

## תרגיל 1: היכרות עם Squeak

### הוראות כלליות

1. לפני הרצת קוד, שמרו את השינויים שעשיתם בקובץ נפרד על-ידי fileout (קליק ימין). אנו ממליצים לכם לבצע שמירות אלו באופן תכוף כיוון שלסקוויק יש נטיה לקרוס בזמנים שאינם צפויים.
2. במידה וחלקי קוד נמחקים לכם מה-workspace (כולל מחלקות או מתודות שנמחקות), אתם יכולים לנסות לשחזר את הקוד מקובץ log שנמצא בתיקייה: Contents\Resources
3. מטרת התרגיל: היכרות קצרה ובסיסית עם התכנות ב Squeak. תרגיל זה אמור להקל בצורה משמעותית על ההתמודדות עם התרגילים הבאים.
4. בכדי להימנע מטעויות, אנא עיינו ברשימת ה-FAQ המתפרסמת באתר באופן שוטף.
5. את תרגיל זה עליכם לממש ב-Squeak 5.1, אותו ניתן להוריד מהאתר הרשמי. שימו לב שזוהי אינה הגרסה האחרונה!
6. אחראי על התרגיל: דניאל בוגצ'נקו. שאלות על התרגיל יש לשלוח למייל danytnt@campus עם הנושא: "HW1 236703".
7. שאלות אדמיניסטרטיביות (כגון מילואים) יש להפנות למתרגל האחראי, נתן, במייל natanb@cs.technion.ac.il תחת אותה כותרת.
8. מועד הגשה: 19/11/2019 בשעה 23:59.
8. הגשה בזוגות בלבד.

### הקדמה

בתרגיל בית זה נעסוק במימוש של פולינומים ב-Squeak, ונבחן שני סוגים של מימושים. הפולינומים אותם נבחן יהיו פולינומים במשתנה אחד, אינסופיים ולא מוגבלים בגודלם.

המקדמים בפולינום יהיו מספרים שלמים בלבד (Integer), והמעריכים בפולינום יהיו מספרים שלמים בלבד (Integer) הגדולים שווים לאפס. דרישה זו תקפה לשני החלקים!

### שימו לב –

1. על המחלקות בתרגיל בית זה להשתייך לקטגוריה חדשה בשם 00P1.

2. בתרגיל בית זה, זריקת חריגות מתבצעת על ידי:

```
self error: 'invalid input'.
```

# חלק א' – המחלקות Monom ו-Polynom

בחלק זה נגדיר שתי מחלקות: האחת Monom, המייצגת פולינום בעל איבר אחד בודד (לדוגמא:  $-8x^{14}$ ,  $-x$ ,  $42x^2$ ), והשנייה Polynom המייצגת פולינום בעל מספר בלתי מוגבל של מונומים (לדוגמא:  $6x^2 + 5x - 9x^2 - 2$ ).

## המחלקה Monom

### Instance variable

1. `exp`

מחזיק את המעריך של המונום ( $x^{exp}$ ). מאותחל ל-0.

2. `coef`

מחזיק את המקדם של המונום ( $coef * x$ ). מאותחל ל-0.

**אין להגדיר שדות נוספים מעבר לשדות שהוגדרו לעיל!**

### instance methods

1. `exp`

מחזירה את ערך המעריך.

2. `exp: anInteger`

מציבה את המספר `anInteger` כמעריך החדש.

במידה והערך לא מספר שלם או שהערך שלילי, תזרק המחרוזת "invalid input" על ידי שימוש במתודה: `error` (המוגדרת ב-`Object`, כפי שהוצג בעמוד הקודם).

3. `coef`

מחזירה את ערך המקדם.

4. `coef: anInteger`

מציבה את המספר `anInteger` כמקדם החדש.

במידה והערך לא מספר שלם, **תזרק חריגה**.

הנחיה: כדי לבדוק שערך מסוים הוא מטיפוס היורש ממחלקה אחרת (או טיפוס של מחלקה מסוימת), יש להשתמש בהודעה `isKindOf: class`. כאשר `class` הוא שם המחלקה המבוקשת.

5. `derivative`

מחזיר מונום חדש שיהיה הנגזרת של מקבל ההודעה.

# המחלקה Polynom

מחלקה זו מייצגת פולינום. פולינום חדש מאותחל להיות פולינום האפס  $P(x) = 0$ .

## instance variables

monoms .1

אוסף (collection) של **אובייקטים** מטיפוס Monom. עליכם לבחור את האוסף המתאים ביותר מבין האוספים שלמדנו.

שמורות שחייבות להתקיים במהלך החיים של מופע של פולינום:

· אין לשמור שני מונומים שונים באוסף בעלי אותו המעריך.

· אין לשמור מונום באוסף שערך המקדם שלו הוא אפס.

**אין להגדיר שדות נוספים מעבר לשדות שהוגדרו לעיל!**

## instance methods

addMonom: aMonom .1

מתודה זו **משנה** את מקבל ההודעה ומחברת את הפולינום הנוכחי עם מונום. אם aMonom אינו מופע של המחלקה Monom, יש **לזרוק חריגה**.

הפולינום לא אמור להיות מושפע משינויים עתידיים של aMonom לאחר הפעולה.

multiplyByMonom: aMonom .2

מתודה זו **משנה** את מקבל ההודעה ומכפילה את הפולינום הנוכחי במונום.

אם aMonom אינו מופע של המחלקה Monom, יש **לזרוק חריגה**.

הפולינום לא אמור להיות מושפע משינויים עתידיים של aMonom לאחר הפעולה.

asDictionary .3

מתודה זו מחזירה את הפולינום מיוצג בתור מילון (כלומר, חוזר מילון), כאשר המפתחות הם המעריכים של המונומים והערכים הם המקדמים שלהם בהתאמה. לדוגמא: עבור  $P(x) = x^4 + 2$ , יוחזר המילון:

$\{0 \rightarrow 2, 4 \rightarrow 1\}$ . **עבור פולינום ה-0, יש להחזיר מילון ריק.**

derivative .4

מתודה זו מחזירה **פולינום חדש** שיהיה הנגזרת של מקבל ההודעה.

eval: anInteger .5

מתודה זו מחזירה את השערוך של הפולינום בנקודה anInteger.

במידה ו-anInteger אינו מספר שלם, יש **לזרוק חריגה**.

add: aPolynom .6

מתודה זו מחזירה **פולינום חדש** השווה לסכום של הפולינום הנוכחי ופולינום הקלט.

במידה ו-aPolynom לא מופע של המחלקה Polynom, יש **לזרוק חריגה**.

יש לשים לב ולגרום לכך שפולינום הסכום לא יושפע משינויים עתידיים של aPolynom או של מקבל ההודעה.

## חלק ב' – המחלקה PolyStream

מחלקה זו תייצג פולינום שניתן לבצע עליו מספר פעולות מתמטיות במהירות, בניגוד לפולינום שהוגדר על ידי המחלקה הקודמת, אך השערוך שלו יהיה מורכב יותר.  
פולינום חדש מאותחל להיות פולינום האפס  $P(x) = 0$ .  
מחלקה זו לא תשתמש במחלקות Monom ו-Polynomial.

ייתכן שבחלק זה תצטרכו להעתיק Collection בצורה עמוקה (Deep Copy) – כלומר המבנה יועתק וגם הנתונים בו יועתקו. כדי לעשות זאת ב-Squeak משתמשים בהודעה `deepCopy`.  
דוגמת שימוש: `myCollection := aCollection deepCopy`. נושא זה יועבר בהמשך באופן מסודר בתרגול המתקדם של Squeak.

## המחלקה PolyStream

### :instance variables

block.1  
שדה המחזיק בלוק כלשהו (טיפוסו יהיה **BlockClosure**). תוכנו נתון לבחירתכם, קבעו אותו בחוכמה.  
באמצעות שדה זה עליכם לייצג את הפולינום.  
כמו כן, אפשר להשתמש בבלוק המקבל מספר ארגומנטים שונים (עד 3 ארגומנטים). ניתן לעשות זאת בצורה הבאה:  
`b := [:first :second :third | "code goes here"].`  
`b value:1 value:2 value:42.`

אין להגדיר שדות נוספים מעבר לשדה block שהוגדר לעיל!

### :instance methods

addCoef: coef withExp: exp.1  
מתודה זו מוסיפה לפולינום הנוכחי מונם חדש -  $coef * x^{exp}$ .  
אם coef או exp לא מספרים שלמים, או ש-exp שלילי, יש לזרוק חריגה.  
סיבוכיות הזמן הנדרשת מפעולה זו היא  $O(1)$ .

block.2  
מתודה זו תחזיר את השדה block.

add: aPolyStream .3

מתודה זו משנה את הפולינום הקיים לפולינום השווה לסכום של הפולינום הנוכחי ופולינום הקלט.  
במידה ו-aPolyStream לא מופע של המחלקה PolyStream, יש לזרוק חריגה.  
ניתן להניח ש-aPolyStream לא ישתנה לאחר המתודה הזו.  
סיבוכיות הזמן הנדרשת מפעולה זו היא  $O(1)$ .

substitute: anInteger .4

מתודה זו משנה את הפולינום הקיים לפולינום בו מציבים  $anInteger * x$  במקום  $x$ . למשל הקריאה  
4 : substitute f(x) תשנה את הפולינום ל- $f(4x)$ .  
השינוי מתבצע רק על הפולינום הנוכחי ולא משפיע על שינויים עתידיים לפולינום. לדוגמה:

p := PolyStream new.

p addCoef: 1 withExp: 1. "f(x) = x"

p substitute: 2. "f(2x)"

p addCoef: 5 withExp: 2. "5x^2+f(2x). NOT 5\*(2x)^2 + f(2x)"

במידה ו-anInteger אינו מספר שלם או שווה ל-0 יש לזרוק חריגה.  
סיבוכיות הזמן הנדרשת מפעולה זו היא  $O(1)$ .

multiplyBy: anInteger .5

מתודה זו משנה את הפולינום הקיים לפולינום השווה למכפלת הפולינום הנוכחי ב-anInteger.  
במידה ו-anInteger אינו מספר שלם יש לזרוק חריגה.  
סיבוכיות הזמן הנדרשת מפעולה זו היא  $O(1)$ .

eval: anInteger .6

מתודה זו תחזיר את השערוך של הפולינום בנקודה anInteger.  
במידה ו-anInteger אינו מספר שלם, יש לזרוק חריגה.

**מפעולה זו אין דרישה על סיבוכיות הזמן. נצלו זאת על מנת לתמוך בדרישות הסיבוכיות של הסעיפים  
הקודמים.**

**בתרגיל זה אין להוסיף שדות, מלבד אלו שצוינו במפורש לאף אחת מהמחלקות! באופן כללי, אין כמובן  
לחסוך בשימוש בשדות בסקוויק.**

## מאיפה מתחילים?

וודאו שעברתם על המדריך לסביבת Squeak אותו תוכלו למצוא בחומר הקורס, ואתם מבינים כיצד להשתמש בסביבת העבודה. חזרו על שני התרגולים הראשונים. מומלץ מאוד לממש את התרגיל לפי סדר המחלקות שהוצגו. לאחר שמימשתם את חלק א', וודאו שאתם מצליחים להגדיר ולהשתמש בפוליוןם. לאחר מכן, **קראו** את כל הדרישות בכל אחת מהמתודות בחלק ב', **תכננו היטב** את המימוש וכתבו אותו.

## טיפים, מגבלות וסייגים

- "פוליוןם בלתי מוגבל" – הכוונה לפוליוןם שאינו מוגבל ע"י התוכנית שכתבתם. היות ומדובר בתכנית שרצה על המחשב (ולא באלגוריתם תיאורטי) ברור כי באופן מעשי קיימת מגבלה המוכתבת ע"י שפת התכנות, כמות הזכרון וכו'.
- מהירות ביצוע אינה נושא מרכזי בתרגילי הבית בקורס תכנות מונחה עצמים. לכן, בכל מקרה של התלבטות בין פשטות לבין ביצועים **העדיפו את המימוש הפשוט**. לא תתקבלנה טענות בסגנון: "העדפתי מהירות ולכן הקוד יצא קצת פחות ברור".
- Squeak היא שפה שבה כל מתודה לא צריכה לקחת יותר מכמה שורות קוד. הימנעו משכפול קוד והשתמשו בקוד שכבר כתבתם כדי לחסוך לכם עבודה. תכנון נכון של הקוד והוספת מתודות-עזר במקרה הצורך יקטינו משמעותית את כמות הקוד שתצטרכו לכתוב ולבדוק.
- הקפידו לתעד את הקוד שלכם. כל חלק לא טריוויאלי בקוד יש לתעד.
- מומלץ לעבור על שני התרגולים הראשונים בסקוויק ועל המעבדה (Squeak Lab, בחומר הקורס) לפני תחילת העבודה.
- מותר לכם להגדיר מתודות כאוות נפשכם. אין להוסיף שדות מעבר לאלה שהוגדרו בתרגיל.
- **אם התוכנית שלכם נתקעה לחצו על ALT+.** בכדי ליצור פסיקת משתמש אשר תפסיק את ביצוע התכנית ותאפשר לכם לראות היכן היא נתקעה.
- **בלוקים ב-Squeak הם closures ויש להם תכונות מיוחדות. קראו שוב את התרגולים כדי לראות דוגמאות שימוש ספציפיות.**
- אופרטור השוואה (=) הדיפולטי של סקוויק מבצע השוואות by reference. נלמד עוד על האופרטור בהמשך הקורס, אך אם רוצים להשתמש בתרגיל ב-collections כגון Dictionary או Set המבצעים השוואות על האובייקטים, יש צורך לממש את אופרטור = וגם את המתודה hash עבור האובייקט שיעבור השוואה (אם טרם מומש). דוגמאות מימוש לשתי הפונקציות עבור המחלקה point2D המכילה שדות x ו-y:

```
= anObject  
  ^ (anObject class = Point2D) and: [(x = anObject getX) and: [y = anObject getY]]
```

```
hash  
  ^ (x + y) hash
```

### לנוחיותכם מספר שגיאות נפוצות:

- אי ציון המילה self כשאובייקט שולח הודעה לעצמו (בניגוד ל-C++ שם ניתן להשמיט את המילה this).
- לא ניתן להציב בתוך פרמטר של מתודה, בתוך גוף המימוש שלה. למשל, אם x הוא פרמטר של המתודה, אזי ההצבה  $x := x * 2$  לא תשנה את ערכו של x.
- אם ברצונכם להשתמש ברשימה מקושרת (LinkedList), עליכם להשתמש באובייקט מסוג Link. כלומר, כל אובייקט שאתם רוצים להכניס לרשימה חייב לרשת מ-Link.

### הוראות הגשה

- שימו לב שבקורס יש מדיניות איחורים, כלומר ניתן להגיש באיחור גם בלי אישור דחייה (עבור קנס בניקוד). פרטים נוספים באתר הקורס תחת General info.
- הגשת התרגיל תתבצע אלקטרונית בלבד (יש לשמור את אישור השליחה!).
- יש להגיש קובץ בשם `<ID2>_<ID1>_OOP1` המכיל:
  - קובץ בשם `readme.txt` המכיל שם, מספר זיהוי וכתובת דואר אלקטרוני עבור כל אחד מהמגישים בפורמט הבא:  
Name1 id1 email1  
Name2 id2 email2
  - קובץ הקוד: `OOP1.st`. על הקובץ להכיל רק את מימוש המחלקות המוזכרות בתרגיל ומתודות לצורך פתרון התרגיל. אין להגיש קוד נוסף, למשל טסטים.
- נקודות יורדו למי שלא יעמוד בדרישות ההגשה (rar במקום zip, קבצים מיותרים נוספים, readme בעל שם לא נכון וכו').

## בהצלחה!

Me:  $7 + 3 * 3$

Squeak: 30

Me:

