

מבוא לתכנות מונחה עצמים Ex0

במטלה זו התבקשנו לממש מחלקות שמהוות פולינום וכוללות מספר יכולות כמו, חיבור, חיסור, כפל, מימוש נגזרת ואינטגרל.

כדי לממש את הפולינום ראשית מימשתי את מחלקת המונום.

במחלקת המונום האובייקט הוא בעל שתי שדות, מקדם (Coefficient) וחזקה (power), הבנאי העיקרי במחלקה מקבל כפרמטר את שתי המשתנים האלו ואיתם מאפשר בנייה של האובייקט.

Constructors
Constructor and Description
<code>Monom(double a, int b)</code>
<code>Monom(Monom ot)</code>

בנוסף ישנם מספר פונקציות המאפשרות חיבור, הכפלה, ערך המונום בנקודה X, נגזרת, וחישוב שטח למונום יחיד.

Method Summary	
All Methods	Instance Methods
Concrete Methods	
Modifier and Type	Method and Description
void	<code>Add(Monom a)</code>
double	<code>area(double x0, double x1, double eps)</code>
void	<code>Derivate()</code>
double	<code>f(double x)</code>
double	<code>get_coefficient()</code>
int	<code>get_power()</code>
boolean	<code>IsZero()</code>
void	<code>Multiply(Monom a)</code>
boolean	<code>PowerIsZero()</code>
void	<code>set_coefficient(double a)</code>
void	<code>set_power(int p)</code>
Methods inherited from class java.lang.Object	
<code>equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait</code>	

מחלקת פולינום

מחלקה זו מממשת פולינום ע"י רשימה של מספר מונומים שמהווים את העצמים של הרשימה, מבנה הנתונים של המחלקה ממומש ע"י ArrayList, ובו בעצם מאוחסנים המונומים ומהווים פולינום.

The Init constructors

void	Init()
void	Init(java.lang.String s)

הבנאי הראשון יוצר פולינום ריק ומאפשר הכנסה של מונומים ע"י האופציה add שקיימת ב ArrayList כברירת מחדל.

הבנאי השני מקבל String כפרמטר (מבנה המחרוזת הוא `String s = "4.0X^6+3.0X^8";`) מפרק את המחרוזת ע"י הפונקציה split למקדמים (חיוביים ושיליים) וחזקות (חיוביות ושלמות בלבד) וכך מכניס את הערכים למונומים בודדים ומצרף אותם לפולינום.

פונקציות החיבור

Modifier and Type	Method and Description
void	<code>add(myMath.Monom m1)</code> Add m1 to this Polynom
void	<code>add(myMath.Polynom_able p1)</code> Add p1 to this Polynom

הפונקציה הראשונה מאפשרת חיבור של מונום בודד לפולינום.

הפונקציה השנייה מאפשרת חיבור של שני פולינומים.

פונקציות נוספות:

Area

חישוב השטח מתחת לפונקציה ומעל ציר האיקס ע"י הצבת שתי נקודות ומספר המלבנים על מנת להגיע לדיוק מרבי.

```
public double area(double x0,
                  double x1,
                  double eps)

Compute Riemann's Integral over this Polynom starting from x0, till x1 using eps size steps, see: https://en.wikipedia.org/wiki/Riemann\_integral

Specified by:
    area in interface myMath.cont_function
Specified by:
    area in interface myMath.Polynom_able

Parameters:
    x0 - starting pooint
    x1 - end point
    eps - positive step value

Returns:
    the approximated area above the x-axis below this Polynom and between the [x0,x1] range.
```

Root

הפונקציה ROOT מאפשרת חישוב של שורש של פונקציה שמהווה את הפתרון של הפונקציה כאשר $F(x)=0$, ז"א פתרון הפונקציה הוא ה x בקירוב עד Eps

```
public double root(double x0,
                  double x1,
                  double eps)
```

Compute a value x' ($x_0 \leq x' \leq x_1$) for with $|f(x')| < eps$ assuming $(f(x_0) * f(x_1) \leq 0)$, returns $f(x_2)$ such that: * (i) $x_0 \leq x_2 \leq x_1$ && (ii) $f(x_2)$

Specified by:

root in interface myMath.Polynom_able

Parameters:

x0 - starting point

x1 - end point

eps - step (positive) value

Returns:

Multiply and Subtract

הפונקציה Multiply מאפשרת הכפלה של פולינום בפולינום אחר, כאשר השינוי מתבצע על הפולינום המקורי, מכפילה כל פעם מונום אחד מהפולינום שמתקבל כפרמטר עם הפולינום המקורי ולבסוף מחזירה את הפולינום המעודכן לאחר ההכפלה.

הפונקציה Subtract מקבלת כפרמטר פולינום אחר, בשלב ראשון היא הופכת את מקדמי הפולינום שמתקבל כפרמטר לשליליים ולאחר מכן מבצעת חיבור בין שני הפולינומים.

subtract

```
public void subtract(myMath.Polynom_able p1)
```

Subtract p_1 from this Polynom

Specified by:

subtract in interface myMath.Polynom_able

Parameters:

p1 -

multiply

```
public void multiply(myMath.Polynom_able p1)
```

Multiply this Polynom by p_1

Specified by:

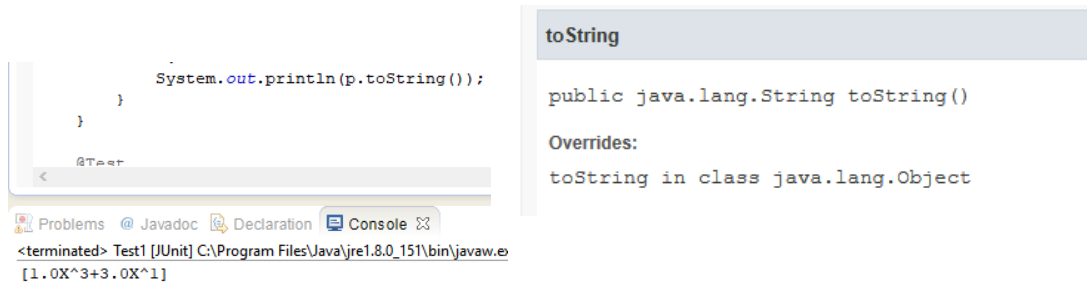
multiply in interface myMath.Polynom_able

Parameters:

p1 -

ToString()

הפונקציה מדפיסה את הפולינום למסך על פי המבנה שהוגדר להכנסת פולינום חדש.



The screenshot shows an IDE with a Java test class on the left and a Javadoc window on the right. The test class contains the following code:

```
System.out.println(p.toString());
}
```

The Javadoc window shows the `toString` method signature and its overrides:

```
public java.lang.String toString()
Overrides:
toString in class java.lang.Object
```

The console output shows the result of the `toString` method call:

```
<terminated> Test1 [JUnit] C:\Program Files\Java\jre1.8.0_151\bin\javaw.e
[1.0X^3+3.0X^1]
```

Area under

הפונקציה `Area under` מחשבת את השטח הכולל של הפונקציה כאשר היא יורדת מתחת לציר ה-X, הפונקציה מחשבת את השטח על ידי העלאה של `Eps` פעם שהלולאה רצה.

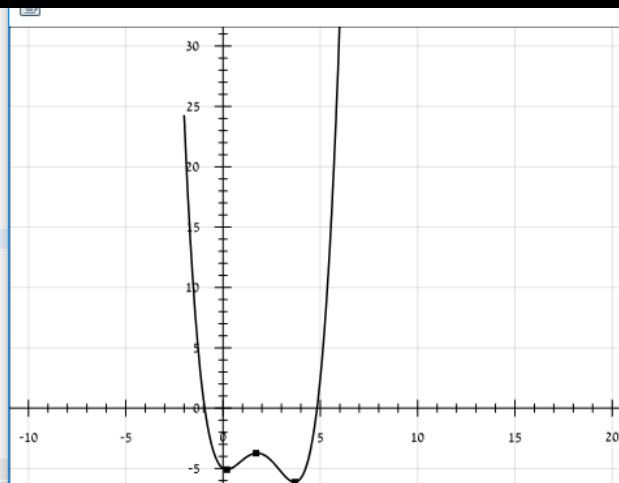
```
/* this function coputes the area under above the function and under the x axis,
 * and adding it to the area that above the axis.
 * @param x0 = the left limit of the funtion
 * @param x1 = the right limit of the funtion
 * @param eps = the eps of the area you want to compute
 * @return the area above and under the axis.
 */
public double areaUnder (double x0,double x1 , double eps) {
    double area = 0;
    for (double i = x0; i < x1; i+=eps) {
        if(this.f(i)<0) {
            area+= eps*Math.abs(f(i));
        }
    }
    return area;
}
```

graphPlot

הפונקציה הנ"ל ממומשת במחלקת פולינום ומאפשרת למשתמש להדפיס גרף בתחום מסוים, הפונקציה מחפשת נוסף נקודות קיצון באותו תחום ומסמנת אותם על הגרף.

```
}  
/**  
 * this funvtiom plot the polynom to the screen and marks the max and min points  
 * by getting the starting x and the end x of the praphic visualition  
 */  
public void graphPlot(double x0, double x1) {  
    LinePlotTest frame = new LinePlotTest(this ,x0 ,x1 );  
    frame.setVisible(true);  
}
```

```
1 package myMath;  
2 import myMath.LinePlotTest;  
3 public class main {  
4  
5     public static void main(String[] args) {  
6         // TODO Auto-generated method stub  
7         String s = "0.2X^4-1.5X^3+3.0X^2-1.0X^1-5";  
8         Polynom a = new Polynom();  
9         a.Init(s);  
10        double x0 = -2;  
11        double x1 =6;  
12        a.graphPlot(x0,x1);  
13    }  
14 }  
15  
16 }  
17
```



הערות נוספות:

- במהלך הפרויקט השתמשתי לראשונה בIterator כמצביע על התקדמות מנה הנתונים, התקשיתי להביאו לידי ביטוי בחלק מהפונקציות, ולכן השתמשתי באינדקס מספרי.
- בפונקציה AREA התייחסתי ל eps כמספר המלבנים שהמשתמש רוצה לחלק את השטח על מנת להגיע לדיוק מרבי.
- הפונקציה Init שמקבלת מחרוזת יכולה לקבל סוג קלט של מספרים חיוביים ושיליים למקדמים ולאחר מכן חייב להופיע הרצף "X^" וסימן המקדם הבא.