מבני נתונים – תרגיל 2 – מעשי

תאריך פרסום: 2/4/17

תאריך הגשה: 30/4/17

מרצה ומתרגלים אחראים: פרופסור פז כרמי, ענר בן-אפרים, ניצן פרחי

1. מבנה העבודה:

חלקים א + ב:

- תכנון מבנה נתונים יעיל לפתרון הבעיה הנתונה (ראו בהמשך) באמצעות שימוש במבני הנתונים שנלמדו בקורס.
- מימוש המבנה המתוכנן ב-Java ובדיקתו.
- ניתוח תיאורטי של זמני הריצה של המימוש שהצעתם.

חלק ג (בונוס) (10 נק'):

- בדיקה של מבנה הנתונים והמימוש שהצעתם (בחלקים א' + ב') לבעיה נוספת (5 נק')
- לענות על השאלה בסעיף 6.5 (5 נק')

2. מבוא (סיפור):

חברת התעופה Nocrash Flying פנתה אליכם כדי לתכנן עבורן מערכת למניעת תאונות אוויריות. תפקידה העיקרי של המערכת הוא לוודא שהמרחק בין כל שני מטוסים לא יורד מתחת לסף מסויים, וכן שהצפיפות של המטוסים באזור בו מופעלת המערכת לא גדול מדי. על המערכת להיות יעילה ככל הניתן. תיאור מפורט של המערכת, כולל זמני הריצה הנדרשים, נמצא בחלק 4.

3. הסבר על המחלקות Point, Container, ו DataStructure ועל הממשק DT:

לתרגיל מצורפת המחלקות Point, Container, ו DataStructure וכן הממשק DT.

המחלקה Point מכילה שלושה שדות: X, Y, ו- name. אתם צריכים להשתמש במחלקה Point כדי לייצג את מיקום המטוסים. שדה ה-X ייצג את מיקום המטוס בציר ה-X, שדה ה-Y ייצג את מיקום המטוס בציר ה-Y, ו name ייצג את המספר הסידורי של הטיסה. **שימו לב: אין לשנות את המחלקה Point!**

בנוסף, לתרגיל מצורפת מחלקה בשם Container, אשר מכילה את השדה Point data. אתם רשאים להוסיף למחלקה שדות ומתודות כרצונכם, אך **<u>עליכם להשאיר את השדה Point data והמתודה getData()</u>**.

המחלקה DataStructure מיצגת את המערכת שעליכם לבנות. עליכם להשלים את המתודות כך שיבצעו את הנדרש בזמן הריצה המצויין. שימו לב שהמחלקה מממשת את הממשק DT, כלומר עליכם לממש את כל המתודות הנדרשות (ראו חלק 4). **אין לשנות את הממשק DT.** חלק מהמתודות שעליכם לממש נקראות עם או מחזירות points, וחלק containers, ואין לשנות את החתימה במתודות אלו (רק לממש אותן). בנוסף, אתם רשאים להוסיף מחלקות ומתודות כרצונכם (אך לא לשנות את המחלקה Point או את הממשק DT)

הערות חשובות והבהרות:

1. **אין לשנות את המחלקה Point או את הממשק DT!**
2. לעומת זאת, כן צריך להשלים את המחלקות Container ו DataStructure.
3. אתם יכולים להניח שלא יהיו שני מטוסים (כלומר שני Points) בעלי אותו ערך X או שתי נקודות בעלי אותו ערך Y. כלומר, ערכי ה-X וה-Y הם ייחודיים.
4. בחלק מהפונקציות מוגדר משתנה בוליאני axis שמגדיר את הציר הנבחר. אם axis=true הכוונה לציר ה X ואם axis=false הכוונה לציר ה Y
5. המרחק בין שתי נקודות $(X_1,Y_1),(X_2,Y_2)$ הינו $\sqrt[2]{(X_1-X_2)^2+(Y_1-Y_2)^2}$
6. החציון עבור ציר נתון הינו הנקודה שתופיע בתא האמצעי במערך בו הנקודות ממוינות לפי הערך של הציר הנבחר, כלומר התא ה $\left\lfloor \frac{n}{2} \right\rfloor$ במערך עם n תאים [0,...,n-1].

דוגמאות עבור axis==false (כלומר, עבור ציר ה- Y):

0	1	2	3	4
(0,0)	(3,1)	(1,2)	(2,3)	(4,4)

0	1	2	3
(1,15)	(4,20)	(3,21)	(19,24)

שימו לב שבדגומאות הללו המערך מכיל את הנקודות ממויינות לפי ערך ה-Y. הנקודה בתא המסומן היא הנקודה החציונית.

4. פירוט דרישות מבנה הנתונים:

להלן הפעולות שמבנה הנתונים צריך לתמוך בהן, והזמן שלהן <u>במקרה הגרוע</u>, כאשר n מייצג את מספר המטוסים הקיימים במערכת (לגבי זמני הריצה של פעולות 6, 8 ו- 9 ראו את הפירוט בחלק 4.1):

#	Returns	OPERATION	Running Time
1	(constructor)	**DataStructure**()	O(1)
2	void	**addPoint**(Point point)	O(n)
3	Point[]	**getPointsInRangeRegAxis**(int min, int max, Boolean axis)	O(n)
4	Point[]	**getPointsInRangeOppAxis**(int min, int max, Boolean axis)	O(n)
5	double	**getDensity**()	O(1)
6	void	**narrowRange**(int min, int max, Boolean axis)	O(|A|)*
7	Boolean	**getLargestAxis**()	O(1)
8**	Container	**getMedian**(Boolean axis)	O(n)
9**	Point[]	**nearestPairInStrip**(Container container, double width, Boolean axis)	O(min(n,|B|log|B|))*
10	Point[]	**nearestPair**()	??***

* ראו הסבר בחלק 4.1

** ניתן להניח שהמתודה nearestPairInStrip תיקרא רק עם container שהוחזר מהפונקציה getMedian

*** ראו הסבר בחלק 4.2

לפרויקט מצורפים קבצים (ראו בהמשך) שבאמצעותם אתם צריכים להגדיר ולממש את הפעולות הנ"ל. קובץ אחד מכיל ממשק בשם DT המגדיר את הפעולות וקובץ נוסף מכיל מחלקה ריקה בשם DataStructure שמממשת את הממשק. הפתרונות שלכם צריכים להכתב במחלקה DataStructure, ואתם רשאים להוסיף מחלקות עזר כרצונכם. לצורך כך יהיה עליכם גם להוסיף שדות ומתודות למחלקה Container.

4.1. פירוט הפעולות

בכל אחת מהפעולות הבאות, n מייצג את מספר הנקודות במבנה. עבור הבנאי, n הוא מספר הנקודות במערך הקלט.

1. DataStructure()

בנאי, יוצר מבנה נתונים ריק ללא נקודות.

זמן ריצה: O(1)

2. `void` addPoint `(Point point)`

שיטה המוסיפה נקודה למבנה הנתונים. ניתן להניח שאין במבנה הנתונים נקודה עם ערך ה- X או ערך ה-Y של הנקודה החדשה

פרמטרים: הנקודה להוספה.

זמן ריצה: O(n)

3. `Point[]` **getPointsInRangeRegAxis**(int min, int max, Boolean axis)

שיטה המחזירה את הנקודות הנמצאות בתחום נתון על הציר axis, ממויינות לפי אותו ציר

פרמטרים: קצוות התחום min (גבול תחתון) ו- max (גבול עליון), והציר axis

ערך מוחזר: מערך המכיל את הנקודות במבנה הנתונים שנמצאות בתחום, ממוינות לפי אותו ציר.

על השיטה לכלול גם נקודות עם ערך X שהוא ממש min או max

זמן ריצה: O(n)

4. `Point[]` **getPointsInRangeOppAxis**(int min, int max, Boolean axis)

שיטה המחזירה את הנקודות הנמצאות בתחום נתון על הציר axis, **ממויינות לפי הציר השני** (כלומר ממויינות לפי הציר axis! , כאשר ה! מסמל not)

פרמטרים: קצוות התחום min (גבול תחתון) ו- max (גבול עליון) , והציר axis

ערך מוחזר: מערך המכיל את הנקודות במבנה הנתונים שנמצאות בתחום, **ממוינות לפי הציר השני**.

על השיטה לכלול גם נקודות עם ערך X שהוא ממש min או max

זמן ריצה: O(n)

5. `Double` **getDensity**()

שיטה המחזירה את צפיפות הנקודות במבנה, כלומר את מספר הנקודות במבנה חלקי שטח המלבן עם צלעות מקבילות לצירים הקטן ביותר שמכיל את הנקודות.

הנוסחה לחישוב הצפיפות הינה $\frac{n}{(X_{max}-X_{min})(Y_{max}-Y_{min})}$ כאשר X_{max} זה הערך ה X המקסימלי של נקודה כלשהי במבנה הנתונים, וכו'

<u>הערה:</u> לא יבדק המקרה עבורו יש במבני הנתונים נקודה אחת בלבד שכן עבורו הנוסחה לצפיפות אינה מוגדרת.

ערך מוחזר: צפיפות הנקודות במבנה.

זמן ריצה: O(1)

6. void **narrowRange**(int min, int max, Boolean axis)

שיטה זו מוחקת ממבנה הנתונים את כל הנקודות שהערך שלהם בציר axis גדול מ-max או קטן מ-min

ערך מוחזר: אין

פרמטרים: קצוות התחום min (גבול תחתון) ו- max (גבול עליון) , והציר axis

זמן ריצה: O(|A|) כאשר |A| הינו מספר הנקודות שיש למחוק ממבנה הנתונים

7. `Boolean` **getLargestAxis**()

שיטה המחזירה את הציר הגדול ביותר עבור הנקודות הנמצאות במבנה הנתונים. כלומר, מחזיר True אם Xmax-Xmin > Ymax-Ymin ו False אחרת, כאשר Xmax ערך ה X המקסימלי של נקודה כלשהי במבנה הנתונים, וכו'.

ערך מוחזר: השיטה מחזירה את הציר הגדול ביותר עבור הנקודות הנמצאות במבנה הנתונים (True עבור ציר ה X ו False עבור ציר ה Y)

זמן ריצה: O(1)

8. `Container` **getMedian**(Boolean axis)

שיטה המחזירה את הנקודה החציונית בציר הנבחר, מבין כל הנקודות במבנה.

ערך מוחזר: הנקודה עם הערך החציוני (מבין הנקודות במבנה) בציר הנבחר (ראו הסבר ב"הערות חשובות והבהרות").

פרמטרים: הציר axis

זמן ריצה: O(n)

9. `Point[]` **nearestPairInStrip**(Container container, double width, Boolean axis)

שיטה המחזירה את שתי הנקודות בתוך הרצועה שרוחבה הוא width ומרכזה הוא ה- container לפי כיוון axis (ציר X אם axis=true וציר Y אם axis=false) שהן הקרובות ביותר. אם אין 2 נקודות ברצועה, על הפונקציה להחזיר מערך ריק (מערך באורך 0. ניתן להחזיר גם null במקרה זה). אפשר להניח שכל זוג נקודות שנמצאות באותו צד של ה- container לפי כיוון axis מרחקם לפחות $\frac{width}{2}$

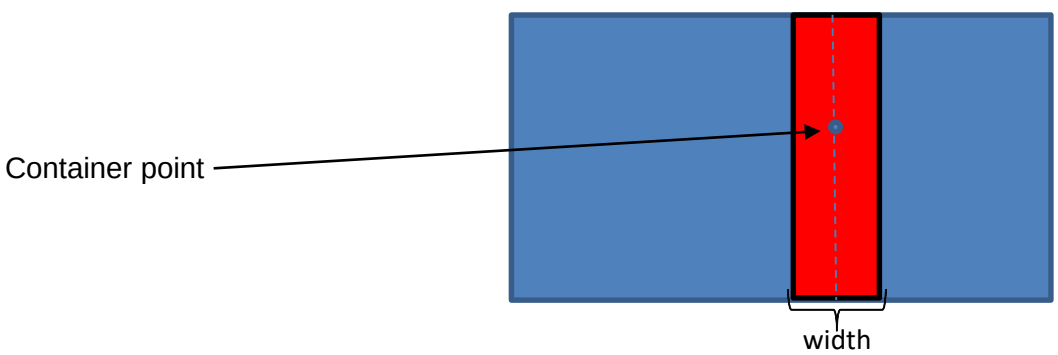
הרצועה מוגדרת ככל הנקודות שהערך שלהם בציר axis הוא $Z_P \pm \frac{width}{2}$, כאשר Z_p הינו הערך של הנקודה ב container בציר axis.

ערך מוחזר: זוג הנקודות הקרובות ביותר – מיוצג ע"י מערך באורך 2 המכיל את 2 הנקודות הקרובות ביותר ברצועה. אם אין זוג נקודות יוחזר מערך ריק (מערך באורך 0 או null).

פרמטרים: Container המכיל נקודה מהתחום (ניתן להניח שהפונקציה תקרא רק עם Container שהוחזר מהפונקציה getMedian), רוחב הרצועה, והציר הנבחר

זמן ריצה: O(min(n,|B|log |B|)), כאשר |B| מספר הנקודות ברצועה. לצרוך כך, אתם רשאים להשתמש בפונקצית המיון Arrays.sort שאתם יכולים להניח רצה בזמן O(N log N), כאשר N גודל מהמערך שהיא ממיינת.

דוגמה:



10. `Point[]` **nearestPair**()

שיטה המחזירה את שתי הנקודות הקרובות ביותר במערך, מיוצג בעזרת מערך של 2 נקודות. ראו פרק 4.2 להסבר מפורט על שיטה זו.

ערך מוחזר: מערך בגודל 2 המכיל את זוג הנקודות הקרובות ביותר במבנה הנתונים. ניתן להניח שקיימות לפחות 2 נקודות במבנה הנתונים.

זמן ריצה: עליכם לחשב את זמן הריצה של פונקציה זו. ראו הסבר בפרק 4.2

4.2. הפונצקיה split, והדרכה והסבר ל nearestPair

הפונקציה split: פונקציה נוספת שעליכם לכתוב פסאודו-קוד בשבילה (אך אין צורך לממש אותה), היא הפונקציה split(int value, Boolean axis), אשר אמורה להחזיר 2 אוספים של נקודות – האוסף של כל הנקודות שהערך שלהן בציר axis גדול מ value (כולל) והאוסף של כל הנקודות שהערך שלהן בציר axis קטן ממש מ value.

זמן הריצה של הפונקציה צריך להיות O(|C|), כאשר |C| מספר הנקודות באוסף **הקטן** (כלומר, האוסף המכיל פחות נקודות). זמן ריצה של O(n) יקבל ניקוד חלקי ואילו זמן ריצה איטי יותר לא יזכה בניקוד.

הוראות:

a. עליכם להסביר מה הערך המוחזר מהפונקציה split וכיצד ניתן לגשת לכל אחת מהנקודות באוסף (אין חובה שהגישה תהיה יעילה)
b. עליכם לכתוב פסואודו-קוד לפונקציה split
c. עליכם לנתח את זמן הריצה של הפונקציה split

הדרכה ל nearestPair: הפונקציה nearestPair מחזירה את זוג הנקודות הקרובות ביותר במבנה הנתונים (כלומר, את שני המטוסים הקרובים ביותר). על הפונקציה לבצע את החישוב באופן הבא:

1. אם יש רק זוג נקודות, החזר אותם. אם יש פחות, החזר זוג ריק (או null).
2. מוצאת את הציר הגדול ביותר axis נניח בלי הגבלת הכלליות ציר X
3. מוצאת את החציון median בציר X
4. מחשבת רקורסיבית את הזוג הקרוב ביותר עבור כל הנקודות הגדולות מ median (כולל) (נקודות שקורדינטת ה-X שלהם גדולה מקורדינטת ה-X של ה- median) ורקורסיבית את הזוג הקרוב ביותר עבור כל הנקודות הקטנות מ median לפי ציר X (נקודות שקורדינטת ה-X שלהם קטנה מקורדינטת ה-X של ה- median)
5. בוחרת את הזוג הקרוב יותר מבין שתי הזוגות בצעד 4 ומחשבת את המרחק ביניהן minDist
6. בודקת האם ברצועה (בציר X) ברוחב 2*minDist אשר האמצע שלה זה ערך ה X של הנקודה median, יש זוג נקודות שמרחקן קטן מ minDist
   a. אם קיימות זוג נקודות כאלו אזי הפונקציה מחזירה את הזוג הזה
   b. אחרת, הפונקציה מחזירה את הזוג מצעד 5

5. חלק א' – חלק התרגיל המעשי:

לתרגיל מצורף קובץ ZIP ובו:

- המחלקה Point והממשק DT שאסור לכם לשנות
- המחלקות Container ו DataStructure שעליכם להשלים
- מחלקה בשם Main עם מספר פונקציות לבדיקת נכונות המבנה. הבדיקות אינן מכסות את כל המקרים ומומלץ להוסיף בדיקות משלכם.
- מחלקה בשם GUI לצורך הבונוס (ראו חלק ג')

מותר לכם (ואף רצוי) לשנות את המחלקה Container ולהוסיף קבצים ומחלקות כרצונכם בהתאם לצרכי המימוש שבחרתם, אך אין לשנות את מה שכבר קיים (כלומר את שם המחלקה, את השדה data ואת המתודה getData())

אתם מקבלים את קובץ המחלקה DataStructure עם מימוש ריק. עליכם להשלים את המחלקה DataStructure כך שתענה על כל המתודות הנדרשות, בזמן שצויין. ניתן להוסיף לה מתודות, בנאים, ושדות כרצונכם. אולם, שימו לב שהבדיקות שלנו יזמנו אך ורק את המתודות והבנאי שצוינו.

6. חלק ב' – הסבר תיאורטי של חלק א':

6.1. תארו בקצרה את המימוש שלכם במסמך מוקלד. הסבירו באילו מבני נתונים השתמשתם ותארו **במילים** את האלגוריתמים שבהם השתמשתם למימוש השיטות של הממשק DT.

6.2. הסבירו **בקצרה** את זמן הריצה של **כל אחת** מן הפעולות.

6.3. ספקו פסאודו-קוד עבור הפונקציה split(int value, Boolean axis), ונתחו את זמן הריצה של הפתרון שהצעתם. ניקוד מלא יתקבל על פתרון שרץ בזמן O(|C|), כאשר |C| מספר הנקודות שבקבוצה הקטנה בחלוקה. פתרון שרץ בזמן O(n) יקבל ניקוד חלקי בלבד, ופתרונות איטיים יותר לא יקבלו ניקוד.

6.4. חשבו והסבירו את זמן הריצה של הפונקציה nearestPair(), (על התשובה להיות אופטימלית ככל האפשר)

6.5. **בונוס (עד 5 נקודות):** הסבירו כיצד ניתן בזמן O(n) לבנות מ DataStructure קיים שני DataStructure, אחד המכיל את כל הנקודות עם ערך X גדול מהחציון (כולל) ואחד המכיל את כל הנקודות עם ערך X קטן מהחציון. על מבני הנתונים החדשים לתמוך כמובן בכל הפונקציות. אופציונלי: הסבירו כיצד ניתן להשתמש (או כיצד השתמשתם) בכך כדי לשפר את זמן הריצה של הפונקציה nearestPair.

על המסמך להיות בפורמט PDF בלבד.

7. חלק ג' – ממשק משתמש גרפי (GUI):

בחלק זה תוכלו לבחון שימוש אפשרי אחר למבנה הנתונים שבניתם.

בקובץ ה-ZIP תמצאו גם מחלקה בשם GUI, שהרצתה תפתח ממשק משתמש. לאחר מימוש חלק א', ניתן להשתמש בממשק זה לבדיקת הקוד שלכם.

האפליקציה יודעת לטעון תמונה (בפורמט JPG) וקובץ TXT המתאר אילו עצמים מופיעים בתמונה. תוך שימוש בפתרון חלק א' - האפליקציה מאפשרת לכם לסמן בצורה גרפית על גבי התמונה תחום מסוים ולהחזיר את העצמים בתמונה שנמצאים בתחום שסימנתם או את מספר העצמים בתחום.

האפליקציה מאוד פשוטה לתפעול. תחילה יש לטעון את התמונה וקובץ ה TXT שנכתב בפורמט מיוחד (ומתואר למטה) המתאים לתמונה המסוימת. לאחר מכן יש לבחור תחום באמצעות לחיצה על העכבר וגרירה על פני התמונה. לאחר עזיבת העכבר הפלט יופיע למטה.

כמו-כן, בחלק מהפונקציות יודפסו הנקודות המתאימות במבנה הנתונים על התמונה (שימו לב שבמצב הסטנדרטי, הנקודות אינן מוצגות על התמונה).

בקובץ ה- zip תוכלו למצוא גם שני סטים המכילים תמונה (בפורמט JPG) וקובץ בפורמט TXT המתאר את העצמים בתמונה. אתם יכולים להשתמש גם בהם כדי לבדוק את נכונות מבנה הנתונים.

כמו כן, אתם יכולים להוסיף סטים של קבצים.

הפורמט של קובץ הנקודות מסוג TXT צריך להיות:

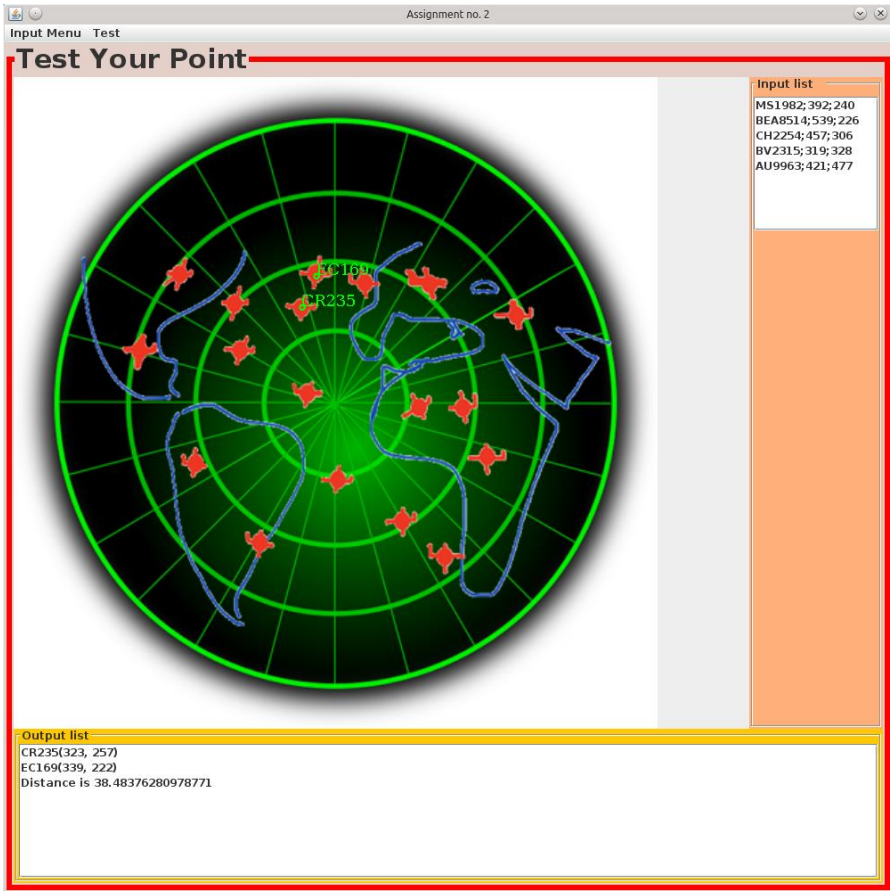
```
<name>;<x-coordinate>;<y-coordinate>
```

לדוגמה:

```
Moshe;30;55
```

אתם יכולים לצרף לעבודה את הקבצים של הסטים שהשתמשתם בהם. הם אינם צריכים בהכרח להתאים לבעית המטוסים. נשקול לתת **בונוס של עד 5 נקודות** לציון העבודה במידה והסטים שתצרפו יהיו מקוריים במיוחד (יש לרשום בקובץ הpdf שצירפתם בונוס וכן תיאור מתומצת שלו. כמו-כן, יש לצרף צילום מסך).

צילום מסך לדוגמה:



הערות חשובות ודרישות הגשה:

1. מומלץ מאוד לקרוא את העבודה מהתחלה עד הסוף לפחות פעמיים לפני שתתחילו לפתור אותה. כדאי לתכנן היטב באילו מבני נתונים להשתמש.
2. אין להשתמש במבני נתונים גנריים הקיימים ב - Java במימוש העבודה. כלומר, עליכם לממש כל מבנה שתרצו בעצמכם. (במחלקת הבדיקה Main אתם רשאים להשתמש במבנים מוכנים של Java, כיוון שאתם לא מגישים את המחלקה הזאת.)
3. ניתן תמיד להניח שהקלט יהיה תקין. לא תתבקשו להוסיף נקודה שכבר קיימת. לא תתבקשו למחוק נקודות ממבנה ריק. לא תקבלו מתכנית הבדיקה שלנו ערך null כפרמטר לאף אחת מהשיטות של הממשק.
4. בקובץ ה- zip יש גם קובץ בשם Main.java שיכול לשמש אתכם לבדיקה ראשונית של העבודה שלכם. לאחר שתסיימו את התכנית אתם יכולים להריץ את ה- main כדי לדעת אם התכנית עובדת כמו שצריך. שימו לב: **הבדיקות אינן מכסות את כל המקרים ומומלץ להוסיף עוד בדיקות משלכם.**
5. את העבודה יש להגיש ב- Submission system.
6. **<u>חשוב: כל הקבצים של העבודה שאתם מגישים צריכים להיות ב- default package</u>** <u>(זה שנוצר אוטומטית ביצירת פרויקט חדש בeclipse). אין ליצור package חדש. package חדש ימנע העלאת הקבצים ל- submission system.</u>
7. עליכם להגיש קובץ מסוג zip בשם assignment2.zip המכיל בתוכו:
   a. תיקיית src ובה קבצי הג'אווה של העבודה, **<u>ללא המחלקות</u>** GUI, Point **<u>ו- Main.</u>**
   b. מסמך PDF המתאר בקצרה את הפתרון ועונה על כל השאלות בחלק 6.
8. סביבת העבודה בה תיבדקנה העבודות הינה JavaSE-1.7
9. עליכם לדאוג כי עבודותיכם יתקמפלו וירוצו בסביבת eclipse תחת גרסאת Java הנזכרת לעיל.
10. עבודות שלא יתקמפלו – יקבלו ציון 0.

11. עבודותיכם יבדקו באמצעות כלי בדיקה אוטומטים הבודקים קורלציה בין עבודות. אין להעתיק! להזכירכם, המחלקה רואה בחומרה רבה העתקות.
12. נרצה לראות קוד מתועד, מתוכנן היטב ויעיל שמייצג הבנה.

בהצלחה!