מטלת מנחה (ממ"ן) 14

הקורס: 20441 - מבוא למדעי המחשב ושפת Java

חומר הלימוד למטלה: יחידה 11 **נושא המטלה:** רשימות מקושרות

מספר השאלות: 2 **משקל המטלה:** 3 נקודות

סמסטר: 2023ב **מועד אחרון להגשה:** 17.6.2023

כזכור, במטלה 12 הגדרנו מחלקה Point המייצגת נקודה במישור, מחלקה Date המייצגת תאריך ומחלקה City המייצגת עיר.

במטלה זו נגדיר מדינה על ידי שימוש ברשימה מקושרת של ערים מהמחלקה City..

שימו לב: חובה עליכם להשתמש במחלקות של Point, Date ו- City ששמנו באתר במטלה 14.

כמו כן, שימו לב : בסוף קובץ זה יש api מפורט של כל השיטות שמתוארות בהמשך. פנו אליו לפני שאתם ממשים כל שיטה, כדי שהמימוש שלכם יהיה בדיוק לפי הממשק הנדרש.

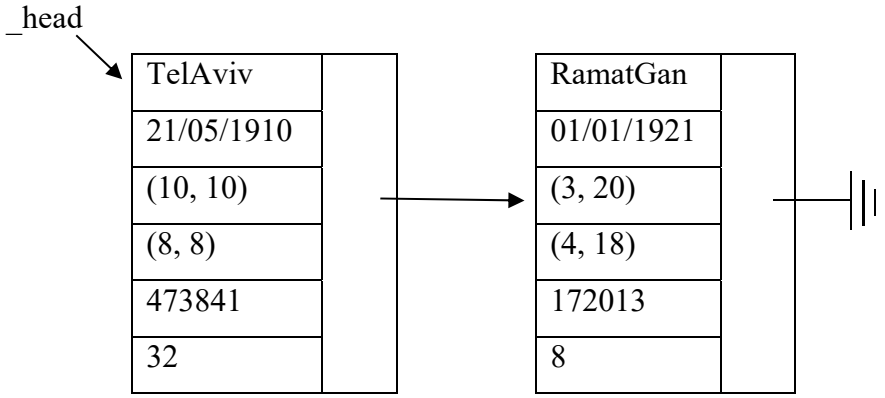
הייצוג נעשה על-ידי רשימה מקושרת ששומרת את רשימת הערים במדינה. אין חשיבות לסדר בין הערים.

אם, לדוגמא, במדינה קיימות הערים תל אביב ורמת גן, שאלו פרטיהן:

```
City name: TelAviv
Date established: 21/05/1910
City center: (10,10)
Central station: (8,8)
Number of residents: 473841
Number of neighborhoods: 32

City name: RamatGan
Date established: 01/01/1921
City center: (3,20)
Central station: (4,18)
Number of residents: 172013
Number of neighborhoods: 8
```

אז הרשימה המייצגת את המדינה היא:



כדי לעשות זאת, עליכם להגדיר שתי מחלקות CityNode ו-Country.

שאלה 1 - להרצה (10%)

המחלקה **CityNode** תייצג עיר אחת במדינה. המחלקה City היא זו שכתבתם במטלה 12, אבל **חובה עליכם להשתמש בקובץ ה- class של המחלקה City שאנחנו שמנו באתר ולא בקובץ שיצרתם ממחלקה שאתם כתבתם.**

לכל אובייקט במחלקה יש שני שדות:

```
1. City _city              //העיר
2. CityNode _next          //מצביע לאיבר הבא
```

למחלקה זו עליכם להגדיר שלושה בנאים:

```
1. public CityNode (City c)
```

בנאי המקבל עיר , שדה ה-_next יאותחל ל-null.

```
2. public CityNode (City c, CityNode next)
```

בנאי המקבל עיר ואיבר נוסף מטיפוס CityNode, ומאתחל את התכונות לפי הפרמטרים.

שימו לב שפה aliasing **למצביע** הוא לא טעות. יש להעתיק את המצביע (next) עצמו ולא עותק של המצביע. שימו לב, בכל הנוגע לעיר עצמה (City) בהחלט כן צריך להימנע מ- aliasing.

```
3. public CityNode (CityNode c)
```

בנאי העתקה. גם כאן שימו לב ש- aliasing למצביע הוא לא טעות. יש להעתיק את המצביע (next) עצמו ולא עותק של המצביע. גם כאן, לא לעשות aliasing לעיר עצמה, אבל כן למצביע.

השיטות במחלקה CityNode הן:

- `public City getCity()` - שיטה המחזירה עותק של העיר שבאיבר.
- `public CityNode getNext()` - שיטה המחזירה מצביע לאיבר הבא. שימו לב שפה aliasing הוא לא טעות. יש להחזיר את המצביע next ולא עותק שלו.
- `public void setCity(City c)` - שיטה המקבלת עיר ומעדכנת את תכונת העיר שבאיבר.
- `public void setNext(CityNode next)` - שיטה המקבלת מצביע ומעדכנת את תכונת המצביע לאיבר הבא. שימו לב שפה aliasing הוא לא טעות. יש לעדכן את המידע (next) עצמו ולא עותק שלו.

שאלה 2 - להרצה (90%)

המחלקה Country מייצגת מדינה.

הייצוג נעשה על-ידי רשימה ששומרת את רשימת הערים שבמדינה.

במחלקה זו מותר להגדיר אך ורק שתי תכונות: _head (חשוב מאוד להקפיד על שם מדויק זה של התכונה) שישמש כראש הרשימה ויצביע לתחילתה, ומחרוזת שמייצגת את שם המדינה. אין להוסיף תכונות מעבר לתכונות אלו.

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

1. הגדרת התכונות של המחלקה.

כאשר אתם מגדירים תכונה עבור ראש הרשימה, עליכם להימנע מלהגדיר אותה כ-private, וזאת כדי להקל על בדיקת הממ"ן שלכם. כלומר, באופן יוצא דופן (ובניגוד למה שעליכם לעשות בכל הקשר אחר), בראש המחלקה שלכם צריך להופיע כך :

```
CityNode _head;
```

ולא כך :

```
private  CityNode _head;
```

2. בנאי שמקבל מחרוזת של שם המדינה ויוצר מדינה ריקה – מאתחל רשימה ריקה ~~את ראש הרשימה להיות null~~, ואת שם המדינה לפי מה שקיבל בפרמטר.
3. שיטה בוליאנית `addCity` שמוסיפה עיר למדינה. היא מקבלת כפרמטרים שם העיר, תאריך הקמת העיר (שלושה מספרים שלמים שמייצגים יום, חודש ושנה),

מיקום מרכז העיר (שני מספרים שלמים שמייצגים את x ו- y), מיקום התחנה המרכזית (שני מספרים שלמים שמייצגים את x ו- y), מספר התושבים ומספר השכונות בעיר, ומכניסה עיר עם תכונות אלו לרשימת הערים. פרמטרים לא תקינים יטופלו לפי הכללים של המחלקה City. **העיר החדשה תוכנס כך שהרשימה תהיה תמיד ממויינת לפי תאריך הייסוד של העיר: בראש הרשימה העיר הותיקה ביותר, בסופה – הצעירה ביותר.** השיטה מחזירה true בהצלחה ו- false בכישלון. אי אפשר להניח שהעיר החדשה לא נמצאת כבר במדינה, אלא צריך לבדוק זאת. אם היא אינה קיימת – היא תתווסף לרשימה במקום המתאים (כאמור, לפי תאריך היווסדה) ויוחזר true . אם העיר כבר קיימת יוחזר false (וכמובן העיר לא תוכנס שוב לרשימה). אם יש ברשימה עיר עם תאריך הקמה זהה לעיר החדשה, העיר שתהיה קרובה יותר לראש הרשימה היא זו ששמה יופיע במילון לפני השם של העיר השניה. להשוואה בין מחרוזות באפשרותכם להשתמש בשיטה compareTo מהמחלקה String. אפשר להניח שלא תהיינה שתי ערים בעלות שם זהה בשום מקרה ~~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 לדעת כמה ערים במדינה נמצאות מצפון לעיר מסוימת (מעל לעיר). יש להתייחס למרכז העיר למטרת החישובים. כתבו את השיטה `numCitiesNorthOf` שמקבלת כפרמטר שם של עיר כלשהי במחרוזת, ומחזירה את מספרן של הערים במדינה שנמצאות מצפון לעיר שהועברה כפרמטר. שימו לב להשתמש בשיטות שכבר קיימות ולא לבצע אותן מחדש.

אם שם העיר לא נמצא במדינה יוחזר הערך 1-.

אם אין ערים מצפון לעיר המבוקשת, יוחזר הערך 0.

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

מדינות במדינה יוחזר null. אם יש יותר מעיר אחת שהיא הכי דרומית תוחזר זו שתאריך הקמתה מוקדם יותר.

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

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

10. כתבו את השיטה `wereCitiesEstablishedBeforeOrAfter` המקבלת שני תאריכים (לא ידוע איזה מהם מוקדם יותר) ומחזירה תשובה האם עיר כלשהי נוסדה ממש לפני המוקדם שבין התאריכים או ממש אחרי המאוחר שבהם.

11. כתבו את השיטה `unifyCities` המבצעת איחוד בין ערים. השיטה מקבלת שני שמות של ערים (נניח "city1", "city2"), מאחדת אותן לעיר אחת ומחזירה את העיר המאוחדת. שם העיר המאוחדת יהיה "city1-city2". תאריך הקמת העיר יהיה המוקדם מבין שני התאריכים של הקמת שתי הערים, 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 מהרשימה את העיר הצעירה יותר (אפשר להניח שאין לשתיהן אותו תאריך ייסוד). **חשוב מאוד : אין להוסיף צומת חדש לרשימה עבור העיר המאוחדת, אלא רק להחליף את הערכים השמורים בצומת של העיר הותיקה בערכים החדשים של העיר המאוחדת.** אפשר להניח שהקלט חוקי, כלומר הפרמטרים הם שמות של ערים שקיימות. אם לשתי התחנות מרכזיות אותו x תיבחר זאת של העיר הצעירה יותר.

12. כתבו את השיטה `establishMaxDiff` המחזירה את ההפרש הגדול ביותר בימים בין תאריכי הקמת שתי ערים כלשהן במדינה. אם במדינה אין ערים יוחזר הערך -1. אם במדינה יש עיר אחת יוחזר הערך 0. שימו לב, אין צורך לכתוב בין אילו ערים קיים ההפרש הגדול ביותר, אלא רק את ערך ההפרש. **שימו לב שהשיטה תהיה יעילה!**

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

לדוגמא, אם במדינה קיימות הערים תל אביב ורמת גן:

```
Cities of Israel:

City name: TelAviv
Date established: 21/05/1910
City center: (10,10)
Central station: (8,8)
Number of residents: 473841
Number of neighborhoods: 32

City name: RamatGan
Date established: 01/01/1921
City center: (3,20)
Central station: (4,18)
Number of residents: 172013
Number of neighborhoods: 8
```

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

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

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

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, int day, int month, int year, int xCenter, int yCenter, int xStation, int yStation, long numOfResidents, int numOfNeighborhoods)
int	establishMaxDiff()
java.lang.String	getCountryName()
int	getNumOfCities()
long	getNumOfResidents()
double	longestDistance()
int	numCitiesNorthOf(java.lang.String cityName)
City	southernmostCity()
java.lang.String	toString()
City	unifyCities(java.lang.String cityName1, java.lang.String cityName2)
boolean	wereCitiesEstablishedBeforeOrAfter(Date date1, Date date2)

שימו לב לכל מקרי השגיאה האפשריים!

דאגו לכך שהקוד יהיה ברור וקריא, וכרגיל, מתועד על-פי כללי javadoc ותיעוד פנימי.

שימו לב,

באתר הקורס תמצאו גם טסטר לבדיקת האיות והפרמטרים של השמות של השיטות והמחלקה שאתם צריכים לכתוב. חובה עליכם לבדוק את המחלקה שכתבתם בטסטר זה, ולהגיש אותה 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 במחלקות כלומר שגיאות קומפילציה. מאד מומלץ להוסיף לו בדיקות.

שימו לב:

1. **אסור להשתמש במחלקות מוכנות כבר של Java.**
2. **מותר ורצוי להשתמש במחלקות שניתנו בהרצאה ונמצאות בחוברת השקפים.**
3. שימו לב לא לכתוב קוד מיותר (שכבר נכתב) אלא להשתמש במחלקות המתאימות.

הגשה

1. הגשת הממ"ן נעשית בצורה אלקטרונית בלבד, דרך מערכת שליחת המטלות.
2. **הקפידו ששמות השיטות והמחלקות יהיו בדיוק לפי הוראות הממ"ן.**
3. את התשובות לשאלות יש להגיש בשני קובצי Java הבאים: CityNode.java, Country.java ארוזים יחד בתוך קובץ zip יחיד. אין לשלוח קבצים נוספים.

ב ה צ ל ח ה