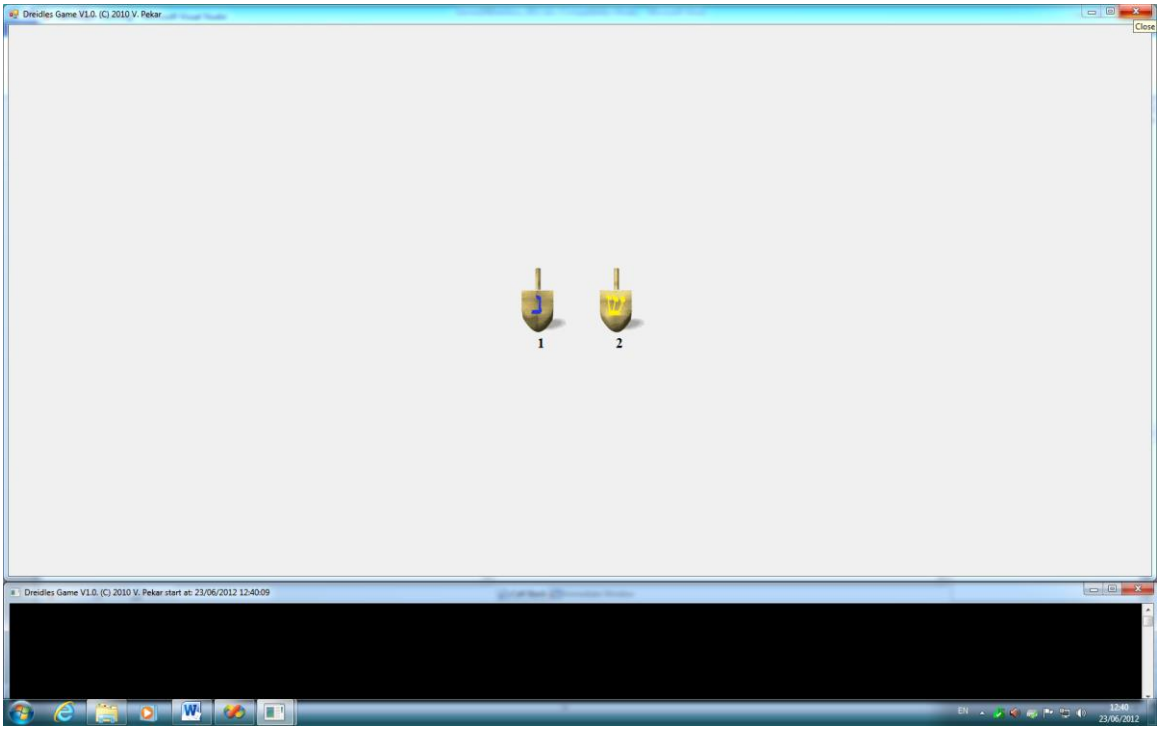
המחלקה *Dreidle*

המחלקה *Dreidle* מגדירה 'סביבון' הנחוצה לסימולציה של "הגרלת סביבון".

<u>הערה</u>: ניתן ליצור רק *5* סביבונים בוזמנית.

*Dreidle (**string** caption)*	הפעולה בונה סביבון שכותרתו *caption*, מתקבל כפרמטר. הסביבון שנוצר מראה אות אקראית בין האותיות **נ ג ה ש**.
***string** GetCaption( )*	הפעולה מחזירה את כותרתו של הסביבון הנוכחי.
***void** SetCaption(**string** caption)*	הפעולה משנה את הכותרת של הסביבון הנוכחי לפי הערך המתקבל כפרמטר.
***void** Roll()*	הפעולה מדמה "הגרלת סביבון" הנוכחי. בתום הפעולה מתעדכנת האות שהסביבון הנוכחי מראה לאחר הגרלה.
***string** GetLetter()*	הפעולה מחזירה מחרוזת שמייצגת את האות שמראה הסביבון: *"nun"* = נ ; *"gimel"* =ג ; *"chay"* =ה ; *"shin"* =ש.

דוגמה: יצירת עצמים – שני סביבונים



```
Dreidle d1 = new Dreidle("1");
Dreidle d2 = new Dreidle("2");
```

דוגמה :

```
using System;
using Games.DieLib;
namespace DieTest
{
    class Program
    {
        static void Main(string[] args)
        {
            Dreidle d1 = new Dreidle("1");
            Dreidle d2 = new Dreidle("2");
            int count = 0;
            string  let1, let2;
            d1.Roll();
            let1 = d1.GetLetter();
            d2.Roll();
            let2 = d2. GetLetter();
            while (let1 != "shin" || let2 != "shin")
            {
                d1.Roll();
                d2.Roll();
                let1 = d1.GetLetter();
                let2 = d2.GetLetter();
                count++;
            }
            Console.WriteLine(count);
        }
    }
}
```

המחלקה *BaseFrog*

מחלקה המייצגת צפרדע מופשטת.

הפעולה מחזירה את כותרתו של הצפרדע הנוכחית.	***string*** *GetCaption( )*
הפעולה משנה את הכותרת של הצפרדע הנוכחית 'צפרדע חום' או 'צפרדע ירוקה' לפי הערך המתקבל כפרמטר.	***void*** *SetCaption(**string** caption)*
הפעולה מחזירה את מיקום הצפרדע הנוכחית 'צפרדע חום' או 'צפרדע ירוקה'.	***int*** *GetPlace( )*
הפעלה מזיזה למיקום סמוך פנוי.	***abstract void*** *Move()*
הפעלה מקפיצה צפרדע למיקום פנוי.	***abstract void*** *Jump()*

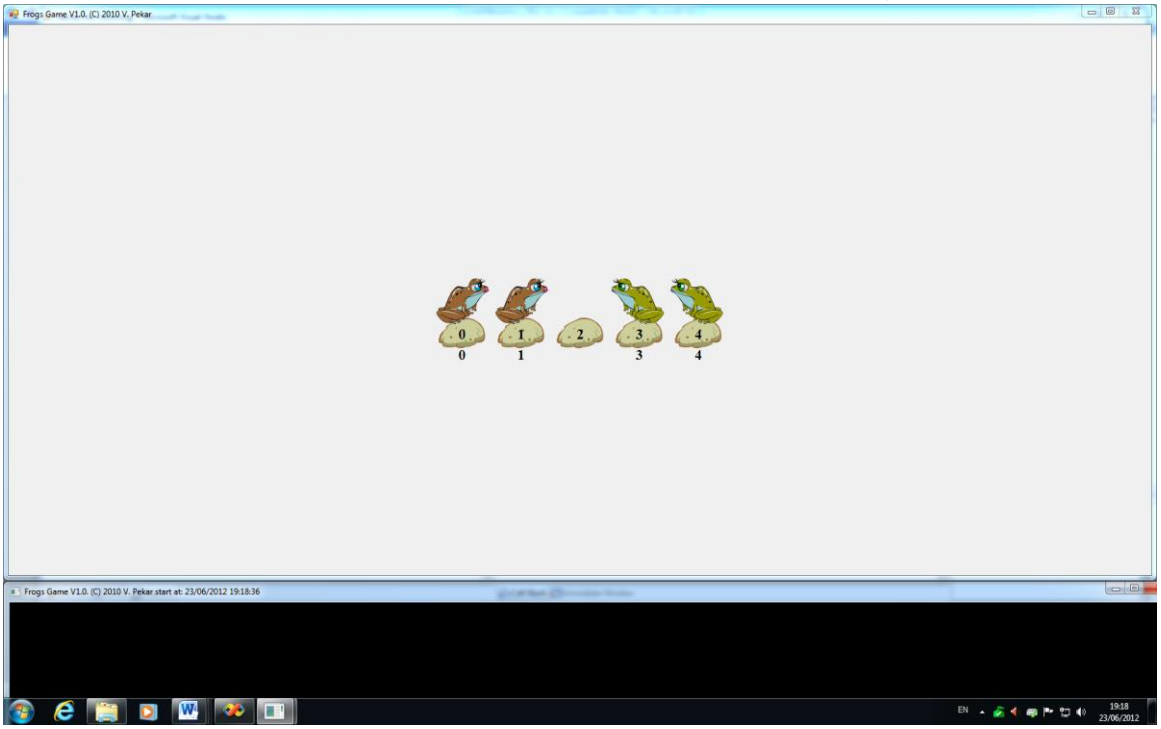
המחלקות *BlueFrog* ו-*GreenFrog*

המחלקות מגדירות 'צפרדע ירוקה' ו-'צפרדע חום' הנחוצות לסימולציה של פיתרון "בעיית הצפרדעים" על שורה של אבנים. בן חמש משבצות.

<u>הערה</u>: הגודל המקסימלי של השורה הוא *18* אבנים.

הפעולה בונה 'צפרדע חום' שכותרתה *caption*. הצפרדע ממוקמת במיקום *place*, ופני הצפרדע פונים לכיוון **שמאל**. המיקום הוא מספר בין *0* ל-*17*. ניתן למקם את הצפרדע רק במקום פנוי.	*BrownFrog(**int** place, **string** caption)*
הפעולה בונה 'צפרדע ירוקה' שכותרתה *caption*. הצפרדע הממוקמת במיקום *place*, ופני הצפרדע פונים לכיוון **ימין**. המיקום הוא מספר בין *0* ל-*17*. ניתן למקם את הצפרדע רק במקום פנוי.	*GreenFrog(**int** place, **string** caption)*
הפעלה מזיזה 'צפרדע ירוקה' או 'צפרדע חום' למיקום סמוך פנוי בכיוון פני הצפרדע. כל הצפרדעים הירוקות או החומות יודעות לזוז **רק בכיוון הפנים**.	***void*** *Move()*
הפעלה מקפיצה 'צפרדע חום' או 'צפרדע ירוקה' בכיוון הפנים. צפרדע בצע אחד יכולה לקפץ מעל צפרדע מצבע אחר אל מיקום פנוי. כאשר כל הצפרדעים הירוקות או החומות יודעות לקפוץ **רק בכיוון הפנים**.	***void*** *Jump()*

דוגמה: יצירת משחק של *4* צפרדעים




```
BlueFrog f1 = new BrownFrog("0");
BlueFrog f2 = new BrownFrog("1");
GreenFrog f3 = new GreenFrog("3");
GreenFrog f4 = new GreenFrog("4");
```

דוגמה

```
using System;
using Games.FrogLib;
namespace DieTest
{
    class Program
    {
        static void Main(string[] args)
        {
            //                  שלב ראשון – יצירת עצמים שמייצגים משחק הצפרדעים
            BlueFrog f1 = new BrownFrog("0");
            BlueFrog f2 = new BrownFrog("1");
            GreenFrog f3 = new GreenFrog("3");
            GreenFrog f4 = new GreenFrog("4");
            Console.WriteLine("To solve Grogs Press Enter !!!");
            Console.ReadLine();
            //                  שלב שני – ביצוע פעולות של עצמים לפתרון משחק הצפרדעים
            f2.Move();
            f3.Jump();
            f4.Move();
            f2.Jump();
            f1.Jump();
            f3.Move();
            f4.Jump();
            f1.Move();
        }
    }
```

מחלקה גרפית לייצוג עץ בינרי

הקובץ *BinTreeCanvasLib.dll* מכיל את מחלקת השירות *TreeCanvas*.

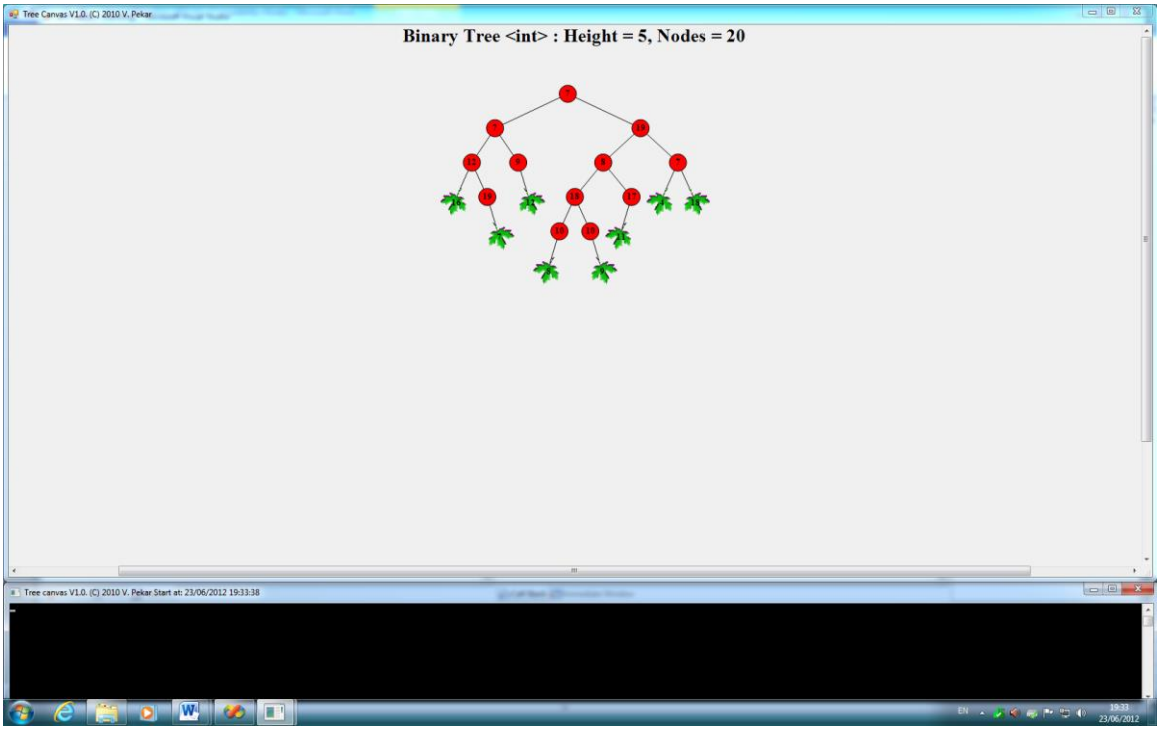
מחלקת השירות מאפשרת הצגה גרפית של עצים בינריים המבוססים על המחלקות הגנריות *BinTreeNode<T>* ו-*BinTreeUtils*, שנמצאות בתוך ספריית העזר *Unit4* "עיצוב תוכנה מבוסס עצמים בשפת *C#*".

static void AddTree<T> (BinTreeNode<T> tree)	הפעולה מעבירה את העץ הבינרי המתקבל כפרמטר לחלון גרפי וגם מציגה אותו.
static void SetAnimastion(AnimationSpeed speed)	הפעולה קובעת מהירות של אנימציה בסריקה של העץ הבינרי שהועבר לחלון גרפי.
static void PrintOutAllElements(bool printOut)	הפעולה קובעת האם להציג תוצאות הסריקה בחלון *Console*.
static void TreeDrawInOrder( )	פעולה מתארת את סריקת העץ הבינרי שהועבר לחלון גרפי בסדר תוכי .
static void TreeDrawPostOrder( )	פעולה מתארת את סריקת העץ הבינרי שהועבר לחלון גרפי בסדר סופי .
static void TreeDrawPostOrder( )	פעולה מתארת את סריקת העץ הבינרי שהועבר לחלון גרפי בסדר תחילי .
static void TreeDrawLevelOrder( )	פעולה מתארת את סריקת העץ הבינרי שהועבר לחלון גרפי לפי רמות .

<u>הערה</u>:

מחלקת השירות *BinTreeUtils* מאפשרת להציג רק עץ בינרי אחד בוזמנית.

דוגמה: להצגה גרפית של עץ בינרי




```
BinTreeNode<int> t = BinTreeUtils.BuildRandomTree(20, 1, 19);
TreeCanvas. AddTree(t);
```

דוגמה :

```
using System;
using Unit4.CollectionsLib;
using Unit4.UtilsLib;
using BinTreeCanvasLib;
namespace BinTreeTest
{
    class Program
    {
        public static void Main()
        {
            BinTreeNode<int> t = BinTreeUtils.BuildRandomTree(50, 1, 3);
            TreeCanvas.TreeDraw(t);
            Console.WriteLine(BinTreeUtils.InOrderTraversal(t));
            Console.WriteLine("To Build next Tree Press Enter");
            Console.ReadLine();
            Console.Clear();
            BinTreeNode<string> t1 = new BinTreeNode<string>("ab");
            TreeCanvas.TreeDraw(t1);
        }
    }
}
```