הטכניון – מכון טכנולוגי לישראל
הפקולטה למדעי הנתונים וההחלטות
הנדסת תוכנה אביב תשפ"ג

הנדסת תוכנה – תרגיל בית 2

דגשים להגשת המטלה

1. תאריך הגשה: יום שלישי 06.06.2023 בשעה 23:59.
2. **הגשה בזוגות בלבד!**
3. הקוד חייב להיכתב בהתאם למוסכמות כתיבת הקוד בקורס כולל תיעוד כנדרש. קוד שלא עומד בדרישות יגרור הורדת ניקוד. ניתן למצוא את קובץ מוסכמות הקידוד באתר הקורס תחת הלשונית "קבצי עזר".
4. ההגשה מתבצעת ב-Moodle באזור המיועד על ידי אחד מהשותפים, לאחר יצירת קבוצה.
5. ניתן להגיש את התרגיל לכל היותר עד 48 שעות לאחר מועד ההגשה ללא הורדת ניקוד. לאחר 48 שעות תיבת ההגשה תיסגר ולא יהיה ניתן להגיש את התרגיל כלל.
6. פורמט הגשת התרגיל נמצא בקובץ ההנחיות ב-Moodle. **כל חריגה מפורמט זה תגרור ציון אפס.**

מטרת התרגיל

עבודה עם מחלקות והורשה.

הכנות טרם תחילת התרגיל

1. פתיחת פרויקט ג'אווה חדש.
2. הורדת קבצי התרגיל, והעתקת הקובץ Main.java **בלבד** אל תוך תיקיית ה-src.

הוראות כלליות

1. מומלץ להריץ את התוכנית עם מספר קלטים שונים ולחשוב על מקרי קצה אפשריים.
2. מומלץ לחזור על התרגולים וההרצאות וכן להיעזר באינטרנט.
3. יש להשתמש בגרסה 9.0.4 של ג'אווה בעת פתרון התרגיל.
4. מומלץ להשתמש ב-Git במהלך כתיבת התרגיל.

הוראות הגשה

1. יש למלא אחר הוראות ההגשה בהתאם למסמך הדרישות "הנחיות כלליות לפתרון והגשת תרגילי הבית" אשר מופיע באתר הקורס.
2. הגשה אלקטרונית **בלבד** דרך אתר הקורס ב-moodle. ההגשה תכלול קובץ ה-zip בלבד בפורמט HW2_<id1>_<id2> כאשר <id1> ו-<id2> הם תעודות הזהות של המגישים. פירוט נוסף נמצא בסוף המסמך.
3. ההגשה מתבצעת על ידי אחד מבני הזוג לאחר שיצר קבוצה ובן הזוג השני הצטרף אליה.
4. תרגיל בית שלא יוגש על פי הוראות ההגשה – **לא ייבדק**.
5. יש להקפיד על יושרת הכנת התרגיל וההגשה.
6. יש לוודא כי הקוד מתקמפל – קוד אשר לא יעבור הידור יקבל ציון אפס.
7. אין צורך להגיש את קובץ הפלט אשר ניתן כחלק מתרגיל זה.

חלק א – שוויון עצמים

בתרגול 4 דיברנו על שוויון עצמים, ובפרט על שוויון עצמים כאשר משתמשים בהורשה.

בחלק זה של התרגיל יהיה עליכם לממש שתי מחלקות ולדרוס בהן את הפעולות equals, hashCode ו-toString.

המחלקה הראשונה הינה מחלקה בשם Date. מחלקה זו מייצגת תאריך ומכילה שלוש תכונות – יום (מספר בין 1 ל-31), חודש (מספר בין 1 ל-12) ושנה (מספר בין 3999- ל-3999).

המחלקה השנייה הינה מחלקה בשם DateTime. מחלקה זו יורשת מן המחלקה Date ומייצגת תאריך ושעה. בנוסף לשלוש התכונות המוגדרות במחלקה Date, המחלקה מכילה שתי תכונות נוספות – שעה (מספר בין 0 ל-23), ודקה (מספר בין 0 ל-59).

על פעולות ה-equals וה-hashCode של המחלקות לעמוד במפרטים של פעולות ה-equals וה-hashCode כפי שהם מוגדרים במחלקה Object (בהתאמה).

בנוסף, על יחס השוויון לקיים מספר תכונות נוספות:

- כל שני מופעים של המחלקה Date יהיו שווים זה לזה אם ורק אם ערכי התכונות שלהם שווים זה לזה בהתאמה.
- כל שני מופעים של המחלקה DateTime יהיו שווים זה לזה אם ורק אם ערכי **כל** התכונות שלהם שווים זה לזה בהתאמה.
- לכל משתנה d בעל טיפוס דינמי Date ולכל משתנה dt בעל טיפוס דינמי DateTime על הקריאות d.equals(dt) ו-dt.equals(d) להחזיר שקר, ללא תלות בערכי התכונות של המופעים.

את פעולת ה-toString במחלקה Date יש לממש כך שהתאריך יוצג בפורמט DD/MM/YYYY ואת פעולת ה-toString במחלקה DateTime יש לממש כך שמופיעה ייוצגו בפורמט DD/MM/YYYY hh:mm. בעת מימוש הפעולה במחלקה DateTime יש להיעזר בפעולת ה-toString של המחלקה Date.

הוראות נוספות:

- בכתיבת פעולות ה-equals חל איסור על שימוש בפעולות אחרות המוגדרות במחלקה Object, ובפרט בפעולה getClass.
- יש לממש את פעולות ה-hashCode באופן בו לכל שני מופעים של **אותה המחלקה** שאינם שווים זה לזה יהיה ערך גיבוב **שונה**.
- במחלקה Date אין לאזכר את המחלקה DateTime, ובמחלקה DateTime אין לאזכר את המחלקה Date פרט להגדרת ההורשה בכותרת המחלקה (כלומר, חל **איסור** שבקוד של מחלקה אחת יופיע השם של המחלקה השנייה, פרט לשורה בה מגדירים כי המחלקה DateTime יורשת מן המחלקה Date).
- בכל מקרה בו מנסים להגדיר לתכונה כלשהי ערך שאינו בטווח החוקי לתכונה זו יש לשנות את הערך של התכונה לערך ברירת המחדל: 1 עבור הימים והחודשים ו-0 עבור שאר התכונות. עם זאת, אין צורך לבדוק את תקינות התאריך עצמו (שאכן התאריך קיים).

חלק ב – פונקציות מתמטיות

בחלק זה של התרגיל תבנו מערכת לייצוג פונקציות מתמטיות.

סיפור המערכת

המערכת אותה תבנו מורכבת מעשרה סוגים של פונקציות:

- פונקציה קבועה: פונקציה מהצורה $f(x) = C$ עבור מספר ממשי כלשהו C.
- פולינום: פונקציה מהצורה $f(x) = a_0 + a_1x + \cdots + a_nx^n$. האיברים $a_0, a_1, \dots, a_n$ הינם מספרים ממשיים אשר נקראים מקדמי הפולינום.
- סכום: פונקציה אשר מורכבת מסכום של שתי פונקציות.
- מולטי-סכום: פונקציה אשר מורכבת מסכום של לפחות שתי פונקציות.
- הפרש: פונקציה אשר מורכבת מהפרש בין שתי פונקציות.
- מכפלה: פונקציה אשר מורכבת ממכפלה של שתי פונקציות.
- מולטי-מכפלה: פונקציה אשר מורכבת ממכפלה של לפחות שתי פונקציות.
- מנה: פונקציה אשר מורכבת ממנה של שתי פונקציות.
- פונקציה נגדית: פונקציה אשר מורכבת מפונקציה אחרת, ומייצגת את הפונקציה הנגדית לה.
- חזקה: פונקציה אשר מורכבת מפונקציה אחרת, ומייצגת את העלאתה בחזקה.

כל פונקציה מתמטית ניתנת לחישוב, הצגה וגזירה:

- חישוב: בהינתן נקודה (מספר ממשי) x, ניתן לחשב את ערך הפונקציה באותה נקודה.
- הצגה: כאשר מציגים פונקציה, התוצאה המתקבלת הינה מחרוזת המציגה את הפונקציה, מוקפת בסוגריים עגולים משני הצדדים.
- גזירה: ניתן לגזור כל פונקציה ולקבל את הנגזרת שלה. על מנת שלא להכעיס את צנזור, נניח שכל הפונקציות שבתרגיל גזירות.

לדוגמה, נתבונן בפונקציה $3x^2(2 + 3x - 4x^3)$. ערכה של הפונקציה בנקודה $x = 1$ הינו 3, והמחרוזת המייצגת את הפונקציה הינה ((3x^2) * (2 + 3x - 4x^3)). את הנגזרת נשאיר כתרגיל עבורכם.

מימוש המערכת

המערכת אותה תבנו תתמוך בפונקציות מתמטיות מסוגים שונים.

על מנת לממש את המערכת, תצטרכו ליצור מספר מחלקות. מחלקות אלו יתבססו אחת על השנייה, וביחד יהוו את מכלול המערכת.

מחלקת Function

מחלקה זו מייצגת פונקציה מתמטית ובה מוגדרות הפעולות הבאות:

- פעולה בשם valueAt אשר מקבלת נקודה (מספר ממשי) ומחזירה את ערך הפונקציה בנקודה זו.
- פעולה בשם toString אשר אינה מקבלת פרמטרים ומחזירה מחרוזת המייצגת את הפונקציה.
- פעולה בשם derivative אשר אינה מקבלת פרמטרים ומחזירה את נגזרת הפונקציה.
- פעולה בשם bisectionMethod אשר מקבלת שלושה מספרים ממשיים a, b ו-epsilon, מחפשת קירוב לשורש הפונקציה בקטע $[a, b]$ עם שגיאה epsilon בעזרת שיטת החצי ומחזירה אותו. בהמשך מופיע פירוט על הפעולה ועל אופן חישוב השורש.
- פעולה בשם bisectionMethod אשר מקבלת שני מספרים ממשיים a, b, מחפשת קירוב לשורש הפונקציה בקטע $[a, b]$ בעזרת שיטת החצייה, מחזירה אותו ומשתמשת בשגיאה עם ערך ברירת מחדל של 10^{-5}.
- פעולה בשם newtonRaphsonMethod אשר מקבלת שני מספרים ממשיים a ו-epsilon, מחפשת קירוב לשורש הפונקציה בסביבת הנקודה a עם שגיאה epsilon בעזרת שיטת ניוטון-רפסון ומחזירה אותו. בהמשך מופיע פירוט על הפעולה ועל אופן חישוב השורש.
- פעולה בשם newtonRaphsonMethod אשר מקבלת מספר ממשי a, מחפשת קירוב לשורש הפונקציה בסביבת הנקודה a בעזרת שיטת ניוטון-רפסון, מחזירה אותו ומשתמשת בשגיאה עם ערך ברירת מחדל של 10^{-5}.
- פעולה בשם taylorPolynomial אשר מקבלת מספר שלם n, ומחזירה את פולינום טיילור של הפונקציה מסדר n. בהמשך מופיע פירוט על הפעולה ועל אופן חישוב הפולינום.

מחלקת Constant

מחלקה זו מייצגת פונקציה קבועה. ידוע כי לאחר יצירת הפונקציה, לא ניתן יהיה לשנות את ערכו של הקבוע לו שווה הפונקציה.

מחלקת Polynomial

מחלקה זו מייצגת פולינום. במחלקה יש להגדיר בנאי שמקבל את מקדמי הפולינום כ-varargs של מספרים ממשיים, כאשר המספר ה-i-י מייצג את המקדם של x^i. ידוע כי לא ניתן לשנות את מקדמי הפולינום לאחר יצירתו.

בעת הצגת פולינום:

- על המקדמים להופיע לפי סדר עולה של חזקתם, בפורמט $a_i x^i$.
- אין לייצג במחרוזת חזקות שהמקדם שלהן שווה 0.
- במידה ומקדם כלשהו שחזקתו אינה 0 שווה ל-1 או למינוס 1, אין לכלול אותו בביטוי. במקרה זה הפורמט יהיה x^i.
- במידה ומקדם כלשהו הוא מספר שלם (שאינו 1, -1 או 0), על המספר להופיע ללא נקודה עשרונית.

מחלקת Sum

מחלקה זו מייצגת פונקציה מסוג סכום.

מחלקת MultiSum

מחלקה זו מייצגת פונקציה מסוג מולטי-סכום.

מחלקת Difference

מחלקה זו מייצגת פונקציה מסוג הפרש.

מחלקת Product

מחלקה זו מייצגת פונקציה מסוג מכפלה.

מחלקת MultiProduct

מחלקה זו מייצגת פונקציה מסוג מולטי-מכפלה.

על מנת לעזור לכם, להלן הנוסחה למכפלה של מספר כלשהו של פונקציות, $f_1(x), \dots f_n(x)$:

$$\left(\prod_{i=1}^{n} f_i(x)\right)' = \sum_{i=1}^{n}\left(f_i'(x)\prod_{i\neq j} f_j(x)\right)$$

שימו לב שביטוי זה הוא מולטי-סכום של מולטי-מכפלות.

מחלקת Quotient

מחלקה זו מייצגת פונקציה מסוג מנה.

מחלקת Negation

מחלקה זו מייצגת פונקציה נגדית.

מחלקת Power

מחלקה זו מייצגת פונקציה מסוג חזקה. ניתן להניח כי המעריך של החזקה הוא מספר שלם וחיובי.

להזכירכם, להלן הנוסחה לגזירת פונקציה המועלית בחזקה, באופן המשאיר את המעריך חיובי:

$$(f(x)^n)' = \begin{cases} n\cdot f(x)^{n-1}\cdot f'(x), & n>1 \\ f'(x) \quad , & n=1 \end{cases}$$

מציאת שורש

בתחום האופטימיזציה, קיימות שיטות רבות למציאת שורשים לפונקציות חד-משתניות.

אחת השיטות למציאת שורש נקראת שיטת החציה, אשר הינה שיטה מסדר אפס המשתמשת רק בערכי הפונקציה. מטרתה של שיטת החצייה היא למצוא קירוב לשורש הפונקציה בקטע נתון $[a,b]$, בעל שגיאה נתונה שאינה עולה על epsilon.

במסגרת מימוש שיטת החצייה, נניח את ההנחות הבאות:

- הפונקציה רציפה בקטע $[a, b]$.
- מתקיים $f(a) \cdot f(b) < 0$.
- בקטע $[a, b]$ יש בדיוק שורש אחד.

שיטת החצייה מתבצעת באופן דומה לחיפוש בינארי, וכוללת את השלבים הבאים:

1. הגדרת שני משתנים חדשים: `left = a, right = b`.
2. חזרה בלולאה על השלבים הבאים כל עוד מתקיים `right - left > epsilon`:
   a. הגדרת משתנה חדש: `mid = (left + right) / 2`.
   b. אם מתקיים `f(left)·f(mid) > 0` אז מבצעים `left = mid`.
   c. אחרת, מבצעים `right = mid`.
3. לבסוף, מחזירים את `(left + right) / 2`.

מטרת פעולות ה-`bisectionMethod` היא לממש את שיטת החצייה.

שיטה נוספת למציאת שורש נקראת שיטת ניוטון-רפסון, אשר הינה שיטה מסדר ראשון המשתמשת גם בערכי הפונקציה וגם בנגזרתה. מטרתה של שיטת ניוטון-רפסון היא למצוא קירוב לשורש הפונקציה בסביבת נקודה נתונה a, בעל שגיאה נתונה שאינה עולה על `epsilon`.

שיטת ניוטון-רפסון הינה שיטה איטרטיבית אשר מגדירה סדרת נקודות המתכנסת לשורש הפונקציה. סדרת הנקודות מוגדרת באופן הבא:

$$x_0 = a\,, \qquad \forall k \geq 0: \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

הגדרת סדרת הנקודות תיעצר כאשר מתקיים $|f(x_k)| < epsilon$, והפלט של השיטה הוא הנקודה האחרונה בסדרה. (הערה למתעניינים: שיטה זו מהירה הרבה יותר משיטת החצייה, והיא נמצאת בשימוש במחשבים רבים במגוון תחומים, כגון מציאת שורשים של פונקציות ומציאת שורש ריבועי של מספר נתון).

מטרת פעולות ה-`newtonRaphsonMethod` היא לממש את שיטת ניוטון-רפסון.

<u>פולינום טיילור</u>

פולינום טיילור הינו פולינום המקרב פונקציה כלשהי באמצעותה ובאמצעות נגזרותיה. פולינום טיילור של פונקציה f מסדר n מוגדר באופן הבא:

$$T_{f,n}(x) \triangleq f(0) + f'(0)x + \frac{1}{2}f''(0)x^2 + \cdots + \frac{1}{n!}f^{(n)}(0)x^n = \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!}x^k$$

כאשר $f^{(k)}$ היא הנגזרת מסדר k של הפונקציה.

מטרת הפעולה `taylorPolynomial` היא לבנות פולינום טיילור של הפונקציה מסדר נתון ולהחזיר אותו.

בעת חישוב העצרת יש להיעזר בטיפוס הנתונים `double` ולא `int`, על מנת לאפשר חישובי פולינום מסדר גבוה.

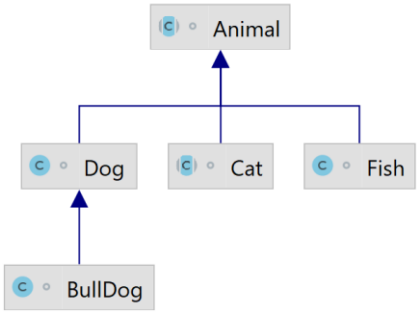
הנחיות לפתרון

- לפני פתרון התרגיל, התחילו בלתכנן את היחסים והקשרים בין המחלקות השונות.
- בעת פתרון התרגיל ניתן ואף מומלץ להגדיר מחלקות נוספות על מנת לחסוך בשכפול קוד כמה שניתן.
- בעת פתרון התרגיל, יש להקפיד על שמות משמעותיים למשתנים, לפעולות ולמחלקות.
- בעת פתרון התרגיל, יש להקפיד על הרשאות הגישה השונות בהתאם לנלמד בקורס.
- בעת פתרון התרגיל, **חשבו האם ישנן מחלקות ופעולות אשר אמורות להיות מופשטות**.
- יש ליצור כל מחלקה בקובץ נפרד.
- **בכל** דריסה ומימוש של פעולה יש להשתמש באנוטציה `@Override` על מנת לוודא שאכן מתבצעת דריסה.
- **יש להשתמש ב-Covariant Return Type במקומות המתאימים**.
- בעת פתרון התרגיל ניתן ואף מומלץ להגדיר קבועים ולא להשתמש במספרי קסם.
- יש לדאוג לכך שניסיון יצירה של פונקציות מסוג מולטי-סכום ומולטי-מכפלה עם פחות משתי פונקציות יוביל **לשגיאת הידור**.
- **אין** לייבא בקוד ספריות או חבילות כלשהן, ואין להיעזר באף פעולה שלא אתם כתבתם, גם לא בפעולות שאינן דורשות ייבוא, פרט לפעולות המוגדרות במחלקה `String` ובמחלקה `Math`.
- בכל מחלקה יש לכלול את התכונות המתאימות לה, ולספק בעבורן פעולות get ו-set **במידת הצורך בלבד**.
- על המחלקות להכיל בנאים. אין חובה שבנאי יקבל כפרמטרים את כל התכונות אשר מוגדרות במחלקה (ואף זה אינו רצוי בעבור חלק מן המחלקות).
- שימו לב כי אין חובה להגדיר את המחלקות על פי הסדר בו הן מופיעות בהנחיות.
- יש לתעד את כל הפעולות והמחלקות אותן אתם מגדירים בעזרת שימוש ב-JavaDoc בהתאם לקובץ מוסכמות התיעוד אשר מופיע באתר הקורס. בנוסף, יש לתעד שורות קוד אשר עשויות להיות קשות להבנה.

הגשת התרגיל

לפני הגשת התרגיל, עליכם ליצור קבצי html מתוך תיעוד ה-JavaDoc שכתבתם בקוד. ניתן ליצור את הקבצים בעזרת ה-IntelliJ על ידי לחיצה על כרטיסיית ה-Tools ובחירת האפשרות Generate JavaDoc. בעת יצירת המסמכים, יש לבחור את הרשאת הגישה private על מנת שכל המחלקות והפעולות (משני חלקי התרגיל) יופיעו במסמכים. את כל הקבצים שנוצרים יש לשמור בתיקייה בשם JavaDoc אשר ממוקמת בתוך תיקיית ה-src.

בנוסף לכך, עליכם ליצור את תרשים המחלקות עבור המחלקות הקיימות בפרויקט. ניתן ליצור את התרשים בעזרת ה-IntelliJ על ידי לחיצה על כפתור ימני בתיקיית ה-src, לחיצה על Diagrams, לחיצה על Show Diagram ובחירת האפשרות Java Classes. ביצוע פעולות אלו יפתח תרשים מחלקות, אותו תוכלו לסדר על מנת שיהיה נוח לקריאה. לאחר מכן, עליכם לשמור את תרשים המחלקות כתמונה: לחצו על כפתור ימני על גבי התרשים, בחרו Export to Image File ושמרו את התמונה המתקבלת. להלן דוגמה לתרשים מחלקות (עם מחלקות אחרות):



כפי שהוזכר, קובץ ההגשה הסופי הינו קובץ zip בודד. שם הקובץ מפורט בתחילת המסמך ובמסמך ההוראות הכללי. קובץ ה-zip יכלול שתי תיקיות - תיקייה בשם src בה יימצאו כל קבצי הקוד אותם כתבתם עבור שני חלקי התרגיל ותיקייה נוספת בשם JavaDoc בה יימצאו כל קבצי התיעוד. בנוסף, קובץ ה-zip יכיל (לא בתוך אף תיקייה) את תרשים המחלקות בשם Diagram_<id1>_<id2>.png, כאשר <id1> ו-<id2> הם תעודות הזהות של המגישים.

<u>הרצת התוכנית וביצוע בדיקות</u>

במחלקה Main קיימות מספר פעולות אשר משמשות לבדיקת הקוד. להזכירכם, חלק מן הבדיקה נעשה באופן אוטומטי, ולכן אין לשנות כלל את המחלקה Main ואת הפעולה הראשית, ובפרט אין לשנות את פעולות ההדפסה המתבצעות בה.

כדי לוודא שהקוד שכתבתם עובד כראוי, מצורף לתרגיל זה קובץ הפלט HW2_output.txt, על מנת שתוכלו לבצע את ההשוואה באופן ידני (או על ידי שימוש ב-DiffMerge).

בהצלחה