

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 מד"ר תמר פז.

המעבדה בקובץ זה מיועדת לתלמידים הלומדים מדעי המחשב בשפת התכנות ג'אווה והיא מותאמת לסביבת אקליפס.
המעבדה מיועדת לשיעורי המעבדה והיא מבוססת על שיטת ההוראה לפיה הלימוד של כל נושא חדש יפתח 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)

מעבדה זו היא מעבדת המשך למעבדה שמינית - עצמים.

במעבדה זו נעשה שימוש במחלקה Bucket. הרעיון למחלקה Bucket לקוח מתוך: המרכז להוראת המדעים, האוניברסיטה העברית, עיצוב תכנה מבוסס עצמים, 2005.
אבל,
מחלקה Bucket במעבדות אלו איננה זהה למחלקה Bucket של האוניברסיטה העברית.
מצורף קובץ מחלקה Bucket המתאים למעבדה זו.

מבנה של מחלקה (class)

משימה 13

```

1 // This class represents a Bucket
2 public class Bucket {
3 // The attributes of the class
4 private double capacity;
5 private double currentAmount;
6 /* A constructor for the class. receives one parameter for capacity.
7 Builds a new empty Bucket according to this parameter */
8 public Bucket (double capacity)
9 {
10 this.capacity = capacity;
11 this.currentAmount = 0;
12 }
13 // Methods:
14 /* returns the capacity of the bucket */
15 public double getCapacity()
16 {
17 return this.capacity;
18 }
19 /* returns the current amount in the bucket */
20 public double getCurrentAmount()
21 {
22 return this.currentAmount;
23 }
24 /* receives amount and add it to the amount in the bucket */
25 public void addAmount (double amountToAdd)
26 {
27 // if the capacity is too small
28 if (this.capacity < this.currentAmount + amountToAdd)
29 {
30 this.currentAmount = this.capacity;
31 }
32 else
33 this.currentAmount = this.currentAmount+ amountToAdd;
34 }
35 /* receives amount and remove it from the bucket */
36 public void removeAmount (double amountToRemove)
37 {
38 if (this.currentAmount < amountToRemove)
39 this.currentAmount = 0;
40 else
41 this.currentAmount = this.currentAmount- amountToRemove;
42 }
43 /* empties the bucket completely */
44 public void emptyAll()
45 {
46 this.currentAmount = 0;
47 }
48 /* checks if the bucket is empty */
49 public boolean isEmpty()
50 {
51 return this.currentAmount == 0;
52 }
53 /* builds and returns a String from the bucket's attributes and their value */
54 public String toString()
55 {
56 return "(my capacity=" + this.capacity + " my current amount=" + this.currentAmount+");";
57 }
58 } // end of class

```

this = העצם הנוכחי
עליו מתבצעת הפעולה

במחלקה (class) Bucket.java מוגדרות התכונות של עצם מטיפוס Bucket, ומוגדרות הפעולות שניתן לבצע עליו ועימו.

פתחו את המחלקה Bucket (בחלון השמאלי, הקליקו פעמיים עם המקש השמאלי של העכבר על Bucket.java). מספור השורות איננו חלק מהמחלקה והוא מופיע פה לצורך הדיון!

- בשורה 2 מופיעה הכותרת של המחלקה:

```
public class Bucket {
    משמעותה: תוגדר כאן מחלקה
    (class) ששמה Bucket.
```

המילה public מציינת שנוכל להשתמש במחלקה הזו גם ממחלקות אחרות.

- בשורות 4,5 מוגדרות התכונות

(attributes) של המחלקה.

התכונות הללו הן התכונות

שהיו לכל העצמים מטיפוס

המחלקה Bucket. במקרה

שלנו, לכל עצם שיוגדר מטיפוס

המחלקה Bucket יהיו שתי

תכונות: capacity ו-

currentAmount, שתיהן

מטיפוס _____.

שימו לב ש-String נכתב עם S גדולה. נעסוק בכך בהמשך

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

- התכונות מוגדרות כך: **private** *טכונה* *טיפוס התכונה* **private**
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 המאחזרות.
 - בשורות 8-12 מופיעה **הפעולה הבונה**. פעולה זו בונה עצם יחיד מטיפוס המחלקה Bucket ונותנת ערכים לתכונות של העצם הנוכחי אותו היא בונה. במקרה שלנו, **הפעולה הבונה נותנת ערכים לתכונות של העצם הנוכחי אותו היא בונה באמצעות הוראות השמה**.
 - **החתימה של הפעולה הבונה: (רשימת פרמטרים וטיפוסיהם) *public* *המחלקה* *public***
 - **בפעולה הבונה, הערכים של התכונות יכולים להקבע על-ידי מי שמשתמש במחלקה, ויכולים להיות קבועים עבור כל העצמים שייבנו מטיפוס המחלקה**. במקרה שלנו, הערך של התכונה capacity נקבע על-ידי מי שמשתמש במחלקה ומועבר אל הפעולה הבונה כפרמטר מטיפוס _____.
 - והערך של התכונה currentAmount הוא קבוע לכל העצמים שייבנו מטיפוס Bucket (הפעולה הבונה, נותנת לתכונה זו תמיד את הערך _____).
 - המילה **public** בחתימה של הפעולה הבונה מציינת כי ניתן להפעיל את הפעולה ולבנות עצמים מטיפוס המחלקה גם מתוך מחלקות אחרות.
 - משורה 13 ועד לסוף הקובץ מופיעות הפעולות (שיטות = methods) הנוספות שניתן לבצע עם ועל עצם מטיפוס Bucket.
 - **חתימה של פעולה שאיננה הפעולה הבונה** כמו חתימה של הפעולות בהן עסקו עד כה ללא המילה static :
(רשימת פרמטרים וטיפוסיהם) *public* *הפעולה* *טיפוס הצרכ האוחזר* *public*
 - המילה **public** בחתימה של הפעולה מאפשרת גם למחלקות אחרות להשתמש בפעולות הללו.
 - **השמטת המילה static מהחתימה של פעולה, מציינת כי זו פעולה שפועלת על עצם!**
 - כדי להקל את ההתמצאות בקובץ המחלקה, נפתח תמיד בפעולות המאחזרות (שמחזירות את הערכים של התכונות של העצם). במקרה שלנו, _____, ו- _____.
 - לאחריהם יופיעו שאר הפעולות. הפעולה האחרונה תהייה הפעולה toString שתפקידה _____.
- כמו בפעולות בהן עסקנו עד כה:
1. כאשר פעולה לא מקבלת פרמטרים, בחתימה של הפעולה, במקום רשימת הפרמטרים, רושמים סוגיים () למשל `public double getCapacity()`
 2. כאשר פעולה לא מחזירה ערך, בחתימה של הפעולה, במקום הערך המוחזר, רושמים: `void`.
 3. ולכן, החתימה של פעולה שלא מקבלת פרמטרים ולא מחזירה ערך, תראה כך:
`public _____ הפעולה _____`

- שימו לב: התייעוד הוא חלק בלתי נפרד מהמחלקה!

- הגדרה של תכונה: **private** *טכונה טיפוס התכונה*
- חתימה של הפעולה הבונה: **public** *רשימת פרמטרים וטיפוסיהם* **טכונה המחלקה**
- חתימה של פעולה שאיננה הפעולה הבונה: **private** *רשימת פרמטרים וטיפוסיהם* **טכונה הצורך המחלקה**
- 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 14 – 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.java`

הקובץ עימו אנו עובדים `Bucket.java` 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 מחלקה חדשה בתוך הפרויקט `BucketProject` והקלידו בה פעולה ראשית שתקלוט מהמשתמש קיבולת וכמות נוכחית של דלי. הפעולה תיצור עצם מטיפוס דלי ותציג כפלט את התכונות של הדלי וערכיהן (באמצעות זימון הפעולה `toString()`).
- שימרו את המחלקה, הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 14 – 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.

משימה 15

- הפעולה `addAmount` (שורות 25-34) 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 תוצג כפלט הודעה מהי הכמות העודפת.
- כיתבו פעולה ראשית במחלקה אחרת שתבדוק את השינוי שבצעתם במחלקה `Bucket`. המחלקה תיצור דלי ותנסה למלא אותו בכמויות העולות על הקיבולת שלו.
- שימרו, הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 16 – 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.
- כיתבו פעולה ראשית במחלקה אחרת שתיצור דלי (תתן לתכונותיו ערכים שיתקבלו מהמשתמש), תפעיל את הפעולה החדשה `fillAll`, ותציג כפלט את התכונות של הדלי ואת ערכיהם.
- שימרו, הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 16 – 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.
- כיתבו פעולה ראשית במחלקה אחרת שתבדוק את הפעולה `isFull`. המחלקה תיצור דלי (תיתן לתכונותיו ערכים שיתקבלו מהמשתמש). לאחר מכן, המחלקה תקלוט מהמשתמש 3 ערכים למילוי, תוסיף אותם לדלי, תציג כפלט את התכונות של הדלי ואת ערכיהן, וכן את הערך המוחזר מהפעולה `isFull`.
- הריצו מספר פעמים כך שגם יוצג כפלט הערך `true` וגם יוצג כפלט הערך `false`.

בניית מחלקה

משימה 17

המחלקה Bucket וכן המחלקות שמשמשות בה נכתבו כתת מדורים של הפרויקט BucketProject. איחוד כל המחלקות הללו בפרויקט אחד איפשר לנו להגדיר מחלקה בקובץ אחד (Bucket) ולהשתמש בה בקבצים האחרים.

נכתוב כעת מחלקה חדשה **Date**. כדי לכתוב מחלקה חדשה נגדיר תחילה פרויקט חדש. לשם כך, בשורת הכפתורים העליונה בחרו ב **File**, בחלון שנפתח בחרו ב **New** ובחלון שנפתח כעת בחרו ב **Project..**. בתחתית החלון שנפתח כעת לחצו על **> Next**. כעת תתבקשו לתת שם לפרויקט, הקלידו את השם **DateProject** ואשרו (לחצו על **> Next**). אשרו גם בחלון הבא (לחצו על **Finish**).

שימו לב כי בחלון הקבצים (החלון השמאלי) יופיע כעת הפרויקט החדש **DateProject**. נפתח כעת מחלקה חדשה ונגדיר בה את המחלקה החדשה **Date**. הפתיחה של מחלקה שתשמש להגדרת מחלקה חדשה היא בדיוק כמו הפתיחה של מחלקה שמשמשת במחלקה קיימת. לכן, 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** ולחצו על **Finish**.

```
public class Date
{
    public static void main(String[] args)
    {
    }
}
```

בעורך, נפתח כעת קובץ חדש שיש בו השורות הבאות: השאירו רק את השורות המודגשות (הכותרת של המחלקה **Date** וסוגריים לפתיחת המחלקה ולסגירתה).

לפניכם הממשק של המחלקה **Date**.

החתימה של הפעולה	תיאור הפעולה
Date (int day , int month , int year)	פעולה בונה: יוצרת עצם מטיפוס Date על פי פרמטרים נתונים. הנחה: day מספר שלם 1-31, month מספר שלם 1-12, year שלם וחיובי.
int getYear ()	מחזירה את השנה.
int getMonth ()	מחזירה את החודש.
int getDay ()	מחזירה את היום.
int daysInYear()	מחזירה את מספר הימים בשנה (366 אם השנה מתחלקת ב- 4 ללא שארית. 365 אחרת).
boolean isInteresting()	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).
int daysPass()	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).
String toString()	מחזירה מחרוזת המתארת את התאריך בצורה הבאה: <i>je / e7n / יומ</i>

השלימו כעת את המחלקה Date.

• תזכורות:

1. חתימה של פעולה:

public (רשימת פראמטרים וטיפוסיהם) **שם הפעולה** **טיפוס הצרך האחוזר** **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: `day`,

`month`, `year`. 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 18 – חלק א'

כדי לבדוק את תקינות המחלקה שכתבתם, פתחו מחלקה חדשה (כחלק מהפרויקט DateProject) וכתבו

בה פעולה שתקלוט מהמשתמש יום, חודש ושנה, תיצור עצם מטיפוס Date ותציג כפלט 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 18 – חלק ב'

הוסיפו לפעולה הראשית את ההוראות הדרושות כך שתבדוק את כל הפעולות של המחלקה Date ותציג

כפלט את הערכים המוחזרים על-ידן.

• הריצו ובידקו שהתקבל הפלט הרצוי.

משימה 19

כתבו פעולה שתקלוט את תאריכי הלידה של 20 תושבי הבניין. הפעולה תבצע:

- א. תיצור עצם מטיפוס Date עבור כל תאריך.
- ב. תמנה ותודיע כמה מהתאריכים הם "מעניינים" בהתאם להגדרה בממשק המחלקה Date.
- ג. תציג כפלט את תאריך הלידה האחרון שנקלט.

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);**

- פיתחו מחלקה חדשה (כחלק מהפרויקט DateProject), הקלידו בה את הפעולה, שימרו, הריצו ובדקו כי התקבל הפלט מבוקש.

משימה 20

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 תציג כפלט 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.

- פיתחו מחלקה חדשה (כחלק מהפרויקט DateProject), הקלידו בה את הפעולה, שימרו, הריצו ובדקו כי התקבל הפלט מבוקש.

יצירת עצם מטיפוס Date == יצירת מופע של המחלקה Date

משימה 21 – חלק א'

כל תלמידי כיתה י"ב נולדו בשנת 1988.

לפניכם שלד של מחלקה שמכילה פעולה ראשית שמודיעה לכל תלמיד כמה ימים יחלפו מתחילת השנה עד ליום ההולדת שלו. המחלקה תבקש מהמשתמש להקליד את יום הלידה של כל התלמידים שנולדו

בחודש מספר 1,

ולאחריהם את המספר

0. התהליך יחזור עבור

חודש מספר 2, לאחריו

עבור חודש מספר 3 וכן

הלאה.

• השלימו את המחלקה

• הקלידו, שימרו, הריצו

ובדקו שהתקבל הפלט

המבוקש.

• מדוע הפעולה

getValidDay היא

בעלת הרשאת גישה

פרטית? _____

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

```
import java.util.Scanner;
public class Birthday
{
    public static void main (String[] args)
    {
        int _____; // יום
        int _____; // חודש
        int _____; // מספר הימים שחלפו מתחילת השנה לתלמיד הנוכחי
        Date birthday;
        for (month = 1; _____ < 13; _____++)
        {
            // קליטת יום ההולדת הראשון בחודש month
            day = getValidDay(month);
            while (day != 0) // month בחודש הנוספים
            {
                birthday = new Date (day,month,1988);
                pass = birthday.daysPass();
                System.out.println ("birthday is "+ pass+" days after year start");
                day = getValidDay(month);
            } // end while
        } // end for
        _____ // end main
        private static int getValidDay (int month)
        {
            Scanner input = new Scanner(System.in);
            int day;
            do
            {
                System.out.println ("enter day for month "+ month + " or 0 for end ");
                day = input.nextInt();
            } while (day < 0 _____ day > 31);
            return day;
        }
    }
}
```

משימה 21 – 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 כפלט 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 שתייצר רק עצם אחד מטיפוס `Date`, ותעדכן אותו כך שיכיל בכל פעם ערכים של התכונות עבור תלמיד אחר.

ראשית, נוסיף למחלקה `Date` שתי 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>

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

- השלימו את השלד של הפעולה `setDay`
- הכנסו לקובץ מחלקה `Date.java` והוסיפו בו את שתי הפעולות החדשות: `setDay`, `setMonth`

משימה 21 – 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=new 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 **מכוננת** `set` 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 22 – 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.

- **נוסיף** כעת למחלקה `Date` פעולה **בונה** חדשה. בפעולה החדשה: היום, החודש והשנה יתקבלו

```
public Date ()
{
    Scanner input = new Scanner(System.in);
    System.out.println ("enter day");
    this.day = input.nextInt();
    System.out.println ("_____");
    this.month = input.nextInt();
    System.out.println ("enter year");
    this.year = _____;
}
```

באמצעות 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!

- הוסיפו בתחילת קובץ המחלקה `Date` את השורה:

```
import java.util.Scanner;
```

משימה 22 – חלק ב'

- כדי לבדוק את הפעולה הבונה החדשה, פיתחו מחלקה חדשה בתוך הפרויקט `DateProject` והקלידו בה פעולה ראשית שתיצור עצם מטיפוס `Date` ותציג כפלט 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 ג'אווה. 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'אווה מאפשרת להגדיר מספר פעולות בעלות אותו שם בתנאי שהן שונות זו מזו ברשימת הפרמטרים. השינוי יכול להתבטא במספר הפרמטרים ו/ 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 23 – 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 שנקלט.

- לפניכם שלד של הפעולה `getValidNum`. השלימו אותו.

```
private int getValidNum (int num)
```

```
{
    Scanner input = new Scanner(System.in);
    int answer;
    do
    {
        System.out.println ("enter number between 1 to "+ num);
        answer = input.nextInt();
    } while (answer <1 _____ answer > _____);
    return answer;
}
```

כמו כל הפעולות האחרות במחלקה `Date`, גם הפעולה `getValidDay` פועלת על עצם מסוים ולכן לא רשמים `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 10 ב'. 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 23 – חלק ב'

- שנו את משימה 20 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 תציג כפלט את מספר הימים המדויק שעברו עבור כל התאריכים ביחד.
- שימרו, הריצו ובידקו שהתקבל הפלט המבוקש.

```

public class TestParameter
{
    public static void main(String[] args)
    {
        Date d1 = new Date (10,9,2006);
        System.out.println ("before test d1 = " + d1.toString());
        test (d1);
        System.out.println ("after test d1 = " + d1.toString());
    }
    private static void test (Date f1)
    {
        f1.setDay (12);
        f1.setMonth (12);
        f1.setYear (2012);
        System.out.println ("in test f1 = " + f1.toString());
    }
}

```

משימה 24

פיתחו מחלקה חדשה (בפרויקט DateProject) והקלידו בה את המחלקה הבאה.

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

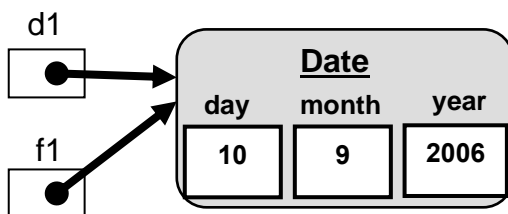
הפעולות שכתבנו עד עכשיו קיבלו כפרמטרים אקטואליים ערכים של משתנים או נתונים ישירים (שאינם שמורים בתוך משתנים) כמו מספרים או תווים.

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)

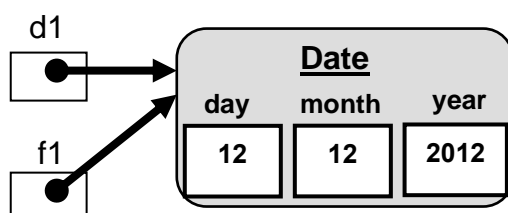
ובמשתנה f1 (של הפעולה test) נמצאת הפניה לאותו

עצם מטיפוס Date.



כתוצאה מכך, כאשר הפעולה test משנה את הערכים של התכונות של העצם שההפניה אליו נמצאת ב f1, היא משנה גם את הערכים של התכונות של העצם שההפניה אליו נמצאת ב d1 (שהרי מדובר באותו עצם!). לכן, לאחר הפעלת הפעולה test, הערכים של התכונות של Date שההפניה אליו

נמצאת ב d1, הם הערכים כפי שנקבעו בפעולה _____.



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

משימה 25 – חלק א'

לפניכם ממשק של המחלקה Time

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

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

- תזכורות: 1. הגדירו תחילה פרויקט חדש (File -- New -- Project.. -- Next > *הקלדת* *ע* *ל* *פ* *ר* *ו* *י* *ק* *ט* -- Finish -- Next >). 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 25 – 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 עבודה עצם מטיפוס Time. לאחר מכן הפעולה תקלוט את זמני ההופעות (בדקות) של כל אחד מהתלמידים. 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 תציג כפלט:
- א. את זמן ההתחלה וזמן הסיום של המסיבה.
- ב. את מספר ההופעות שיתקיימו.
- ג. במידה וההופעה האחרונה תסתיים בשעת מנוחה, הפעולה תציג כפלט 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.