

מאגר מעבדות לשפות התכנות החדשות

ניתן להשתמש בחומרים לצורך הוראה בלבד.

לא ניתן לפרסם את החומרים או לעשות בהם כל שימוש מסחרי

ללא קבלת אישור מראש מצוות הפיתוח בראשות ד"ר תמר פז.

המעבדה בקובץ זה מיועדת לתלמידים הלומדים מדעי המחשב בשפת התכנות C# והיא מותאמת לסביבת Visual C# 2005 Express Edition.

המעבדה מיועדת לשיעורי המעבדה והיא מבוססת על שיטת ההוראה לפיה הלימוד של כל נושא חדש ייפתח בהתנסות אישית במעבדה. לאחריה, יבוא דיון כיתתי, שבעקבותיו ייפתרו משימות שונות.

המעבדה מתרכזת בלימוד נושא מרכזי אחד:

היכרות עם מחלקות (מבנה של מחלקה ובניית מחלקה).

הוספת מחלקה חדשה לפרוייקט

1. מקש ימני בעכבר על שם הפרוייקט

2. בחרו הוספה Add

3. בחרו סוג ההוספה class

4. בחלון שנפתח תנו שם למחלקה

הרעיון למחלקה Bucket לקוח מתוך: המרכז להוראת המדעים, האוניברסיטה העברית, עיצוב תכנה מבוסס עצמים, 2005

מבנה של מחלקה (class)

משימה 1. הקלידו את המחלקה בקובץ Bucket.cs

```
public class Bucket
{
    private int capacity;
    private double currentAmount;
    /// <summary>
    /// פעולה הבונה דלי חדש
    /// </summary>
    /// <param name="capacity">קיבולת הדלי </param>
    public Bucket(int capacity)
    {
        this.capacity = capacity;
        this.currentAmount = 0;
    }
    /// <summary>
    /// פעולה המרוקנת את הדלי
    /// </summary>
    public void Empty()
    {
        this.currentAmount = 0;
    }
    /// <summary>
    /// פעולה הבודקת אם הדלי ריק
    /// </summary>
    /// <returns>מחזירה אמת אם כן, אחרת מחזירה שקר</returns>
    public bool IsEmpty()
    {
        return this.currentAmount == 0;
    }
    /// <summary>
    /// פעולה הממלאת את הדלי בכמות amountToFill
    /// אם כמות הקלט גדולה מקיבולת הדלי, המילוי רק נפח הקיבולת
    /// </summary>
    /// <param name="amountToFill">כמות להוספה</param>
    public void Fill(double amountToFill)
    {
        if (this.capacity < this.currentAmount + amountToFill)
            this.currentAmount = this.capacity;
        else
            this.currentAmount += amountToFill;
    }
    /// <summary>
    /// פעולה המאזנת את קיבולת הדלי
    /// </summary>
    /// <returns>קיבולת הדלי</returns>
}
```

שם המחלקה. מציין שהמחלקה ציבורית וניתן לקרוא לה ממחלקות אחרות

תכונות המחלקה הן התכונות שיהיו לכל העצמים מטיפוס המחלקה Bucket. במקרה שלנו, לכל עצם שיוגדר מטיפוס המחלקה Bucket יהיו שתי תכונות: capacity - המילה המילה **private** מציינת כי התכונות הן משתנים פנימיים של קובץ זה. אי אפשר להשתמש בהן מקבצים אחרים, ואם רוצים לדעת את ערכן מקובץ אחר - משתמשים בפעולות המאזרות.

פעולה זו בונה עצם יחיד מטיפוס המחלקה Bucket ונותנת ערכים לתכונות של העצם הנוכחי אותו היא בונה. החתימה של הפעולה הבונה: (שם הפעולה תמיד זהה לשם המחלקה) **public** *רשימת פרמטרים וטיפוסיהם* **שם המחלקה**

בהמשך מופיעות הפעולות (שיטות = methods) הנוספות לביצוע על עצם מטיפוס Bucket. הפעולות מוגדרות עם **public** בחתימה של הפעולה המאפשרת גם למחלקות אחרות להשתמש בפעולות הללו. השמטת המילה **static** מהחתימה של הפעולה, מציינת כי זו פעולה שפועלת על עצם!

המילה **this** מציינת את העצם הנוכחי עליו מתבצעת הפעולה

תיעוד הפעולה. מתקבל אחרי הקשת (שלושה אלכסונים) לפני פעולה כתובה

----- המשך המחלקה מופיע בעמוד הבא -----

הרעיון למחלקה Bucket לקוח מתוך: המרכז להוראת המדעים, האוניברסיטה העברית, עיצוב תכנה מבוסס עצמים, 2005

```

public int GetCapacity()
{
    return this.capacity;
}
/// <summary>
/// פעולה המאחזרת את הכמות הנוכחית בדלי
/// </summary>
/// <returns>כמות נוכחית בדלי</returns>
public double GetCurrentAmount()
{
    return this.currentAmount;
}
/// <summary>
/// פעולה המעבירה כמות מקסימלית מהדלי הנוכחי לדלי שהתקבל כפרמטר
/// </summary>
/// <param name="bucketInto">הדלי אליו להעביר</param>
public void FourInto(Buket bucketInto)
{
    double freespace = bucketInto.GetCapacity() -
        bucketInto.GetCurrentAmount();
    if (this.currentAmount < freespace)
    {
        bucketInto.Fill(this.currentAmount);
        this.currentAmount = 0;
    }
    else
    {
        bucketInto.Fill(freespace);
        this.currentAmount -= freespace;
    }
}
/// <summary>
/// פעולה המחזירה את תאור הדלי
/// </summary>
/// <returns>מחרוזת המתארת את הדלי</returns>
public override string ToString()
{
    string st = "The capacity is :" + this.capacity + "\n" + "The current amount is :" + this.currentAmount;
    return st;
}
}

```

- כדי להקל את ההתמצאות בקובץ המחלקה, נפתח תמיד בפעולות המאחזרות (שמחזירות את הערכים של התכונות של העצם). במקרה שלנו, _____ -I _____ לאחריהם יופיעו שאר הפעולות. הפעולה האחרונה תהיה הפעולה ToString שתפקידה _____.

- כמו בפעולות בהן עסקנו עד כה:

1. כאשר פעולה לא מקבלת פרמטרים, בחתימה של הפעולה, במקום רשימת הפרמטרים, רושמים סוגריים ()

למשל `public double GetCapacity()`

2. כאשר פעולה לא מחזירה ערך, בחתימה של הפעולה, במקום הערך המוחזר, רושמים: `void`. למשל,

`public void Fill (double amountToFill)`

3. ולכן, החתימה של פעולה שלא מקבלת פרמטרים ולא מחזירה ערך, תראה כך:

public _____ **פעולה** _____

הרעיון למחלקה Bucket לקוח מתוך: המרכז להוראת המדעים, האוניברסיטה העברית, עיצוב תכנה מבוסס עצמים, 2005

- שימו לב: התיעוד הוא חלק בלתי נפרד מהמחלקה!

- הגדרה של תכונה: **private** *טכונה* **טכונה** *טכונה* **private**
- חתימה של הפעולה הבונה: **public** *טכונה* **טכונה** *טכונה* **public**
- חתימה של פעולה שאיננה הפעולה הבונה:
public *טכונה* **טכונה** *טכונה* **public**
- private** = הרשאת גישה פרטית. ניתן לגשת לתכונות רק מקובץ המחלקה הנוכחי.
- public** = הרשאת גישה פומבית. ניתן לגשת לפעולות גם ממחלקות אחרות.
- בכל הפעולות, ההתייחסות לתכונה של העצם עליו מתבצעת הפעולה, היא באמצעות המילה **this**.
- המילה **this** מציינת את העצם הנוכחי עליו מתבצעת הפעולה.

משימה 2 – חלק א'

- נשנה כעת את הפעולה הבונה כך שגם הקיבולת של הדלי (capacity) וגם הכמות הנוכחית יקבעו ע"י

```
public Bucket (double capacity, double currentAmount)
{
    this.capacity = capacity;
    this.currentAmount = _____;
}
```

- המשתמש בעת יצירת הדלי.
- לפניכם שלד של הפעולה הבונה החדשה. השלימו אותו.
- שנו את הפעולה הבונה בקובץ **Bucket.cs**

הקובץ עימו אנו עובדים **Bucket.cs** הוא קובץ המחלקה **Bucket**. בקובץ זה נמצאות ההגדרות של המחלקה **Bucket**. כלומר, אילו תכונות יש לעצם מטיפוס **Bucket** ואילו פעולות ניתן לבצע עימו ועליו. כדי לבדוק את השינוי שביצענו במחלקה, נכתוב מחלקה ובה פעולה ראשית שמשתמשת במחלקה **Bucket**.

- פיתחו את הקובץ **Program.cs** בפרויקט **BucketProject** והקלידו בפעולה ראשית תכנית שתקלוט מהמשתמש קיבולת וכמות נוכחית של דלי. הפעולה תיצור עצם מטיפוס דלי ותדפיס את התכונות של הדלי וערכיה (באמצעות זימון הפעולה **ToString()**).
- שימרו את המחלקה, הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 2 – חלק ב'

- הוסיפו לפעולה הראשית מסננות קלט כך שהקיבולת שתיקלט מהמשתמש לא תהייה שלילית, והכמות הנוכחית שתקלט מהמשתמש לא תהייה שלילית ותהייה קטנה או שווה לקיבולת של הדלי.
- שימרו, הריצו ובידקו שהתקבל הפלט הרצוי.

הרעיון למחלקה **Bucket** לקוח מתוך: המרכז להוראת המדעים, האוניברסיטה העברית, עיצוב תכנה מבוסס עצמים, 2005

משימה 3

- הפעולה Fill מוסיפה לכמות הנוכחית של הדלי את הכמות שמתקבלת כפרמטר. במידה שהתוספת תגרום לכמות הגדולה מהקיבולת של הדלי, הפעולה דואגת למלא את הקיבולת בלבד. שנו את הפעולה כך שאם התוספת תגרום לכמות הגדולה מהקיבולת של הדלי, בנוסף לכך שתתמלא רק הקיבולת של הדלי, גם תפלט הודעה מהי הכמות העודפת.
- בתכנית הראשית כיתבו את הפקודות שיבדקו את השינוי שבצעתם במחלקה Bucket. התכנית תיצור דלי ותנסה למלא אותו בכמויות העולות על הקיבולת שלו.
- שימרו, הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 4 – חלק א'

- הוסיפו למחלקה Bucket פעולה FillAll שממלאת את הדלי בכל הקיבולת שלו.
- רמז: הפעולה החדשה לא צריכה לקבל אף פרמטר וגם לא צריכה להחזיר ערך.
- בתכנית הראשית כיתבו את הפקודות שייצרו דלי (תתן לתכונותיו ערכים שיתקבלו מהמשתמש), תפעיל את הפעולה החדשה FillAll, ותציג את התכונות של הדלי ואת ערכיהם.
- שימרו, הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 4 – חלק ב'

- הוסיפו למחלקה Bucket פעולה isFull שמחזירה true אם הדלי מלא בכמות השווה לקיבולת שלו, ומחזירה false אחרת.
- חישוב תחילה האם הפעולה צריכה לקבל ערכים ומדוע.
- בתכנית הראשית כיתבו את הפקודות שיבדקו את הפעולה isFull. תיצרו דלי (תתן לתכונותיו ערכים שיתקבלו מהמשתמש). לאחר מכן, תקלוט מהמשתמש 3 ערכים למילוי, תוסיף אותם לדלי, תציג את התכונות של הדלי ואת ערכיהן, וכן את הערך המוחזר מהפעולה isFull.
- הריצו מספר פעמים כך שגם יוצג הערך true וגם יוצג הערך false.

בניית מחלקה

משימה 5

המחלקה Bucket וכן התוכניות שמשמשות בה נכתבו כתת מדורים (קבצים) של הפרויקט BucketProject. שיוך של כל הקבצים הללו לפרויקט אחד איפשר לנו להגדיר מחלקה בקובץ אחד (Bucket) ולהשתמש בה בקבצים האחרים.

נפתח כעת פרוייקט חדש DateProject.

בתפריט הראשי נבחר File -> New Project

בחלון שנפתח נבחר את הסוג Console Application,

וניתן לפרויקט את שמו DateProject

שימו לב כי בחלון הקבצים (החלון הימני) יופיע כעת הפרויקט החדש DateProject.

נפתח כעת מחלקה חדשה ונגדיר בה את המחלקה החדשה Date.

לפתיחה, עימדו עם העכבר על DateProject, ולחצו על המקש הימני של העכבר. בחלון שנפתח בחרו ב-New. בחלון שנפתח עכשיו בחרו ב-Class. כעת, תתבקשו לתת שם, תנו את השם Date. (לפי ההנחיות בתחילת הפרק)

לפניכם הממשק של המחלקה Date.

תיאור הפעולה	החתימה של הפעולה
הפעולה הבונה: יוצרת עצם מטיפוס Date על פי פרמטרים נתונים. הנחה: day מספר שלם 1-31, month מספר שלם 1-12, year שלם וחיובי.	Date (int day , int month , int year)
מחזירה את השנה.	int GetYear ()
מחזירה את החודש.	int GetMonth ()
מחזירה את היום.	int GetDay ()
מחזירה את מספר הימים בשנה (366 אם השנה מתחלקת ב-4 ללא שארית. 365 אחרת).	int DaysInYear()
מחזירה true אם התאריך "מעניין", ו- false אחרת. תאריך "מעניין" הוא תאריך בו היום שווה לחודש ובנוסף, אם היום הוא בן ספרה אחת אז היום שווה גם לספרה האחרונה של השנה, ואם היום הוא בן שתי ספרות אז היום שווה גם לשתי הספרות האחרונות של השנה (תאריכים "מעניינים": 6.6.2006 , 12.12.1912).	boolean IsInteresting()
מחזירה את מספר הימים שחלפו מתחילת השנה עד לתאריך הנוכחי. למשל אם התאריך הנוכחי הוא 6.3.2006 אז מספר הימים שחלפו הוא: 31+28+6=65 (ינואר 31, פברואר 28).	int DaysPass()
מחזירה מחרוזת המתארת את התאריך בצורה הבאה: <i>שנה / חודש / יום</i>	string ToString()

השלימו כעת את המחלקה Date.

• תזכורות:

1. חתימה של פעולה:

(רשימת פראמטרים וטיפוסיהם) שם הפעולה טיפוס הצרך האוחזר public

למשל, public int GetYear() .

2. בכל הפעולות, ההתייחסות לתכונה של העצם עליו מתבצעת הפעולה, היא באמצעות המילה **this**. המילה **this** מציינת את העצם הנוכחי עליו מתבצעת הפעולה.

3. כדי לכתוב את המחלקה צריך לדעת אילו תכונות יש לה! התכונות אינן חלק מהממשק משום שהתכונות הן פרטיות למחלקה והן לא מענייניו של מי שמשתמש במחלקה. במקרה שלנו, למחלקה Date יש שלוש תכונות: month , year , day . לכן, קובץ המחלקה יתחיל כך:

```
public class Date {  
  
    // The attributes of the class  
    private int day;  
    private int month;  
    private int year;  
}
```

4. הפעולה הבונה תמיד דואגת לתת ערך לכל התכונות של העצם הנבנה אבל הערכים לא מוכרחים להופיע בחתימה שלה. למשל הפעולה הבונה של המחלקה Bucket כפי שהוצגה בתחילת המעבדה, קיבלה רק פרמטר אחד ובכל זאת היא דאגה לתת ערכים לשתי התכונות של הדלי.

משימה 6 – חלק א'

כדי לבדוק את תקינות המחלקה שכתבתם, פתחו את הקובץ program (כחלק מהפרויקט DateProject) וכתבו תכנית שתקלוט מהמשתמש יום, חודש ושנה, תיצור עצם מטיפוס Date ותציג את התאריך המוחזר מהפעולה ToString. הוסיפו למחלקה מסננת קלט שתדאג כי היום יהיה בתחום 1-31, החודש בתחום 1-12 והשנה חיובית.

- הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 6 – חלק ב'

הוסיפו לפעולה הראשית את ההוראות הדרושות כך שתבדוק את כל הפעולות של המחלקה Date ותציג את הערכים המוחזרים על-ידן.

- הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 7

כתבו פעולה שתקלוט את תאריכי ימי ההולדת של 20 תושבי הבניין ותודיע כמה מהתאריכים הם "מעניינים" בהתאם להגדרה בממשק המחלקה Date. הפעולה תיצור עצם מטיפוס Date עבור כל תאריך ותפעיל עליו את הפעולה המתאימה.

שימו לב:

כאשר ההוראה `d1=new Date (day, month , year);` מתבצעת במחלקה יותר מפעם אחת (למשל, כאשר היא רשומה בתוך לולאה), אין צורך להכריז יותר מפעם אחת על `d1` כעל משתנה שתהיה בו הפניה לעצם מטיפוס Date, כלומר **אין צורך** שביצוע המחלקה יתקל יותר מפעם אחת בהוראה: `Date d1; או בהוראה: Date d1= new Date (day, month , year);` (בדיוק כמו שלא נכתוב בתוך לולאה `int sum=sum+count;`). ולכן,

הצהירו על המשתנה רק פעם אחת בתחילת הפעולה! כלומר, לפני הלולאה רישמו: `Date d1;` ובעת יצירת העצם רישמו: `d1=new Date (day, month, year);`

משימה 8

כתבו פעולה ראשית שקולטת נתונים של תאריכים ויוצרת עצם מטיפוס Date עבור כל אחד מהתאריכים. הפעולה תסכם את הימים שעברו מתחילת השנה עד לכל תאריך. קליטת הנתונים תפסק כאשר סך הימים שעברו מתחילת השנה עבור כל התאריכים יהיה 400 או יותר. הפעולה תדפיס את מספר הימים המדויק שעברו עבור כל התאריכים ביחד.

למשל, עבור הקלט: תאריך ראשון: 1.3.2000, תאריך שני: 20.4.2002, תאריך שלישי: 12.12.2006 קליטת הנתונים תפסק לאחר קליטת התאריך השלישי כיוון שסך הימים שעברו מתחילת השנה עבור שלושת התאריכים הוא יותר מ-400.

יצירת עצם מטיפוס Date == יצירת מופע של המחלקה Date

משימה 9 – חלק א'

כל תלמידי כיתה י"ב נולדו בשנת 1988.

לפניכם שלד של תכנית שמודיעה לכל תלמיד כמה ימים יחלפו מתחילת השנה עד ליום ההולדת שלו. התכנית תבקש מהמשתמש להקליד את יום הלידה של כל התלמידים שנולדו בחודש מספר 1, לסיום

המשתמש יקליד את המספר 0, והתהליך יחזור עבור החודשים הנותרים.

- השלימו את התכנית
- הקלידו, שימרו, הריצו ובדקו שהתקבל הפלט המבוקש.
- מדוע הפעולה GetValidDay היא בעלת הרשאת גישה פרטית? _____

```
namespace DateProject
{
class Program
{
static void Main(string[] args)
{
int _____, _____;
int _____;
Date birthday;
for (month = 1; _____ < 13; _____++)
{
// month קליטת יום ההולדת הראשון בחודש
day = GetValidDay(month);
while (day != 0) // month טיפול בתאריך הראשון, קליטה וטיפול בתאריכים הנוספים בחודש
{
birthday = new Date (day,month,1988);
pass = birthday.DaysPass();
Console.WriteLine ("birthday is "+ pass+" days after year start");
day = GetValidDay(month);
} // end while
} // end for
} // end main
private static int GetValidDay (int month)
{
int day;
do
{
Console.WriteLine ("enter day for month "+ month + " or 0 for end ");
day = int.Parse(Console.ReadLine());
} while (day < 0 _____ day > 31);
return day;
}
}
```

כמו במחלקות שפגשנו עד עכשיו, גם במחלקה שיש בה הפניות לעצם, ניתן להגדיר מספר פעולות בעלות הרשאות גישה שונות. במקרה הנוכחי, המחלקה Birthday מכילה _____ פעולות: main בעלת הרשאת גישה פומבית ו- _____ בעלת הרשאת גישה _____.

משימה 9 – חלק ב'

ההוראה: `birthday = new Date (day,month,1988)`; דואגת ליצירת עצם מטיפוס `Date` והצבת הפניה אליו במשתנה `birthday`. בעת ביצוע הפעולה, מספר הפעמים שההוראה מתבצעת הוא כמספר _____ (המחלקה יוצרת עצם מטיפוס `Date` עבור כל תלמיד). לאחר יצירת

כל עצם, המחלקה מפעילה עליו את הפעולה `DaysPass`, מעדכנת את `pass` בהתאם לערך המוחזר, מציגה את הערך של `pass` והמשתנה `birthday` מקבל הפניה לעצם אחר (העצם שיווצר עבור התלמיד הבא). לכן, נוצרים מספר רב של עצמים, שלא ניתן לגשת אליהם!

נשנה כעת את המחלקה כך שתיצור רק עצם אחד מטיפוס `Date` ותעדכן אותו כך שיכיל בכל פעם ערכים של התכונות עבור תלמיד אחר.

ראשית, נוסיף למחלקה `Date` שלוש פעולות שדואגות לעדכן את התכונות של העצם הנוכחי:

תיאור הפעולה	החתימה של הפעולה
הפעולה מעדכנת את היום בהתאם לפרמטר הנתון. הנחה: הערך של הפרמטר הוא מספר שלם 1-31	<code>void SetDay (int newDay)</code>
הפעולה מעדכנת את החודש בהתאם לפרמטר הנתון. הנחה: הערך של הפרמטר הוא מספר שלם 1-12	<code>void SetMonth (int newMonth)</code>
הפעולה מעדכנת את השנה בהתאם לפרמטר הנתון. הנחה: הערך של הפרמטר הוא מספר שלם וחיובי	<code>void SetYear (int newYear)</code>

```
public void SetDay (int newDay)
{
    this._____ = _____ ;
}
```

- השלימו את השלד של הפעולה `setDay`
- היכנסו לקובץ מחלקה `Date.cs` והוסיפו בו את שלוש הפעולות החדשות:
`SetDay`, `SetMonth`, `SetYear`

משימה 9 – חלק ג'

שנו כעת את המחלקה מסעיף א' כך שהעצם `Date` יוגדר רק פעם אחת.

רמזים:

1. אפשר להתחיל ביצירת עצם שלא נשתמש בו, למשל כך: `birthday = Date(0,0,1988)`, ובמהלך הפעולה, לעדכן את העצם באמצעות הפעולות החדשות.
 2. הפעלה (זימון) של הפעולה `SetMonth` תהיה כך: `birthday.SetMonth (_____)`;
- שימרו את המחלקה, הריצו ובדקו שהתקבל הפלט המבוקש.

ההוראה **תכונה** `get` מחזירה את הערך של התכונה עבור העצם הנוכחי.
ברוב המחלקות שנכתוב, נרצה לאפשר למשתמש לשנות את הערכים של התכונות. לכן, נגדיר בהן את ההוראה `set` עבור כל אחת מהתכונות של העצם.

משימה 10 – חלק א'

עד עכשיו, קליטת הנתונים מהמשתמש, נעשתה במחלקה שלא מגדירה עצמים, והנתונים הועברו כפרמטרים לפעולה הבונה של המחלקה המתאימה. אבל- אפשר גם לקלוט נתונים מהמשתמש בפעולה הבונה עצמה.

• **נוסף** כעת למחלקה `Date` **פעולה בונה** חדשה. בפעולה החדשה: היום, החודש והשנה יתקבלו באמצעות

```
public Date ()
{

    Console.WriteLine ("enter day");
    this.day = int.Parse(Console.ReadLine());
    Console.WriteLine ("_____");
    this.month = int.Parse(Console.ReadLine());
    Console.WriteLine ("enter year");
    this.year = _____;
}
```

הוראת קלט מהמשתמש.

לפניכם שלד של הפעולה הבונה החדשה.

השלימו אותו.

• הקלידו את הפעולה החדשה בקובץ

המחלקה `Date`.

אל תמחקו את הפעולה הבונה שכבר קיימת במחלקה!

משימה 10 – חלק ב'

• כדי לבדוק את הפעולה הבונה החדשה,

פיתחו מחלקה חדשה בתוך הפרויקט `DateProject` והקלידו בה פעולה ראשית שתיצור עצם מטיפוס `Date` ותדפיס את התכונות שלו ואת ערכיהן (באמצעות זימון הפעולה `ToString()`). יצירת העצם תהיה

באמצעות הפעולה הבונה החדשה.

רמז: הפעלה של הפעולה הבונה החדשה: `new Date();`

• שימרו, הריצו ובידקו שהתקבל הפלט המבוקש.

במחלקה שמגדירה עצם כמו המחלקה `Date` או המחלקה `Bucket`, אפשר להשתמש בכל ההוראות המוכרות בשפת `C#`. לכן, ניתן גם לקלוט בה נתונים מהמשתמש.

למעשה, המחלקה `Date` מכילה כעת **שתי פעולות בונות**. ההבדל ביניהן הוא ברשימת הפרמטרים. הפעולה הבונה הישנה קיבלה _____ פרמטרים, והפעולה הבונה החדשה לא מקבלת אף פרמטר.

ג'אוה מאפשרת להגדיר מספר פעולות בעלות אותו שם בתנאי שהן שונות זו מזו ברשימת הפרמטרים. השינוי יכול להתבטא במספר הפרמטרים / או בטיפוסים שלהם. במקרה שלנו, הגדרנו במחלקה `Date` שתי פעולות בונות ששונות זו מזו ב _____ . בעת הזימון של הפעולה המהדר (הקומפיילר) יבצע את הפעולה המתאימה לרשימת הפרמטרים שתסופק לו.

משימה 11 – חלק א'

נוסיף כעת למחלקה Date פעולה GetValidNum שתקבל מספר טבעי ותדאג לקלוט מהמשתמש מספר בתחום שבין 1 לבין המספר שקיבלה. הפעולה תחזיר את המספר הנקלט. למשל, אם הפעולה תקבל את הערך 12, היא תדאג לקלוט מהמשתמש מספר בתחום שבין 1 לבין 12 ותחזיר את הערך הנקלט.

- לפניכם שלד של הפעולה GetValidNum. השלימו אותו.

```
private int GetValidNum (int num)
```

```
{
```

```
    int answer;
```

```
    do
```

```
    {
```

```
        Console.WriteLine ("enter number between 1 to "+ num);
```

```
        answer = = int.Parse(Console.ReadLine());
```

```
    } while (answer <1 _____ answer > _____);
```

```
    return answer;
```

```
}
```

כמו כל הפעולות האחרות במחלקה Date, גם הפעולה GetValidDay פועלת על עצם מסוים ולכן לא רושמים static בחתימה שלה.

- הוסיפו את הפעולה החדשה לקובץ המחלקה Date.

- שנו את הפעולה הבונה Date() כך שתשתמש בפעולה החדשה.

- כדי לוודא שהמחלקה Date תקינה, הריצו פעם נוספת את המחלקה שכתבתם במשימה 10ב'.

- כשתתבקשו להקליד נתונים, הקלידו גם נתונים שגויים.

- מדוע הפעולה GetValidNum היא בעלת הרשאת גישה פרטית private? _____

גם במחלקה שמגדירה עצם (כמו המחלקה Date), אפשר להגדיר פעולות עזר בעלות הרשאת גישה פרטית (private) שניתן להשתמש בהן רק מתוך המחלקה בה הן מוגדרות.

משימה 11 – חלק ב'

- שנו את משימה 8 כך שתשתמש בפעולה הבונה החדשה.

המשימה: לכתוב פעולה ראשית שקולטת נתונים של תאריכים ויוצרת עצם מטיפוס Date עבור כל אחד מהתאריכים. הפעולה תסכם את הימים שעברו מתחילת השנה עד לכל תאריך. קליטת הנתונים תפסק כאשר סך הימים שעברו מתחילת השנה עבור כל התאריכים יהיה 400 או יותר. הפעולה תדפיס את מספר הימים המדויק שעברו עבור כל התאריכים ביחד.

- שימרו, הריצו ובידקו שהתקבל הפלט המבוקש.

משימה 12

```
namespace DateProject
{
class Program
{
public static void main(String[] args)
{
Date d1 = new Date (10,9,2006);
Console.WriteLine ("before test d1 = " + d1.ToString());
Test (d1);
Console.WriteLine ("after test d1 = " + d1.ToString());
}
private static void Test (Date f1)
{
f1.SetDay (12);
f1.SetMonth (12);
f1.SetYear (2012);
Console.WriteLine ("in test f1 = " + f1.ToString());
}
}
}
```

בפרויקט DateProject הקלידו את התכנית הבאה. (את התכנית קודמת אפשר לסמן כהערה) שימרו, הריצו והשלימו: (שימו לב שהפעם משתמשים בפעולה הבונה הראשונה).

• המחלקה כוללת שתי פעולות.

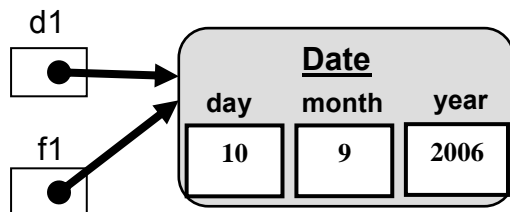
פעולה _____ ופעולה

משנית Test .

• הפעולה test מקבלת כפרמטר

הפניה לעצם מטיפוס

הפעולות שכתבנו עד עכשיו קיבלו כפרמטרים אקטואליים ערכים של משתנים או נתונים ישירים (שאינם שמורים בתוך משתנים) כמו מספרים או תווים. במקרה של עצמים, הפרמטר המועבר הוא הפניה לעצם.

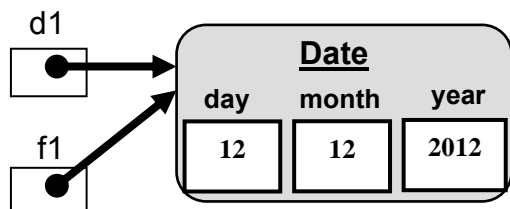


הפעולה מקבלת ממי שמזמן אותה הפניה לעצם ולא את העצם עצמו!

לכן, במקרה שלנו, במשתנה d1 (של הפעולה הראשית)

ובמשתנה f1 (של הפעולה Test) נמצאת הפניה לאותו עצם מטיפוס Date.

כתוצאה מכך, כאשר הפעולה Test משנה את הערכים של התכונות של העצם שהפניה אליו נמצאת ב f1, היא משנה גם את הערכים של התכונות של העצם שהפניה אליו נמצאת ב d1 (שהרי מדובר באותו עצם!).



לכן, לאחר הפעלת הפעולה Test,

הערכים של התכונות של Date שהפניה אליו נמצאת ב d1, הם הערכים כפי שנקבעו

בפעולה _____.

בהעברת עצם כפרמטר, אנחנו מעבירים את הפניה לעצם ולא את העצם עצמו!

משימה 13 – חלק א'

לפניכם ממשק של המחלקה Time

כותרת הפעולה	תיאור הפעולה
Time (int hour, int minute)	הפעולה הבונה: יוצרת עצם מטיפוס Time על פי פרמטרים נתונים. הנחות: hour שלם 0-23, minute שלם 0-59
int GetHour()	מחזירה את השעה.
int GetMinute()	מחזירה את הדקות.
boolean RestTime ()	מחזירה true אם זו שעת מנוחה, ומחזירה false אחרת. שעת מנוחה: 00:30-7:30, 14:00-15:59
void Add (int minutesToAdd)	מוסיפה minutesToAdd לזמן הנוכחי. minutesToAdd יכול להיות גדול מ-60
String ToString()	מחזירה מחרוזת הכוללת את הזמן המדויק בצורה הבאה: <i>דקות:שעה</i>

ממשו את המחלקה Time. כלומר, כתבו כעת ההוראות למימוש המחלקה.

- תזכורות: 1. הגדירו תחילה פרויקט חדש (File -- New -- Project.. -- Next > -- *הקלדת* *ע* *ל* *פ* *ר* *ו* *י* *ק* *ט* *ט*)
 -- Next > -- Finish). ובפרויקט החדש הגדירו מחלקה חדשה בשם Time.
 2. התכונות של המחלקה אינן חלק מהממשק משום שהתכונות הן פרטיות למחלקה והן לא מענייניו של מי שמשתמש במחלקה. אך קובץ המחלקה צריך להתחיל בתיאור התכונות.

משימה 13 – חלק ב'

- תלמידי הכיתה מתכננים מסיבת סיום. כל תלמיד שמעוניין יכין הופעה קצרה.
 פיתחו מחלקה חדשה וכיתבו בה פעולה ראשית שתקלוט את שעת ההתחלה של המסיבה (שעה ודקות) ותיצור עבורה עצם מטיפוס Time. לאחר מכן הפעולה תקלוט את זמני ההופעות (בדקות) של כל התלמידים. לאחר קליטת כל זמן כזה, הפעולה תבדוק האם ההופעה תסתיים בשעת מנוחה. התהליך יפסק כאשר ייקלט זמן הופעה 0, או כאשר ייקלט זמן הופעה שיסתיים בשעת מנוחה.
 לאחר קליטת הנתונים, הפעולה :mhd,
 א. את זמן ההתחלה וזמן הסיום של המסיבה.
 ב. את מספר ההופעות שיתקיימו.
 ג. במידה וההופעה האחרונה תסתיים בשעת מנוחה, הפעולה, תציג הודעה מתאימה.
 דוגמה 1: קלט: שעת התחלה: 23:00, זמני הופעות (משמאל לימין): 0 8 14 20 12 10
 הפלט: שעת התחלה: 23:00 שעת סיום: 00:04 יתקיימו 5 הופעות.
 דוגמה 2: קלט: שעת התחלה: 23:10, זמני הופעות (משמאל לימין): 15 22 25 15 12
 הפלט: שעת התחלה: 23:10 שעת סיום: 00:39 יתקיימו 5 הופעות. ההופעה האחרונה תסתיים בשעת מנוחה.
 • שימרו, הריצו ובדקו שהתקבל הפלט המבוקש.