תרגיל בית 0

מגישים:

נעה אנגלנדר 307875302

איתי קינן 305186900

שאלה 1-

Naming conventions

1. Classes or interfaces
    a. CamelCase - First letter of each word is capital.
2. Methods
    a. camelCase - First letter of first word is lowercase, first letter of every other word in capital.
3. Member variables
    a. _camelCase - Begin with underscore, first letter is lowercase and the first letter of every other word begins with a capital letter.
4. Local variables
    a. camelCase - First letter of first word is small, first letter of every other word in capital.
5. Final variables
    a. ALL_CAPITAL

Code layout

1. Indentation
    a. All indents are 4 spaces (not with tabs).
2. Brackets
    a. All method names should be immediately followed by a left parenthesis.
    b. All array dereferences should be immediately followed by a left square bracket.
    c. Every block should begin with a left brace ‘{‘ in the same line.
3. Line length
    a. Will be maximum of 120 characters.
4. White spaces
    a. Binary operators should have a space on either side.
    b. Commas and semicolons are always followed by whitespace.
    c. All casts should be written with no spaces.
    d. The keywords if, while, for, switch, and catch must be followed by a space.

Documentation

1. Implementation documentation
    a. Every unique logical implementation should be explained with comments '// …'
2. Specifications
    a. Before each method and class, a JavaDoc spec should be supplied using '/** … */'
    b. @requires, @modifies, @effects fields are mandatory

שאלה 3-

א. פלט תכנית הבדיקה:

```
Creating ball 1: Volume of ball 1 is 1.0
Changing volume of ball 1: New volume of ball 1 is 2.2
Creating ball 2: Volume of ball 2 is 3.4

After creating container:
Volume of balls in container is 0.0
Number of balls in container is 0

After adding ball 1:
ballContainer.contains(ball1) = true
Volume of balls in container is 2.2
Number of balls in container is 1

Adding ball 1 again:
ballContainer.add(ball1) = false
ballContainer.contains(ball1) = true
Volume of balls in container is 2.2
Number of balls in container is 1

After adding ball 2:
ballContainer.contains(ball2) = true
Volume of balls in container is 5.6
Number of balls in container is 2

Removing ball 1 from container
After removing ball 1:
ballContainer.contains(ball1) = false
Volume of balls in container is 3.4
Number of balls in container is 1

After clearing container:
ballContainer.contains(ball1) = false
ballContainer.contains(ball2) = false
Volume of balls in container is 0.0
Number of balls in container is 0

Adding null ball:
ballContainer.add(null) = false

Removing non-existing ball:
ballContainer.remove(ball1) = false
```

ב. בסעיף א' מימשנו את BallContainer כך שכאשר קוראים ל-()getVolume עוברים על כל רשימת הכדורים וסוכמים את נפחם. בסעיף ב' מימשנו את המחלקה כך שהסכימה נעשית בהוספת והסרת כדורים מהמיכל.
היתרון של המימוש הראשון (סעיף א') הוא פשטות המימוש.
חיסרון המימוש הוא שסיבוכיות ()getVolume היא O(n) כאשר n מספר הכדורים במיכל.

היתרון של המימוש השני הוא שסיבוכיות getVolume() היא O(1), אך החיסרון הוא שכעת יש צורך בשימרת משתנה נוסף בזיכרון. (נפח המיכל)

ג.

1. אין צורך בשינויים נוספים למפרט. ב- @return מצוין שמחזירים true אם הצלחנו להוסיף את הכדור, ו-false אחרת. המקרה שבו ball = null נכלל ב"אחרת" ולכן אין צורך לפרט מה לעשות במקרה זה.

2. במימוש שלנו בדקנו ש-ball אינו null.

```
/**
 * @modifies this
 * @effects Adds ball to the container.
 * @return true if ball was successfully added to the container,
 *         i.e. ball is not already in the container; false otherwise.
 */
public boolean add(Ball ball) {
    if (ball == null)
        return false;
    else if (contains(ball))
        return false;
    else {
        _ballList.add(ball);
        return true;
    }
}
```

אם @requires ball != null אז אין צורך בבדיקה if (ball == null) וניתן למחוק אותה.

3. המפרט החדש **חלש** יותר, מכיוון שדורשים יותר מהמשתמש.

שאלה 4

חלק א'

תשובה א – לא ניתן להגדיר יחס של חוזק בין S1 ל – S2

הסבר:

בתרגול 2 למדנו כי על מנת שמפרט ייחשב חזק יותר ממפרט אחר, הוא צריך לקבל יותר קלטים (@requires) או להיות יותר ספציפי לגבי הפלט שהוא יוצר (@effects).

מכיוון שבדוגמה זו התנאי מתקיים עבור כל אחד מהמפרטים כלפי השני (S1 חזק יותר מבחינת @requires ו – S2 חזק יותר מבחינת @effects), לא ניתן לקבוע מי חזק יותר ממי.

חלק ב'

1. **תשובה ד – S2, S4**

נשים לב כי ההבדל בין מפרט S2 ל – S1 הוא ש – S2 דורש כקלט רשימה ממוינת, בעוד S1 לא דורש זאת, כלומר S1 חזק יותר (פחות דרישות על הקלט). לכן בפרט S1 יכול גם לקבל רשימה ממוינת, כלומר הוא יכול לקבל קלט רחב יותר. לכן אם מימוש מסוים מקיים את S1, הרי שהוא מקיים גם את S2.

בנוסף, S1 גם חזק יותר מ – S4, שכן הוא מסוגל לקבל רשימה שהצבע הדרוש אינו בהכרח נמצא בה, כלומר יש בו פחות תנאים על הקלט. לכן, מאותה סיבה, המימוש מקיים גם את S4.

2. **תשובה ו – אף מפרט חוץ מ - S2**

אם מימוש מסוים מקיים מפרט מסוים, הרי שהוא מקיים גם מפרטים חלשים יותר. אך לגבי מפרטים חזקים יותר, לא ניתן לקבוע מבלי לדעת פרטים נוספים על המימוש. במקרה זה, לא ניתן לקבוע האם הוא מקיים את S1, שכן S1 חזק מ – S2.

3. **תשובה ג – S4**

מכיוון ש – S3 חזק יותר מ – S4, ומהסיבה שצוינה בסעיף הקודם, המימוש מקיים גם את מפרט S4.

4. **תשובה ו – אף מפרט חוץ מ – S4**

S4 אינו חזק יותר מאף מפרט שהוצג, לכן לא ניתן לקבוע האם המימוש מקיים מפרט נוסף.