

פרק 3 מחלקות

1. הזמן הנדרש

6 שעות.

2. מטרות

1. היכרות עם תכונות מחלקה ועם פעולותיה.
2. היכרות עם תרשימי העצמים והמחלקות.
3. הגדרת מחלקה וכתיבתה.
4. תיעוד מחלקה.

3. מבנה הפרק

א – הכרזה על מחלקה

א.1. יש לחזור על חשיבותה של כתיבה נכונה על פי מוסכמות, ולהסביר כי המוסכמות אמנם אינן מחייבות מבחינת התחביר של השפה, אך הן חלק חשוב בתכנות ובעיצוב נכון וקריא של התוכנית. אנו נקפיד להשתמש בהן בכל הדוגמאות והפתרונות שנביא, ונדרוש שגם התלמידים יקפידו עליהן בתרגילים שהם מגישים.

שמות "עבריים" (כגון: Dli) אינם מקובלים, שכן הם אינם מתאימים לשימוש רחב ובינלאומי. לא כל התלמידים יכירו בהכרח את כל המילים באנגלית שיופיעו בשיעור ובתרגילים, ולכן יש לעודד אותם לשאול או לחפש במילון. יש להסב את תשומת לבם של התלמידים לעובדה שהשימוש באותיות לועזיות גדולות במקום קטנות, או להפך, משנה את משמעות המילה או הפעולה (שכן שפת ג'אווה היא case sensitive), כך למשל הרצף box אינו זהה ל-Box.

ב – מצב של עצם

בפרק הקודם עסקנו במצב של עצם ללא התייחסות למושג תכונות, משום שהפרק הקודם עסק בשימוש במחלקה ולא בבנייתה. בפרק הזה אנו מגדירים שמצב של עצם נקבע על ידי תכונות של אותו העצם.

תהליך נכון בתכנון מחלקה מתחיל בהגדרת המצב, כלומר מה משמעותו של העצם למשתמש; מהי רשימת תלמידים, ולא מה הייצוג שלה. צריך להגדיר מהן הפעולות הנדרשות ומה הן עושות, ורק בשלב שני לחפש ייצוג שיתמוך במה שנרצה שהעצם יבצע.

ג – פעולה בונה

1. הסעיף מבהיר כי הפעולה הבונה, למרות שמה, אינה בונה את העצם. למעשה הפעולה מאתחלת את העצם (ייתכן שהשם הנכון יותר הוא "פעולה מאתחלת" ולא פעולה בונה). יש

לשים לב לכך שלעיתים התלמידים טועים וחושבים שהמטרה של הפעולה הבונה היא אכן לבנות עצם. טעות זו יכולה להוביל אותם לנסות לייצר עצם באמצעות האופרטור `new` בגוף הפעולה הבונה ולנסות לבצע פקודת `return` של העצם שנוצר. חשוב להבהיר כי הפעולה הבונה רק מאתחלת את התכונות. האופרטור `new` הוא זה שבונה את העצם לפני הזימון של הפעולה הבונה.

2. בדרך המקובלת, קליטת הערכים מהמשתמש אינה נעשית בפעולה הבונה, אלא לפני זימון הפעולה הבונה (למשל בפעולה הראשית). רק אחר כך מזמנים את הפעולה הבונה עם הערכים שנקלטו. זאת כיוון שאנו רוצים לייצר מחלקות שאינן תלויות חומרה, ומחלקה שמשמשת בקליטת נתונים נחשבת לתלויה חומרה.

3. נקודה חשובה נוספת בסעיף זה: עבור פעולה בונה ללא פרמטרים, המתכנת מחליט מהם הערכים של התכונות. התכונות אינן חייבות לקבל ערכי ברירת מחדל, שהם 0 לטיפוסים הפשוטים או `null` לעצמים (אף שלרוב זה יהיה המצב). למשל, בדוגמה שלנו אין זה הגיוני שהתכונה `capacity` תקבל ערך 0 (שהרי מכך משתמע שנפחם של הדליים הוא 0). לכן הערך שאלינו יאותחלו הדליים יהיה 10 (כמובן שיכולנו לבחור כל ערך אחר). במקרה של זימון הפעולה הבונה ללא פרמטרים, יש לתעד כל ערך שהתכונות מאותחלות אליו.

4. כאשר אנו פונים לתכונות בתוך הפעולות נשתמש במילה השמורה `this`. למעשה, שימוש זה אינו הכרחי.

אם נוותר על השימוש ב-`this` ונכתוב, למשל, בפעולת ריקון הדלי:

```
public void empty() /Empty()  
{  
    capacity = 0;  
}
```

התוכנית תחפש משתנה בשם `capacity` בטווח (scope) הקרוב ביותר. משתנה שכזה נמצא רק בתכונות. התוכנית תזהה את המשתנה כתכונה, ותמשיך להתנהל כרגיל. דווקא כאשר בפעולה מוגדר פרמטר או משתנה בעל שם התכונה של המחלקה, עלולה להתעורר בעיה. לדוגמה, התבוננו בפעולה הבונה של הדלי:

```
public Bucket (int capacity)  
{  
    Capacity = capacity;  
    currentAmount = 0;  
}
```



התכונה `currentAmount` תאותחל באופן נכון, כיוון שבטווח הקרוב היא המשתנה היחיד בעל שם זה.

לעומת זאת התכונה capacity תאוחל באופן שגוי. בטווח הקרוב ביותר נמצא פרמטר בשם זה ולא התכונה. במקרה זה התוכנית תתייחס לשני האזכורים של capacity כאילו הם אותו הפרמטר. התוכנית תציב את הפרמטר בתוך עצמו, וערך התכונה לא ישתנה כלל! לפיכך, נקפיד תמיד על פנייה לתכונה באמצעות `this`. כך נימנע מטעויות ונחזק אצל התלמידים את התחושה שתכונה אינה משתנה "תלוש", והיא שייכת תמיד לעצם מסוים. כפי שאנו כותבים פנייה לפעולות פומביות: `b1.empty() / b1.Empty()`, כך נכתוב פנייה לתכונות בתוך המחלקה: `this.capacity`.

ביחידה זו חובה לשמור על מוסכמת השימוש ב-`this`. יש להציג לתלמידים אופן כתיבה זה כהרגל. רק אם התלמידים עצמם גילו שאפשר לפנות לתכונה ללא `this`, יש להסביר שאמנם מוסכמה זו אינה הכרחית מבחינת השפה, אך חשוב להבהיר לתלמידים את יתרונותיה.

כמו כן, שימו לב כי באמצעות המילה השמורה `this` ניתן לפנות לא רק לתכונות של העצם הנוכחי אלא גם לפעולות שלו (למשל `this.empty() / this.Empty()`).

5. פעמים רבות פעולה בונה מאתחלת את כל התכונות לפי פרמטרים המועברים אליה. יש לשים לב שלא ייווצר אצל התלמידים הרושם המוטעה שכל פעולה בונה בנויה כך. חשוב להדגיש לתלמידים שניתן לאתחל את התכונות לפי פרמטרים, לפי ערכים קבועים מראש או על ידי חישוב כלשהו. דוגמה טובה לעניין זה הן הפעולות הבונות של הדלי:

```
public Bucket (int capacity)
{
    this.capacity = capacity;
    this.currentAmount = 0;
}
```

ניתן לראות כי קיים רק פרמטר אחד: `capacity`, והתכונה `currentAmount` מאותחלת ל-0 (כיוון שהדלי ריק).

ד – תרשימים

איננו מציינים הרשאות גישה ב-UML. לרוב התכונות הן פרטיות והפעולות פומביות. אם ישנה פעולה פרטית או תכונה פומבית, יש צורך לציין זאת.

ציון הרשאות גישה בתוך תרשימים UML הוא אופציונלי, והמורה יכול להחליט שבכיתתו יקפידו על כתיבת הרשאות גישה בתוך תרשימי UML.

1.4. בסעיף זה אנו מזמנים שתי פעולות בונות שונות באותו האופן בדיוק. בזימון הבא אנו מדגימים את הזימון של הפעולה ללא הפרמטרים שהגדרנו:

```
Bucket b2 = new Bucket ();
```

ואילו בדוגמה הזו אנו מדגימים את הזימון של הפעולה הבונה ברירת המחדל:

```
Bucket b3 = new Bucket ();
```

הכוונה היא כמובן לכך שרק אחת הפעולות האלה מוגדרת במחלקה, ורק אחד הזימונים מתבצע באותה תוכנית. אין שום אפשרות להגדיר שתי פעולות שונות בעלות אותה כותרת במחלקה אחת.

ה – פעולות נוספות

1. היכרות טובה עם מבנה הכותרות של הפעולות תאפשר לתלמידים לזכור איך מגדירים פעולה, וכן להבין פעולה מתוך ממשק נתון.

2. כדאי להתעכב על הפעולה `isEmpty() / IsEmpty()` ולהסביר את התנאי המקוצר.

3. חשוב להתעכב על הפעולה `pourInto(...) / PourInto(...)`. פעולה זו מעבירה מים מהדלי הנוכחי לדלי אחר. הפעולה מקבלת כפרמטר עצם מטיפוס המחלקה עצמה. העצם נוצר לפני זימון הפעולה. הנחה סמויה שתלווה אותנו לכל אורך יחידת הלימוד היא שפרמטר מטיפוס עצם הנשלח לפעולה לעולם אינו שווה `null` אלא אם צוין אחרת במפורש (ראו הערה 1 בסעיף ט.5).

בקוד המופיע בפרק אנו מזמנים את הפעולות של העצם שהתקבל כפרמטר:

```
public void pourInto(Bucket bucketInto)
{
    double freeSpace = bucketInto.getCapacity() -
                        bucketInto.getCurrentAmount();
    ...
}
```

ובשפת סישרפ:

```
public void PourInto(Bucket bucketInto)
{
    double freeSpace = bucketInto.GetCapacity() -
                        bucketInto.GetCurrentAmount();
    ...
}
```

באותה מידה, מכיוון שאנו מצויים בתוך המחלקה `Bucket` עצמה, ניתן לפנות ישירות אל התכונות של העצם `bucketInto`, כיוון שהוא מטיפוס המחלקה:

```
public void pourInto(Bucket bucketInto)
{
    double freeSpace = bucketInto.capacity -
                        bucketInto.currentAmount;
}
```

ובשפת סישרפ:

```
public void PourInto(Bucket bucketInto)
{
    double freeSpace = bucketInto.capacity -
                        bucketInto.currentAmount;
}
```

אחרי שקוראים את הגרסה המופיעה בפרק אפשר לשאול את התלמידים האם לדעתם ניתן לפנות ישירות לתכונות ולעורר דיון.

4. הפעולה toString() / ToString() אינה מדפיסה את המחרוזת המייצגת את העצם, אלא מחזירה אותה. ניתן לקלוט את המחרוזת ולהדפיסה במקום אחר.

ו – העמסת פעולות

בפעולה בונה מעתיקה קיים חיסרון תכנותי: כאשר בתכנות מתקדם רוצים לבצע overriding של פעולת ההעתקה, אין אפשרות לעשות זאת לפעולה הבונה. אך כיוון שנושא הירושה אינה בתוכנית הלימודים חיסרון זה אינו מורגש ביחידה.

לעתים, כאשר משתמשים בפעולה בונה מעתיקה נוטים לחשוב כי זו היא האפשרות היחידה להעתקה של עצמים. ניתן במקום פעולה בונה מעתיקה להגדיר פעולה פשוטה בשם copy למשל, אשר תעתיק את כל תכונות העצם ותחזיר עצם המהווה עותק של העצם המקורי:

```
public Bucket copy(Bucket b)
{
    Bucket copyBucket = new Bucket(b.capacity);
    copyBucket.currentAmount = b.currentAmount;
}
```

ובשפת סישרפ:

```
public Bucket Copy(Bucket b)
{
    Bucket copyBucket = new Bucket(b.capacity);
    copyBucket.currentAmount = b.currentAmount;
}
```

פעולה זו מחזירה את העותק של הפרמטר.

ז – הכמסה

3.ז. בסעיף זה אנו מדגימים את השינוי של ייצוג ללא שינוי המימוש ומשתמשים לצורך הדוגמה במלבן. זוהי דוגמה חדשה לגמרי. ניתן היה לתרגל אותו רעיון בדלי המוכר, ולשנות את הייצוג של הדלי לייצוג בעזרת מערך חד-ממדי שבו שני תאים: תא אחד לקיבולת ותא אחר לכמות הנוכחית. שינוי כזה של הייצוג מעלה את הצורך לדון בעצם המורכב, דבר שאינו נכלל בתכני פרק זה. מומלץ מאוד להתעכב על המושג "טיפוס נתונים מופשט". מושג זה ילווה אותנו לכל אורך היחידה וישמש אותנו להבחנה בין סוגי אוספים שונים בהמשך. חשוב שההגדרה של טיפוס שאינו חשוף למשתמש, המוגדר באמצעות פעולותיו ולא באמצעות הייצוג שלו, תהיה ברורה לתלמידים כבר בשלב זה, שבו למעשה הם נתקלים רק במחלקות שהן טיפוסים נתונים מופשטים.

ח – תיעוד מחלקה

נושא התיעוד הוא נושא חשוב ביותר. כיוון שסביבות העבודה החינמיות של סישרפ אינן תומכות בדרך פשוטה בהפקת תיעוד מלא ממחלקה מתועדת, אנו נאלצים להציג את הנושא החשוב הזה אך לא לשים עליו דגש ברמת בחינה ושאלות. בכל זאת, עקב העבודה המרובה עם ממשקים, כדאי להדגיש את חשיבות הנושא ואף לדרוש לבצע אותו בחלק מהתרגילים. למלמדי ג'אווה קיימת מוטיבציה נוספת והיא הפקת מסמך תיעוד באמצעות מנגנון javadoc.

ט – איברי מחלקה

1. כתיבת הפעולה הסטטית לאחר פעולת ה-main(...) / Main(...) באותו הקובץ מתגלה לעתים כמסובכת עבור התלמידים. זאת משום שהתלמידים רואים בפעולת ה-main(...) / Main(...) את התוכנית כולה, וקשה להם להבין איך ניתן להגדיר דברים לאחר הפעולה הזו. יש להבהיר כי זו פעולה רגילה בעלת תפקיד ייחודי. מאותה הסיבה התלמידים עלולים להתקשות בהגדרת משתנים גלובליים מחוץ ל-main(...) / Main(...), ולחשוב שאם הם הגדירו את המשתנים בתוך ה-main(...)/Main(...) כל יתר הפעולות בקובץ צריכות להכיר אותם.
2. **ט.4.** עלינו לחזור לדוגמה הקודמת לפני קביעת הקיבולת כתכונה סטטית-קבועה, אחרת לא ניתן יהיה להגדיר פעולת Set / set על התכונה (תכונה קבועה אינה ניתנת לשינוי).
3. **ט.5.** בסעיף זה מוצגת פעולה סטטית שעניינה מילוי שני דליים בכמות מים שהיא הממוצע של תכולתם ההתחלתית. את המשימה ביצענו על ידי ריקון הדליים ומילויים מחדש. במציאות היינו שופכים את המים מהדלי שמכיל יותר מים לדלי שמכיל פחות מים עד להשוואת הכמויות. דוגמה זו ממחישה הבדל מהותי שקיים בין עצמים במציאות לעצמים בתכנות. בזמן בחינה יש להבהיר היטב לתלמידים מה נדרש מהם. האם מספיקה ה"אמת התכנותית" המוכחת על ידי מבחן התוצאה (אם השגנו את התוצאה, אזי התהליך אינו מעניין אותנו), או שנדרשת "אמת מציאותית" המגיעה אל התוצאה בעזרת התהליך המתבצע בעולם המעשה.
- ההערה הראשונה המובאת בסעיף זה תלווה את יחידת הלימוד כולה ויש לחזור ולשנן אותה באוזני התלמידים בכל אחד מהממשקים שיוצגו בפרקים הבאים.
4. מתי מגדירים פעולה כסטטית, ומתי לא? ניתן להבחין בין פעולות סטטיות לפעולות שאינן סטטיות באופן זה: אם הפעולה משתמשת בתכונות מחלקה שאינן סטטיות, היא אינה פעולה סטטית. לעומת זאת, אם הפעולה משתמשת רק בפרמטרים ובתכונות סטטיות, יש להגדירה כפעולה סטטית.

4. קשיים צפויים

1. "מחלקה" היא מושג מופשט שהתלמידים מתקשים לתפוס, ולכן המעבר משימוש בעצמים (פרק 2) לשלב בניית מחלקה, וממנו לשלב של יצירת עצמים מטיפוס מחלקה, אינו פשוט ודורש תשומת לב מיוחדת של המורה.
2. הפרק ארוך ומתרגל מספר רב של נושאים. ניתן ללמדו בחלקים, ובסוף כל חלק לתרגל אותו בעזרת התרגילים המתאימים.

5. דפי העבודה

פתרונות לכל דפי העבודה מופיעים באתר המרכז להוראת המדעים, האוניברסיטה העברית בירושלים ונגישים רק לציבור המורים:

http://sites.huji.ac.il/science/unit4_2007/

התרגילים מסודרים על פי מבנה זה:

בגרסת ג'אווה:

דפי עבודה 1-5, 7-8: תרגילים מעשיים

דפי עבודה 6, 9: תרגילים תיאורטיים

בגרסת ס'שרפ:

דפי עבודה 1-4, 6-7: תרגילים מעשיים

דפי עבודה 5, 8: תרגילים תיאורטיים

דף עבודה מספר 1 – נקודת התחלה

מומלץ.

דף עבודה מספר 2 / לא קיים בגרסת C# – תיעוד javadoc

מומלץ.

דף עבודה מספר 3 / דף 2 בגרסת C# – משחק בקוביות

מומלץ.

דף עבודה מספר 4 / דף 3 בגרסת C# – תאריך

מומלץ.

דף עבודה מספר 5 / דף 4 בגרסת C# – מספר רציונלי

מומלץ.

דף עבודה מספר 6 / דף 5 בגרסת C# – הרשאות גישה

רשות.

דף עבודה מספר 7 / דף 6 בגרסת C# – מונה

רשות.

דף עבודה מספר 8 / דף 7 בגרסת C# – פעולת מחלקה

רשות.

דף עבודה מספר 9 / דף 8 בגרסת C# – כיתה מוזיקלית

רשות.

דף עבודה מספר 1 – נקודת התחלה

תרגיל מעשי המתרגל הגדרת מחלקה ויצירת עצמים ממנה.
מומלץ להכין בכיתה בהנחיית המורה.

דגשים:

בחלק הראשון יש לכתוב מחלקה פשוטה מאוד.
בחלק השני יש לכתוב שתי פעולות, ששתייהן מקבלות עצם כפרמטר. הפעולה אף יוצרת עצם ומחזירה אותו כערך החזרה. בכיתות חלשות ניתן לדגל בשלב זה על חלק ב, ולחזור אליו מאוחר יותר. קודם לביצוע החלק השני של הדף יש לדבר עליו ולהסביר שעצמים נוצרים לא רק בפעולה הראשית, אלא, כפי שרואים זאת בפעולה המחזירה את אמצע הקטע, ניתן ליצור את הנקודה בתוך פעולה אחרת (נושא זה של יצירת עצם חדש כערך החזרה של פעולה ישמש אותנו רבות בהמשך כאשר נדון בנושא של הפניות ושמירה על כך שלא יהיו הפניות רבות מדי לעצמים כדי למנוע פגיעה בהם. במצבים כאלה נמליץ על פעולות `get()` / `Get()` שיחזירו עצם חדש בעל ערכי תכונות מסוימים ולא הפניה לעצם קיים).
שתי הפעולות הנוספות מתרגלות דברים שונים:
הפעולה מרחק: מתרגלת את השימוש בפעולות של המחלקה `Math`.
הפעולה אמצע קטע: מתרגלת בניית עצם בתוך הפעולה והחזרתו מתוכה.

דף עבודה מספר 2 / לא קיים בגרסת C# – תיעוד javadoc

תרגיל מעשי המהווה בסיס לכל נושא התיעוד בהמשך.
מומלץ להכין בכיתה בהנחיית המורה.

דגשים:

כדאי לדרוש מהתלמידים מרגע זה ואילך להקפיד לייצר דף HTML לכל תרגיל שהם עושים.

קשיים צפויים:

1. אם שיניתם את ברירות המחדל של ההתקנה, יהיה עליכם למצוא בעזרת ה-`Configure` את מקומה של פקודת `Javadoc.exe`, שתמצא בדרך כלל ב-
`Y:\eclipse\jre\bin\javadoc.exe`

Y הוא הכוון שעליו התקנתם את ג'אווה.

מקומה של הפקודה על פי ברירת המחדל יהיה ב:

`C:\Program Files\Java\jdk1.5.0_04\bin\javadoc.exe`

כפי שמופיע בראש המסך הזה.

2. לעובדים בסישרפ מומלץ לתרגל את אופן התיעוד הנכון והמלא של מחלקה על פי המתואר בפרק, אך הם לא יוכלו לקבל מהמחלקה המתועדת קובץ הנראה כמו קובץ MSDN הקיים עבור מחלקות הקיימות בשפה. בכל זאת מומלץ לא לוותר על התרגול ועל הבהרת חשיבות

הנושא, וכן אין לוותר על ההבהרה שהפקת מסמכי תיעוד היא מוגבלת ונובעת מיכולות סביבת העבודה שעמה הם עובדים.

דף עבודה מספר 3 / דף 2 בגרסת C# – משחק בקוביות

תרגיל מעשי.

מומלץ להכין בכיתה בהנחיית המורה.

דגשים:

1. בתרגיל זה התלמידים נדרשים להחליט לבד על התכונות של מחלקת קובייה. החשיבה העצמית יכולה לעזור בהבנת המושג: "המצב הפנימי של העצם".
2. אתחול התכונה נעשה בדרך בלתי שגרתית: לא מקבלים פרמטר בעת יצירת הקובייה, אלא מגרילים מספר אקראי ומאתחלים על פיו. נקודה זו יכולה לעזור בהפרדה תפיסתית בין הפרמטרים שהפעולה הבונה מקבלת לבין התכונות של המחלקה.
3. לעובדים בסישרפ: יש לשמור אובייקט מטיפוס Random בתור תכונה סטטית של המחלקה. מספיק שעצם אחד כזה ישרת את כל הקוביות שייווצרו מהמחלקה. ללא שימוש באובייקט שכזה, המספרים שייווצרו לא יהיו ממש אקראיים אלא יחזרו על עצמם.

דף עבודה מספר 4 / דף 3 בגרסת C# – תאריך

תרגיל מעשי. מומלץ מאוד.

מומלץ לתת כשיעורי בית.

דגשים:

חלק א – כתיבת מחלקה פשוטה.

פעולת `compareTo(...)` / `CompareTo(...)` – פעולה המקבלת עצם כפרמטר.

פעולת `toString(...)` / `ToString(...)` – כדאי לחזור על משמעות הפעולה: היא אינה מדפיסה את העצם אלא מחזירה מחרוזת המייצגת אותו.

חלק ב – תרגול הפניות.

תרגיל "נוסע ודרכון" בפרק 6 (הפניות ועצמים מורכבים) הוא תרגיל מסכם וחשוב, ומומלץ מאוד לבצעו. כיוון שהמחלקות נוסע ודרכון משתמשות במחלקה `Date`, חשוב שהתלמידים יכתבו את המחלקה וישמרו אותה כך שיוכלו להשתמש בה בהמשך.

דף עבודה מספר 5 / דף 4 בגרסת C# – מספר רציונלי

תרגיל מעשי.

ניתן לתת אותו כשיעורי בית, אך לפני כן יש לקרוא אותו יחד עם התלמידים בכיתה.

דגשים :

1. בתרגיל זה חשוב להבהיר לתלמידים את גישתנו למקרי הקצה.
2. כאשר כתובה הנחה בתוך הממשק אין צורך לבדוק את המקרה הזה בתוך הפעולה. האחריות לעבוד לפי ההנחות היא של המשתמש בפעולות. במקרה של אי-ציות להנחה, אין אחריות על תוצאות הפעולה. יש להקפיד שהתלמיד ירשום את ההנחות בתוך תיעוד הפעולה הבונה.
3. בפעולת החלוקה מתבצעת בדיקה שהמחלק אינו שווה אפס. אם בכל זאת מנסים לחלק באפס, הפעולה מחזירה null. יש לבדוק שהתלמיד זכר לבדוק בפעולת החלוקה שהמונה שונה מ-0.

דף עבודה מספר 6 / דף 5 בגרסת C# – הרשאות גישה

תרגיל תיאורטי.

תרגיל חשוב. מומלץ להכין בכיתה בהנחיית המורה ולפתח סביבו דיון.

תשובות לתרגיל:

1. כן, המחלקה Alice תעבור הידור.
2. א. חוקי. התכונה a1 של Alice היא פומבית.
ב. חוקי. הפעולה היא פעולה פומבית של Alice.
- ג. לא חוקי. לא ניתן לפנות לתכונה פרטית a2 של Alice.
- ד. לא חוקי. לא ניתן לפנות לפעולה פרטית findA2(...) / FindA2(...) של Alice.

דגשים :

1. בתרגיל זה התלמידים פוגשים בפעם הראשונה עצם מורכב, כלומר עצם שהתכונה שלו היא מטיפוס המחלקה.
2. ניתן לשאול את התלמידים: מהן התכונות של המחלקה Bob, ולוודא שהם מבינים שהתכונה היחידה היא עצם מטיפוס Ally, ולא התכונות של Ally.
3. התרגיל הזה ממחיש את העובדה שהתכונות יכולות להיות פומביות והפעולות פרטיות.
4. הפעולה findA2(...) / FindA2(...) במחלקה Alice יכולה להיות פעולה סטטית, כיוון שאינה משתמשת בתכונות העצם.

דף עבודה מספר 7 / דף 6 בגרסת C# – מונה

תרגיל מעשי.

מומלץ להכין בכיתה בהנחיית המורה ולפתח דיון סביב התרגיל.

דגשים:

תרגיל זה מדגיש את ההבדל שבין תכונה רגילה, השייכת למופע מסוים, ובין תכונה סטטית, השייכת למחלקה.

דף עבודה מספר 8 / דף 7 בגרסת C# – פעולת מחלקה

תרגיל מעשי.

מומלץ להכין בכיתה בהנחיית המורה, ולדון יחד עם התלמידים באלגוריתם.

דגשים:

בדומה להעברת כמות המים הממוצעת הנדונה בפרק, גם כאן, העברת כמות המים הרצויה תיעשה בצורה שאינה טבעית לעולם האמיתי. אין בידינו פעולה המאפשרת להוציא כמות מים רצויה מדלי אחד ולהוסיף אותה לדלי אחר. ולכן האלגוריתם יתבצע באופן הזה:

שפוך מדלי ראשון כמות מים x לדלי שני:

רוקן את הדלי הראשון ושמור את כמות המים שהייתה בו ב- a .

רוקן את הדלי השני ושמור את כמות המים שהייתה בו ב- b .

בצע $a - x$ והחזר את התוצאה לדלי הראשון.

בצע $b + x$ והחזר את התוצאה לדלי השני.

מקרי קצה שיש לבדוק:

כדאי לפתח דיון עם התלמידים מה לעשות במקרי הקצה:

אם $a - x$ מספר שלילי, פירוש הדבר שלא היו מספיק מים בדלי הראשון.

יש לבדוק שהכמות $b + x$ אינה מעבר לקיבולת של הדלי שני.

דף עבודה מספר 9 / דף 8 בגרסת C# – כיתה מוזיקלית

תרגיל מעשי.

ניתן לתת את דף העבודה כשיעורי בית.

תשובות לדף העבודה:

1. ההדפסות שיתקבלו:

5

12

2. המשתנה הוגדר סטטי כדי שכל המופעים יוכלו לפנות אליו ולעדכנו. רק כך תוכל להצטבר בו הכמות הכללית של הדיסקים שייאספו בכיתה.