**Object Oriented programmin – Homework 0**

Tzahi Peretz 203840095

Shimi Haleluya 205836455

**Q0 – code guideline**

1.Naming conventions :

1.1 package representing - all should be in lower case . for example

1.2 representing types must written in mixed case startin with an upper case

For example ,

1.3 Variables must be written in mixed case , starting with lower case .

For example ,

1.4 representing constants would be in in uppercase

For example ,

1.5 method names must contain a verb , and written in mixed case start lower case   
 For example ,

1.6Abbreviations and acronyms should not be uppercase when used as name

For example , // not exportHTML

1.7 private class variable should have underscore suffix

For example :

|  |
| --- |
|  |
| 1.8 generic variable should have the same name as their type  For example ,  1.9 the name of an objecet is impilicit , and should avoided in a method name  For example , // not string.getStrngLength() |
|  |

2. Formatting

2.1 Indentation – defined as 4 spaces . done with spaces , not TABS .

2.2 matching braces always will be at the same column

2.3 if ,and loop statement should has braces always .

2.4 class member ordering – privates at the top and the constructor is below them

2.5 maximum line length – maximum line legnth is 120 characters , if the line start to get indented to the right , consdier break your code to moremethods ,

3.comments and documentation

3.1 tricky code shouldn’t be commented but rewritten

3.2 comments in English

3.3 javDoc comments shold have the following form :

*/\*\**

*\* Return lateral location of the specified position.*

*\* If the position is unset, NaN is returned.*

*\**

*\* @param x X coordinate of position.*

*\* @param y Y coordinate of position.*

*\* @param zone Zone of position.*

*\* @return Lateral location.*

*\* @throws IllegalArgumentException If zone is <= 0.*

*\*/*

public double computeLocation(double x, double y, int zone)

throws IllegalArgumentException

{

...

}

2.

The input we used : the output :

Text

Description automatically generated

Line1

Line2 // comment a

// comment b

line 4

line 5 /\* cooment c \*/

/\* comment d \*/

line 7

// comment e

/\*

line 9

line10

\*/

// commetn shimi

/\*

lalal

lalal

\*/

line22

3א.

The input we used :

import java.io.IOException;

import hw0\_new.Ball;

import hw0\_new.BallContainer;

import hw0\_new.BallContainerB;

public class testFor3 {

public static void main(String [] args) throws IOException {

Ball a = new Ball(1.1);

// getvolume at the begining

System.***out***.println(a.getVolume()); // return the volum of ball a - 1.1

// set

a.setVolume(5.5);

// get after set

System.***out***.println(a.getVolume()); // suppose to returnn the ew valume (after set)= 5.5

BallContainerB a\_container = new BallContainerB(10);

System.***out***.println(a\_container.getContainerSize()); // suppose to return 10

Ball b = new Ball(1);

Ball c = new Ball(2);

Ball d = new Ball(4);

a\_container.add(b);

System.***out***.println(a\_container.getVolume()); // suppose to return 1 - only b

System.***out***.println(a\_container.size()); // suppose to return 1 - only 1 elemnt in the list

a\_container.add(a);

a\_container.add(a); // suppose to print invalid - already in the list

a\_container.add(d); // suppose to fail - not enough space

a\_container.remove(a);

a\_container.add(d);

System.***out***.println(a\_container.getVolume());// b+d=5 in the container - suppose to return 5

System.***out***.println(a\_container.contains(b)); // check if b contained - print true

System.***out***.println(a\_container.contains(c)); // check if c contained - print false

a\_container.clear();

System.***out***.println(a\_container.size()); // suppose to return 0 - empty list .

}

}

The output :

1.1

5.5

10.0

1.0

1

Add failed - ball already in the list .

Add failed - not enough place .

5.0

true

false

0

3ב.

לפי המימוש הראשון השתמשנו במשתנה בשם unused שתפקידו לתת חיווי כמה נפח פנוי יש בקונטיינר

ואילו במימוש השני לא השתמשנו במשתנה כזה אלא ביצענו את החישוב בפונקציית getVolume

במימוש א‘ פעולה getVolume מהירה כי אין צורך לבצע לולאה על כל הרשימה אבל כאמור יש לה תפיסת זיכרון גדולה יותר ב64 ביט של טיפוס double.

ואילו במימוש ב‘ getVolume איטית יותר ופונקציית add מהירה כי חוסכים את העדכון של המשתנה unused.

3ג.1.

שינוי זה לא יגרור שינויים נוספים במפרט מכיוון של כעת אנו סומכים על המשתמש כי לא יוזן כinput כדור שהוא null ולכן אין צורך לפרט על המקרה הנ“ל במפרט של return.

(מלכתחילה לא התייחסו למקרה זה בreturn@ לכן אדרבה אין סיבה לשנות כעת כי החלשנו את המתודה)

3ג.2

כן שינוי זה כן יגרור שינויים בקוד מכיוון שכעת אנו סומכים על המשתמש כי לא יזין כinput כדור שהוא null ולכן אין צורך בבדיקה בתחילה המתודה שאכן הכדור לא null.

3ג.3

בהוספת requires@ כלומר אילוצים נוספים שאנו כופים על המשתמש במתודה אנו מחלישים את המתודה שלנו.

**שאלה 4:**

התשובה: **התשובה הנכונה היא א‘.**

נסביר, מצד אחד נתון לנו שמבחינת הrequires המפרט של S1 חזק יותר מהמפרט של S2 אבל מאחר וS2 נותן פלט יותר ספציפי מהפלט של S1 לא ניתן באמת להשוות ביו שני המפרטים שבהחלפה של אחד בשני לא נקבל את אותן התוצאות ולכן למרות הנתון של החוזקה בrequires לא ניתן להגדיר יחס של חוזק בין S1 S2.

התשובה: **התשובה היא ד‘.**

כאן מדובר לפי הבנתינו במפרטים של מחלקות S1 וS2 שמכילות מתודות מסויימות.

תשובות א׳ ,ב׳ וג׳ לא נכונות – לפי הדוגמא הבאה .   
אם יש לנו שתי קלאסים בעלות אותה מתודה ואותה חתימה , אך אחת מהן יכולה לזרוק excpection והשניה איננה . במצב זה , לא ניתן להשוות בין המחלקות .

במקרה הזה , א׳ נפסל כי המחלקות לא ריקות .

ב׳ נפסל , כי המתודות בעלות אותה חתימה.

ג׳ נפסל , בדוגמא שנתנו יכול להיות שלכל המתודות יש אותה חתימה .

לכן ד׳ נכון – יכול להיות קלט מסוים עבורו מימוש שלא יזרוק לנו exception , יתאים לשני המפרטים.

**חלק ב‘**

**1.**

התשובה היא **ד‘**  מכיוון שהמפרט effects@ הוא זהה לS1 ולS2 ואילו בS2 יש רק החלשה של המפרט ומאחר והמימוש של 1 אמור לעבוד גם בלי ההבטחה של מערך ממויין.

כמו כן המימוש יהיה טוב גם למפרט של S4 מכיוון שהדרישות של S4 מובטח שהצבע קיים ברשימה ולכן לא יכול להיות פלט שונה (במצב שבוא הצבע לא קיים ברשימה).

**2.**

התשובה היא **ו‘** מכיוון שהמימוש של S2 הינו מימוש חלש ביחס לאחרים מהבחינה שהוא סומך על המשתמש שיזין מערך ממויין ואילו כל שאר המפרטים לא דורשים את התכונה הזו ונדרשים לטפל במצב שהקלט אינו ממויין.

**3.**

התשובה היא **ג‘**  זאת מכיוון שהrequires של S4 מבטיח לנו מצב שלא נצטרך להכנס לקטע קוד במימוש 3 של הקריאה לחריגה ולכן אכן הפלט יהיה זהה והמפרט של 4 גם יוכל להתקיים מהמימוש של מפרט 3.

**4.**

התשובה היא **ו‘**  נשים לב שהמימוש של S4 לא מטפל במצב שבו הצבע לא קיים במערך ולכן לא מחזיר -1 כמו S1 ו S2 ולכן המימוש של S4 לא יעיל למפרטים S1 וS2.

כמו כן המימוש של S4 לא קורא לחריגה במצב שהצבע לא נמצא (מה שעושה המימוש של S3) ןלכן המימוש של S$ לא מספיק מחסה ולא מקיים אף אחד מהמפרטים מלבד S$ עצמו.