מה ה- this הזה לכם?

ד"ר נוע רגוניס
המכללה האקדמית בית ברל, הטכניון

על בסיס מחקר שנעשה עם ד"ר רונית שמלו

סירקו את הקוד

הפעילו QR Code Reader

אפשר להוריד אפליקציה למי שאין...

למה חשוב לדבר על this ?

- אפשר בלעדיו, אבל נהוג להשתמש בו
- שימוש טכני ללא הבנה – אף פעם לא מומלץ
- הפגנת הבנה מעידה בדרך כלל על הבנה של מהות העצמים
- התייחסות שגויה מעידה בדרך כלל על טעות בתפיסה שתעלה גם במקומות אחרים
- מפגיש הבנה של מושגים (עצם / טיפוס) עם אופני יישום שונים בשפת התכנות – בשני ההיבטים יש הפשטה
- הגישה שלי: להשתמש היכן שנחוץ
- שימוש גורף לעיתים מאד מכביד על הקוד

דגשים ממחקרים קודמים

- קשיים בהבנה של יסודות תכנות מונחה עצמים קיימים אצל לומדים, תלמידים וסטודנטים, ויש מחקרים רבים המזהים את הקשיים וחלקם מציעים דרכים להתגבר עליהם.
- קשיים בהבנה של this אינם רק שלכם עם תלמידיכם, אלא של עמיתים ברחבי העולם (דוגמאות של מחקרים בסוף המצגת שתופץ).
- הבנה של "this" נוגעת בהבנה של מושגים רבים בהם יש קשיים.

- אציג בקצרה דגשים מעבודת הדוקטורט שלי, 2005; ומעבודת הדוקטורט של רונית, 2012;

תפיסות הנוגעות להבנת העצם הנוכחי

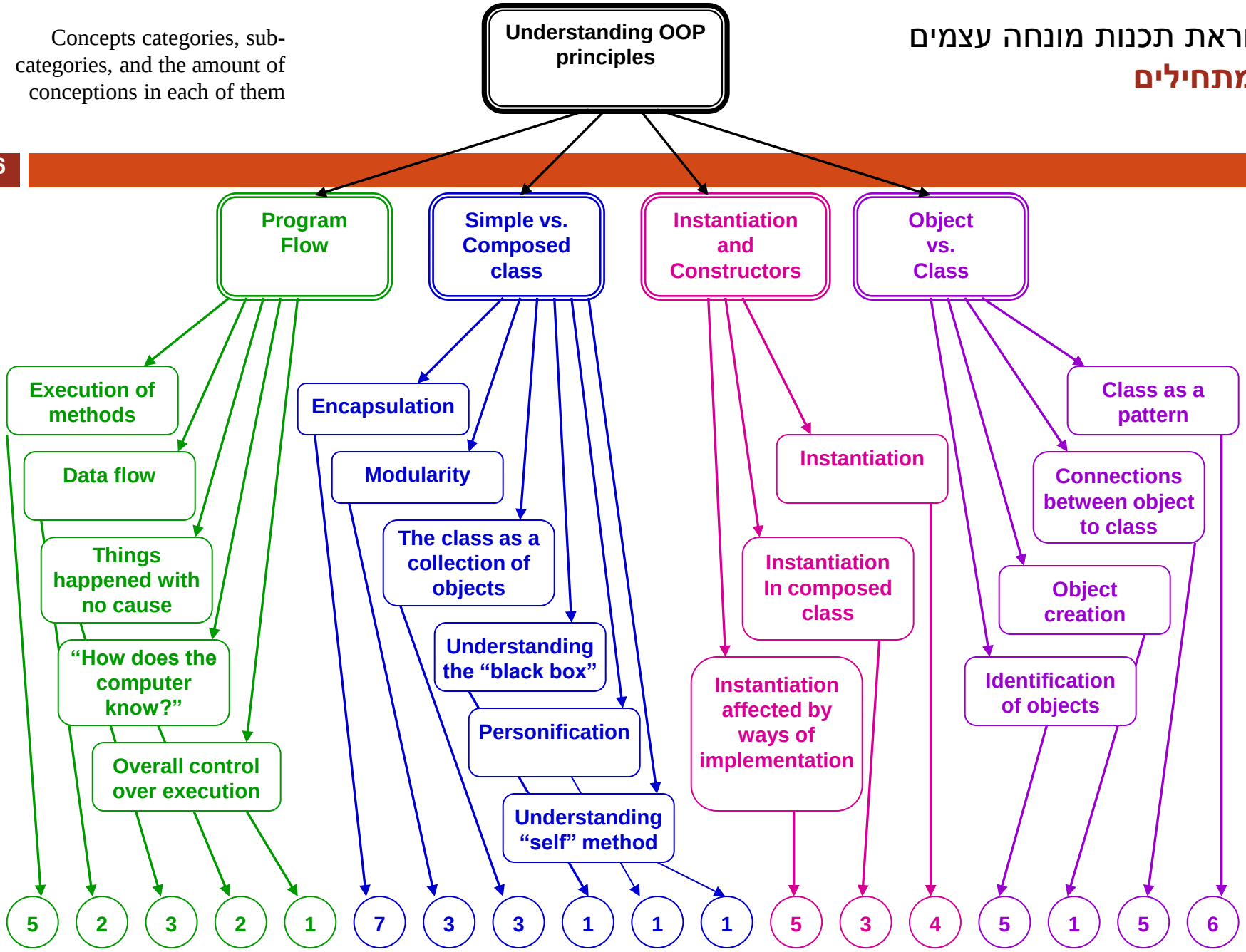
- **נוע רגוניס, 2005**
- **הוראת תכנות מונחה עצמים למתחילים**
  - התייחסות לעצמים כאל "משתנים רגילים".
  - בלבול בין עצם לטיפוס (מחלקה).
  - הבנה של הפעולה הבונה.

    למשל: פעולה בונה רק מאתחלת ערכי משתנים

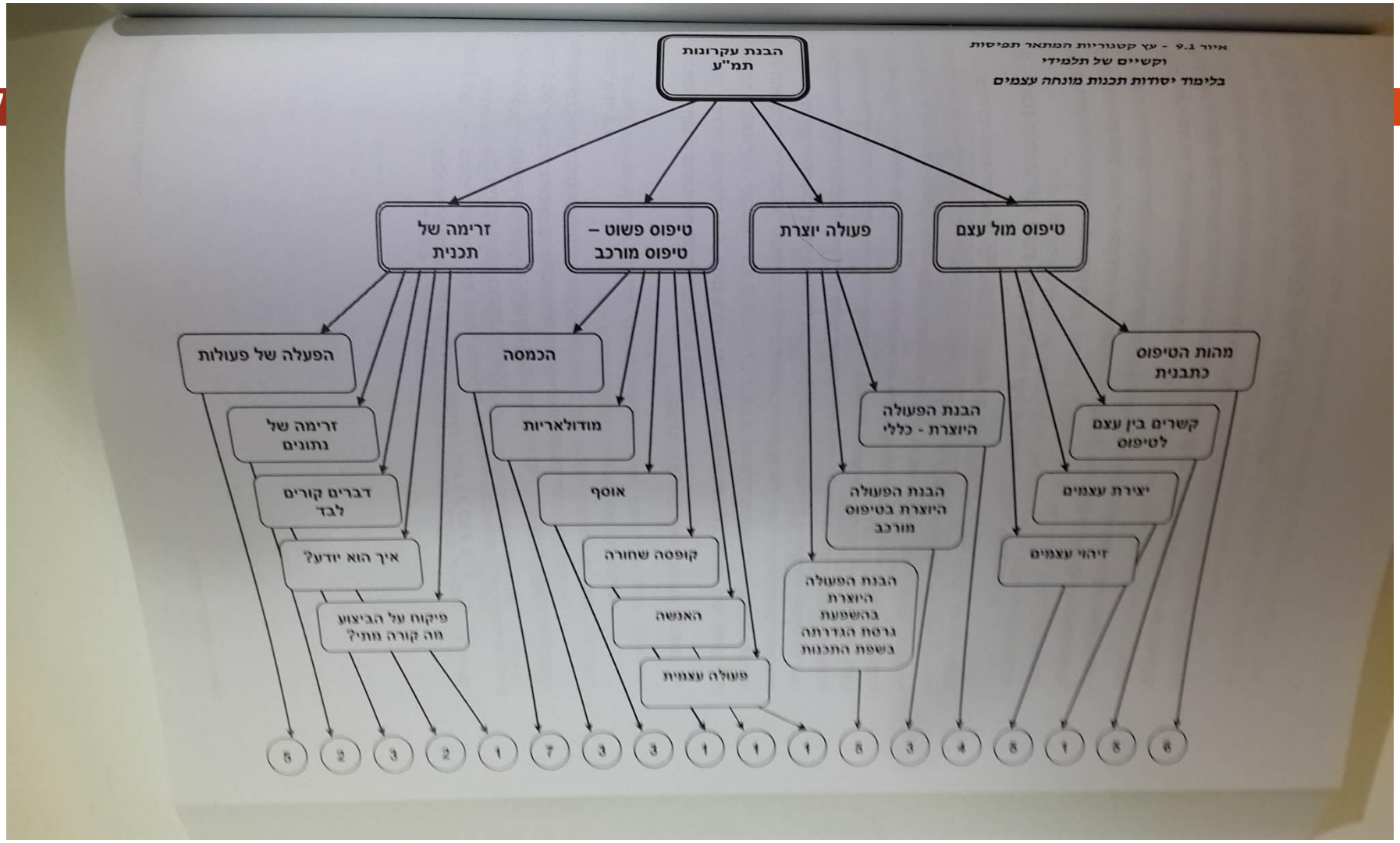
    או לחלופין: "זה רק" ביצוע של גוף הפעולה
  - התייחסות לעצם רק כאוסף של שדות.
  - בלבול בין "העצם עצמו" – למשתנה/מזהה המתייחס אליו.
  - התייחסות לתכונות כאל משתנים בגוף המחלקה (ולא כמאפיינים של עצמים מהטיפוס).
  - איך this יכול להתייחס בכל פעם למשהו אחר.
  - איתחול תכונות בהגדרתן, משחרר מבניית עצמים.

Concepts categories, sub-categories, and the amount of conceptions in each of them

הוראת תכנות מונחה עצמים
למתחילים



איור 9.1 - עץ קטגוריות המתאר תפיסות וקשיים של תלמידי בלימוד יסודות תכנות מונחה עצמים



תפיסות הקשורות להבנת העצם הנוכחי

- **רונית שמלו, 2012:**
- **אבני בניין של תכנות מונחה עצמים באמצעות לימוד מטעויות**
  - אי הבחנה בין עצם למשתנה
  - "פירוק" מרכיבי העצם כשיש להתייחס אליו כמכלול
  - אין צורך לציין מזהה של עצם בעת זימון פעולה
  - אם תכונות העצם מאותחלות, אזי הוא נוצר
  - השמה בין עצמים מייצרת עצם חדש
  - השוואה בין מזהי עצמים היא השוואה של ערכי התכונות שלהם
  - הצהרה על עצם מייצרת אותו
  - בניית עצם היא רק ביצוע גוף הפעולה הבונה
- וחלק שלם המתייחס פדגוגית ללמידה מטעויות

מה אנחנו בדקנו?

- מחקר שהועבר בקרב סטודנטים, תלמידי תיכון ומורים.
- כאן נתאר חלק מן הממצאים של מחקר התלמידים, שנערך לקראת בחינת הבגרות בשנה שעברה:
  - N=86, מ- 7 בתי ספר
  - 31 – תלמידי יוד; 45 – תלמידי יא; 10 – תלמידי יב
- + מעט התייחסויות שלכם אל תשובות התלמידים.

שאלות השאלון התייחסו אל:

- **שאלה 1**: על בסיס פרויקט הכולל טיפוס פשוט, טיפוס מורכב ומחלקה ראשית: לסמן היכן נחוץ this והיכן הוא מיותר; התייחסות ל- this בטיפוס הפשוט ובטיפוס המורכב; המרת פעולה פנימית עם this לפעולה חיצונית; התייחסות ל- this ב- main.

- **שאלה 2**: this כפרמטר (לא נלמד בתיכון, אבל יש הקשר להבנה...)

- **שאלה 3**: this בפרויקט עם הורשה, מופיע כשם של פעולה.

- **שאלה 4**: "טעם אישי" בשימוש ב- this – תוצג בהרחבה

- **שאלה 5**: שאלה פתוחה, הבנה: מתי חובה להשתמש?; מתי היית ממליץ להשתמש? מתי אסור להשתמש?; "מי הוא this?"

שאלה 4 (א): כל העצמים הנבנים זהים

גרסה 1

```
public class Point
{
    private int x = 0;
    private int y = 0;

    public Point()
    {
    }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        this.x = 0;
        this.y = 0;
    }
}
```

גרסה 3

```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        x = 0;
        y = 0;
    }
}
```

- דרגו את הגרסאות על פי ההעדפה שלכם, החל מהגרסה המועדפת.
- רשמו את הקריטריונים שהשפיעו על הבחירה.

שאלה 4 (ב): אתחול תכונות בהתאם לפרמטרים

גרסה 1	גרסה 2

public class **Point**
{
    private int x;
    private int y;
 	
    public Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }
}
	public class **Point**
	{
	    private int x;
	    private int y;
	
	    public Point(int x, int y)
	    {
	        this.x = x;
	        this.y = y;
	    }
	}

- דרגו את הגרסאות על פי ההעדפה שלכם, החל מהגרסה המועדפת.
- רשמו את הקריטריונים שהשפיעו על הבחירה.

שאלה 4 (ג): פעולה בונה מעתיקה

גרסה 3

```
public class Point
{
   private int x;
   private int y;

   public Point(Point p)
   {
      x = p.x;
      y = p.y;
   }
}
```

גרסה 2

```
public class Point
{
   private int x;
   private int y;

   public Point(Point p)
   {
      this.x = p.x;
      this.y = p.y;
   }
}
```

גרסה 1

```
public class Point
{
   private int x;
   private int y;

   public Point(Point p)
   {
      this.x = p.getX();
      this.y = p.getY();
   }
   public int getX()
   {  return x;   }

   public int getY()
   {  return y;   }
}
```

- דרגו את הגרסאות על פי ההעדפה שלכם, החל מהגרסה המועדפת.
- רשמו את הקריטריונים שהשפיעו על הבחירה.

שאלה 4 (ד): משחקים עם equals ו- toString


```
public class Point
{
    private int x;
    private int y;

    public String toString()
    {
       return "<" + this.x + "," + this.y + ">";
    }
}
```

גרסה 1

```
public boolean equals(Point p)
{
  return this.x == p.getX()&& this.y == p.getY();
}
```

גרסה 2

```
public boolean equals(Point p)
{
   return this.toString().equals(p.toString());
}
```

גרסה 3

```
public boolean equals(Point p)
{
   return toString().equals(p.toString());
}
```

- דרגו את הגרסאות על פי ההעדפה שלכם, החל מהגרסה המועדפת.
- רשמו את הקריטריונים שהשפיעו על הבחירה.

שאלה 4 (א): כל העצמים הנבנים זהים

גרסה 1

```
public class Point
{
    private int x = 0;
    private int y = 0;

    public Point()
    {
    }
}
```

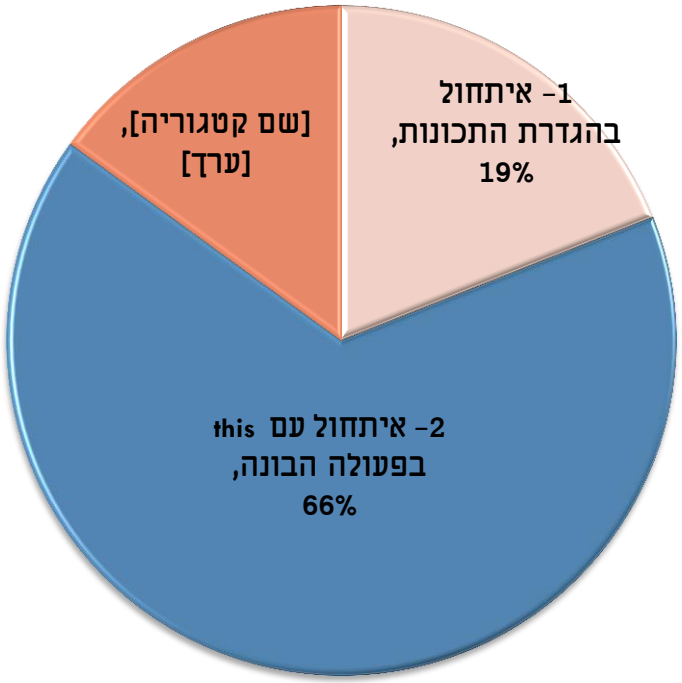
גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        this.x = 0;
        this.y = 0;
    }
}
```

גרסה 3

```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        x = 0;
        y = 0;
    }
}
```


% התלמידים שבחרו בגירסה בעדיפות ראשונה



עדיפות 1	% בוחרים	עדיפות 2	% בוחרים
גרסה 1	19%	גרסה 1	6%
		גרסה 2	5%
		גרסה 3	8%
גרסה 2	66%	גרסה 1	16%
		גרסה 2	1%
		גרסה 3	49%
גרסה 3	15%	גרסה 1	1%
		גרסה 2	12%
		גרסה 3	2%

שאלה 4 (א) - נימוקים

עדיפות 1	% בוחרים
גרסה 1	19%
גרסה 2	66%
גרסה 3	15%

גרסה 1
```
public class Point
{
    private int x = 0;
    private int y = 0;

    public Point()
    {
    }
}
```

גרסה 2
```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        this.x = 0;
        this.y = 0;
    }
}
```

גרסה 3
```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        x = 0;
        y = 0;
    }
}
```


- הנימוק השכיח של הרוב **שבחר תחילה בגרסה 2** ואז בגרסה 3:
  - גרסה 2 היא הכי נכונה / מקובלת / מוסכמות / מה שלמדנו / יותר קריא / יותר מובן / ככה יבגני אמר ☺
  - ברור יותר שהמשתנים (המונח שגוי) מוגדרים ל-0
  - כי אם היו משתנים בבנאי שהם באותו שם כמו המשתנים (המונח שגוי) שבתוך המחלקה אז ה this מבחין ביניהם
  - מציגה ומסבירה את פעולת הבנאי הכי טוב
  - אם נרצה לערוך שינויים בפעולה הבונה, יהיה יותר קל (לא ניגע בצד שמאל)
  - גרסה 3 נכונה אבל פחות / לא משתמשת ב this / נכונה אבל עדיף עם this

- הנימוק השכיח של אלו **שבחרו תחילה בגרסה 3** ואז בגרסה 2:

  אין צורך בשימוש ב this

  ויש גם מי שחשב שגרסה3 שגויה כי אין בה this

שאלה 4 (א) -
נימוקים

עדיפות 1	% בוחרים
גרסה 1	19%
גרסה 2	66%
גרסה 3	15%

גרסה 1
```
public class Point
{
    private int x = 0;
    private int y = 0;

    public Point()
    {
    }
}
```

גרסה 2
```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        this.x = 0;
        this.y = 0;
    }
}
```

גרסה 3
```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        x = 0;
        y = 0;
    }
}
```


- **נימוקים בעד גרסה 1:**
  - הכי טובה כי זה בעת ההגדרה
  - אם ידועים הערכים ההתחלתיים של העצם אין צורך להגדיר אותם בבנאי
  - השימוש ב this מיותר
  - ככל שיותר קצר יותר טוב / קצרה וקולעת
  - יותר יעיל מאשר לזמן פעולה

שאלה 4 (א) -
נימוקים

עדיפות 1	% בוחרים
גרסה 1	19%
גרסה 2	66%
גרסה 3	15%

גרסה 1
```
public class Point
{
    private int x = 0;
    private int y = 0;

    public Point()
    {
    }
}
```

גרסה 2
```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        this.x = 0;
        this.y = 0;
    }
}
```

גרסה 3
```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        x = 0;
        y = 0;
    }
}
```


- **נימוקים נגד גרסה 1:**
  - לא כותבים ככה / לא מקובל לכתוב ככה / לא למדנו שאפשר לכתוב ככה
  - האתחול צריך להיות בבנאי / הבנאי מאבד את המשמעות שלו
  - לא יפה בעיני / לא הגיוני / חסרת פואנטה
  - נותן לי תחושה שה- X וה- Y קבועים (נוע: תחושה נכונה דווקא)
  - לא טוב שהפעולה הבונה לא מזמנת את המשתנים (מונח שגוי)

- לא מאתחלת את המשתנים (מונח שגוי) בבנאי (נוע: הסבר לא קריטריון)
- שמים ערך במשתנה (מונח שגוי) לפני הגדרת עצם מהמחלקה
- הפעולה שנכתבה שם לא מבצעת דבר / לא יעבור קומפילציה
- עלול ליצור בעיות בהמשך
- זו פעולה בונה ברירת מחדל – ומוחזר null שלא מאפשר לנו יותר מידי

שאלה 4 (ב): אתחול תכונות בהתאם לפרמטרים

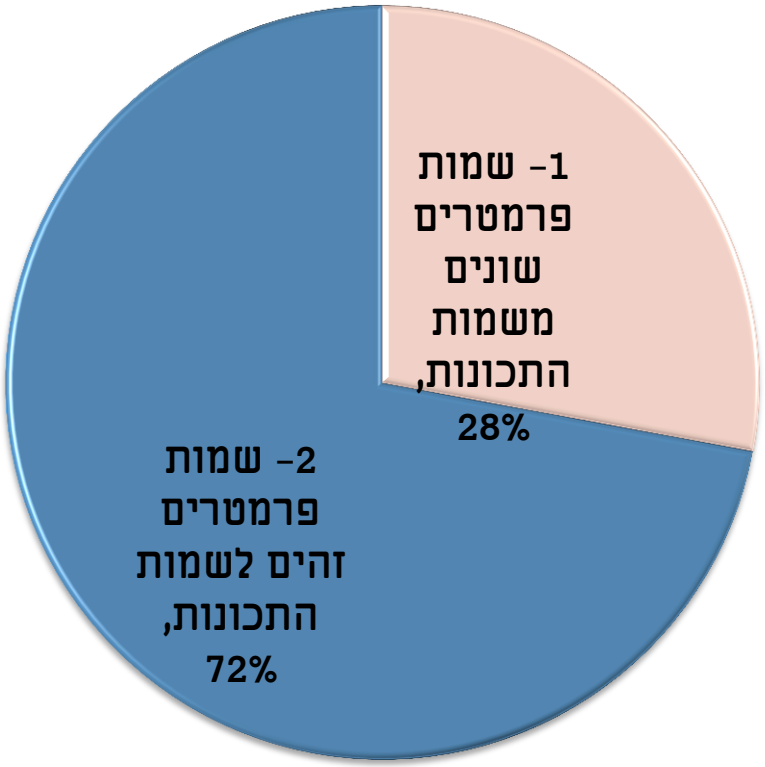
גרסה 1

```
public class Point
{
    private int x;
    private int y;

    public Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

% התלמידים שבחרו בגירסה בעדיפות ראשונה



עדיפות 1	% בוחרים
גרסה 1	28%
גרסה 2	72%

שאלה 4 (ב): נימוקים

עדיפות 1	% בוחרים
גרסה 1	28%
גרסה 2	72%

גרסה 1

```
public class Point
{
    private int x;
    private int y;

    public Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```


- הנימוקים של הרוב **שבחר בגרסה 2**:
  - יותר ברורה / יותר קריא / כי יבגני...
  - אני מעדיף ששמות המשתנים המתקבלים בבנאי יהיו זהים לשמות המשתנים של המחלקה
  - עדיף ליצור הבחנה בין משתנים מאשר לתת להם שמות שונים
  - כי יש this לכל מקרה... / ה this מעניק לי יותר הבנה / ניתנת הפרדה והבנה לאיזה כל אחד מהם מתייחס
  - כי בגרסה 1 – אין this / פחות מובנת
  - יותר יעיל

שאלה 4 (ב): נימוקים

עדיפות 1	% בוחרים
גרסה 1	28%
גרסה 2	72%

גרסה 1

```
public class Point
{
    private int x;
    private int y;

    public Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```


- הנימוקים של אלו **שבחרו בגרסה 1**:
  - קצר יותר / יותר אחלה / העדפה אישית / יותר אסטתי
  - לא נחוץ this
  - בגלל שהשמות שונים קל יותר להבין את התכנית
  - עדיף בבנאי להשתמש במשתנים ששמם דומה אך לא זהה למשתני המחלקה במקום להשתמש ב this
  - כי יותר נכון לכתוב ככה / הכי טוב ועובד ככה, וככה למדנו
  - כדי לא ליצור בלבול
  - אחרת יכול להיות בלבול בנתונים ואיבוד
  - יותר יעילה כי לא משתמשת ב this

שאלה 4 (ג):
פעולה בונה מעתיקה

גרסה 1

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
       this.x = p.getX();
       this.y = p.getY();
    }
    public int getX()
    {  return x;   }

    public int getY()
    {  return y;   }
}
```

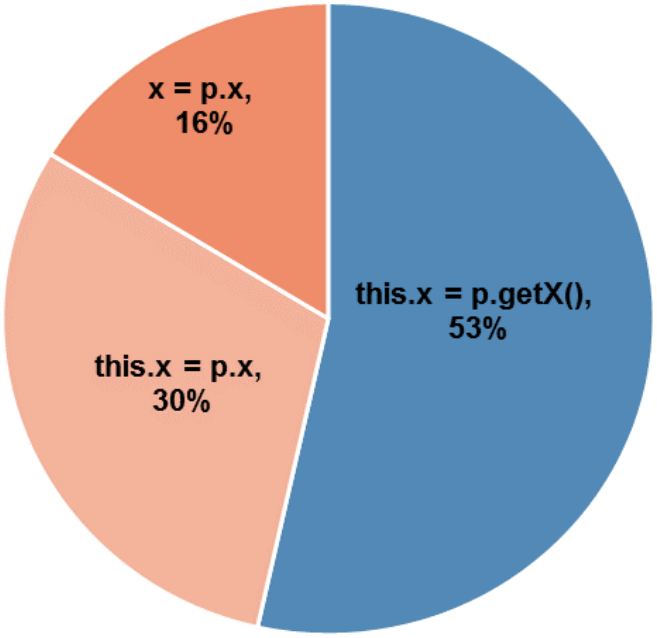
גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
       this.x = p.x;
       this.y = p.y;
    }
}
```

גרסה 3

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
       x = p.x;
       y = p.y;
    }
}
```

% התלמידים שבחרו בגירסה בעדיפות ראשונה



עדיפות 1	% בוחרים	עדיפות 2	% בוחרים
גרסה 1	53%	גרסה 1	7%
		גרסה 2	42%
		גרסה 3	5%
גרסה 2	30%	גרסה 1	15%
		גרסה 2	1%
		גרסה 3	14%
גרסה 3	16%	גרסה 1	2%
		גרסה 2	14%
		גרסה 3	0%

שאלה 4 (ג):
פעולה בונה מעתיקה

עדיפות 1	% בוחרים
גרסה 1	53%
גרסה 2	30%
גרסה 3	16%

גרסה 1
```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        this.x = p.getX();
        this.y = p.getY();
    }
    public int getX()
    {  return x;   }

    public int getY()
    {  return y;   }
}
```

גרסה 2
```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        this.x = p.x;
        this.y = p.y;
    }
}
```

גרסה 3
```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        x = p.x;
        y = p.y;
    }
}
```


- הנימוקים של הרוב **שבחר בגרסה 1**:
  - כי יש get ויש this
  - יותר מקצועי / הקריאה למתכנת יותר ברורה
  - השימוש ב this מדגיש שמדובר בעצם מהמחלקה הנוכחית
  - אני מעדיף שתהיינה פעולות מאחזרות ואני מעדיף שיהיה this/ ככה מרגיש לי / הכי מסודר / ככה למדנו
  - **נגד**: סתם ארוכה / לא יעיל ובזבזני
  - זה הקוד היחיד שהיה עובד / צריך...
- הנימוקים של אלו **שבחרו בגרסה 2**:
  - כאשר משתמשים ב this זה יותר קריא / מובן / ברור
  - מיותר להשתמש ב get, זה מתבצע בתוך המחלקה
  - לא יעיל להשתמש ב get, לשם מה לזמן פעולה?
  - חלק לא נכונים

שאלה 4 (ג):
פעולה בונה מעתיקה

עדיפות 1	% בוחרים
גרסה 1	53%
גרסה 2	30%
גרסה 3	16%

גרסה 1

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        this.x = p.getX();
        this.y = p.getY();
    }
    public int getX()
    {  return x;   }

    public int getY()
    {  return y;   }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        this.x = p.x;
        this.y = p.y;
    }
}
```

גרסה 3

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        x = p.x;
        y = p.y;
    }
}
```


- הנימוקים של אלו **שבחרו בגרסה 3**:
  - הכי קצר ונוח לכתיבה
  - לא נחוץ get ולא נחוץ this / שימוש מיותר /
  - אין צורך בפעולות נוספות / הכי יעיל /
  - **נגד**: הפנייה לתכונות או לפרמטרים לא ברורה כאן

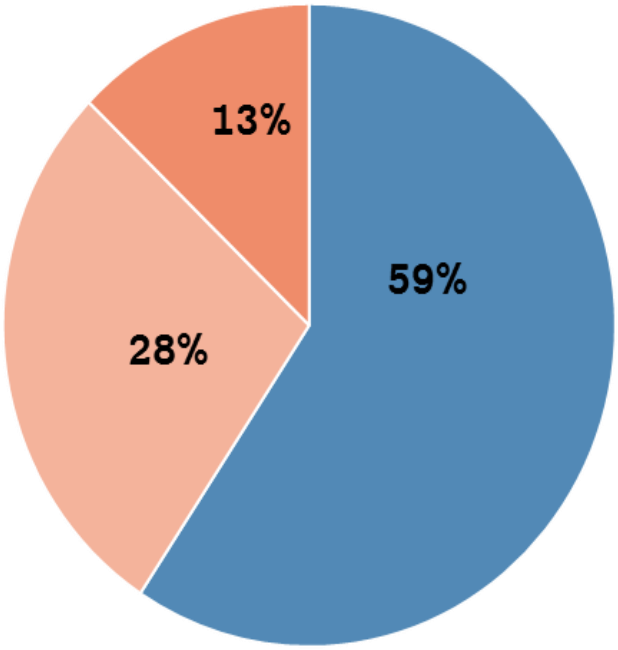
```
public String toString()
{ return "<" + this.x + "," + this.y + ">";}
```

equals:

```
return this.x == p.getX() && this.y == p.getY();
return this.toString().equals(p.toString());
return toString().equals(p.toString());
```


שאלה 4 (ד):
משחקי toString I equals

% התלמידים שבחרו בגירסה בעדיפות ראשונה



עדיפות 1	% בוחרים	עדיפות 2	% בוחרים
גרסה 1	59%	גרסה 1	5%
		גרסה 2	44%
		גרסה 3	10%
גרסה 2	28%	גרסה 1	13%
		גרסה 2	2%
		גרסה 3	13%
גרסה 3	13%	גרסה 1	0%
		גרסה 2	13%
		גרסה 3	0%

שאלה 4 (ד): משחקי equals וtoString I

```
public String toString()
{ return "<" + this.x + "," + this.y + ">";}
equals:
return this.x == p.getX() && this.y == p.getY();
return this.toString().equals(p.toString());
return toString().equals(p.toString());
```

עדיפות 1	% בוחרים
גרסה 1	59%
גרסה 2	28%
גרסה 3	13%

- הנימוקים של הרוב **שבחר בגרסה 1 או 2**:
  - רוב הנימוקים דומים לסעיף ג, כלומר:
    - מעדיפים שימוש ב get ומעדיפים שימוש ב this
      (ג4- 53%, ד4- 59%)
    - או חושבים ש get מיותר, ומעדיפים שימוש ב this כדי להבחין בברור בין התכונות
      (ג4- 30%, ד4- 28%)
- התייחסויות נוספות:
  - עדיף לבדוק את המשתנים עצמם מאשר לקרוא לפעולה
  - יש יותר הפרדה בין משתנה המחלקה למשתנה שקיבלנו דרך הפעולה
  - בגרסאות האחרות [2,3] משווים את הסטרינגים ולא את העצמים עצמם / לא הבנתי את סיבת השימוש ב toString / זה לא נהוג / זה לא עובד...
  - גרסה 2 לא מובנת לי, 3 מובנת אבל פחות טובה מ-1.

שאלה 4 (ד): משחקי equals ו toString

```
public String toString()
{ return "<" + this.x + "," + this.y + ">";}
```

equals:

```
return this.x == p.getX() && this.y == p.getY();
return this.toString().equals(p.toString());
return toString().equals(p.toString());
```

עדיפות 1	% בוחרים
גרסה 1	59%
גרסה 2	28%
גרסה 3	13%

- הנימוקים של אלו **שבחרו בגרסה 3**:
  - לטעמי [נוע] "טריקית" והיה מעניין לראות אם היא בכלל מובנת
  - יותר קצר יותר טוב / הכי פשוטה / המהדר יודע...
  - לא צריך סתם this ולא צריך סתם get / גרסה 1 ארוכה ומסורבלת...

- הנימוקים **נגד** גרסה 3:
  - **נגד**: אין סיבה סתם להפעיל equals בין מחרוזות
  - **נגד**: אי השימוש ב this עלול לגרום בלבול
  - **נגד**: אין התייחסות לפרמטרים של המחלקה שצריכים להיות מוחזרים
  - **נגד**: שגוי
  - לטעמי [נוע] דווקא כאשר מדובר בעצמים עם הרבה תכונות, זה נוח יותר
  - כמו שאני מבין בגרסאות 2 ו- 3 תתבצע רקורסיה
  - הכי טובה מבחינת יעילות (? שווה דיון)

מה אני חושבת (?)

- לא בטוח שזה מה שחשוב...
- אני בעד שימוש ב this במקומות ההכרחיים, אחרת לטעמי הוא מעיק (בפעולה בונה, בפעולות קובעות).
- בכל מקרה חשוב שתלמידים לא יעבדו רק מתוך הרגל... אלא יבינו. ממליצה להציג חלופות נכונות ולדון בהן (לא בהתחלה... בהתחלה שימוש אחיד ועיקבי בגירסה אחת הוא זה שנחוץ).
- הוכנסו שיקולי יעילות שאינם רלוונטיים ברוב המקומות, אבל בחלקם כן – שווה לנהל דיון בכיתה.
- אם מבינים: אין סיבה לקביעה על חלק מן הגרסאות שאינן נכונות (מעבר לכך שנכתב במפורש שהכל תקין).
- כפי שנראה בהמשך, למרות שעיקר נימוקי התלמידים בשאלה זו הם נכונים או מובנים, כפי הנראה זה נובע בעיקר "מהרגלים" ו הסברים בכיתה (שכמובן חשובים!) אבל לא תמיד **מהבנה משמעותית של this כמתייחס לעצם הנוכחי עליו מופעלת הפעולה בה הוא מופיע** – כפי שנראה בניתוח השאלה הבאה.

שאלה 5: שאלה פתוחה

- **א.** מתי חובה להשתמש ב- this?
- **ב.** מתי היית ממליץ להשתמש ב- this?
- **ג.** מתי אסור להשתמש ב- this?
- **ד.** "מי הוא this?"

שאלה 5 ד: מי הוא this – קטגוריות של תשובות

מס'	קטגוריה	% משיבים (N=86)
1	העצם הנוכחי (לא הכל מדויק)	38%
2	הפנייה אל תכונה (לא מהות העצם, אלא התייחסות אופרטיבית)	21%
3	הפנייה אל פעולה (לא מהות העצם, אלא התייחסות אופרטיבית)	7%
4	פעולה	6%
5	מחלקה	5%
6	תכונה	2%
7	מצביע	2%
8	לא יודע (או הזוי)	19%

שאלה 5 ד: מי הוא this – העצם הנוכחי (38%)

- דוגמאות לתשובות מדוייקות
  - האובייקט הנוכחי
  - העצם עליו מופעלת הפעולה
  - מציין שתפקידו לציין כי מדובר בעצם הנוכחי
  - העצם מסוג המחלקה בה נכתבה המילה this
  - This הוא "עצמי"
- דוגמאות לתשובות לא מדויקות (נכונות?)
  - העצם של המחלקה
  - האובייקט שקורא לפעולה/לתכונה ששייכת לו
  - הכתובת של האובייקט בו מתבצעת הפעולה במחלקה
  - מצביע על העצם הראשי (נכונה?)
  - מצביע שמראה לנו על מי מדברים
  - מבדיל בין מאפיינים בעלי אותו שם
  - אופרטור שמתייחס לעצם הנוכחי או למשתנים בו שדרכו המחלקה מופעלת

שאלה 5 ד: מי הוא this – הפנייה אל תכונה (21%)

- דוגמאות לתשובות שגויות, אך עם התיחסות אופרטיבית נכונה
  - מפנה למשתני המחלקה בתוך המחלקה עצמה (מה זה משתני מחלקה?)
  - הפנייה למשתנה מקומי
  - הוא מגדיר את החלק הנוכחי מהעצם בו אנו רוצים להשתמש
  - הוא המקשר בין העצם לתכונות
  - כלי שעוזר להתמצא באובייקטים, הוא מפנה לתכונות וגורם לקוד להיות מובן
  - עזר המבדיל בין פרמטר לבין תכונה או פעולה של מחלקה מסויימת
- דוגמאות לתשובות שגויות
  - הפנייה לפרמטרים
  - התכונה של העצם עליו מתבצעת הפעולה

שאלה 5 ד: מי הוא this – הפנייה אל פעולה (7%)

- דוגמאות לתשובות שגויות

- this הוא אמצעי העוזר לתת גישה לפעולות ולתכונות המחלקה במחלקה
- כאשר נכתוב משתנה של מחלקה נקודה THIS אנו מבינים שאנו מדברים על THIS, כאשר מדובר על המחלקה ועל התוכן הפנימי שלה
- פנייה לפעולה של המחלקה
- כשפעולה חיצונית מבצעת קריאה לפעולה במחלקה של אובייקט ורוצים לפנות ישירות לאובייקט אז שמו יהיה דיס

שאלה 5 ד: מי הוא this

- התייחסות כפעולה (6%)
  - הוא פעולה שקוראת למשתנים שיש במחלקה
  - סוג של פעולה
  - פעולה הקוראת לתכונה
- התייחסות כמחלקה (5%)
  - המחלקה בה עושים את הפעולה
- התייחסות כאל תכונות/משתנים (2%)
  - ערך המשתנים שהתקבלו
- התייחסות כמצביע (2%)
  - מצביע על פרמטר במחלקה הנוכחית
  - מצביע על העצם הראשי

שאלה 5 ד: מי הוא this - לא יודע והזוי (19%) (סליחה...)

- לא יודע (9%)
  - לא יודע / אינני יודע להגדיר / ...
- זה / אתה / מילת יחס (6%)
  - this הוא כולם, this הוא אני, this הוא אתה, this הוא אנחנו...
- אחרים (3%)
  - העצם או המשתנה אותו הגדרת להיות this
  - האזור שבו מגדירים משתנה, למשל ב program או הקלאס

מה התלמידים אמרו? (מדגם שניתן להערכה שלכם)

- 1. מצביע על העצם הראשי
- 2. הוא מתייחס לאובייקט ספציפי במקרה הצורך
- 3. הוא קובע באיזה משתנה להשתמש מאיזו פעולה
- 4. פעולה הקוראת לתכונה בעצם הנוכחי/מפעילה פעולה על העצם הנוכחי
- 5. ערך המשתנים שהתקבלו
- 6. הוראה שמראה לפעולה באיזה משתנה להשתמש
- 7. this הוא כתובת של האובייקט בו מתבצעת הפעולה במחלקה
- 8. מפנה למשתני המחלקה בתוך המחלקה עצמה הוא המקשר בין העצם לתכונות עזר המבדיל בין פרמטר לבין תכונה או פעולה של מחלקה מסוימת
- 9. משמש כדי לזמן עצמים/פעולות וכו... של אותה מחלקה שאנחנו משתמשים this הוא סוג של פעולה2. הוא מתייחס לאובייקט ספציפי במקרה הצורך

מה התלמידים אמרו? (מדגם)

- 10. המחלקה בה עושים את הפעולה
- 11. העצם הנוכחי שעליו הפעולה מתבצעת
- 12. הוא אופרטור המייצג את המופע הנוכחי(!) של האובייקט במחלקה
- 13. תכונות המחלקה
- 14. העצם או המשתנה שאותו הגדרת להיות הthis
- 15. העצם שהמשתמש הכניס ביצירת הפעולה שלו ובכך כאשר מזמנים את this מזמנים משהו מתוך העצם שהמשתמש יצר
- 16. this הוא תיאור של האובייקט הנוכחי שמדובר עליו בפעולה
- 17. הוא אובייקט אשר נותן למתכנתים במערכת הבחנה בין משתני המחלקה למשתנים חיצוניים

מה התלמידים אמרו? (הסיווג שלנו)

1. מצביע על העצם הראשי (נכון)
2. הוא מתייחס לאובייקט ספציפי במקרה הצורך (פרשנות כנכון)
3. הוא קובע באיזה משתנה להשתמש מאיזו פעולה (מפנה/אופרטיבי)
4. פעולה הקוראת לתכונה בעצם הנוכחי/מפעילה פעולה על העצם הנוכחי (פעולה)
5. ערך המשתנים שהתקבלו (תכונה/משתנה)
6. הוראה שמראה לפעולה באיזה משתנה להשתמש (מפנה/אופרטיבי)
7. this הוא כתובת של האובייקט בו מתבצעת הפעולה במחלקה (נכון)
8. מפנה למשתני המחלקה בתוך המחלקה עצמה הוא המקשר בין העצם לתכונות עזר המבדיל בין פרמטר לבין תכונה או פעולה של מחלקה מסוימת (מפנה/אופרטיבי)
9. משמש כדי לזמן עצמים/פעולות וכו... של אותה מחלקה שאנחנו משתמשים this הוא סוג של פעולה2. הוא מתייחס לאובייקט ספציפי במקרה הצורך (מפנה אל/לא מראה הבנה)

מה התלמידים אמרו? (הסיווג שלנו)

- 10. המחלקה בה עושים את הפעולה (מחלקה)
- 11. העצם הנוכחי שעליו הפעולה מתבצעת (נכון)
- 12. הוא אופרטור המייצג את המופע הנוכחי (!) של האובייקט במחלקה (פרשנות כנכון, אך אופרטור?)
- 13. תכונות המחלקה (תכונות)
- 14. העצם או המשתנה שאותו הגדרת להיות הthis (שגוי)
- 15. העצם שהמשתמש הכניס ביצירת הפעולה שלו ובכך כאשר מזמנים את this מזמנים משהו מתוך העצם שהמשתמש יצר (מפנה/אופרטיבי)
- 16. this הוא תיאור של האובייקט הנוכחי שמדובר עליו בפעולה (נכונה~)
- 17. הוא אובייקט אשר נותן למתכנתים במערכת הבחנה בין משתני המחלקה למשתנים חיצוניים (מפנה/אופרטיבי)

מה אתם אמרתם על מה שהתלמידים אמרו? (N=47)

.

.

תוצאות הסקר המקדים שלכם

מה אתם אמרתם על מה שהתלמידים אמרו? (N=47)

היגד	נכון	ניתן לפרשנות כנכון	נכון~	שגוי	נכון לדעתנו
1	6%	43%	49%	51%	כן אמיתי
2	21%	47%	68%	32%	כן פרשנות
3	4%	30%	34%	66%	כן אופרטיבי
4	40%	36%	77%	23%	לא
5	0%	11%	11%	89%	לא
6	6%	47%	53%	47%	כן אופרטיבי
7	45%	38%	83%	17%	כן אמיתי
8	28%	55%	83%	17%	לא
9	6%	34%	40%	60%	לא
10	23%	49%	72%	28%	לא

פעולה

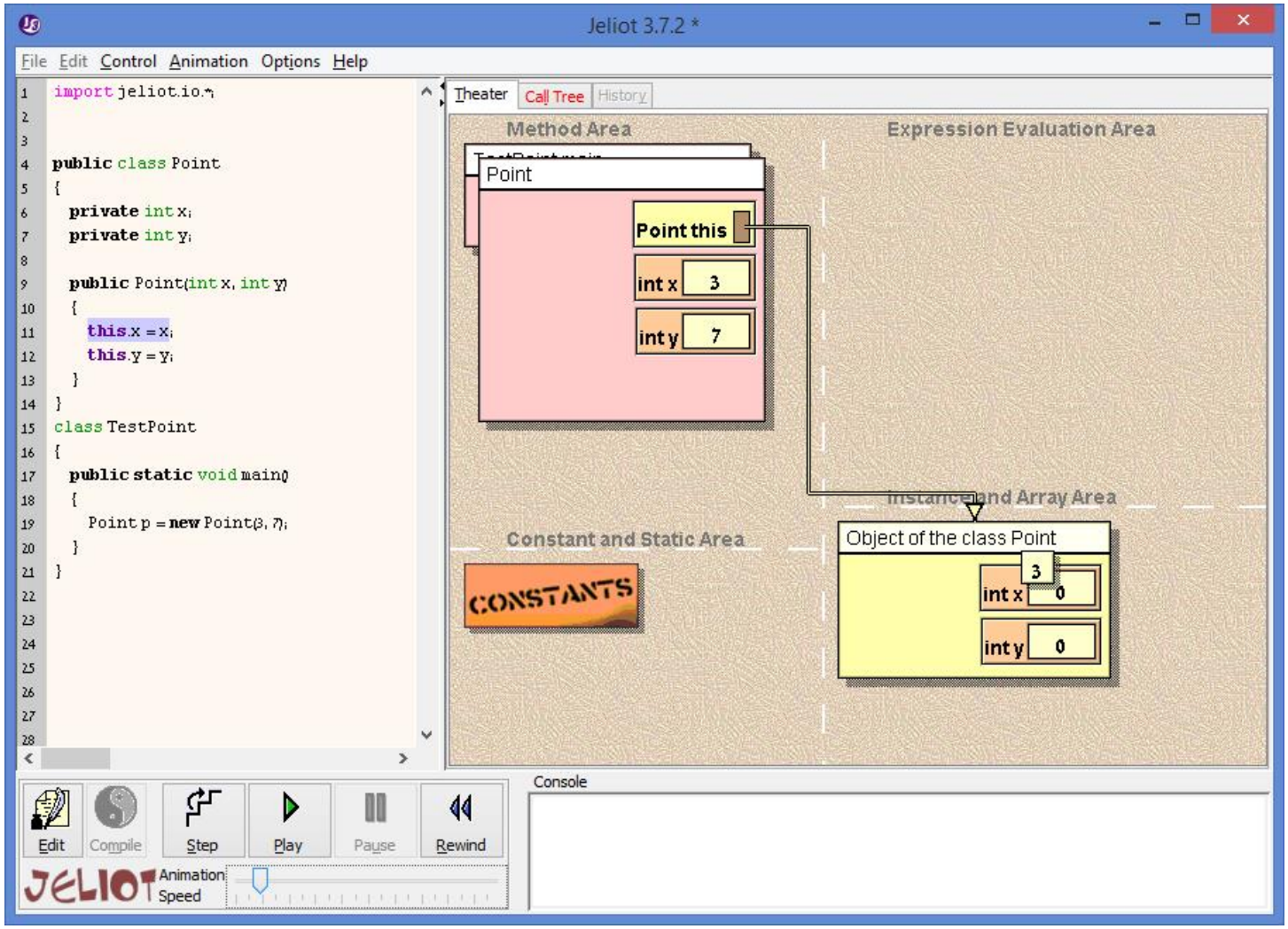
אופרטיבי

מחלקה

מה אתם אמרתם על מה שהתלמידים אמרו? (N=47)

היגד	נכון	ניתן לפרשנות כנכון	נכון~	שגוי	נכון לדעתנו
11	81%	13%	94%	6%	כן
12	28%	45%	72%	28%	כן (אופרטור)
13	6%	15%	21%	79%	לא
14	4%	28%	32%	68%	לא
15	6%	57%	64%	36%	כן אופרטיבי
16	49%	38%	87%	13%	כן (תיאור)
17	23%	51%	74%	26%	כן אופרטיבי

איך ויזואליזציה יכולה לסייע בהבנה



References

Biddle, R. & Tempero, E. (1998). Java pitfalls for beginners. *ACM SIGCSE Bulletin, 30*(2), 48-52.

Chen, C., Cheng, S. and Lin, J.Ṁ. (2012). A Study of Misconceptions and Missing Conceptions of Novice Java Programmers., *Proceedings of the 2012 international conference on frontiers in education: computer sceince & computer engineering*, July 16-19,2012, 307-313.

Eckerdal, A. & Thuné, M. (2005). Novice Java programmers' conceptions of "object" and "class", and variation theory, *SIGCSE Bulletin*, 37(3), 89-93.

Fleury, A. E. (2000). Programming in Java: student-constructed rules. *SIGCSE Bulletin, 32*(1), 197-201.

Holland, S., Griffiths, R. & Woodman, M. (1997). Avoiding object misconceptions. *ACM SIGCSE Bulletin*, 29(1), 131-134.

Ragonis, N. & Ben-Ari, M. (2005a). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education*, *15*(3), 203-221.

Ragonis, N. & Ben-Ari, M. (2005b). On understanding the statics and dynamics of object-oriented programs. *SIGCSE Bull. 37*(1) (February 2005), 226-230.

Sanders, K., Thomas, L. (2007). Checklists for grading object-oriented CS1 programs: concepts and misconceptions, *SIGCSE Bulletin*., *39*(3), 166–170.

Sanders, K., Boustendt, J., Eckerdal, A., McCartney, R., Moström, J. E., Thomas, L., & Zander, C. (2008). Student understanding of Object-Oriented programming as expressed in concept maps. *Proceeding of SIGCSE 2008*, 332-336.

Shmallo, R., Ragonis, N. & Ginat, D. (2012). Fuzzy OOP: expanded and reduced term interpretations. *Proceedings of ITiCSE 12*, ACM Press, 309-314.

Sorva, J. (2008). The same but different – students' understandings of primitive and object variables, *Proceedings of the 8th International Conference on Computing Education Research (Koli Calling '08)*, New York, 5-15.

Thomasson, B., Ratcliffe, M., & Thomas, L. (2006). Identifying novice difficulties in object oriented design, *SIGCSE Bulleti*n., *38*(3), 28–32.

שאלה 1

גרסה 1

```
public class Point
{
    private int x = 0;
    private int y = 0;

    public Point()
    {
    }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        this.x = 0;
        this.y = 0;
    }
}
```

גרסה 3

```
public class Point
{
    private int x;
    private int y;

    public Point()
    {
        x = 0;
        y = 0;
    }
}
```

שאלה 2

גרסה 1

```
public class Point
{
    private int x;
    private int y;

    public Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

שאלה 3

גרסה 1

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        this.x = p.getX();
        this.y = p.getY();
    }
    public int getX()
    {  return x;   }

    public int getY()
    {  return y;   }
}
```

גרסה 2

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        this.x = p.x;
        this.y = p.y;
    }
}
```

גרסה 3

```
public class Point
{
    private int x;
    private int y;

    public Point(Point p)
    {
        x = p.x;
        y = p.y;
    }
}
```