**שמות המגישים:** איתי שיף 209280247 , אלעד משה 316224732

**הסבר קצרצר על כל class/enum שהגדרנו**

Namespace GarageLogic:

**Garage:** מחלקה המכילה מידע לגבי המוסך – רשימה של סוגי רכבים שניתן להכניס למוסך (עם מידע של נתונים יבשים שהמשתמש לא אמור להכניס לגביהם כמו כמות גלגלים, וגם מידע המשמש לתקינות קלט), רשימה של כלי הרכב הנוכחיים הנמצאים במוסך.

במחלקה זו מתודות שמפעילות את כל המתרחש במוסך, כלומר כל ה- execute של התפריט מתבצע כאן.

**Vehicle:** מחלקה המכילה מידע על סוג כלי רכב כללי, בתוכה יש גם 2 nested class: **VehicleOwnerData** שבתוכה את פרטי שם הבעלים והפלאפון שלו של אותו סוג רכב, **Wheel** שבתוכה יש פרטים על הגלגלים של אותו סוג רכב: שם היצרן, לחץ אוויר נוכחי, מקסימום לחץ אוויר אפשרי. במחלקה Vehicle נשמר מידע על שם מודל הרכב, מספר רישוי, רשימת הגלגלים ופרטים עליהם, פרטי הבעלים, Specificiations כלומר פרטים רלוונטיים בהתאם לסוג הרכב הספציפי(הסבר בהמשך), ומצב תיקון נוכחי במוסך. כאשר משווים בין אובייקטים של Vehicle ההשוואה מתבצעת באמצעות מספר הרישוי שלהם.

**GasolineFueledVehicle:** מחלקה היורשת מVehicle- ואותה יוצרים לכל סוג כלי רכב המשתמש בדלק. היא מכילה מידע על איזה סוג דלק יש להשתמש בו בהתאם לאותו סוג רכב. כמות דלק נוכחית בליטרים וכמות דלק מקסימלית בליטרים.

**ElectricVehicle:** מחלקה היורשת מVehicle- ואותה יוצרים לכל סוג כלי רכב חשמלי המשתמש במצבר. היא מכילה מידע על כמה זמן מצבר נותר לה בשעות וזמן מצבר מקסימלי בשעות.

**Specifications:** מחלקה אבסטרקטית השומרת את שם סוג הרכב שנשמר. בעלת מתודה אבסטרקטית InitSpecifications הנועדה לוודא שכל מי שיורש ממנה יממש את התכונות הפרטיות שלו בהתאם ורק באופן זה יווצרו אובייקטים בצורה תקינה לפי הקלט המשתמש.

**BikeSpecficiations / CarSpecifications / TruckSpecifications:** מחלקה זו יורשת מ-Specifications ושומרת שם את שם סוג הרכב שנשמר. במחלקות אלו יש פרטים הרלוונטים אך ורק לאותו סוג כלי רכב ספציפי, כלומר תכונות שאין אותם לסוג רכב כללי(Vehicle). לדוגמה רק על Car יש מידע על כמות הדלתות שלו ואיזה סוג צבע הוא.

**ValueOutOfRangeException:** מחלקה היורשת מException- השומרת מינימום ומקסימום ומשתמשים בה לזריקת Exception כאשר ניתן ערך שאינו בטווח הרצוי.

**NoSuchVehicleException:** מחלקה היורשת מ-Exception, משתמשים בה לזריקת Exception כאשר מחפשים כלי רכב כלשהו לפי מספר רישוי והוא לא נמצא במוסך.

**VehicleFactory:** מחלקה המייצרת אובייקטים חדשים של מחלקות כלי הרכבים הסוגים בצורה דינמית עם Reflection. בתוכה יש מחלקה בשם **VehicleTypes** שהיא מנהלת את כל סוגי הרכבים האפשריים שיכולים להיות במוסך עם פרטים יבשים שהמשתמש לא אמור להכניס וגם פרטים המשמשים לתקינות קלט.

**VehicleUtils:** מחלקת עזר לכלי רכב. שם מתבצע הקריאה באמצעות Reflection של סוגי כלי הרכב האפשריים שיכולים להיות במוסך ובדיקת תקינויות קלט של פרמטרים השונים הקשורים לכלי רכב.

Namespace UI:

**GarageMenuExecutorUI:** המחלקה המבצעת את כל המתרחש במוסך מבחינת ה-UI.

**GarageMenuForm:** מחלקה המציגה את תפריט המוסך ומחזירה קלט תקין של המשתמש לאחר בחירה מוצלחת שלו. בתוכה יש enum eGarageMenuOption שהוא מרכז את כל סוגי האופציות השונות שניתן לבחור בתפריט המוסך, מטרת מחלקה זו להחזיר eGarageMenuOption לאחר קליטת קלט תקין מהמשתמש בבחירת אופציה התפריט.

**LicensePlateForm:** מחלקה המציגה למשתמש להכניס מספר רישוי. מטרת מחלקה זו להחזיר מספר רישוי לאחר שהמשתמש הכניס קלט תקין.

**OwnerDataForm:** מחלקה המציגה למשתמש להכניס את פרטי הבעלים של אותו סוג רכב. מטרת מחלקה זו להחזיר OwnerData לאחר שהמשתמש הכניס קלט תקין.

**RemainingEnergyOrFuelForm:** מחלקה המציגה למשתמש אופציה לבחור כמה דלק או זמן מצבר נותר לו בהתאם לסוג כלי הרכב הספציפי שהוא בחר. מטרת מחלקה זו להחזיר float מספר תקין בטווח עד המקסימום בהתאם לסוג כלי הרכב שנבחר.

**SpecificationForm:** מחלקה שמטרתה הוא להחזיר את Specifications עם פולימופוריזם בהתאם לסוג כלי הרכב שהמשתמש בחר, לאחר שהמשתמש הזין קלט תקין.

**VehicleTypeForm:** מחלקה המציגה למשתמש את אופציות סוגי כלי הרכב השונים שניתן לבחור(כאשר ברצונו להכניס כלי רכב חדש למוסך). מטרת מחלקה זו להחזיר את סוג כלי הרכב שהמשתמש בחר לאחר קליטת קלט תקין.

**WheelsForm:** מחלקה המציגה למשתמש להכניס מידע על פרטי הגלגלים של כלי הרכב שלו. מטרת מחלקה זו הוא להחזיר רשימה של גלגלים עם פרטים מלאים עליהם לאחר שהמשתמש הזין קלט תקין.

**YesNoForm:** מחלקה כללית המקבלת קלט על איזה הודעה להציג למשתמש, שאלה של כן/לא. מטרת מחלקה זו היא להחזיר את בחירת המשתמש לאחר שהזין קלט תקין.

**EnumForm:** מחלקה כללית המקבלת רשימה של ערכי enum שונים להציג למשתמש, על המשתמש לבחור אחד מבין האופציות הללו. מטרת מחלקה זו היא להחזיר את האופציה שהמשתמש בחר לאחר שהזין קלט תקין.

**Program:** מחלקה שבתוכה נמצא רק ,Main בה היא קוראת ליצירת GarageMenuExecutorUI ועושה Execute.

Diagram, schematic

Description automatically generated**דיאגרמה שמציגה את היררכיות הירושה והקשר בין הרכיבים**

**הוראות להוספה של סוג כלי רכב חדש במוסך**

אם ברצונך להוסיף סוג כלי רכב חדש שהמוסך יקבל, לדוגמה טרקטור Tractor.

1. עליך ליצור קובץ מחלקה בשם כלשהו שתבחר (לא משנה איזה) שיתווסף ל- project GarageLogic.

נניח שנקרא לו .**TractorSpecifications** על מחלקה זו לרשת מ**Specifications**.

נניח כי לטרקטור (רגיל/חשמלי), בנוסף לתכונות של רכב, יש גם את התכונות: x, y, z.

נוסיף אותם בתור private data members עם properties get מתאימים למחלקה TractorSpecifications (שם יהיה כל תכונה כלשהי הייחודית לו שאין בתכונות של הרכב הרגילות, גם אם זה Enum, int, float).

נוסיף את ה-constructor הבא למחלקה שיצרנו:

public TractorSpecifications(string i\_TypeOfVehicle) : base(i\_TypeOfVehicle) { }

נממש את המתודה הזו:

public override void InitSpecifications(List<object> i\_SpecificationAnswers)

מימוש מתודה זו מתבצע כך שיתבצעו בה השמות באופן הבא (דוגמה בהמשך):

ה- data member ה- i שהוכרז בתוך המחלקה הזו יהיה שווה לערך i\_SpecificationAnswers[i] עם casting מתאים לסוג הtype של הdata member הספציפי הזה.

כאשר i=0 מייצג את ה- data member הראשון שהוכרז , i=1 מייצג את ה- data member השני שהוכרז, וכך הלאה על כל ה data members.

יש לציין כי במתודה זו, לפני שמתבצעים כל ההשמות ניתן לבצע בדיקות על SpecificationAnswers[i] כלשהו ולזרוק Exception מתאים, מה שנועד למנוע מהמשתמש ליצור אובייקטים מסוג זה עם קלט לא תקין.

(דוגמה מלאה למימוש מתודה זו על מכונית: סדר ה data members שהוכרזו הוא חשוב:

קודם הוכרז שם eColor m\_Color (מסוג public eColor enum שהוכרז במחלקה זו),

ואז הוכרז שם int m\_DoorCount (שאמור להכיל ערך בין 2 ל-5 כולל),

לכן כך נראה המימוש, לפי אותו סדר מתבצעת השמה הזו:

{

if ((int)i\_SpecificationAnswers[1] < 2 || (int)i\_SpecificationAnswers[1] > 5)

{

throw new ValueOutOfRangeException(2, 5);

}

m\_Color = (eColor)i\_SpecificationAnswers[0];

m\_DoorCount = (int)i\_SpecificationAnswers[1];

}

סוף דוגמה על מכונית)

(הערה: יש לציין כי הdowncasting הנ"ל תמיד בהכרח יעבוד כי i\_SpecificationAnswers הוא מערך של קלט מהמשתמש המתבצע באופן דינמי עם Reflection על מחלקה זו, שם מתקבל קלט לפי ה-type הספציפי של הdata member והוא נשמר בתור .object typeלכן סה"כ מתבצע פה upcasting & downcasting)

1. נפתח את המחלקה VehicleFactory הנמצאת תחת project/namespace GarageLogic.

שם נמצא תת מחלקה בשם VehicleTypes ונוסיף בתוכה nested class בשם Tractor (השם שיוצג למשתמש בתור סוג הרכב שהתווסף, ניתן לבחור כל שם) בדיוק בפורמט הבא:

public class Tractor

{

public const int k\_WheelsNumber = 4;

public const float k\_MaxTirePressure = 31f;

public static readonly Type sr\_SpecificationType = typeof(TractorSpecifications);

private Tractor() { }

public class Fueled

{

public const GasolineFueledVehicle.eFuelType k\_CurrentFuelType = GasolineFueledVehicle.eFuelType.OCTAN98;

public const float k\_FuelTankCapacityInLiters = 6.2f;

private Fueled() { }

}

public class Electric

{

public const float k\_MaxBatteryTimeInHours = 2.5f;

private Electric() { }

}

}

ניתן למחוק את כל Nested class Fueled אם יש לנו במוסך רק electric tractor, ובאותו אופן ניתן למחוק את כל Nested class Electric אם יש לנו במוסך רק fueled by gasoline tractor.

את שאר ערכי השדות פה יש לשנות בהתאם שיהיו נכונים לסוג הרכב שהתווסף, חוץ מ sr\_SpecificationType שהוא בהכרח יהיה שווה ל:

(שם המחלקה שהתווספה לפרוייקט מסעיף 1typeof(

**הוראות למחיקה של סוג כלי רכב קיים במוסך**

אם ברצונך למחוק סוג כלי רכב קיים שהמוסך מקבל, לדוגמה טרקטור Tractor.

1. נמחק את קובץ הspecification class הרלוונטי שיצרנו לסוג הרכב שאותו אנו רוצים למחוק. במקרה שלנו נמחק את הקובץ TractorSpecifications.
2. נפתח את המחלקה VehicleFactory הנמצאת תחת project/namespace GarageLogic.

שם נמצא מחלקה בשם VehicleTypes ונמחק משם nested class בשם סוג הרכב שאותו אנו רוצים למחוק. במקרה שלנו נמחק את כל nested class Tractor.