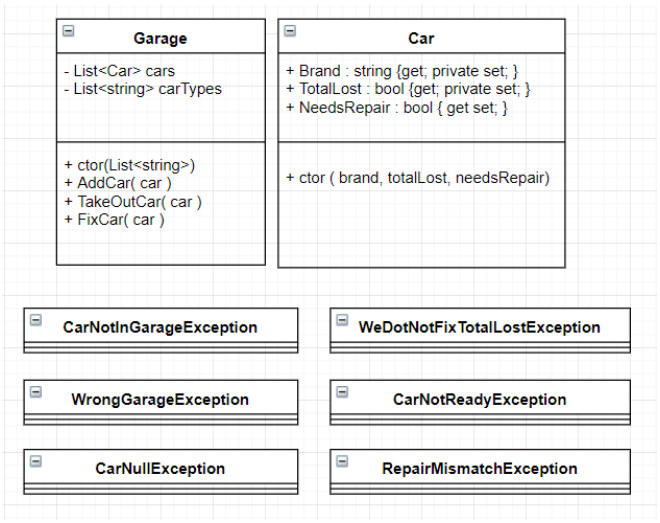
שיעורי בית

1. מהו **CustomException**? מתי נייצר אותו?
2. מאיזה מחלקה נירש כאשר נייצר **CustomException**? מדוע?
3. מדוע כדאי שה- **Exceptions** יהיו **Serializable**?
4. מהו **inner exception**? מתי נשתמש בו?
5. מי אמור לעשות את ה **catch**? (כאשר יש מספר רכיבים בשרשרת המובילים לרכיב שזורק את ה **Exception** לאוויר)
6. מה משמעות המילה **when** ב catch של exception?
7. מדוע כדאי לכתוב טסטים?
8. מה הרעיון העומד מאחורי ה- **TDD**? כיצד יכול ה- **TDD** לחסוך זמן פיתוח?
9. איזה סוג פרוייקט נייצר לצורך טסטים?
10. איזה מתודה בתוך טסט משווה בין ערכים (ומכשילה את הטסט אם אינם מה שציפיתי)?
11. איזה **Attribute** מעיד על כך שהמתודה היא מתודת טסט?
12. איזה **Attribute** מעיד על כך שהמחלקה היא מחלקת טסט?
13. מה ה- **Attribute** שיש לרשום מעל מתודה כדי לצפות לכך שייזרק **Exception** במהלך המתודה, ואם לא ייזרק אז הטסט ייכשל?

המשך בעמוד הבא...



- ממש את הדיאגרמה בצורה הנכונה ביותר.
- הוסף *interface* בשם **IGarage** אשר מכיל את שלושת הפונקציות AddCar, TakeOutCar, FixCar וממש אותו במחלקת Garage
- <u>מחלקת **Garage**:</u>

זוהי מחלקה אשר מחזיקה "בבטן" רשימה של מכוניות. המוסך יודע לטפל רק במכוניות **מהסוג** שניתן ברשימה בבנאי.
למוסך יש כללים, כגון :המוסך אינו מטפל במכוניות total lost. המוסך אינו מוכן להוציא רכב אשר טרם תוקן וכו'

- פונקציית **בנאי**- מקבלת רשימה של סוגי מכוניות אשר רק בהם המוסך יודע לטפל
- **<u>**אתגר אסתר המלכה:</u> נסה לכתוב את הטסטים לפני כתיבת המחלקה (TDD)**
- פונקציה **AddCar** מקבלת כארגומנט רכב:
  - אם הרכב כבר קיים ברשימה זרוק *CarAlreadyHereException* (שים לב: האקספשיין הזה **חסר** בדיאגרמה למעלה)
  - אם הרכב הוא Total Lost זרוק *WeDoNotFixTotalLostException*
  - אם סוג הרכב אינו מופיע ברשימת סוגי הרכבים בהם מטפל המוסך, זרוק *WrongGarageException*
  - אם הרכב הוא null זרוק *CarNullException*
  - אם הרכב אינו זקוק לטיפול (ראה בוליאני), זרוק *RepairMismatchException*
  - אחרת, אם הכול תקין: הוסף את הרכב לרשימה
- פונקציה **TakeOutCar** מקבלת כארגומנט רכב:
  - אם הרכב הוא null זרוק *CarNullException*
  - אם הרכב לא קיים ברשימת הרכבים, זרוק *CarNotInGarageException*
  - אם הרכב עדיין זקוק לטיפול (ראה שדה בוליאני), זרוק *CarNotReadyException*
  - אחרת, אם הכול תקין: הסר את הרכב מהרשימה
- פונקציה **FixCar** מקבלת כארגומנט רכב:
  - אם הרכב שהתקבל הוא null זרוק *CarNullException*
  - אם הרכב לא קיים ברשימת הרכבים של המוסך, זרוק *CarNotInGarageException*
  - אם הרכב לא זקוק לטיפול (ראה שדה בוליאני), זרוק *RepairMismatchException*
  - אחרת, אם הכול תקין: שנה את סטטוס ה- *NeedsRepair* של הרכב מ- true ל- false
- <u>מחלקת **Car**:</u>

בבנאי, אם *TotalLost* הוא אמת ו *NeedsRepair* הוא שקר, זרוק *RepairMismatchException*

- ב **Main** גרום לאקספשיין להיזרק מבלי להיתפס. חפש את פרטי האקספשיין ב *event viewer* (רמז: ראה סירטון)

- כעת כתוב פרוייקט טסט עם טסטים הבודקים את **כל הפונקציות** ואת **כל סוגי האקספשיינס**

- ***<u>**אתגר מרדכי היהודי (לא לבעלי לב חלש):**</u> ייצר מחלקה נוספת בשם **Bike** אשר היא דומה למחלקת **Car** פרט לעובדה שקיים לה שדה מידע נוסף בשם **EngineVolume** (נפח מנוע). הפוך את מחלקת **Garage** לגנרית (באמצעות T) ולכן:
  - עדכן את האינטרפייס **IGarage** שיהיה גנרי (באמצעות T), אל תישכח להחליף את המילה Car במילה Vehicle
  - עדכן את כל הפונקציות במחלקת **Garage**, אל תישכח להחליף את המילה Car במילה Vehicle במקומות הנכונים
  - עדכן את כל החריגות <u>הרלוונטיות</u> אשר מכילות את המילה **Car** שיהיו גם גנריות (באמצעות T). אל תישכח להחליף את המילה Car במילה Vehicle בשם החריגה.
  - הבדל נוסף בין הטיפול במכונית לטיפול באופנוע: שאופנוע ניתן לתקן TotalLost (ומכונית לא). ולכן ייצר interface בשם IFixTotalLost אשר מכיל פונקציה בשם CanFixTotalLost המחזירה בוליאני. במימוש של אינטרפייס זה במחלקת Car יוחזר שקר ובמימוש במחלקת Bike יוחזר אמת. וכך ה Garage ידע אם ניתן ברכב זה לתקן total lost או לא... <u>כמובן שזה אומר שבמחלקת Garage ה- T חייב לממש IFixTotalLost</u> (רמז: ראה פיתרון שיעורי בית עבור רשימה שבה הוספנו INamable)

כעת כמובן, עדכן את כל הטסטים שיבדקו גם את מוסך המכוניות וגם את מוסך האופנועים...

<u>פורים שמח!</u>

בהצלחה