

## תרגיל מס' 3 – מערכת לניהול מוסך

### מטרות

- הטמעה של עבודה עם מחלקות, הורשה ופולימורפיזם
- שימוש ב- Collections
- מימוש enums
- פיתוח ושימוש ב- Dll (אסמבלי) חיצוני
- עבודה עם מספר פרויקטים
- עבודה עם Exceptions

### ידע נדרש

- תכנות מונחה עצמים תוך שימוש בהורשה ופולימורפיזם ב- C#
- שימוש ב- Collections
- פיתוח ושימוש ב- Dll (אסמבלי) חיצוני
- עבודה עם מספר פרויקטים
- עבודה עם Exceptions
- הירשה מ- Object

### התרגיל

המטרה הסופית: מערכת קטנה ש"מנהלת" מוסך.  
המערכת תדע לנהל מוסך שמטפל כיום בחמישה סוגי רכבים –

- אופנוע רגיל
  - (2 גלגלים עם לחץ אוויר מקסימאלי 32, דלק מסוג Octan98, 6.2 ליטר טנק דלק)
- אופנוע חשמלי
  - (2 גלגלים עם לחץ אוויר מקסימאלי 32, זמן מצבר מקסימאלי – 2.9 שעות)
- מכונית רגילה
  - (5 גלגלים עם לחץ אוויר מקסימאלי 34, דלק מסוג Octan95, 52 ליטר טנק דלק)
- מכונית חשמלית
  - (5 גלגלים עם לחץ אוויר מקסימאלי 34, זמן מצבר מקסימאלי – 5.4 שעות)
- משאית
  - (14 גלגלים עם לחץ אוויר מקסימאלי 29, דלק מסוג Soler, 125 ליטר טנק דלק)

לכל כלי רכב יש את התכונות:

- שם דגם (string)
- מספר רישוי (string)
- אחוז האנרגיה שנותרה במקור האנרגיה שלו (בשביל מד הדלק/חשמל) (float)
- אוסף של גלגלים

לכל גלגל יש את התכונות הבאות:

- שם יצרן (string)
  - לחץ אוויר נוכחי (float)
  - לחץ אוויר מקסימאלי שקבע היצרן (float)
- פעולת ניפוח (מתודה שמקבלת נתון לגבי כמה אוויר להוסיף לגלגל, ומשנה את מצב לחץ האוויר אם הוא לא חורג מהמקסימום)

לאופנוע (רגיל/חשמלי), בנוסף לתכונות של רכב, יש גם את התכונות:

- סוג רישיון: A1, A2, B1, B2
- נפח מנוע בסמ"ק (int)

למכונית (רגילה/חשמלית), בנוסף לתכונות של רכב, יש גם את התכונות:

- צבע (הצבעים האפשריים הם: כחול, שחור, לבן, אפור)
- כמות דלתות (2, 3, 4, או 5)

למשאית, בנוסף לתכונות של רכב, יש גם את התכונות:

- האם מוביל חומרים בקירור (bool)
- נפח מטען - float

**בכלי רכב שעובד על דלק ניתן למצוא את המידע הבא ולהפעיל את הפעולות הבאות:**

- סוג הדלק (Soler, Octan95, Octan96, Octan98)
- כמות הדלק הנוכחית בליטרים (float)
- כמות הדלק המקסימאלית בליטרים (float)
- פעולת תדלוק (מתודה שמקבלת כמות ליטרים להוספה וסוג דלק, ומשנה את מצב הדלק במידה וסוג הדלק מתאים ואין חריגה מגודל הטנק)

**בכלי רכב חשמלי ניתן למצוא את המידע הבא ולהפעיל את הפעולות הבאות:**

- זמן מצבר שנותר בשעות (float)
- זמן מצבר מקסימאלי בשעות (float)
- פעולת טעינת מצבר (מתודה שמקבלת נתון שהוא מספר שעות להוסיף למצבר ו"טוענת" את המצבר בהתאמה כל עוד מספר השעות לא חורג מהמקסימום)

**על כל רכב שנמצא במוסך יש לשמור את הנתונים הבאים:**

- שם בעלים (string)
- טלפון בעלים (string)
- מצב הרכב במוסך (המצבים האפשריים: בתיקון, תוקן, שולם)
- כל רכב שנכנס למוסך מצבו ההתחלתי הוא 'בתיקון'.

### **המערכת תספק את הפונקציונאליות הבאה למשתמש בה:**

1. "להכניס" רכב חדש למוסך -

המשתמש מזין את מספר הרישוי של הרכב שמעוניינים להכניס למוסך. אם מדובר ברכב שכבר נמצא במוסך (עפ"י מספר רישוי), המערכת תוציא הודעה מתאימה ותשתמש ברכב שכבר נמצא במוסך (ותעביר את מצב הרכב ל- "בתיקון"). אם לא -

המשתמש בוחר את סוג הרכב שהוא מעוניין "להכניס למוסך", ואז מזין את "מצבו" (כמות הדלק שיש בו או מצב המצבר, כמות האוויר בגלגלים, צבע (אם מדובר במכונית), סוג רישיון (אם מדובר באופנוע), האם מסיע בקירור (אם מדובר במשאית), וכיו"ב. \* כדי להקל על המשתמש, ניתן לאפשר לו להזין מצב זהה לכל הגלגלים בפעם אחת (במקום לבקש ממנו להזין את המצב של כל אחד מהגלגלים בנפרד). \* כדי להקל על המימוש (למרות שזה מקשה על המשתמש), ניתן לממש באופן שהודעה על קלט/ים לא תקינים תוצג רק אחרי שהמשתמש הזין את כל הקלטים, ולא אחרי הקלט השגוי.

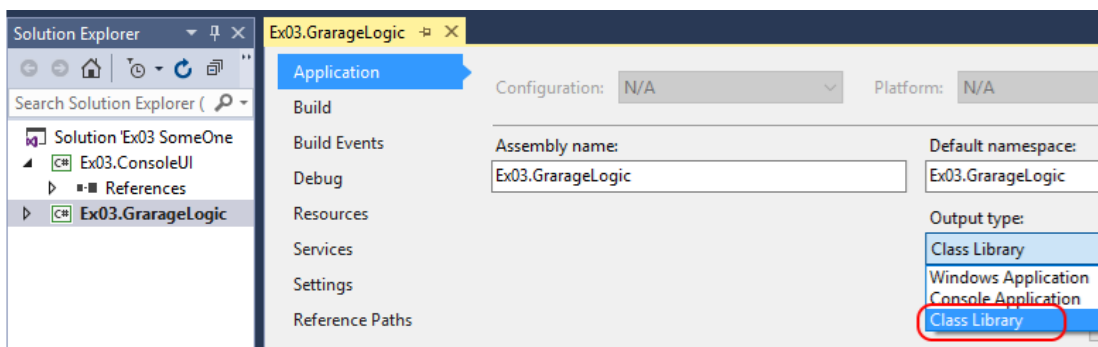
2. להציג למשתמש את רשימת את מספרי הרישוי של הרכבים במוסך, עם אפשרות לסינון לפי המצב שלהם במוסך.

3. לשנות מצב של רכב במוסך (הנתונים המבוקשים מהמשתמש הם מספר רישוי והמצב החדש).
4. לנפח אוויר בגלגלים של רכב למקסימום (המשתמש מזין מספר רישוי)
5. לתדלק רכב שמונע ע"י דלק (המשתמש מזין מספר רישוי, סוג דלק למילוי, כמות למילוי)
6. להטעין רכב חשמלי (המשתמש מזין הנתונים הם מספר רישוי, כמות דקות להטענה)
7. להציג נתונים מלאים של רכב לפי מספר רישוי (מספר רישוי, שם דגם, שם בעלים, מצב במוסך, פירוט הגלגלים (לחץ אוויר ויצרן), מצב דלק + סוג דלק / מצב מצבר, ושאר הפרטים הרלוונטיים לסוג הרכב הספציפי)

לצורך פיתוח המערכת, יש ליצור Solution שיכיל שני פרויקטים:

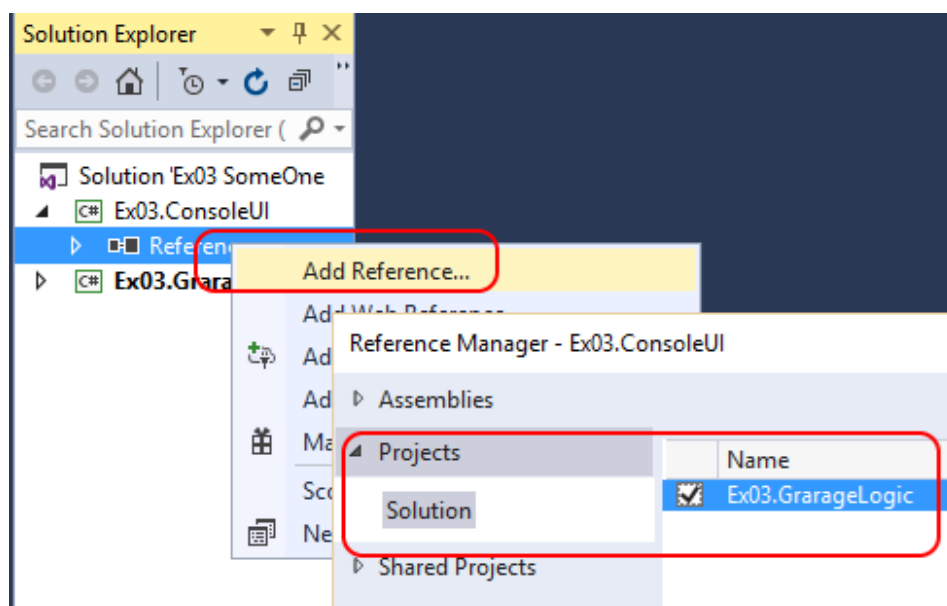
### 1. Ex03.GarageLogic

זהו פרויקט שמייצר dll (אסמבלי שאינו PE) שמכיל רק את מודל האובייקטים והשכבה הלוגית של המערכת (השכבה שנרצה לעשות בה שימוש חוזר גם אם נפתח ממשק משתמש אחר מאשר Console).  
השכבה הזו אינה יודעת להדפיס למסך או לקלוט נתונים מקונסול.  
כדי לייצר dll יש ללכת למאפיינים (Properties) של הפרויקט ולבחור באפשרות:  
Output type: Class Library



### 2. Ex03.ConsoleUI

זהו פרויקט שמייצר את ה- exe ובו יש מימוש של ממשק המשתמש במערכת לניהול מוסך בקונסול. מערכת זו עושה שימוש במודל שמומש בפרויקט הראשון ע"י reference לפרויקט הראשון:



## הערות:

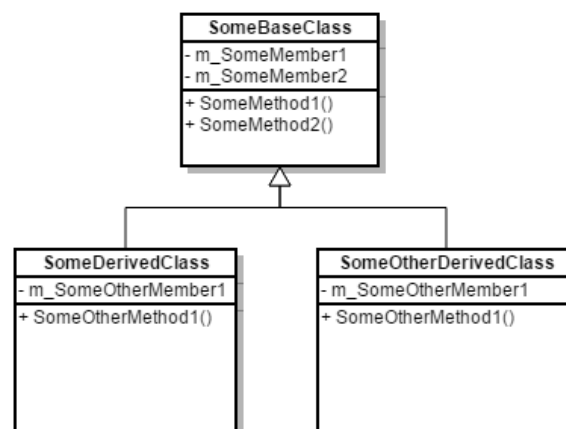
1. יש לתכנן את המערכת בצורה הטובה ביותר של קוד מבחינת חלוקה למחלקות ולמתודות, הורשה ופולימורפיזם.
2. יש להפריד בין ממשק המשתמש והשכבה הלוגית אשר מנהלת את מודל האובייקטים ואת הישיות הלוגיות של המערכת (מחלקה כמו Vehicle לא אמורה להכיר את המחלקה Console, גם לא באופן עקיף).  
יש לנסות ולחשוב על פתרונות לצורך הפרדה זו.
3. יש למקם את הקוד שמייצר אובייקטים (new) של מחלקות הרכבים, ורק אותו, ברכיב קוד מבודד (מחלקה נפרדת) בחלק הלוגי של המערכת (ברכיב זה מותר גם לשמור את רשימת סוגי הרכבים הנתמכים כרגע במערכת). רכיב זה אינו פונה למשתמש לא באופן ישיר ולא באופן עקיף.
4. יש לכתוב את המערכת באופן כזה שהוספה של סוג רכב חדש (נניח "טרקטור") תוכל להתבצע **ללא שינוי בקוד** (למעט תוספת מינימאלית במחלקה מסעיף 3 שמייצרת אובייקטים).
5. יש להימנע משכפול קוד/לוגיקה.
6. חובה להשתמש ב:

- List<T> ו/או Dictionary<K,V>
- Enum
- String formatting
- FormatException (תיזרק במקרה של קלט לא מתאים ברמת ה-Parsing).
- ArgumentException (תיזרק במקרה של קלט לא מתאים מבחינה לוגית, למשל בחירה למילוי דלק מהסוג הלא נכון)
- ValueOutOfRangeException – **מחלקה שאתם תכתבו** (תיזרק במקרה של קלט לא מתאים מבחינה של טווח ערכים רצוי, למשל מילוי דלק מעל המותר).
- זוהי מחלקה שיורשת מהמחלקה Exception ומחזיקה את השדות הנוספים הבאים:
  - float MaxValue
  - float MinValue

## הגשה:

**יש לצרף להגשה בתוך התיקיה קובץ word ובו:**

- שמות המגשים + ת"ז
- רשימת הטיפוסים שהגדרתם (class/enum) עם הסבר קצרצר על כל אחד מהם.
- דיאגרמה שמציגה את היררכיות הירושה והקשר בין הרכיבים. לדוגמא:



- שם הקובץ יהיה כשם קובץ ה- solution (עם סיומת doc/docx).

בהצלחה!