בס"ד

מיני פרוייקט במערכות חלונות - תרגיל מספר 2

מטרת התרגיל:

- היכרות עם Object Members כדוגמת: Fields , Methods , Properties
- מימוש של collections שונים.
- תכנון נכון של מחלקות.
- מימוש הממשקים , IComparable, IEnumerator IEnumerable
- בנוסף הכרת מושגים חדשים כגון: , Casting Operator Indexers

הנחיות לביצוע התרגיל והגשתו

✓ העבודה תתבצע באותם הזוגות כמו בתרגילים הקודמים

✓ חובה להשתמש במערכת ניהול גרסאות Git ומאגר מרוחק משותף של GitHub – חובה להשתמש באותו המאגר כמו התרגילים הקודמים

✓ פרויקט התרגיל ייפתח באותו ה-Solution יחד עם התרגילים הקודמים

✓ יש להגיש במודל קישור על פי ההנחיות בקובץ "הגשת מטלות בקורס מיני פרויקט במערכות חלונות 153007"!

✓ נא להקפיד על פורמט זה על מנת למנוע מצב של אי קבלת ציון על תרגיל מסוים

המטלה

בתרגיל זה ניצור חלק של מערכת מידע על קווי אוטובוס וזמני ההגעה לתחנות

1. הגדירו מחלקה שתייצג ***תחנת אוטובוס***
תחנת אוטובוס מכילה *לפחות* את הנתונים הבאים:
   - *קוד תחנה, מספר ייחודי בן 6 ספרות לכל היותר*
   - *מיקום התחנה*
   הערה: את המיקום יש לשמור באמצעות 2 ערכים:
     - *קו רוחב Latitude – מספר ממשי בתחום [-90,90]*
     - *קו אורך Longitude – מספר ממשי בתחום [-180,180]*
   *מומלץ **להגריל** ערכים שנמצאים בתחומי מדינת ישראל,*
     - *קו רוחב בתחום [31,33.3]*
     - *קו אורך בתחום [34.3,35.5]*
   בנוסף, ניתן להגדיר גם את כתובת התחנה. כתובת התחנה יכולה להיות ריקה.
   אין חובה להשתמש במיקומים של תחנות אוטובוס אמיתיות.

שם התכונה	דוגמא / אפשרויות	תיאור התכונה
BusStationKey	765432	קוד תחנה, מספר מזהה ייחודי בן עד 6 ספרות
Latitude	31.234567	קו רוחב, מספר ממשי בין 90- ל-90
Longitude	34.56789	קו אורך, מספר ממשי בין 180- ל-180
	"רח' פלוני אלמוני 12, תל חורף"	כתובת התחנה, מחרוזת

יש לדרוס (override) את מתודת ToString – כך שתציג את תכונות התחנה.
למשל ע"פ השדות שמופיעים בדוגמא:

```
Bus Station Code: 765432, 31.234567°N 34.56789°E
```

.2 הגדירו מחלקה שתייצג **תחנת קו אוטובוס**
תחנת קו אוטובוס מכילה את כל נתוני תחנת האוטובוס כנ"ל ובנוסף *לפחות* את הנתונים הבאים:

- *מרחק מתחנת קו אוטובוס הקודמת*
- *זמן נסיעה מתחנת קו אוטובוס הקודמת*

ניתן לבנות תחנת קו אוטובוס על בסיס אובייקט של תחנת אוטובוס ונתונים הנ"ל.

.3 הגדירו מחלקה שתייצג **קו אוטובוס** בודד
קו אוטובוס מוגדר כמסלול של תחנות קו אוטובוס שונות. ניתן לייצג מסלול כרשימה של תחנות קו אוטובוס. ניתן להשתמש במבנה נתונים של רשימה גנרית List.

בנוסף לקו אוטובוס יש:

- *מספר המזהה אותו*
- *תחנת התחלה ותחנת סיום*
- *קו אוטובוס יכול להיות משויך לאזור מסוים מתוך רשימת אזורים מוגדרת או להיות חוצה אזורים ובמקרה כזה יוגדר באזור כללי*
  הערה: *הגדירו enum עבור האזורים השונים כולל האזור הכללי*

תיאור התכונה	דוגמא / אפשרויות	שם התכונה
מספר קו	39	BusLine
תחנת מוצא		FirstStation
תחנת סופית		LastStation
אזור בארץ	General\North\South\Center\Jerusalem\...	Area
רשימות התחנות		Stations

הערה: תחנות קו ראשונה ואחרונה במסלול חייבות להיות תואמות לתחנת מוצא ותחנה סופית בהתאם

כמו כן תכלול המחלקה את המתודות הבאות:

1. דריסה של ToString המציגה עבור קו אוטובוס את מספר הקו, את האזור בו הקו פועל ו~~שתי~~-רשימות מספרי התחנות ~~במסלולי הקו בהלוך ובחזור~~.

2. הוספת/ גריעת תחנה ממסלול הקו
   הערה: יש לשים לב שניתן להוסיף תחנה גם בתחילה גם באמצע וגם בסוף המסלול!

3. מתודה בוליאנית שבודקת האם תחנה מסוימת נמצאת במסלול הקו

4. מתודה שתחזיר את המרחק בין 2 תחנות שנמצאות על הקו (התחנות לא בהכרח צמודות)

5. מתודה שתחזיר את זמן הנסיעה בין 2 תחנות שנמצאות על הקו (התחנות לא בהכרח צמודות)

6. מתודה שמחזירה **תת-מסלול** של הקו, המתודה מקבלת 2 תחנות ומחזירה אובייקט מסוג קו אוטובוס שייצג בפועל את מקטע הקו בין שתי התחנות (כולל)

7. נוסע שנמצא בתחנת מסוימת מעוניין לבחור בין 2 קווי אוטובוס שמגיעים לתחנת היעד שלו באמצעות השוואת זמני הנסיעה בין תחנות כלשהן. לכם יש לאפשר השוואה בין שני אובייקטים של קו אוטובוס ע"י השוואת זמן כולל של נסיעה בקו ולכן יש לממש את הממשק **IComparable**.

4. הגדירו מחלקה שתטפל באוסף של קווי אוטובוס.
המחלקה תכלול אוסף קוי אוטובוס.
באוסף יכול להופיע אותו קו פעמיים (לכל היותר) כאשר תחנות המוצא והסופית שלהם הפוכות (זאת אומרת קוי הלוך וחזור).
שימו לב שהמרחק וזמן הנסיעה בין אותן שתי תחנות אבל בכיוונים שונים עשויים להיות שונים, מכיוון שמסלול הנסיעה וגורמים מעכבים יכולים להיות שונים (למשל רחובות חד סטריים, כיוונוני רמזורים, וכו'). כמו כן התחנות ברשימות הלוך וחזור עשויות להיות שונות לפעמים בגלל אילוצים תעבורתיים ואחרים.

המחלקה תכלול את תבנית איטרטור (תממש ממשק **IEnumerable**). כמובן הקוד שמשתמש במחלקה ייבנה תוך התחשבות בדרישה הזו.

המחלקה תבטיח את הביצוע של הפעולות הבאות:

1. הוספת / גריעת קו לאוסף, תוך בדיקת הדרישות הנ"ל

2. מתודה שמקבלת מספר מזהה (קוד) של תחנת אוטובוס ומחזירה את רשימת הקווים העוברים בתחנה זו. במידה ואין קווים שעוברים בתחנה תיזרק חריגה.

3. מתודה שמחזירה רשימת כל הקוים הממוינת לפי משך הנסיעה הכולל, מהקצר לארוך (ראה גם סעיף 7 ברשימת המתודות של המחלקה קו אוטובוס)

4. Indexer המקבל מספר קו ומחזיר את המופע. אם לא קיים קו כזה תיזרק חריגה.

5. בתוכנית הראשית יש לאתחל את המחלקה של אוסף הקווים עם לפחות 10 קווי אוטובוס, וכן להשתמש בלפחות 40 תחנות אוטובוס שונות עבור הקווים השונים.

- אין צורך שכל הקווים יעברו בכל התחנות.
- יש לוודא שבכל תחנה יעבור לפחות קו אוטובוס אחד
- לפחות ב-10 תחנות יעבור יותר מקו אוטובוס אחד

התוכנית הראשית תציג למשתמש תפריט שבו לפחות הפעולות הבאות:

1. אפשרויות הוספה:
   - קו אוטובוס חדש
   - הוספת תחנה לקו אוטובוס

2. אפשרויות מחיקה:
   - גריעת קו אוטובוס
   - מחיקת תחנה ממסלול קו אוטובוס

3. אפשרויות חיפוש:
   - קווים שעוברים בתחנה ע"פ מספר תחנה
   - הדפסת האפשרויות לנסיעה בין 2 תחנות, ללא החלפת אוטובוס
     יש לקלוט תחנת מוצא ותחנת יעד ולהחזיר את התוצאות הממוינות לפי זמן הנסיעה.

4. אפשרויות הדפסה:
   - כל קווי האוטובוס במערכת
   - רשימת כל התחנות ומספרי הקווים שעוברים דרכם

5. יציאה

<u>הערות:</u>

1. אפשר אך אין חובה לשמור את התחנות הקיימות במבנה נתונים נפרד
2. יש לוודא שלא קיימות 2 תחנות עם אותו מספר מזהה במיקומים שונים.
3. יש לוודא שבכל קו אוטובוס יש לפחות 2 תחנות.
4. במקרה של קלט לא תקין / לא קיים אובייקט מבוקש, יש לזרוק חריגה מתאימה.

הקפידו על קוד קריא וקצר, ושדות רק לפי הנדרש.
ניתן להוסיף פונקציות או תכונות (properties) בהתאם לצורך.

בעניין החריגות:

- איפה שמתאים ניתן להשתמש בחריגות מתאימות שכבר קיימות בדוט-נט
- יש ליצור חריגות מותאמות איפה שלא נמצאו חריגות מתאימות בדוט-נט. כמובן הגדירו וממשו את החריגות הומותאמות
- אסור לזרוק חריגה מסוג **Exception**

לא לשכוח לתעד את המחלקות השונות!

בהצלחה רבה!