מטלת מנחה (ממ"ן) 15

הקורס: 20453 - מבוא למדעי המחשב ושפת Java א

חומר הלימוד למטלה: יחידות 5 – 6 **נושא המטלה:** לולאות ומערכים

מספר השאלות: 1 **משקל המטלה:** 5 נקודות

סמסטר: **2019א** **מועד אחרון להגשה:** 5.1.2019

(ת)

במטלה זו אנו משתמשים במחלקות Point ו- City שכתבנו בממ"ן 14.

אתם יכולים להשתמש במחלקות Point ו- City **שכתבתם**, או בקבצים Point.class ו- City.class שאנו שמנו באתר בספר הדיגיטלי של יחידות 5 - 6, בפרק של מטלה 15.

אנא קראו את הכתוב במדריך creating_a _project_and_using_existing_classes שנמצא בלשונית "מדריכי עזר" במשאבי הלמידה בדף הבית של האתר הקורס. כך תדעו איך להשתמש במחלקה שכבר כתובה, וניתנת לכם כקובץ class ללא הקוד.

שמנו באתר טסטר בסיסי לבדיקה ראשונית של המטלה. **חובה** להריץ את המטלה מול הטסטר ולבדוק שאין טעויות קומפילציה.

שאלה 1 - להרצה (100%)

כזכור, בממ"ן 14 עסקנו במדינת ישראל שרוצה למפות את הערים שבשטחה. הפעם נוסיף מחלקה בשם Country.

המחלקה Country מייצגת את המדינה ומכילה את רשימת הערים.

הייצוג נעשה על-ידי מערך ששומר את רשימת הערים. התכונות במחלקה הן:

- שם המדינה String _countryName
- מערך של הערים City [] _cities
- מספר הערים במדינה int _noOfCities

כמו כן קיים קבוע מספרי MAX_NUM_CITIES המציין את המספר המקסימלי של הערים במדינה – 1000.

הערים (כלומר האובייקטים מהמחלקה City) נמצאים במערך ברצף, **ללא "חורים" מתחילת המערך. המערך צריך להישאר כך (ללא חורים) לאחר כל פעולה.**

עליכם לממש ב- Java את המחלקה Country לפי הסעיפים להלן:

1. הגדרת התכונות של המחלקה.

2. בנאי שמקבל את שם המדינה, ומאתחל את תכונות המחלקה כך שהמערך יהיה בגודל מקסימלי.

3. שיטה בוליאנית `addCity` שמוסיפה עיר למדינה. היא מקבלת כפרמטרים שם העיר, מיקום מרכז העיר (שני ממשיים שמייצגים את x ו- y), מיקום התחנה המרכזית (שני ממשיים שמייצגים את x ו- y), מספר התושבים ומספר השכונות בעיר, ומכניסה עיר עם תכונות אלו למערך הערים. אין חשיבות לסדר בו הערים שמורות במערך הערים. השיטה מחזירה true בהצלחה (יש מקום במערך להוסיף עיר) ו- false בכישלון. אפשר להניח שהעיר החדשה לא נמצאת כבר במדינה. כמו כן לא יהיו שתי ערים בעלות שם זהה כשהפרמטרים האחרים שונים. אין צורך לבדוק זאת.

4. בזמן מפקד האוכלוסין רוצים לדעת מה מספר התושבים הכולל במדינה. לשם כך כתבו את השיטה `getNumOfResidents` שמחזירה את מספרם הכולל של התושבים שבמדינה.

5. סוקרי המפקד רוצים לענות על השאלה מהו המרחק הגדול ביותר בין שתי ערים במדינה. כלומר אם ניקח את שתי הערים שהמרחק ביניהם הוא הגדול ביותר (מרחק בין מרכזי הערים) – מה יהיה מרחק זה. כתבו את השיטה `longestDistance` שמחזירה מרחק זה. שימו לב שעליכם להחזיר את המרחק, ולא את הערים. אם מספר הערים במדינה קטן מ-2 יוחזר 0.

6. באותו מפקד אוכלוסין רוצים גם להדפיס את המידע על כל הערים במדינה שנמצאות מצפון לעיר מסוימת (מעל לעיר). יש להתייחס למרכז העיר למטרת החישובים. כתבו את השיטה `citiesNorthOf` שמקבלת כפרמטר שם של עיר כלשהי כמחרוזת תווים, ומחזירה מחרוזת עם פרטים המלאים של כל הערים במדינה שנמצאות מצפון לעיר שהועברה כפרמטר. מידע על כל עיר יופיע בשורה נפרדת. שימו לב להשתמש בשיטות שכבר קיימות ולא לבצע אותן מחדש.

   אם שם העיר לא נמצא במדינה תוחזר המחרוזת:

   "There is no city with the name ..."

   אם אין ערים מצפון לעיר המבוקשת, תוחזר המחרוזת:

   "There are no cities north of …"

   כאשר במקום שלוש הנקודות יופיע השם של העיר.

אם יש ערים מצפון לעיר המבוקשת יש לרשום כותרת ואחריה רשימת הערים.

לדוגמא:

The cities north of Eilat are:

```
City Name: TelAviv
City Center: (10.0,10.0)
Central Station: (8.0,8.0)
Number of Residents: 10000
Number of Neighborhoods: 5

City Name: Jerusalem
City Center: (3.0,20.0)
Central Station: (4.0,18.0)
Number of Residents: 20000
Number of Neighborhoods: 8
```

7. כתבו את השיטה `southernmostCity` שמחזירה את **העיר** הדרומית (הנמוכה) ביותר במדינה. . יש להתייחס למרכז העיר למטרת החישובים . אם אין ערים במדינה יוחזר null.

8. כתבו את השיטה `getCountryName` המחזירה את שם המדינה.

9. כתבו את השיטה `getNumOfCities` המחזירה את מספר הערים במדינה.

10. כתבו את השיטה `getCities` המחזירה מערך של ערים בגודל מספר הערים במדינה ובו עותק של כל אחת מן הערים.

11. כתבו שיטה לאיחוד ערים `unifyCities`. השיטה מקבלת שני שמות של ערים (נניח "city1", "city2"), מאחדת אותן לעיר אחת ומחזירה את העיר המאוחדת. שם העיר המאוחדת יהיה "city1-city2". מספר התושבים בעיר המשותפת הוא סכום מספרי התושבים, מספר השכונות בעיר המשותפת הוא סכום מספרי השכונות, מיקום מרכז העיר החדשה הוא באמצע הדרך בין שני מרכזי הערים, ומיקום התחנה המרכזית המשותפת הוא בתחנה המערבית יותר מבין השתיים (שמאלית יותר). שימו לב שעליכם להסיר מהמערך את העיר הקטנה יותר (במספר התושבים. אם מספר התושבים בשתי הערים זהה, תוסר city2) ואת פרטי העיר החדשה המשותפת יש לשים המשותפת יש לשים במקום בו היתה העיר הגדולה יותר (במספר התושבים). אפשר להניח שהקלט חוקי, כלומר הפרמטרים הם שמות של ערים שקיימות.

12. כתבו את השיטה `toString` המחזירה מחרוזת המכילה את המידע על כל הערים במדינה. שימו לב להשתמש בשיטות שכבר קיימות.

לדוגמא, ישראל עם הערים תל אביב וירושלים:

```
Cities of Israel :
```

```
City Name: TelAviv
City Center: (10.0,10.0)
Central Station: (8.0,8.0)
Number of Residents: 10000
Number of Neighborhoods: 5

City Name: Jerusalem
City Center: (3.0,20.0)
Central Station: (4.0,18.0)
Number of Residents: 20000
Number of Neighborhoods: 8
```

אם אין ערים במדינה יוחזר המחרוזת: "There are no cities in this country."

שימו לב - כאשר משווים בין אובייקטים ובפרט מחרוזות יש להשתמש בשיטה equals ולא ב-== על מנת להשוות בין תוכן האובייקטים, ולא הכתובות שלהם.

שימו לב לא לבצע aliasing במקומות המועדים.

מותר להוסיף שיטות נוספות (פרטיות), לפי ראות עיניכם. אסור להוסיף תכונות.

אתם צריכים לכתוב בעצמכם API למחלקה, לבנאים ולשיטות לפי הנהוג בכתיבת API. כמו כן, עליכם לתעד בתיעוד פנימי כל מה שדורש הבהרה ואינו פשוט.

המימוש אשר תכתבו צריך להיות בהתאם ל- API אשר נמצא כאן להלן. את הערות ה- API אתם צריכים לכתוב בעצמכם.

Constructor Summary
Country(java.lang.String countryName)

Method Summary	
boolean	**addCity**(java.lang.String name, double xCenter, double yCenter, double xStation, double yStation, long numOfResidents, int noOfNeighborhoods)
java.lang.String	**citiesNorthOf**(java.lang.String cityName)
City[]	**getCities**()
java.lang.String	**getCountryName**()
int	**getNumOfCities**()
long	**getNumOfResidents**()
double	**longestDistance**()
City	**southernmostCity**()
java.lang.String	**toString**()
City	**unifyCities**(java.lang.String cityName1, java.lang.String cityName2)

שימו לב,

באתר הקורס תמצאו גם טסטר לבדיקת האיות והפרמטרים של השמות של השיטות במחלקה שאתם צריכים לכתוב. חובה עליכם לבדוק את המחלקה שכתבתם בטסטר זה, ולהגיש אותן רק אם הטסטר עובר קומפילציה. שימו לב שהטסטר לא מכסה את כל האפשרויות, ובפרט לא את מקרי הקצה. הוא רק בודק את השמות של השיטות במחלקה כלומר שגיאות קומפילציה. מאד מומלץ להוסיף לו בדיקות.

הגשה

1. הגשת הממ״ן נעשית בצורה אלקטרונית בלבד, דרך מערכת שליחת המטלות.
2. הקפידו לתעד בתיעוד פנימי וב- API את כל השיטות שיש במחלקות השונות.
3. הקפידו ששמות השיטות יהיו בדיוק כפי שכתוב במטלה. וכן שההדפסות יהיו בדיוק כפי שמופיע במטלה.
4. עליכם להגיש את הקובץ Country.java, עטפו אותו בקובץ zip ושלחו. אין לשלוח קבצים נוספים.

בהצלחה