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: 2023א | **מועד אחרון להגשה:** 28.1.2023

כזכור, במטלה 12 הגדרנו מחלקה בשם Date המייצגת תאריך, מחלקה בשם Car המייצגת מכונית ומחלקה בשם Rent המייצגת השכרת רכב.

במטלה זו נכתוב (בין היתר) מחלקה בשם Company המייצגת חברה להשכרת רכבים על ידי שימוש ברשימה מקושרת.

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

המחלקה Company מייצגת חברה להשכרת רכבים.

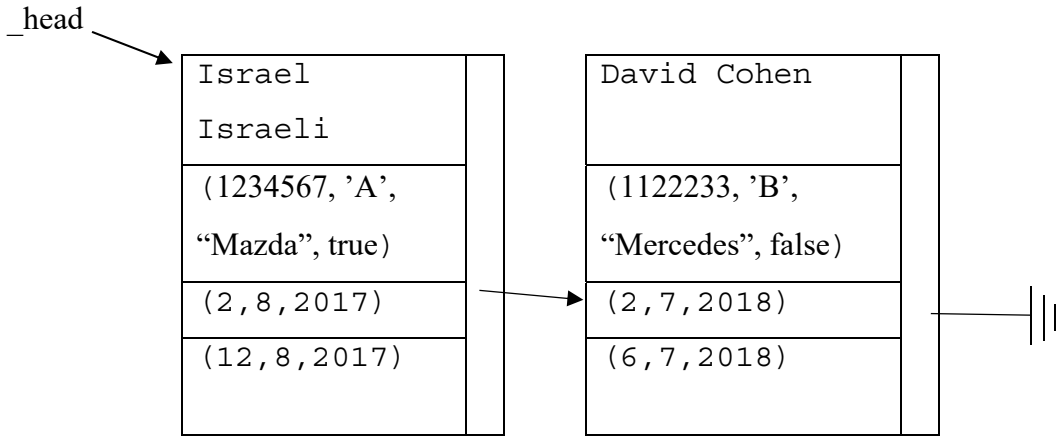
הייצוג נעשה על-ידי רשימה מקושרת ששומרת את רשימת ההשכרות מסוג Rent. **רשימה זו תהיה ממוינת כך שההשכרות יופיעו לפי סדר תאריך קבלת המכונית (pickDate). אם יש יותר מהשכרה אחת עם תאריך קבלה זהה אזי ההשכרה שמשך זמנה ארוך יותר תופיע קודם ברשימה.**

לדוגמא, אם בחברה להשכרת רכב (Company) קיימות רשומות ה-Rent המייצגות השכרת רכב שאלו פרטיהן:

```
String _name: David Cohen
Car _car: (1122233, 'B', "Mercedes", false)
Date _pickDate: (2,7,2018)
Date _returnDate: (6,7,2018)

String _name: Israel Israeli
Car _car: (1234567, 'A', "Mazda", true)
Date _pickDate: (2,8,2017)
Date _returnDate: (12,8,2017)
```

אז הרשימה המייצגת את החברה להשכרה (Company) היא:



כדי לעשות זאת, עליכם להגדיר את שתי המחלקות RentNode ו-Company.

שאלה 1 - להרצה (10%)

המחלקה **RentNode** תייצג השכרה אחת בחברה. המחלקה Rent היא זו שכתבתם במטלה 12 וקובץ ה-class שלה מופיע באתר במטלה 14. **חובה עליכם להשתמש בקובץ ה- class שאנחנו שמנו באתר ולא בקובץ שיצרתם ממחלקה שאתם כתבתם.**

לכל אובייקט במחלקה יש שני שדות:

```
1. Rent _rent                    //אובייקט ההשכרה
2. RentNode  _next               //מצביע לאיבר הבא
```

למחלקה זו עליכם להגדיר שלושה בנאים:

```
1. public RentNode (Rent r)
```

בנאי המקבל השכרה בלבד, שדה ה-_next יאותחל ל-null.

```
2. public RentNode (Rent r, RentNode next)
```

בנאי המקבל השכרה ואיבר נוסף מטיפוס RentNode, ומאתחל את התכונות לפי הפרמטרים.

שימו לב שפה aliasing ~~למצביע~~ **לשדה ה- _next** הוא לא טעות. יש להעתיק את ~~המידע~~ next עצמו ולא לייצר עותק שלו ~~המצביע~~.

```
3. public RentNode (RentNode other)
```

- בנאי העתקה. שימו לב שגם פה aliasing ~~למצביע~~ **לשדה ה- _next** הוא לא טעות. יש להעתיק את ~~המידע~~ next עצמו ולא לייצר עותק שלו ~~המצביע~~.

השיטות במחלקה RentNode הן :

- `public Rent getRent()` - שיטה המחזירה עותק של ההשכרה שבאיבר.
- `public RentNode getNext()` - שיטה המחזירה מצביע לאיבר הבא. שימו לב שפה aliasing הוא לא טעות. יש להחזיר את המצביע next ולא עותק של המצביע.
- `public void setRent(Rent r)` - שיטה המקבלת השכרה ומעדכנת את תכונת ההשכרה שבאיבר.
- `public void setNext( RentNode 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.

שאלה 2 - להרצה (90%)

המחלקה Company מייצגת חברה להשכרת רכבים על ידי שימוש ברשימה מקושרת וכיוון שכך אין כאן מגבלה על מספר ההשכרות המיוצגות ברשימה.

במחלקה זו מותר להגדיר אך ורק תכונה פרטית אחת והיא: ראש הרשימה, שתצביע להתחלת הרשימה. אין להוסיף תכונות מעבר לתכונה זו.

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

1. הגדירו את התכונה של המחלקה .
2. כתבו בנאי היוצר חברה (Company) ריקה – כלומר מאתחל את ראש הרשימה להיות null.
3. כתבו שיטה בוליאנית addRent שמוסיפה השכרה (Rent) לחברה (Company). השיטה מקבלת כפרמטרים את שם המשכיר, מכונית (אובייקט המחלקה Car), אוביקט מטיפוס Date המייצג את תאריך תחילת ההשכרה ואובייקט נוסף מטיפוס המחלקה Date המייצג את תאריך ההחזרה של הרכב המושכר. השיטה מכניסה השכרה (Rent) עם תכונות אלו לרשימת ההשכרות. **שימו לב כי יש חשיבות לסדר בו ההשכרות שמורות ברשימת ההשכרות במחלקה Company. ההשכרות יופיעו לפי סדר תאריך קבלת המכונית (pickDate). אם ישנה יותר מהשכרה אחת עם תאריך קבלה זהה אזי ההשכרה שמשך זמנה ארוך יותר תופיע קודם ברשימה.** אפשר להניח שלא תהיינה שתי השכרות שונות עם תאריך לקיחה (pickDate) וגם תאריך החזרה (returnDate) זהה, אך לא ניתן להניח שההשכרה החדשה לא נמצאת כבר בחברת ההשכרה . צריך לבדוק זאת, ואם היא כבר קיימת (זהות השכרות נקבעת על פי הזהות שהוגדרה במחלקה Rent), השיטה מחזירה false. אם ההשכרה החדשה אינה נמצאת ברשימה, והיא הוספה בהצלחה, השיטה מחזירה true.

4. כתבו שיטה בוליאנית בשם removeRent שמקבלת כפרמטר תאריך d ומסירה מהרשימה את ההשכרה הראשונה שתאריך ההחזרה שלה זהה לתאריך d שהתקבל כפרמטר. אם התבצעה הסרה - השיטה מחזירה true. במקרה ואין השכרה מתאימה, אין לעשות דבר, והשיטה מחזירה false.

5. כתבו את השיטה getNumOfRents המחזירה את מספר ההשכרות בחברה.

6. כתבו שיטה בשם getSumOfPrices המחשבת ומחזירה את הרווח הכולל של כל תקופות ההשכרה המיוצגות ברשימה.

7. כתבו שיטה בשם getSumOfDays המחשבת ומחזירה את מספר ימי ההשכרה הכולל של החברה.

8. כתבו את השיטה averageRent המחזירה את משך ימי השכרה הממוצעים. אם אין השכרות כלל, יש להחזיר 0.

9. כתבו את השיטה lastCarRent המחזירה את המכונית שתאריך החזרה שלה לחברה הוא המאוחר ביותר. אם ישנן כמה השכרות עם תאריך החזרה זהה תוחזר ההשכרה המופיעה ראשונה ברשימה. אם אין השכרות כלל, יש להחזיר null.

10. כתבו את השיטה longestRent המחזירה את ההשכרה (Rent) בה מספר ימי ההשכרה הוא מקסימלי. במקרה של מספר השכרות עם מספר מקסימלי זהה, יש להחזיר את ההשכרה הראשונה. אם אין השכרות כלל, יש להחזיר null.

11. כתבו את השיטה mostCommonRate המחזירה את הדירוג הפופולרי בקרב כלל ההשכרות. במקרה של מספר זהה בין יותר מדירוג אחד, יש להחזיר את הדירוג הגבוה יותר. להכירכם : B נחשב דירוג גבוה יותר מאשר A. אם אין השכרות כלל, יש להחזיר את התו 'N'.

12. כתבו את השיטה includes המקבלת רשימת השכרות נוספת, ובודקת האם הרשימה שהתקבלה כפרמטר לשיטה מוכלת לחלוטין ברשימה עליה הופעלה השיטה. כלומר השיטה תחזיר true אם ורק אם כל אחת ואחת מההשכרות המופיעות ברשימה שניתנה כפרמטר, מופיעה באופן זהה (כפי שהוגדר במחלקה Rent) גם ברשימה עליה הופעלה השיטה. אם הרשימה שהתקבלה כפרמטר היא ריקה אזי השיטה תחזיר true. אם הרשימה עליה מופעלת השיטה היא ריקה אז יוחזר false (אלא אם כן גם הרשימה שהתקבלה כפרמטר ריקה) . על מנת לפשט את הפתרון של שיטה זו, ניתן להניח שאין שתי השכרות שונות, שכל אחת מהן מרשימה אחרת, שתאריך הלקיחה (pickDate) שלהן זהה וכן תאריך ההחזרה (returnDate) שלהן זהה גם כן.

13. כתבו את השיטה merge הממזגת שתי רשימות השכרות מסוג Company. השיטה מקבלת רשימת השכרות, ומכניסה את כל ההשכרות הנמצאות בה לתוך הרשימה עליה הופעלה השיטה. **תוך שמירה על המיון לפי סדר תאריך קבלת המכונית** כפי שתואר

לעיל. להזכירכם, הרשימה איננה יכולה להכיל השכרה "כפולה" כלומר, על כל השכרה להופיע פעם אחת בלבד ברשימה. על מנת לפשט את הפתרון של שיטה זו, ניתן להניח שאין שתי השכרות שונות, שכל אחת מהן מרשימה אחרת, שתאריך הלקיחה (pickDate) שלהן זהה וכן תאריך ההחזרה (returnDate) שלהן זהה גם כן.

14. ~~כתבו את השיטה unifyRents הדואגת לכך שלא יהיו חפיפות בין רשומות השכרה ברשימה המיוצגת במחלקה Company. במידה שישנן רשומות חופפות הן יאוחדו לרשומה אחת על פי העקרונות המנחים שהוגדרו בעניין חפיפה במחלקה Rent.~~ **השיטה הזו מבוטלת ואין צורך לכתוב אותה.**

15. השיטה toString המחזירה מחרוזת תווים המתארת את כל ההשכרות הקיימות בחברה. המחרוזת צריכה להיות **בדיוק** בפורמט הבא:

```
The company has 3 rents:
Name:Ruthi From:10/03/2022 To:14/03/2022 Type:A Days:4 Price:400
Name:Lior From:11/03/2022 To:18/03/2022 Type:C Days:7 Price:1134
Name:Rama From:30/10/2022 To:12/11/2022 Type:B Days:13 Price:1845
```

אם אין השכרות, השיטה תחזיר מחרוזת **בדיוק** בפורמט הבא (כולל הנקודה):

```
The company has 0 rents.
```

מותר לפתור המטלה בעזרת לולאות או בעזרת שימוש ברקורסיה.

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

יש להימנע מ aliasing פרט למקומות בהם נכתב אחרת.

שימו לב לא לכתוב קוד מיותר (שכבר נכתב) אלא להשתמש במחלקות המתאימות.

כאשר משווים בין אובייקטים ובפרט מחרוזות יש להשתמש בשיטה equals ולא ב- == על מנת להשוות בין תוכן האובייקטים, ולא בין 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	
Company()	
Method Summary	
boolean	addRent(java.lang.String name, Car car, Date pick, Date ret)
boolean	removeRent(Date ret)
int	getNumOfRents()
int	getSumOfPrices()
int	getSumOfDays()
double	averageRent()
Car	lastCarRent()
Rent	longestRent()
char	mostCommonRate()
boolean	includes(Company other)
void	merge(Company other)
~~void~~	~~unifyRents()~~ השיטה מבוטלת
java.lang.String	toString()

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.**

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: RentNode.java, Company.java ארוזים יחד בתוך קובץ zip יחיד. אין לשלוח קבצים נוספים.

ב ה צ ל ח ה