חוברת הדרכה ותרגילים מלווה ליחידה תמ"ע

פותח על ידי:
אביטל גרינוולד
ענת שלוס
משה שטיינר
מיכל מלכה

לצורך נוחיות, הדוגמאות וההסברים עבור C# ו-Java מופיעים ביחד. במקומות בהם יש הבדלים, הם יצוינו במפורש. הבדלים שלא יצוינו במפורש הם: שמות פעולות המתחילות באות גדולה או קטנה בהתאם למקובל בשפה, והטיפוס בוליאן בו נשתמש לחלופין ב-bool או boolean.

1- ירושה
2 - הרשאות גישה [משה]
3 - פעולה בונה
4 - הגדרה מחדש של פעולות
5 - מידול [EVI] 7
6 - פולימורפיזם והמרות [EVI] 15
7 - מעקב תרשים עצמים [EVI] 21
7 - מעקב אחרי זימון פעולות (הגדרה מחדש של פעולות ופולימורפיזם) [משה] 24
8 - ממשקים 30
9- מחלקות מופשטות 34
10 - מילון מושגים (כולל מילים נרדפות ושמות המושגים באנגלית) [מיכל + Evi]
11 - נספח - תמ"ע באנדרואיד 37
12 - נספח - **WinForm** - שימוש בירושה (**C#** בלבד) [משה] 38
13 - נספח - תרגיל מסכם [משה] 40

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

1. ירושה

ירושה מגדירה יחס היררכי בין מחלקות.
תהליך הירושה בשפות מונחות עצמים מאפשר ליצור מחלקות חדשות על בסיס מחלקות קיימות.
המחלקה החדשה יורשת את כל התכונות והפעולות של המחלקה המורישה, וניתן להוסיף לה תכונות ופעולות המייחדות אותה. ירושה היא אם כן סוג של הרחבה.
מחלקה היורשת ממחלקה אחרת נקראת תת מחלקה.(subclass)
המחלקה המורישה נקראת מחלקת-על.(super class)
הורשה מתקיימת בין מחלקות ולא בין אובייקטים. אנחנו מגדירים יחס ירושה כאשר מצהירים על המחלקה.
עצמים של תת-המחלקה הם גם עצמים מטיפוס מחלקות-העל שלה.
בשפות C# ו- Java ניתן לרשת רק ממחלקה אחת.

שאלה מתגלגלת...

[ענת]
חברת Ishimoto מייצרת רכבים פרטיים, משאיות ואוטובוסים.

לפניכם כותרת המחלקה **Vehicle**

```
public class Vehicle
{
   protected static string MANUF = "Ishimoto";//  יצרן
}
```

הוסיפו למחלקת הבסיס **Vehicle** עם התכונות הבאות: דגם, שנת ייצור ונפח מנוע.

הגדירו תת מחלקה **Truck** ולה התכונות הבאות: דגם, שנת ייצור, נפח מנוע, משקל מטען מקסימלי.

הגדירו תת מחלקה **Bus** ולה התכונות הבאות: דגם, שנת ייצור, נפח מנוע, מספר נוסעים מקסימלי.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

2. הרשאות גישה [משה]

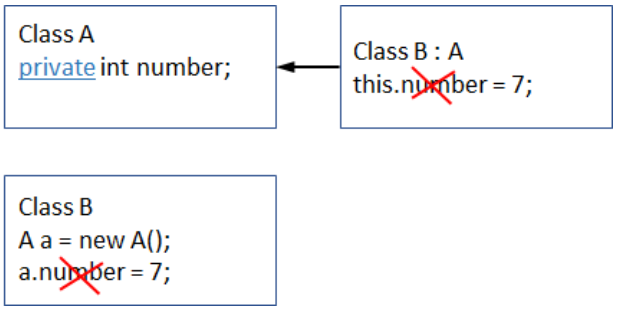
קיימות 3 רמות של הרשאות גישה למשתנים ומתודות:

- פרטי, private, הרשאת גישה אך ורק מתוך המחלקה בה המשתנה מוגדר. זוהי גם ברירת המחדל.
- ציבורי, public, יש גישה כללית מכל מקום ברחבי הקוד, כולל לשינויים.
- יורש/חסוי, protected, מחלקות היורשות מהמחלקה בה מוגדר משתנה מסוג proteced, ניגשות אליו כאילו הוגדר בתוך המחלקה. עבור כל יתר המחלקות, המשתנה בחזקת פרטי.

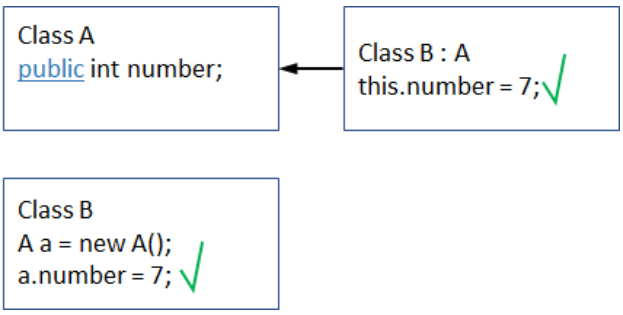
בנוסף, כאשר אובייקט ניגש למשתנה פרטי באובייקט אחר מאותו הסוג (אותה המחלקה), אין חסימת פרטיות.

הערה: בג'אוה הרחבת המחלקה נעשית על ידי החלפת נקודותיים (:) ב-extends. לדוגמה: class B extends A

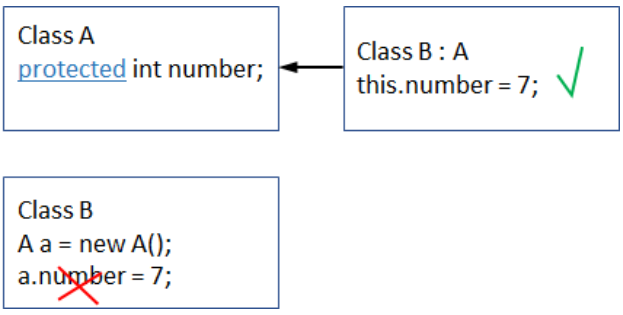
private:



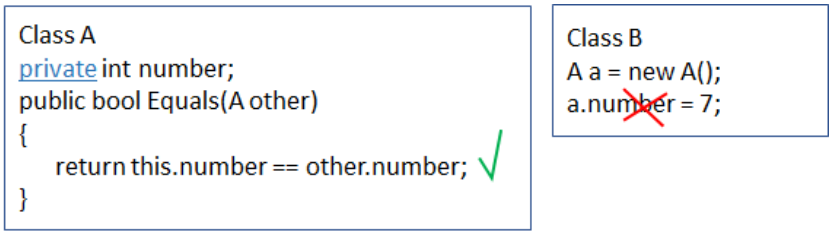
public:



protected:



private in the same class:



ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

3. פעולה בונה

פעולה בונה אינה עוברת בירושה.
כדי לאתחל עצם מטיפוס תת-מחלקה צריך לאתחל את כל התכונות שלו. הן של מחלקת-העל והן את התכונות הנוספות של תת המחלקה.
לשם כך נרצה לעשות שימוש בפעולה הבונה של מחלקת העל, באמצעות המילה השמורה . **base**

שאלה מתגלגלת...

[ענת]
הוסיפו פעולות בונות בשלושת המחלקות: Truck, Vehicle ו- Bus

נקודה למחשבה
איזו הרשאת גישה נתתם לתכונות במחלקה Vehicle?
האם ניתן להגדירם כ private?

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

שאלת אתגר/העשרה

[ענת]

לפניך המחלקה WhoAmI

```
public class WhoAmI
{
   private static WhoAmI me = null;

   private WhoAmI() { }

   public static WhoAmI GetMe()
   {
      if (me == null)
         me = new WhoAmI();
      return me;
   }
}
```

(1) לפניך מספר הוראות שנכתבו הפעולה הראשית במחלקה Program. עבור כל הוראה ציין האם היא חוקית או לא. במידה וכן שרטט את תרשים העצמים המתקבל. במידה ולא נמק.

(i) WhoAmI i1 = new WhoAmI();

(ii) WhoAmI i2 = WhoAmI.GetMe();

(iii) WhoAmI i3 = WhoAmI.me;

(2) כמה אובייקטים מסוג WhoAmI ניתן לבנות? נמק את תשובתך.

Singleton- תבנית אשר נועדה למקרים בהם מעוניינים להגביל את יצירת המופעים של מחלקה. נעשה בה שימוש במקרים בהם נהיה מעוניינים להגביל את יצירת המופעים של מחלקה מסוימת למופע יחיד.
מימוש התבנית חייב לעמוד בשני עקרונות: מופע יחיד של המחלקה, וגישה גלובלית האפשרות לגשת למופע מבלי ליצור מופע נוסף של המחלקה

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

4. הגדרה מחדש של פעולות – דריסת פעולות [משה]

לעיתים ההתנהגות שמתאימה למחלקת-העל אינה מתאימה לתת-המחלקה שלה. כדי לשנות פעולות שעברו בירושה, ניתן להגדירן מחדש בתת-המחלקה.

כאשר אנו מגדירים בתת-מחלקה פעולה שחתימתה זהה לזו של פעולה המוגדרת במחלקת-העל שלה אנו מבצעים הגדרה מחדש **Overriding**

לאחר שהגדרנו פעולה מחדש, הפעולה היא מחליפה את הפעולה המקורית.

הגדרה מחדש של פעולה בתת-מחלקה גורמת "להסתרת" הגדרתה המקורית. כל זימון של הפעולה בתת-המחלקה יביא להפעלת מימוש הפעולה לפי ההגדרה החדשה בתת-המחלקה, ולא לפי הגדרתה במחלקת-העל.

לעתים נרצה לגשת אל פעולה "מוסתרת", לשם כך נעשה שימוש במילה השמורה **base**, ובג'אווה **super**

באמצעות **base/super** ניתן לפנות אך ורק לאיבר במחלקת העל הישירה של מחלקה, כלומר: רק רמה אחת למעלה. מכיוון שמחלקת העל כבר דרסה את הפעולה שלפניה, או השתמשה בפעולה בצורתה המורשת, לא נוכל לגשת אל אבות קדמונים, כלומר: השימוש ב- base.base או super.super אינו אפשרי.

שאלה מתגלגלת...

במחלקות Bus, Truck שהוזכרו למעלה נרצה להוסיף פעולה מתארת, הפעולה ()ToString המחזירה מחרוזת המתארת את האובייקט. על המחרוזת להכיל את גם את שם היצרן, הנמצא במחלקת Vehicle.

1. כיתבו את ToString במחלקות היורשות המציגה את כל תכונות האובייקט בפורמט הבא:
Vehicle: <Manuf>, model: <model>, year: <year>, engine: <volume>, weight: <weight>
את שם היצרן יש לשלוף ממחלקת העל.

2. חיזרו על סעיף 1, ועכשיו השתמשו ב-ToString של מחלקת העל לשליפת שם היצרן.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

5. מידול [EVI]

שאלה פתוחה בה התלמידים יחליטו בעצמם על המידול הנכון, יבנו תרשים UML ויכתבו את כותרות המחלקות

שאלה 1: מתאימה לשאלה במבחן מתכונת. ניתן להרחיב למשימה מורכבת יותר במעבדה.

כתבה EVI מרעיון של שאלה של מרה, כמו בגרות 2012/19

בבית מלון "מנוחה לכל כיס" ישנם 50 חדרים משלושה סוגים: חדר רגיל, חדר אכסניה וחדר משודרג.
במלון 5 קומות כאשר קומה 5 היא קומת VIP והיא ניתנת רק לחדר רגיל או לחדר משודרג.
מחיר חדר בקומה 5 הוא בתוספת של 10% על מחיר הבסיס.

<u>המאפיינים של כל חדר:</u>

חדר-רגיל: מספר חדר, מספר קומה, מחיר בסיס, כמות מיטות, האם-פנוי?, האם-נקי? יחיד או זוג?
חדר-אכסניה: מספר חדר, מחיר בסיס, כמות מיטות, האם-פנוי?, האם-נקי? מספר מתארחים
חדר-משודרג: מספר חדר, מספר קומה, מחיר בסיס, כמות מיטות, האם-פנוי?, האם-נקי? יחיד או זוג? מספר-תוספות

<u>הפעולות שניתן לבצע על חדר:</u>

הזמנת חדר מתאפשרת במידה וכמות המוזמנים לחדר לא עוברת על כמות המקומות לאכלוס.
במידה וניתן לבצע הזמנה לחדר הוא הופך להיות תפוס ולא נקי.
חישוב מחיר מתבצע עבור חדר מאוכלס על פי הסוג שלו וכמות המתארחים בו.
ניקוי חדר מתאפשר עבור חדר רגיל ומשודרג בלבד.

<u>מה קיים בכל סוג של חדר?</u>

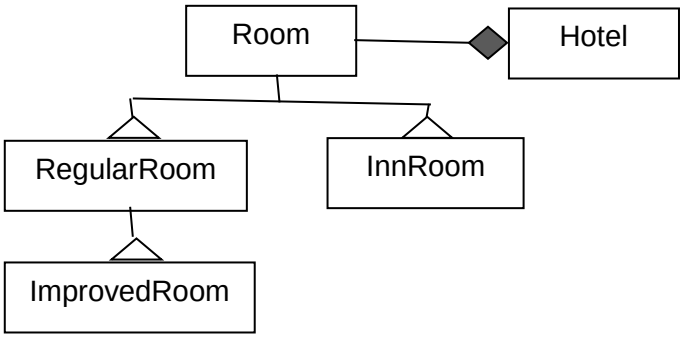
חדר-רגיל מכיל 2 מיטות ומחירו משתנה בהתאם למספר האנשים בו- זוג או יחיד.
ליחיד מחיר בסיס. עבור זוג יש לשלם תוספת של 50% על מחיר בסיס.
חדר-אכסניה הינו חדר המכיל מיטות קומתיים ומחיר למיטה. כל מיטה מאכלסת שני מתארחים.
חדר זה אין המלון דואג לניקיון. המחיר לתשלום הוא מחיר בסיס x מספר המיטות התפוסות.
כלומר, מספר המיטות שיש בהם מתארחים.
חדר-משודרג הוא חדר רגיל עם תוספות.
מחירו מחושב על פי מחיר בסיסי ומספר התוספות. על כל תוספת יש להוסיף 2% למחיר הבסיס.

בית המלון שומר מידע על כל החדרים במלון.
פעולות הניתנות לביצוע:

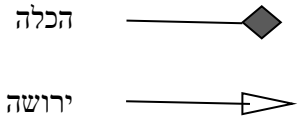
- הזמנת חדר במידה ופנוי ומספר המתארחים אינו גדול ממספר המיטות בחדר.
- ניקוי חדר על ידי צוות המלון.
- יצירת חשבון עבור הזמנת חדר ליום על פי סוגו וכמות המוזמנים.
- בדיקה אם יש חדר פנוי.
- בדיקה אם המלון בתפוסה מלאה.

לקוחות יכולים לבצע הזמנה לחדר במידה ופנוי. בהזמנה יש לציין את פרטי החדר.

לפניך תרשים היררכית המחלקות הנחוצות למחשוב הזמנת חדרים במלון.



מקרא:



נוסף על המחלקות המתוארות בתרשים, נתונים 3 ממשקים

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

```
interface IOrder {  boolean order(int numPersons);  } // הזמנת חדר
interface IPrice  {  double calcPrice();             } // חישוב מחיר
interface IClean {  void clean();                    } // ניקוי חדר
```

1- העתק לדף הבחינה את תרשים היררכיית המחלקות והוסף אליו את הממשקים במקומות המתאימים ביותר על פי העקרונות של תכנות מונחה עצמים.
השתמש בסימון ----------➔ לציון מימוש ממשק.

2- לכל מחלקה כתוב את:

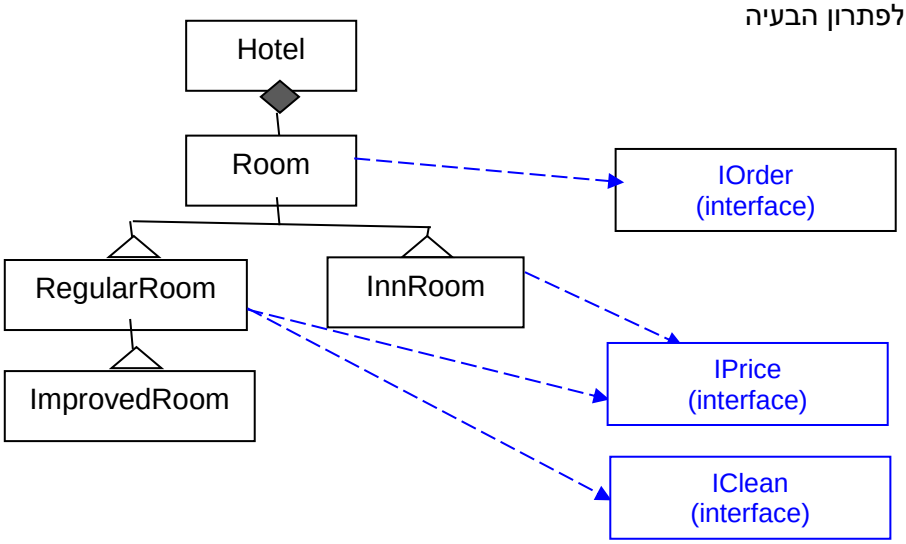
- הכותרת שלה בשפת ג'אווה.
- התכונות שלה. לכל תכונה רשום את ההגדרה ב java כולל הרשאת גישה ותיעוד התכונה.
- הפעולות שלה. לכל פעולה רשום את הכותרת ב java כולל תנאי קדם ובתר של הפעולה.
- אין צורך לממש את הפעולות.
- אין צורך לכתוב פעולות בונות , פעולות קובעות (set) פעולות מאחזרות (get)

3- ממש בג'אווה את הפעולה לחישוב המחיר עבור החדר בכל אחת מהמחלקות בהן היא מופיעה.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פתרון שאלה 1

א- היררכית המחלקות והממשקים לפתרון הבעיה



ב-

public class **RegularRoom** extends Room implements IClean, IPrice { protected int floorNumber protected boolean couple; public boolean order(int numPersons) {…} public void clean() {…} public double calcPrice() {…}	 מספר קומה 'אמת' – אם זוג בחדר, 'שקר'-אחרת תנאי קדם: מספר אנשים תנאי בתר: הזמנת חדר אם אפשר ומחזיר 'אמת' , 'שקר' – אחרת ללא הזמנת חדר. תנאי קדם: אין תנאי בתר: מנקה חדר ומשנה מצב נקי ל- 'אמת' תנאי קדם: אין תנאי בתר: מחזיר חישוב תשלום עבור החדר

public class **ImprovedRoom** extends
RegularRoom
{
private int numExtras;
 	
public double calcPrice() {…}
}
	
	מספר תוספות
	
	תנאי קדם: אין
	תנאי בתר: מחזיר חישוב תשלום עבור החדר

public class **InnRoom** extends Room
implements IPrice
{
public double calcPrice()
}
	
	תנאי קדם: אין
	תנאי בתר: מחזיר חישוב תשלום עבור החדר

public class **Hotel** { private Room[] rooms; public void orderRoom (int roomNumber, int numOfpersons) {…} public double calcPrice(int roomNumber){..}	 מערך חדרים תנאי קדם: מספר חדר ומספר אנשים תנאי בתר: הזמנת חדר על פי הפרמטרים תנאי קדם: מספר חדר תנאי בתר: החזרת מחיר על החדר במידה וניתן לחדש מחיר חדר אחרת יחזיר0

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

תנאי קדם: מספר חדר תנאי בתר: ניקוי חדר במידה וניתן תנאי קדם: אין תנאי בתר: אמת אם מלון בתפוסה מלאה, שקר – אחרת.	public void clean(int roomNumber) public boolean isFull()

ג- פעולה לחישוב המחיר עבור החדר בכל אחת מהמחלקות בהן היא מופיעה.

מחלקת **RegularRoom**	public double **calcPrice**()
	{
	double price = this.basePrice;
	if (this.floorNumber == 5)
	price *= 1.1;
	if (this.couple)
	price = price *1.5;
	return price;
	}

מחלקת **ImprovedRoom**	public double **calcPrice**()
	{
	double price = super.calcPrice();
	price = price + this.basePrice * this.numExtras * 0.02;
	return price;
	}

מחלקת **InnRoom**	public double **calcPrice**()
	{
	int numBads = this.numPersons/2;
	if (numBads%2==1)
	numBads++;
	return this.basePrice * numBads;
	}

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

שאלה 2 : [ירושה , ממשקים, הכמסה, ,פולימורפיזם, כתבה EVI מבוסס על 2008, 19 ו 2014, 20]

מתאימה לשאלה במבחן מתכונת

ברשת חנויות מחשבים "פיסיטון" יש 20 חנויות. הרשת מוכרת ציוד היקפי למחשבים.
מנהל הרשת מעוניין בתכנה שבעזרתה הוא יוכל לנהל טוב יותר את המלאי בחנויות הרשת.
במיוחד הוא מתעניין במלאי של מדפסות, סורקים , פקסים ומדפסות משולבות.

להלן המאפיינים של פריטים אלו:

- מדפסת: קוד מוצר, דגם, מחיר, כמות במלאי, לייזר או דיו, מהירות הדפסה
- סורק: קוד מוצר, דגם, מחיר, כמות במלאי, מהירות סריקה
- פקס: קוד מוצר, דגם, מחיר, כמות במלאי, מספר טלפון
- מדפסת משולבת: קוד מוצר, דגם, מחיר, כמות במלאי, לייזר או דיו, מהירות הדפסה או סריקה

להלן הפעולות שיכולים לבצע כל אחד מהפריטים:

- מדפסת: מדפיסה מסמכים
- סורק: סורק מסמכים
- פקס: שולח ומקבל פקסים
- מדפסת משולבת: מבצעת על כל הפעולות הנ"ל

קוד מוצר נקבע אופן אוטומטי על ידי המערכת. מספר הקוד המינימלי הוא 100.
כל אחד מהמוצרים נמצא בכל אחת מחנויות הרשת ומכל מוצר יש דגמים שונים בכמויות שונות.

הדרישות מהתוכנה ברמת החנות הן:

- ✓ מי הפריטים בחנות שהכמות שלהם קטנה ממספר מסוים limit
- ✓ מה השווי הכספי של כל המלאי בחנות

הדרישות מהתוכנה ברמת הרשת הן:

- ✓ מי הפריטים שהכמות שלהם קטנה ממספר מסוים limit בכל הרשת.
- ✓ מה השווי הכספי של כל המלאי מכל חנויות הרשת

עליך לתכנן את המחלקות והממשקים עבור כתיבת התוכנה לניהול מלאי החנות בנושא מדפסות, סורקים, פקסים ומדפסות משולבות באופן המתאים ביותר לעקרונות של תכנות מונחה עצמים: הכמסה, ירושה ופולימורפיזם.

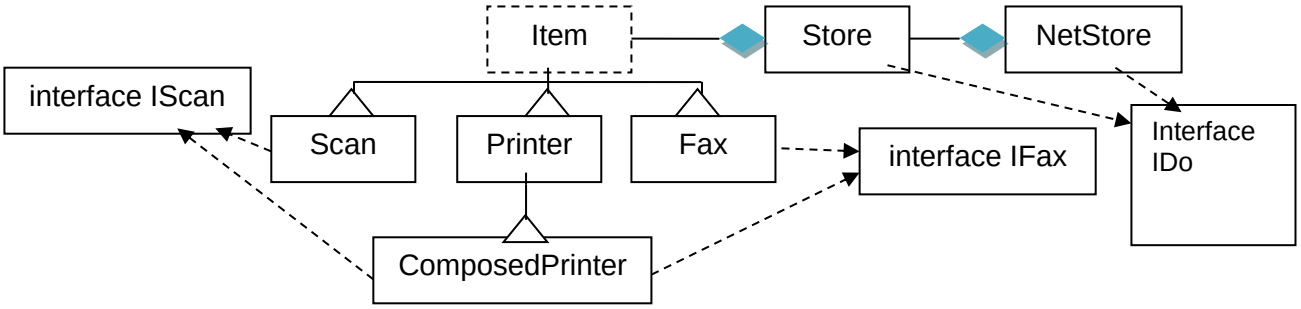
א- סרטט את היררכית המחלקות והממשקים. סמן את הקשרים בין המחלקות ואת הקשרים בין המחלקות והממשקים. אין צורך לכלול בסרטוט את התכונות והפעולות.

ב- בעבור כל מחלקה שכללת בסרטוט רשום בג'אווה את כותרת המחלקה, התכונות והפעולות שלה.
בעבור כל ממשק שכללת בסרטוט רשום בג'אווה את הכותרת וכותרות לפעולות.
יש לכלול אך ורק את הפעולות הנחוצות כדי לענות על הדרישות מהתוכנה כפי שתוארו לעיל.
יש לתעד כל תכונה וכל פעולה. אין צורך לממש את הפעולות במחלקות.
אין צורך לרשום פעולות בונות, מאחזרות get וקובעות set ו toString .

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פתרון שאלה 2

א- היררכית המחלקות הנדרשות.




```
public abstract class Item
{
    protected static int code=99;
    protected String model;
    protected double price;
    protected int amount;
}
```

```
public class Fax extends Item implements IFax
{
    private String phone;

    public void sendFax(Sting phone) {}
    public void recieveFax(String phone) {}
```

```
public class Scan extends Item implements
ISan
{
        protected double velocity;
        public void scan() {}
}
```

```
public class Printer extends Item
{
        private double velocity;
        private char layzerInc;
        public void print() {}
}
```

```
public class ComposedPrinter extends Printer implements IFax, ISan
{
        public void sendFax(Sting phone) {}
        public void recieveFax(String phone) {}
        public void scan() {}
}
```

```
public class Store implements IDo
{
    private Vector<Item> items;
}
```

```
public class NetStore implements IDo
{
    private Store[] stores;
}
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

הממשקים- כותרת ופעולות:

```
public interface IFax
{
   public void sendFax(String phone); // לטלפון פקס שליחת
   public void recieveFax(String phone); // מטרפון ממספר פקס קבלת
}
```

```
public interface ISan
{
   public void scan();// מסמך סריקת
}
```

```
public interface IDo
{
   public void printAmountLessLimit(int limit); // limit מ פחות שהכמות בחנות פריטים מדפיס
   public double totalPrice();      // המוצרים של הכולל השווי את מחזיר
}
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

תכונות ופעולות המחלקות:

המחלקה	התכונה	תיעוד התכונה
Item	קוד מוצר	protected static int code=99;
	מודל	protected String mode;
	מחיר	protected double price;
	כמות במלאי	protected int amount;
	הפעולה	**תיעוד הפעולה**
המחלקה	**התכונה**	**תיעוד התכונה**
Scan	מהירות סריקה	private double velocity;
	הפעולה	**תיעוד הפעולה**
	מבצע סריקה	public void scan()
המחלקה	**התכונה**	**תיעוד התכונה**
Printer	האם לייזר או דיו. 'l' עבור לייזר, 'i' עבור דיו	private char layzerInc;
	מהירות הדפסה	private double velocity;
	הפעולה	**תיעוד הפעולה**
	מבצע הדפסה	public void print()
המחלקה	**התכונה**	**תיעוד התכונה**
Fax	מספר טלפון	private String phone
	הפעולה	**תיעוד הפעולה**
	שליחת פקס לטלפון phone	public void sendFax(String phone)
	קבלת פקס מטלפון phone	public void recieveFax(String phone)
המחלקה	**התכונה**	**תיעוד התכונה**
Composed Printyer	מהירות הדפסה או סריקה	private double velocity;
	'אמת' אם לייזר, 'שקר'- אם דיו	private boolean layzerInc;
	הפעולה	**תיעוד הפעולה**
	שליחת פקס	public void sendFax(String phone)
	קבלת פקס	public void recieveFax(String phone)
	סריקה	public void scan()
	הדפסה	public void print()
המחלקה	**התכונה**	**תיעוד התכונה**
Store	וקטור של פריטים	private Vector<Item> items;
	הפעולה	**תיעוד הפעולה**
	הפעולה מקבלת מספר שלם שמהווה כמות – limit הפעולה מדפיסה את כל המוצרים שהכמות שלהם קטנה מ limit בחנות	public void **printAmountLessLimit** (int limit)
	הפעולה מחזירה את השווי הכללי של כל הפריטים בחנות	public double **totalPrice**()
NetStore	מערך חנויות	private Store[] stores;
	הפעולה מקבלת מספר שלם שמהווה כמות – limit הפעולה מדפיסה את כל המוצרים שהכמות שלהם קטנה מ limit בכל הרשת	public void **printAmountLessLimit** (int limit)
	הפעולה מחזירה את השווי הכללי של כל הפריטים ברשת	public double **totalPrice**()

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פולימורפיזם והמרות **זיהוי שגיאות** [EVI]

שאלה 1: [EVI] , ירושה, מעקב , זיהוי עקרונות מונחה עצמים , מבוסס על שאלה 19א בגרות 2013]

<u>חלק א:</u> לפניך המחלקות הבאות:

```
public class T
{
   protected String name;

   public T(String name)
   {
      this.name = name;
   }
   public String toString()
   {
      return "T:"+this.name;
   }
}
```

```
public class G extends T
{
   public G(String name)
   {
      super(name);
   }
   public String toString()
   {
      return "G:"+ super.toString();
   }
   public T getFather()
   {
      return new T(this.name);
   }
}
```

```
public class RunTG
{
   public static void main(String[] args)
   {
      T a1 = new T("dog");
      T a2 = new G("cat");
      G a3 = new G("me");
      System.out.println(a1);
      System.out.println(a2);
      System.out.println(a3);
      System.out.println((T)a3);
      System.out.println(a3.getFather());
   }
}
```

(I) עקוב אחר הפעולה הראשית בעזרת תרשים עצמים ורשום את הפלט.

(ii) לפניך רשימה של מושגים הקשורים לתכנות למונחה עצמים:

- העמסת פעולות overloading
- דריסת פעולות overriding
- פולימורפיזם
- המרות. המרה כלפי מעלה והמרה כלפי מטה
- ירושה
- הרשאות גישה

לגבי כל מחלקה מבין המחלקות: T, G, RunTG
בחר מושג או מושגים הבאים לידי ביטוי במחלקה. הסבר את קביעתך.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

<u>חלק ב</u> (אין קשר לסעיף א) [interfaces , דומה לבגרות 2013, שאלה 18, סעיף א]

לפניך ההגדרות הבאות:

```
public class Car { }
public class Hybrid extends Car { }
public class Engine extends Car implements IGazUsage{ }
public class FordMustang extends Engine implements IFordable{ }
```

1. שרטט היררכית המחלקות.
2. לפניך 6 שורות קוד, עבור כל אחת מהן קבע אם היא תקינה או לא. נמק קביעתך.
   במידה ואינה תקינה ציין אם זו שגיאת הידור או ריצה. אם יש המרה ציין מאיזה סוג ומאיזו מחלקה לאיזו.

a) `Car c1 = new FordMustang();`

b) `IGazUsage e1 = new Engine();`

c) `IGazUsage e2 = new Hybrid();`

d) `FordMustang f1 = (FordMustang)e1;`

e) `e2 = (Engine)c1;`

f) `f1 = e1;`

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פתרון שאלה 1 – חלק א

(I)מעקב ופלט

```
public class RunTG
{
   public static void main(String[] args)
   {
      T a1 = new T("dog");
      T a2 = new G("cat");
      G a3 = new G("me");
      System.out.println(a1);
      System.out.println(a2);
      System.out.println(a3);
      System.out.println((T)a3);
      System.out.println(a3.getFather());
   }
}
```

<T> a1 → **T** name: dog

<T> a2 → **G** name: cat

<G> a3 → **G** name: me

T:dog
G:T:cat
G:T:me
G:T:me
T:me

(ii) זיהוי עקרונות מונחה עצמים

במחלקה :T

הרשאות גישה

במחלקה G:

ירושה, דריסת פעולות,

במחלקה Main :

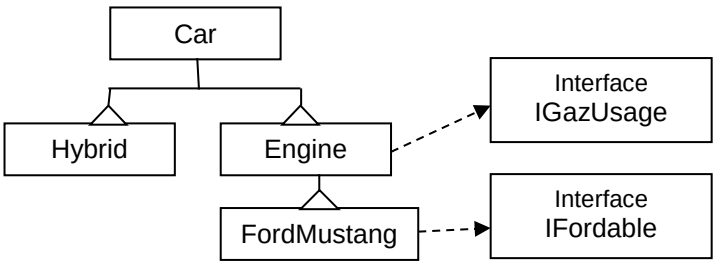
פולימורפיזם והמרות כלפי מעלה

פתרון שאלה 1 חלק ב

לפניך ההגדרות הבאות:

```
public class Car { }
public class Hybrid extends Car { }
public class Engine extends Car implements IGazUsage{ }
public class FordMustang extends Engine implements IFordable{ }
```

1. שרטט היררכית המחלקות.



ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

2. לפניך 6 שורות קוד, עבור כל אחת מהן קבע אם היא תקינה או לא. נמק קביעתך.
במידה ואינה תקינה ציין אם זו שגיאת הידור או ריצה. אם יש המרה ציין מאיזה סוג ומאיזו מחלקה לאיזו.

a)	Car c1 = new FordMustang();	תקין. FordMustang . הוא סוג של Car . ניתן ליצור עצם מתת-מחלקה והציבו במשתנה מטיפוס מחלקת העל. זו המרה כלפי מעלה מ FordMustang ל Car .
b)	IGazUsage e1 = new Engine();	תקין. Engine . מממש את הממשק IGazUsage . ניתן ליצור עצם ממחלקה והציבו במשתנה מטיפוס הממשק אותו הוא מממש. המרה כלפי מעלה מטיפוס Engine לטיפוס הממשק IGazUsage .
c)	IGazUsage e2 = new Hybrid();	שגיאת תחביר. Hybrid . אינו ממש את הממשק IGazUsage ולכן אי אפשר לבצע השמה זו. אין התאמה בטיפוסים.
d)	FordMustang f1 = (FordMustang)e1;	שגיאת ריצה. e1 נוצר כ Engine . יש כאן נסיון לבצע המרה מטיפוס מחלקת העל לטיפוס מחלקת הבת ForMustang . בזמן ריצה התכנית תגלה שאי אפשר לבצע המרה זו.
e)	e2 = (Engine)c1;	תקין. c1 נוצר כ FordMustang שהוא יורד מ Engine ולכן ניתן להמיר אותו למחלקת על. Engine מממשת הממשק IGazUsage ולכדן ניתן לבצע המרה למטיפוס המחלקה של טיפוס הממשק אותו היא מממשת. יש כאן שתי המרות: האחת מ FordMustang ל Engine השנייה: מ Engine ל IGazUsage . שתי ההמרות הן המרות כלפי מעלה.
f)	f1 = e1;	שגיאת תחביר. חוסר התאמה בטיפוסים. e1 נוצר כ Engine ואילו f1 הוא עצם מטיפוס FordMustang . Engine לא מכיר את הטיפוס FordMystang . הערה: השמה הפוכה היא נכונה. f = e1

שאלה 2 [Evi , איתור שגיאות, ירושה, העמסת פעולות ודריסה פעולות, פולימורפיזם, מעקב]

בעמוד הבא קוד המחלקות **Position, Position2, Position3, TestPosition**

א- שרטט תרשים היררכית המחלקות

ב- קבע לגבי כל אחת מהשורות הבאות: 3, 5, 7, 9, 11, 13, 14, 15 אם הן תקינות או לא. נמק תשובתך בכל מקרה!
במידה וההוראה שגוייה . רשום האם שגיאת תחביר או ושגיאת ריצה. הצע דרך לתקן הוראות שגויות אם ניתן.

ג- עקוב אחר הפעולה הראשית במחלקה TestPosition עבור שורות 1-12 לאחר שתיקנת את כל השגיאות שגילית בסעיף ב' מבחן שורות אלו. חובה ללוות את המעקב בתרשים עצמים.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

המשך שאלה 2

```
public class Position
{// מתאר מיקום בממד אחד
   protected int x;
   public Position(int x)
   {
      this.x = x;
   }
   public void moveAbsolute(int x)
   {
      this.x = x;
   }
   public void moveRelative(int dx)
   {
      this.x = this.x + dx;
   }
   public String toString()
   {
      return "("+this.x+")";
   }
   public void decribe()
   {
      System.out.println("Position in (x) axis");
   }
}
```

```
public class Position2 extends Position
{// מתאר מיקום בשני ממדים
   protected int y;
   public Position2(int x, int y)
   {
      super(x);
      this.y = y;
   }
   public void moveAbsolute(int x, int y)
   {
      super.moveAbsolute(x);
      this.y = y;
   }
   public void moveRelative(int dx, int dy)
   {
      super.moveRelative(dx);
      this.y = this.y + dy;
   }
   public String toString()
   {
      return "("+super.toString()+","+this.y+")";
   }
   public void decribe()
   {
      System.out.println("Position in (x,y) axis");
   }
}
```

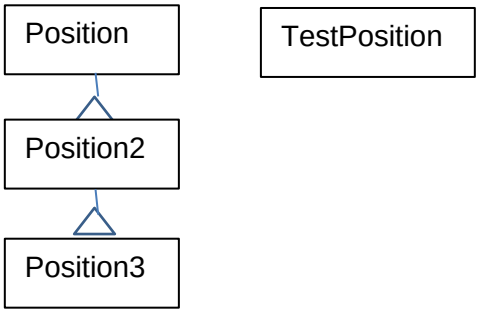
```
public class Position3 extends Position2
{// מתאר מיקום בשלושה ממדים
   private int z;
   public Position3(int x, int y, int z)
   {
      super(x, y);
      this.z = z;
   }
   public void moveAbsolute(int x, int y, int z)
   {
      super.moveAbsolute(x, y);
      this.z = z;
   }
   public void moveRelative(int dx, int dy, int dz)
   {
      super.moveRelative(dx, dy);
      this.z = this.z + dz;
   }
   public String toString()
   {
      return super.toString()+","+this.z+")";
   }
   public void decribe()
   {
      System.out.println("Position in (x,y,z) axis");
   }
}
```

```
public class TestPosition
{
   public static void main(String[] args)
   {
        1)   Position pos1 = new Position(3);
        2)   System.out.println(pos1);
        3)   pos1.moveRelative(2);
        4)   System.out.println(pos1);
        5)   Position pos2 = new Position2(-1, 4);
        6)   System.out.println(pos2);
        7)   pos2.moveAbsolute(3,0);
        8)   System.out.println(pos2);
        9)   Position3 pos3 = new Position3(2, 0, -3);
        10)  System.out.println(pos3);
        11)  pos3.moveRelative(0, -2);
        12)  System.out.println(pos3);
        13)  Position3 posC =(Position3)pos2;
        14)  pos2 = pos3;
        15)  pos3 = pos1;
   }
}
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פתרון שאלה

1- שרטט תרשים היררכית המחלקות



ב- קבע לגבי כל אחת מהשורות הבאות: 3, 5, 7, 9, 11, 13, 14, 15 אם הן תקינות או לא. נמק תשובתך בכל מקרה!

במידה וההוראה שגוייה . רשום האם שגיאת תחביר או ושגיאת ריצה. הצע דרך לתקן הוראות שגויות אם ניתן.

שורה3	שורה תקינה .pos1 נוצר כעצם מטיפוס Position ול Position יש פעולה כזו עם חתימה כזו.
שורה5	שורה תקינה .ל Position2 יש פעולה בונה כזו Position2 , יורש מ Position ולכן אפשר להציג אותו במשתנה מטיפוס זה.
שורה7	הוראה שגוייה .מסתכלים על pos2 מנקודת מבט של . Position למחלקה Position אין פעולה כזו. אמנם יש פעולה עם שם כזה אבל חתימה שונה, לכן זו שגיאת הידור. תיקון ((Position2)pos2).moveAbsolute(3,0) :המרה לטיפוס Position2 ואז הפעלת הפעולה.
שורה9	שורה תקינה .לעצם מטיפוס Position3 יש פעולה בונה כזו וניתן להציג אותה במשתנה מטיפוס Position3
שורה11	שורה תקינה .יורש מ Position2 ולכם יורש את כל הפעולות שלו. ל Position2 יש פעולה כזו עם חתימה כזו ולכן ניתן להפעיל אותה על pos3 שנוצר ומסתכלים עליו כ Position3
שורה13	שגיאת ריצה .נוצר כ Position3 . Position2 יורש מ Position2 ולכן ניתן לכתוב הוראת המרה כלפי מטה למחלקת הבת, כלומר מבחינה תחבירית אין בעיה. בזמן ריצה הקומפיילר יגלה שהעצם נוצר מטיפוס Position2 ולא יוכל להמירו לטיפוס . Position3 המרות מתבצעות בזמן ריצה
שורה14	שורה תקינה .Position3 יורש מ position2 ולכן ניתן להמיר המרה מפורשת כלפי מעלה ממחלקת הבת למחלקת העל.
שורה15	הוראה שגוייה .שגיאת הידור .pos1 נוצר כ Position , אי אפשר להציב אותו במשתנה מטיפוס .Position3 אין התאמה בטיפוסים.

ג- עקוב אחר הפעולה הראשית במחלקה TestPosition עבור שורות 1-12 לאחר שתיקנת את כל השגיאות שגילית בסעיף ב' מבחן שורות אלו. חובה ללוות את המעקב בתרשים עצמים.

1) <Position> pos1 → | **Position** x: 3 | 2) (3) פלט

3) <Position> pos1 → | **Position** x: ~~3~~ 5 | 4) (5) פלט

5) <Position> pos2 → | **Position2** x: -1 y: 4 | 6) ((-1),4) פלט

Correction: ((Position2)pos2).moveAbsolute(3,0)

7) <Position2> pos2 → | **Position2** x: 3 y: 0 | 8) ((3),0) פלט

9) <Position3> pos3 → | **Position3** x: 2 y: 0 z: -3 | 10) ((2),0),-3) פלט

11) <Position3> pos3 → | **Position3** x: 2 y: -2 z: -3 | 12) ((2),-2),-3) פלט

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

7. מעקב תרשים עצמים כולל דריסת פעולות [EVI + Moshe]

שאלה 1 [מעקב עם תרשים עצמים+ממשקים, מבוסס על בגרות 2008, שאלה 20 , שינוי של EVI]

לפניך פרויקט ובו המחלקות הבאות:

```
public class Base
{
    protected int x;
    public Base(int x)
    {       this.x = x;      }
     protected void doSomething()
    {
        System.out.println("x is: "+this.x);
    }
}
```

```
public class Derived1 extends Base
{
    private int y;
    public Derived1(int x, int y)
    {
        super(x);
        this.y = y;
    }
    protected void doSomething()
    {
        super.doSomething();
        this.multiplication();
    }
    public void multiplication()
    {
        System.out.println("y is: "+this.y);
        System.out.println("x*y is: "+this.x*this.y);
    }
}
```

```
public class MainBase
{
    public static void main(String[] args)
    {
        Base[] arr = new Base[5];
        arr[0] = new Derived1(3,11);
        arr[1] = new Derived2(15,0);
        arr[2] = new Derived3(-15, 5);
        arr[3] = new Base(120);
        arr[4] = new Derived2(13,3);
        for (int index=0; index<arr.length; index++)
        {
            System.out.println("index is: "+index);
            arr[index].doSomething();
        }
    }
}
```

```
public class Derived2 extends Base
{
    private int z;

    public Derived2(int x, int z)
    {
        super(x);
        this.z = z;
    }
    protected void doSomething()
    {
        super.doSomething();
        if (this.z != 0)
            this.division();
    }
    public void division()
    {
        System.out.println("z is: "+this.z);
        System.out.println("x/z is: "+this.x/this.z);
    }
}
```

```
public class Derived3 extends Base
{
    private int w;
    public Derived3(int x, int w)
    {
        super(x);
        this.w = w;
    }
    protected void doSomething()
    {
        super.doSomething();
        System.out.println("w is : "+this.w);
        this.multiplication();
        if (this.w !=0)
            this.division();
    }
    public void multiplication()
    {
        System.out.println("x*w is: "+ this.x*this.w);
    }
    public void division()
    {
        System.out.println("x/w is: "+this.x/this.w);
    }
}
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל

א- שרטט תרשים היררכית המחלקות

ב- בצע מעקב אחר הפעולה **main** במחלקה **MainBase** וכתוב את הפלט.
במעקב <u>חובה</u> להראות תרשים עצמים.

ג- הוסיפו למחלקות שני מימשקים : 1Inter ו 2Inter .
להלן קוד הממשקים:

```
public interface Inter1
{    public void multiplication();  }
```

```
public interface Inter2
{    public void division();   }
```

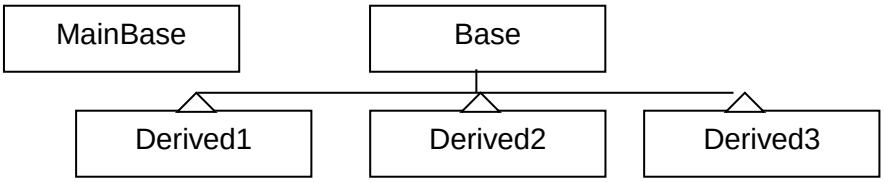
שנה את היררכיות המחלקות כך שיממשו את הממשקים 2Inter1, Inter

- עבור מחלקות שעברו שינוי, רשום את השינוי והסבר אותו.
- שרטט את היררכיות המחלקות בהתאם לשינוי.
- כתוב את כותרות המחלקות שעברו שינוי.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

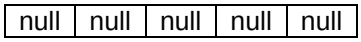
פתרון שאלה 1 [מעקב עם תרשים עצמים+ממשקים, מבוסס על בגרות 2008, שאלה 20 , שינוי של EVI]

א- שרטט תרשים היררכית המחלקות

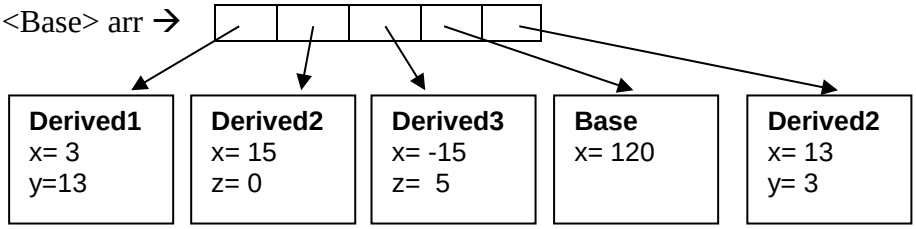


ב- מעקב אחר הפעולה **main** במחלקה **MainBase** וכתוב את הפלט. (4+10=14)

1) <Base> arr →



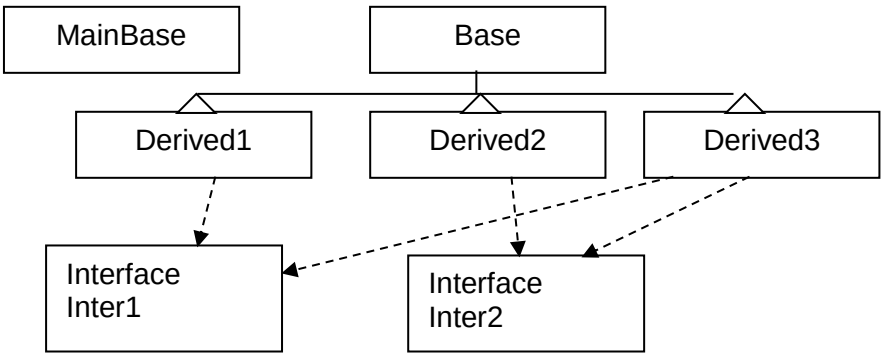
2) <Base> arr →



3)

index	index<arr.length	output
0	true	index is: 0 x is: 3 y is: 11 x*y is: 33
1	true	index is: 1 x is: 15
2	true	index is: 2 x is: -15 w is : 5 x*w is: -75 x/w is: -3
3	true	index is: 3 x is: 120
4	true	index is: 4 x is: 13 z is: 3 x/z is: 4

ג- היררכיות המחלקות כולל הממשקים Inter1, Inter2

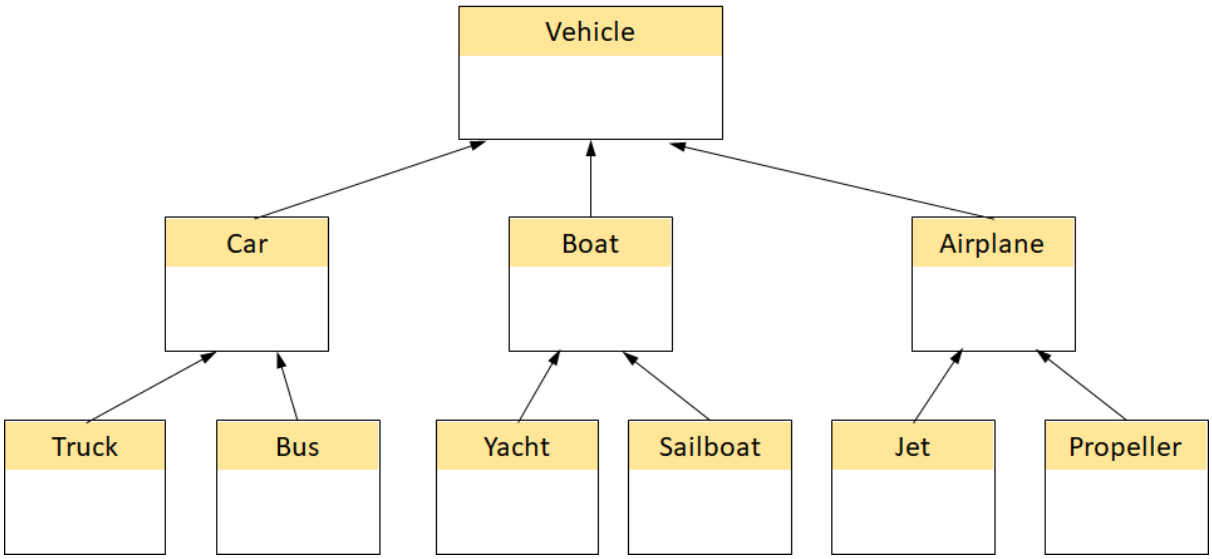


public class **Derived1** extends Base implements Inter1
public class **Derived2** extends Base implements Inter2
public class **Derived3** extends Base implements Inter1, Inter2

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

שאלה 2 [משה]

פולימורפיזם, או רב צורתיות, היא היכולת לראות אובייקט במספר צורות בו זמנית.
כל מחלקה יורשת ממחלקת Object, ואין צורך לכתוב זאת במפורש. כלומר כל אובייקט הוא סוג של Object.
בדוגמה שלפנינו, כלי תחבורה יכול להיות רכב, או סירה או מטוס.
רכב יכול להיות משאית או אוטובוס. סירה יכולה להיות יאכטה או סירת מפרש. ומטוס יכול להיות מטוס סילון או מטוס בוכנה.



כלומר, אם נגדיר משאית, נוכל להסתכל עליה באופנים הבאים:

```
Truck truck = new Truck();
Car car = truck;
Vehicle vehicle = truck;
Object obj = truck;
```

יצרנו עצם מסוג משאית, ואנחנו יכולים להשתמש בו ברמות שונות של הפשטה. בכל רמה יכולות להיות מוגדרות פעולות/שיטות שונות הזמינות לנו.
למשל, אם המתודה GetCost() מוגדרת ב-Truck בלבד, אזי השורות הבאות שגויות:

```
int cost = car.GetCost();
int cost = vehicle.GetCost();
```

הגדרה מחדש של פעולות (דריסת פעולות) אם נגדיר פעולה במחלקה Vehicle ונגדיר אותה מחדש במחלקות היורשות, אזי הפעולה המוגדרת מחדש "דורסת" את קודמתה, והיא הקובעת בכל הרמות.
הפעולה של מחלקת העל מוסתרת.

```
public class Vehicle
{
    public virtual string GetMyClass() { return “I am a Vehicle”; }
}
public class Car :  Vehichle
{
    public override string GetMyClass() { return base.ToString() + “: Car”; }
}
public class Truck : Car
{
    public override string GetMyClass() { return base.ToString() + “ [Truck]”; }
}
```

הערה: מילות המפתח virtual, ו- override אינן קיימות ואינן נחוצות ב-Java

בדוגמה שלמעלה, כל המשתנים: vehicle, car, truck בהפעילם את GetMyClass יקבלו אותה תשובה....

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

בהגדרת פעולה דורסת, ניתן לזמן את הפעולה אותה דרסנו/הסתרנו, על ידי הזימון:
base.GetMyClass(), ובג'אווה: super.GetMyClass(). מה זימון זה יעשה?

<u>תרגלו את עצמכם</u>:

נתונות המחלקות Vehicle, Car, Truck. להלן תיאור חלקי שלהן:

```
public class Vehicle
{
  private Date manufDate;
  public Vehicle(Date date)
  {
    this.manufDate = date;
  }
  public override string ToString()
  {
    return "production day: " + manufDate;
  }
}
```

```
public class Car : Vehicle
{
  protected string brand;


            base(date)
  {
    this.brand = brand;
  }
}
```

```
public class Truck : Car
{
  private string model;
  public Truck(string brand, string model, Date date) : base(brand, date)
  {
    this.model = model;
  }
  public override string ToString()
  {
     return "Truck: " + brand + " " + model + "\n" + base.ToString();
  }
}
```

מה יודפס בביצוע הקוד הבא, בשורות הממוספרות 1 עד 6.

```
Truck truck = new Truck("Volvo", "V5000", new Date(25,1,2010));
Console.WriteLine (truck);          // (1)
Console.WriteLine ((Car)truck);  // (2)
Console.WriteLine ((Vehicle)truck);   // (3)
Car c = new Car("Mercedes", new Date(17,7,2011));
Console.WriteLine (c);              // (4)
Console.WriteLine ((Object)c);   // (5)
Console.WriteLine ((Truck)c);   // (6)
```

פתרון שאלה 2:

(1) Truck: Volvo V5000
production day: 25/1/2010
(2) Truck: Volvo V5000
production day: 25/1/2010
(3) Truck: Volvo V5000
production day: 25/1/2010
(4) production day: 17/7/2011
(5) production day: 17/7/2011
(6) שגיאת זמן ריצה. המרה שגויה

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

שאלה 3: [EVI: ירושה, דריסה, זיהוי שגיאות, מעקב, על בסיס 19/2013, 14/2016]

לפניך המחלקות FF, SS .

שני הסעיפים הבאים א ו ב מתייחסים למחלקות אלו, אך אינם קשורים זה לזה. ענה על שניהם.

```
public class FF
{
    protected int num;
    public FF()
    {
        this(10);
        System.out.println("FF(0)");
    }
    public FF(int num)
    {
        this.num = num;
        System.out.print("FF(1): ");
    }
    public void setNum(int num)
    {
        this.num = num;
    }
    public void add( int num)
    {
        this.num += num;
    }
    public String toString()
    {
        return "FF: num = " + this.num;
    }
}
```

```
public class SS extends FF
{
    private int num;
    public SS()
    {
        this(5);
        System.out.println("SS(0)");
    }
    public SS(int num)
    {
        super(num*2);
        this.num = num;
        System.out.print("SS(1): ");
    }
    public SS(int num1, int num2)
    {
        super(num1);
        this.num = num2;
        System.out.print("SS(2): ");
    }
    public void setNum(int num)
    {
        this.num = num;
    }
    public void setSupNum(int num)
    {
        super.num = num;
    }
    public void add(int num)
    {
        this.num += num;
    }
    public String toString()
    {
        String str = "SS: num = " + this.num;
        str += ", " + super.toString();
        return str;
    }
}
```

```
public class Program
{
   public static void main(String[] args)
   {
(1)    FF f1 = new FF(4);
(2)    Object o1 = f1;
(3)    o1.setNum(7);
(4)    SS s1 = new SS(13);
(5)    FF f2 = s1;
(6)    if (f2 instanceof SS)
           f2.setSupNum(7);
(7)    f2 = o1;
(8)    SS s2 = (SS)f1;
   }
}
```

<u>סעיף א</u>

במחלקה Program בפעולה הראשית כתבו את ההוראות הבאות.

קבע לגבי כל אחת מהשורות (1) – (8) אם היא תקינה או לא.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

אם היא תקינה , רשום את הפלט שמתקבל, אם אינה תקינה, ציין אם זו שגיאת הידור (קומפילציה) או שגיאת זמן ריצה. חובה לנמק תשובתך. במידה ושגיאת הידור, הצע דרך לתיקון.

סעיף ב

לפניך סדרה הוראות בפעולה הראשית, עקוב אחר ההוראות בעזרת תרשים עצמים ורשום את הפלט.

```
(1)   FF a = new FF();
(2)   FF b = new SS();
(3)   System.out.println("a:"+a);
(4)   System.out.println("b:"+b);
(5)   a.setNum(-2);
(6)   b.setNum(12);
(7)   System.out.println("a:"+a);
(8)   System.out.println("b:"+b);
(9)   b.add(2);
(10)  System.out.println("b:"+b);
(11)  ( (SS)b ).setSupNum(12);
(12)  System.out.println("b:"+b);
(13)  SS c = new SS(1, 2);
(14)  System.out.println("c:"+c);
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פתרון שאלה 3: [Evi]

סעיף א: איתור שגיאות והצעה לתיקון 12=1.5x8

	ההוראה	תקין/לא תקין	הצעת תיקון
1	FF f1 = new FF(4);	תקין. יש פעולה בונה כזו, הצבת עצם מטיפוס FF במשתנה מאותו טיפוס. פלט FF(1):	
2	Object o1 = f1;	תקין. כל המחלקות יורשות ממחלקה Object ולכן ניתן להציב במשתנה זה. יש המרה בלתי מפורשת כלפי מעלה.	
3	o1.setNum(7);	שגיאת הידור. ל Object אין פעולה כזו setNum .	((FF) o1).setNum(7);
4	SS s1 = new SS(13);	תקין. יש פעולה בונה כזו ומציבים במשתנה מאותו הטיפוס. פלט FF(1):SS(1)	
5	FF f2 = s1;	תקין. SS יורשת מ FF . ניתן להציב משתנה ממחלקה יורשת במחלקה העל. המרה בלתי מפורשת כלפי מעלה	
6	if (f2 instanceof SS) f2.setSupNum(7);	שגיאת הידור. נקודת המבט של f2 היא מטיפוס המחלקה FF . למחלקה זו אין פעולה setSupNum	if (f2 instanceof SS) ((SS)f2).setSupNum(7);
7	f2 = o1;	שגיאת הידור. אין התאמה בטיפוסים. 2F מטיפוס FF אי אפשר להציב אליו עצם מטיפוס Object	יש לבצע המרה מפורשת כלפי מטה f2 = (FF) o1;
8	SS s2 = (SS)f1;	שגיאת זמן ריצה. נוצר כ FF . המרות מתבצעות בזמן ריצה.	

סעיף ב מעקב עם תרשים עצמים

```
(1)  FF a = new FF();
(2)  FF b = new SS();
(3)  System.out.println("a:"+a);
(4)  System.out.println("b:"+b);
(5)  a.setNum(-2);
(6)  b.setNum(12);
(7)  System.out.println("a:"+a);
(8)  System.out.println("b:"+b);
(9)  b.add(2);
(10) System.out.println("b:"+b);
(11) ( (SS)b ).setSupNum(12);
(12) System.out.println("b:"+b);
(13) SS c = new SS(1, 2);
(14) System.out.println("c:"+c);
```

1) <FF> a → FF num = 10 output: FF(1): FF(0)

2) <FF> b → SS <FF>num = 10 <SS>num = 5 output: FF(1): SS(1): SS(0)

3) output: a:FF: num = 10

4) output; b:SS: num = 5, FF: num = 10

5) <FF> a → FF num = -2

6) <FF> b → SS <FF>num = 10 <SS>num = 12

7) output: a:FF: num = -2

8) output; b:SS: num = 12, FF: num = 10

9) <FF> b → SS <FF>num = 10 <SS>num = 14

10) output; b:SS: num = 14, FF: num = 10

11) <FF> b → SS <FF>num = 12 <SS>num = 14

12) output: b:SS: num = 14, FF: num = 12

13) <SS> c → SS <FF>num = 1 <SS>num = 2

output: FF(1): SS(2):

14) output: c:SS: num = 2, FF: num = 1

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פלט הקוד

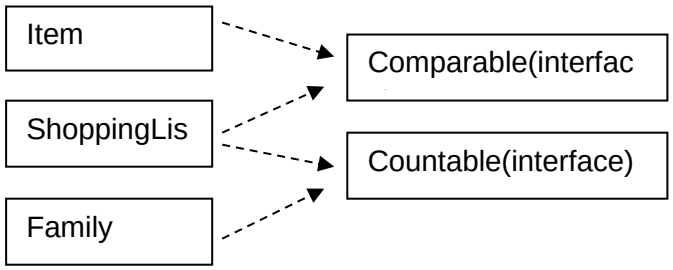
```
FF(1): FF(0)
FF(1): SS(1): SS(0)
a:FF: num = 10
b:SS: num = 5, FF: num = 10
a:FF: num = -2
b:SS: num = 12, FF: num = 10
b:SS: num = 14, FF: num = 10
b:SS: num = 14, FF: num = 12
FF(1): SS(2): c:SS: num = 2, FF: num = 1
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

8. ממשקים [EVI , לקוח מספר תכנות מבנה נתונים של הוראת המדעים ובגרויות]

שאלה 1 [יחסים בין ממשקים למחלקות, המרות כלפי טיפוס הממשק, הגדרת מחלקה המממשת ממשק]
השאלה לקוחה ממבחני חזרה , צוות מדעי המחשב , האוניברסיטה העברית בירושלים, הוראת המדעים.

א- התבונן בתרשים הבא המתאר יחסים בין מחלקות וממשקים



כתוב אילו מחלקות מתוארות בתרשים,
ואילו מימשקים ממשת כל מחלקה.

ב- נתונות שלוש פעולות בונות למחלקות Family, Item, ShoppingList

public **Family** (int numOfParents, int numOfChildren)	בונה משפחה על פי הפרמטרים הנתונים
public **Item**(String name, double price)	בונה מוצר על פי הפרמטרים הנתונים
public **ShoppingList**(Item[] itemList)	בונה רשימת קניות על פי מערך נתון של מוצרים

כמו כן, נתונים שני המימשקים Comparable בר-השוואה ו Countable בר-מנייה

```
public interface Comparable
{
      public boolean isEqual(Object other);
      public boolean isLessThan( Object other);
}
```

```
public interface Countable
{
      public int count();
}
```

בהינתן ההוראות הבאות והיחסים שהוגדרו בסעיף א',
קבע עבור כל אחת מ 6 השורות הממוספרות
האם היא:
(1) תגרום לשגיאה (2) לא תגרום לשגיאה.
הנח כי כל יתר השורות תקינות לחלוטין.
בכל מקרה בו יש שגיאה, הסבר בקצרה מהי השגיאה.
לא יתקבל ניקוד ללא מתן הסבר!

```
Family myFamily = new Family(2,3);
Item item1 = new Item("butter",4.5);
Item item2 = new Item("bread", 7.5);
Item[] itemList = {item1,item2};
ShoppingList sList = new
ShoppingList(itemList);
Comparable comp1;
Countable count1;

comp1 = sList;                    // -- 1 --
count1 = item1;                   // -- 2 --
count1 = sList;
Item myItem = (Item)count1;       // -- 3 --
count1 = (ShoppingList)comp1;     // -- 4 --
int c1 = sList.count();           // -- 5 --
int c2 = comp1.count();           // -- 6 --
```

ג- ממש את המחלקה **Family** במלואה.
על המחלקה לקיים את היחס המתואר בסעיף א'
ולהכיל את הפעולה הבונה שמוגדרת בסעיף ב'.
הוסף פעולות או תכונות הנראות לך הכרחיות
למימוש המחלקה.
הנח שפעולת המנייה מחזירה את מספר הנפשות
במשפחה (הורים + ילדים)

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פתרון שאלה 1 [נכתב על ידי Evi]

א-

המחלקה	ממשת מימשק
Item	ממשת מימשק Comparable
ShoppingList	ממשת את המימשקים Comparable , Countable
Family	ממשת מימשק Countable

ב- שורה 1: **תקינה** . sList הוא עצם מטיפוס המחלקה ShoppingList אשר ממשת את המימשק Comaprable . comp1 הוא עצם מטיפוס המימשק Comparable לכן ההמרה לעצם מטיפוס המימשק תקינה

שורה 2: **לא תקינה ותהייה שגיאת הידור.** Item1 הוא עצם מטיפוס המחלקה Item אשר איננה ממשת את המימשק Countable. Count1 הוא עצם מטיפוס המימשק Countable . המרה כזו אינה חוקית.

שורה 3: **לא תקינה** ותגרור שגיאת ריצה**.** count1 הוא עצם מטיפוס המימשק Countable והוא מכיל הפנייה לעצם מטיפוס ShoppingList . ShoppingList מממשת את המימשק Countable ולכן הפעולה תעבור הידור אבל בזמן ריצה כאשר יש נסיון לעשות המרה לטיפוס מהמחלקה Item תופיע שגיאת ריצה. אי אפשר להמיר עצם מטיפוס ShoppingList לעצם מטיפוס Item .

שורה 4: **תקינה**. comp1 מכיל את העצם sList שהוא עצם מטיפוס ShoppingList ולכן ניתן לבצע המרה חזרה כלפי מטה אל הטיפוס שממנו נוצר.

שורה 5: **תקינה**. sList הוא עצם מטיפוס המחלקה ShoppingList אשר מממשת את המימשק Countable ולכן היא מכירה את הפעולה count ויכולה להפעיל אותה.

שורה 6: **לא תקינה ותהיה שגיאת הידור**. Comp1 הוא עצם מטיפוס המימשק Comparable . מימשק זה אינו מכיל את הפעולה count לכן לא יכיר אותה ולא יכול לזמן אותה.

ג- מימוש המחלקה **Family**

```
public class Family implements Countable
{
      private int numOfParents;          // מספר ההורים
      private int numOfChildren;         // מספר הילדים

   public Family(int numOfParents, int numOfChildren)
   {// פעולה בונה על פי מספר הורים ומספר ילדים
      this.numOfParents = numOfParents;
      this.numOfChildren = numOfChildren;
   }

   public int count()
   {// מחזיר את מספר הנפשות במשפחה
      return this.numOfParents+this.numOfChildren;
   }
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

שאלה 2 [EVI, בהשראת בגרות 381_15_2017 , 205-19_2017]

לפניך הממשקים **Inter1**, **Inter2**, **Inter3** והמחלקות **A**, **B**, **C**, **D** ו **Program** .

public interface **Inter11**
{
public boolean oneA(Object o);
public int oneB(int num);
}
	public class **A** implements Inter1
	{
	….
	}

public interface **Inter2**
{
public void two();
}
	public class **B** implements Inter2
	{
	….
	}

public interface **Inter3**
{
public void three();
}
	public class **C** extends B implements Inter3
	{
	…
	}

public class **Program**
{
public static void main(String[] args)
{ … }
	public class **D** implements Inter3
	{
	….
	}

א- צייר היררכית המחלקות והממשקים.

ב- כתוב את שמות הפעולות שחייבים לממש בכל אחת מהמחלקות A, B , C , D . נמק תשובתך.

ג- עבור כל אחד מהקטעים הבאים (a) – (d) שלפניך, אם נכתוב אותם בפעולה הראשית במחלקה Program האם הקוד תקין או לא.

נמק תשובתך בכל מקרה !

במידה וההוראה שגוייה, ציין אם זו שגיאת תחביר (הידור) או ריצה.

a	Inter1 i1 = new Inter1();
b	Inter2 i2 = new C();
c	B b = new C();
d	D d = b;
e	Inter3 i3 = (C)b;
f	D d = i3;
g	i3.three();

ד- הוסיפו לפעולה הראשית במחלקה Program את ההוראות הבאות:

```
Inter2[] arr = new Inter2[2];
arr[0] = new B();
arr[1] = new C();
for (int k = 0; k < arr.length; k++)
    arr[k].two();
```

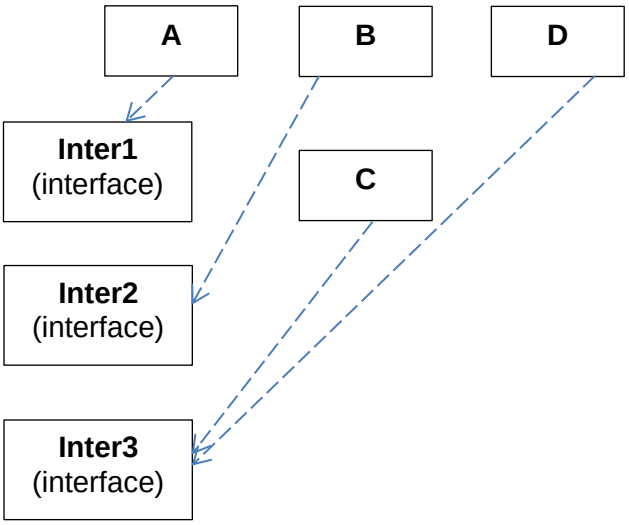
בצע שינויים מתאימים במחלקות הנדרשות בפרויקט כדי שיתקבל הפלט הבא בעקבות הרצת הפעולה הראשית במחלקה Program .
כתוב את השינוי וציין באיזו מחלקה בצעת אותו.

```
I am two of B
I am two of C
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

פתרון שאלה 2 [פתרון של EVI]

א- היררכית המחלקות והממשקים.



ב- כתוב את שמות הפעולות שחייבים לממש בכל אחת מהמחלקות A, B , C , D . נמק תשובתך.

המחלקה	הפעולות	נימוק
A	public boolean oneA(Object o) { } public int oneB(int num) { }	ממשת את הממשק Inter1
B	public void two { }	ממשת את הממשק Inter2
C	public void three() { }	אין צורך לממש את הפעולה two כי יורשת אותה מהמחלקה B אבל יש צורך לממש את הפעולות מהממשק Inter3 , אותו היא ממשת
D	public void three() { }	ממשת את הממשק Inter3

ג- עבור כל אחד מהקטעים הבאים (a) – (d) שלפניך, אם נכתוב אותם בפעולה הראשית במחלקה Program האם הקוד תקין או לא. במידה וההוראה שגוייה, ציין אם זו שגיאת תחביר (הידור) או ריצה.

a	Inter1 i1 = new Inter1();	**שגיאת תחביר**. אין ליצור עצם מטיפוס הממשק
b	Inter2 i2 = new C();	**תקין**. המחלקה C ממשת את הממשק Inter2 ולכן ניתן לבצע המרה לא מפורשת כלפי מעלה מטיפוס המחלקה לטיפוס הממשק אותו היא ממשת.
c	B b = new C();	**תקין**. C יורש מ B ולכן ניתן ליצור עצם ממחלקת הבת ולהמיר אותו המרה לא מפורשת כלפי מעלה למחלקה אותה מרחיב.
d	D d = b;	**שגיאת תחביר**. אין קשר בין המחלקות. חוסר התאמה בטיפוסים.
e	Inter3 i3 = (C)b;	**תקין**. b נוצר מטיפוס המחלקה C ולכן ניתן לבצע המרה מפורשת ממחלקת העל B אל מחלקת הבת אשר ממנה נוצר. בנוסף C ממש את הממשק Inter3 ולכן ניתן לבצע עליו המרה מהמחלקה לטיפוס הממשק אותו הוא ממש.
f	D d = i3;	**שגיאת תחביר**. אמנם גם מחלקה D ממשת את הממשק Inter3 אבל i3 נוצר מטיפוס C. אין קשר בין הטיפוסים.
g	i3.three();	**תקין**. לממשק inter3 יש פעולה כזו three() ולכן יכול לבצע אותה.

ד- הוסיפו לפעולה הראשית במחלקה Program את ההוראות הבאות:

```
Inter2[] arr = new Inter2[2];
arr[0] = new B();
arr[1] = new C();
for (int k = 0; k < arr.length;
k++)
```

בצע שינויים מתאימים במחלקות הנדרשות בפרויקט כדי שיתקבל הפלט הבא

```
I am two of B
I am two of C
```

בעקבות הרצת הפעולה הראשית במחלקה Program . כתוב את השינוי וציין באיזו מחלקה בצעת אותו.

המחלקה	השינוי
B	public void **two**() { System.out.println("I am two of B"); }
C	public void **two**() { System.out.println("I am two of C"); }

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

9. מחלקות מופשטות [לא חובה, אבל מומלץ אם יש זמן ללמד]

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

10. מילון מושגים (כולל מילים נרדפות ושמות המושגים באנגלית) [מיכל + EVI]

מושג	באנגלית	משמעות	דוגמא
הכמסה	Encapsulation	מימוש עקרון הסתרת המידע. חסימת גישה למשתני המחלקה מבחוץ. ישנן מספר הרשאות גישה: **public** – ציבורי, כל אחד יכול לגשת **protected** מאפשר גישה למחלקות היורשות **private** – פרטי, גישה רק בתוך המחלקה עצמה	נפוץ בפרק זה: מחלקת מכונית יורשת מכלי תחבורה, אם נרצה שלמחלקת מכונית תוכל לעשות שימוש בתכונות של מחלקת כלי תחבורה- נגדיר את התכונות של מחלקת כלי תחבורה כ-protected.
ירושה	Inheritance	היכולת להרחיב (extends) מחלקה קיימת. המחלקה החדשה מקבלת בירושה את כל התכונות והפעולות של מחלקת הבסיס. בין המחלקה היורשת למחלקת הבסיס מתקיים יחס "**סוג של**".	מחלקת "מכונית" יורשת ממחלקת "כלי תחבורה" ולכן מקבלת את כל התכונות והפעולות שלה. מקיים בין המחלקות היחס: מכונית היא סוג של כלי תחבורה
רב צורתיות- פולימורפיזם	Polymorphism	• היכולת להסתכל על אותו אובייקט בצורות שונות. • הסתכלות אחידה על אובייקטים שונים ויחד עם זאת להתאים לכל אחד את ההתנהגות שמתאימה לו.	• אובייקט מטיפוס יאכטה, ניתן להסתכל עליו כעל "סירה" או כעל "כלי תחבורה" • אוטובוס, יאכטה ומטוס- ניתן להסתכל עליהם כעל כלי תחבורה ולהפעיל את פונקציית toString שתחזיר מחרוזות שונות המתאימות לכל אחד מהכלי תחבורה.
ממשקים	Interface	הגדרת התנהגות ללא מימוש. בממשק נגדיר פעולות ללא מימוש וקבועים. מחלקה יכולה לבחור לממש את הממשק אך היא מתחייבת לממש את כל הפעולות שהוגדרו בו. "**מתפקד כ**"	
הפשטה	Abstraction	הגדרת מחלקה **מופשטת** מייצגת מושג או רעיון מופשט, שימושי כאשר יש מכנה משותף בין מחלקות שונות. כולן ירשו מהמחלקה המופשטת ולכל אחת מהן מימוש ייחודי משלו לפעולות השונות. לא ניתן ליצור עצם ממחלקה מופשטת.	לדוגמה נגדיר מושג "**צורה**" בעלת שם, מיקום על המסך וצבע. ממנה ירשו צורות שונות, למשל: מלבן, משולש וכדומה. לגבי כל צורה יורשת ניתן לצייר אותה על המסך, לחשב את השטח שלה ועוד.
העמסה	Overloading	**העמסת פעולות**. היכולת להגדיר מספר פעולות בעלות אותו שם הנבדלות ביניהן בפרמטרים המועברים לפעולה במספר או בטפוס.	הגדרה של מספר בנאים במחלקה: בנאי שלא מקבל פרמטרים, בנאי שמקבל פרמטר מסוג שלם, בנאי שמקבל פרמטר מסוג מחרוזת, בנאי שמקבל שני פרמטרים וכו'
דריסה	Overriding	**דריסת פעולות**. מימוש מחדש של פעולה במחלקת הבת שירשה ממחלקת העל. הפעולה זהה בחתימתה לפעולה המקורית., כלומר אותו השם ואותן הפרמטרים המעברים לפעולה.	במחלקת מכונית הוגדרה הפעולה toString המחזירה מחרוזת המתארת את המכונית (יצרן ושנת ייצור). מחלקת אוטובוס היורשת ממחלקת מכונית קיבלה את פעולה זו אך היא לא מתאימה לה. כתיבה מחדש של פעולת toString במחלקת אוטובוס נקראת דריסה.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

המרה	Casting	ישנן שתי סוגי המרות:
		1. **upCasting**- המרה כלפי מעלה. השמה של עצם מסוג תת- מחלקה בתוך עצם ממחלקת האב. נחשבת **להמרה בטוחה**.
		2. **downCasting**- המרה כלפי מטה ממחלקת האב אל מחלקת אל מחלקת הבן. **המרה לא בטוחה.** נעשה זאת בצורה מפורשת (כתיבת שם המחלקה בסוגריים לפני העצם) ורק כשאנחנו יודעים בוודאות שאכן העצם נוצר כעצם ממחלקת הבן.
		יש לבדוק שאכן ניתן לבצע המרה זו בגאווה, בעזרת **instanceof** בסי שארפ, בעזרת **typeof**
			1. דוגמא ל-upCasting:
			Car b1 = new Bus();
			אוטובוס הוא סוג של מכונית וניתן להסתכל עליו מנקודת מבט של מכונית.
			
			2. דוגמא ל -downCasting:
			Car c1 = (Bus) b1;
			b1 נוצר מטיפוס Bus אבל נקודת המבט שלו היה של Car .
			בוצעה המרה מפורשת כלפי מטה מטיפוס Car אל טיפוס מחלקת הבת Car.
			if (c1 instanceof Bus) { }
			
			if (c1 typeof(Bus)) { }

תכונת מופע של עצם	**תכונת מחלקה**
תכונה השייכת למופע של מסוים של העצם שנוצר.	תכונה השייכת למחלקה עצמה
נוצר רק כאשר נוצר העצם	נוצרת כבר בעת הצהרה של שם המחלקה
גישה אליה מחוץ למחלקה תהייה על ידי **שם העצם**	גישה אליה מחוץ למחלקה תהייה על ידי **שם המחלקה**
בתוך המחלקה גישה אליה בעזרת המילה **this**	בתוך המחלקה גישה אליה בעזרת המילה **שם המחלקה**

דוגמה

```
private double currentAmount ;

Bucket b1 ;    Bucket של מופע עדיין אין

b1=new Bucket();      נוצר מופע
b1.fill(3);
System.out.println(b1.getCurrentAmount( ));
```

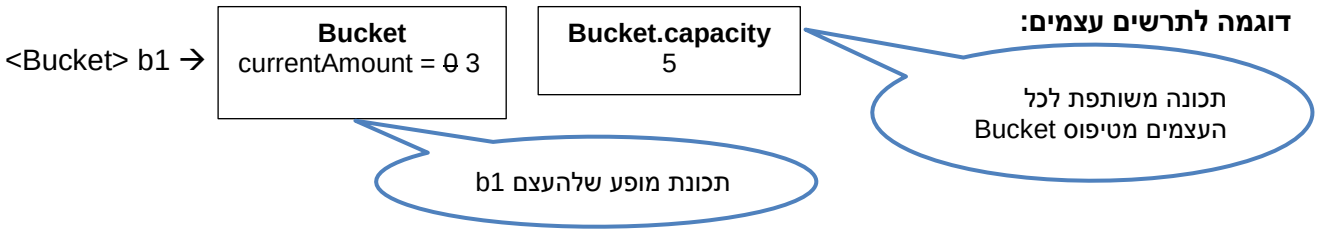
הערות:

גישה אל תכונה ופעולה של עצם

שם העצם.פעולה()
b1.getCurrentAmount()

```
private static int capacity=5;

תכונה של מחלקה נוצרת עוד לפני יצירת המופע הראשון. היא שייכת למחלקה ולא לעצם ספציפי

*static- שייכת למחלקה ולא לעצם ספציפי

פנייה למשתנה מחלקה:
שם המחלקה. שם המשתנה
Bucket . capacity;

public int getCapacity
{
      return Bucket.capacity;
 }

** capacity-    Bucket של תכונה
```

דוגמה לתרשים עצמים:



ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

11. נספח - תמ"ע באנדרואיד [ענת]

כל מסך באפליקציה מוגדר ע"י המחלקה Activity. בבואנו ליצור מסך חדש עלינו לבנות מחלקה חדשה המרחיבה (יורשת) את מהמחלקה Activity.[1]

כאשר נפתח מסך באפליקציה מערכת ההפעלה מזמנת את הפעולה onCreate המוגדרת במחלקה Activity.

ועלינו להגדיר אותה מחדש (**override**) בתת המחלקה שנבנה.[2]

כל רכיב גרפי שנרצה למקם על המסך יהיה מסוג View. כלומר כל הכפתורים, תיבות הטקסט, תמונות וכו' הם כולם סוג של View. כל המחלקות הללו יורשות מהמחלקה View.[3]

כאשר אנו מקבלים הפניה לאותם רכיבים באמצעות הפעולה findViewById אנו מקבלים חזרה הפניה מהטיפוס View. במידה ונרצה לגשת לתכונות של אותו רכיב שמוגדרות בתת המחלקה ולא במחלקת העל View עלינו לבצע המרה כלפי מטה (**downcasting**).[4]

כאשר נרצה להגיב ללחיצה על כפתור יהיה עלינו להגדיר מאזין (Listener). לשם כך נממש את הממשק OnClickListener.[5]

מה שיחייב אותנו לממש את פעולת הממשק onClick.[6]

```
public class MainActivity extends Activity implements View.OnClickListener {[1] [5]

   private TextView tvName;[3]
   private Button btnSave;


 @Override[2]
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);


       tvName = (TextView)findViewById(R.id.tvName);[4]
       btnSave = (Button) findViewById(R.id.btnSave);

       btnSave.setOnClickListener(this);
   }


   @Override[6]
   public void onClick(View view) {
       Toast.makeText(this, "click", Toast.LENGTH_LONG).show();
   }
}
```

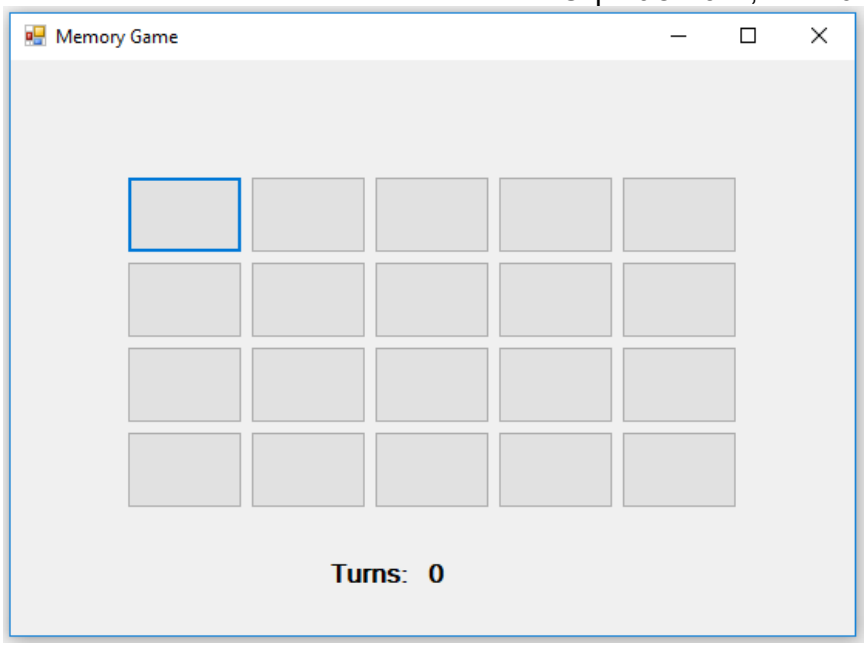
ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

12. נספח - WinForm - שימוש בירושה , C# בלבד [משה]

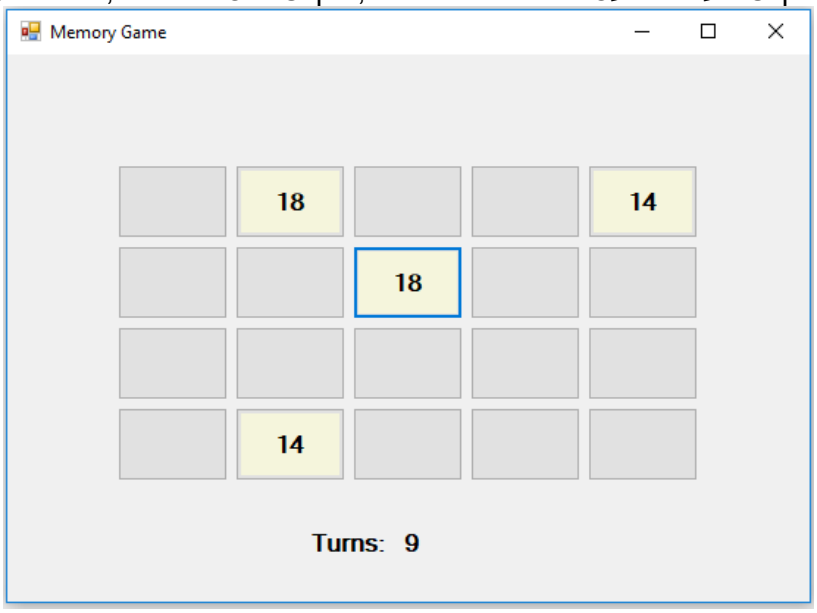
<u>הנחת עבודה:</u> המורה והתלמידים מכירים את סביבת Windows Forms Applications, ברמה בסיסית.

משחק הזכרון: התרגיל אינו מכיל את כל ההנחיות למימוש המשחק עח ידי התלמידים, אלא מתמקד בהיבט של ירושה הרלוונטי לחוברת תרגילים זו.

במשחק הזכרון מחולקים זוגות קלפים באופן אקראי עם פניהם כלפי מטה, ומסודרים במטריצה, מערך דו-מימדי. בדוגמה שלפנינו יש 10 זוגות, ובסה"כ 20 קלפים:



על השחקן לחשוף שני קלפים על ידי העכבר. אם זהו זוג תואם, הקלפים ישארו גלויים, אחרת יוסתרו שנית:



נוכל לממש את המשחק בכמה אופנים, נתמקד בשניים:

1. נגדיר מערך דו-מימדי של כפתורים, ובמקביל אליו מערך דו-מימדי של שלמים:

```
Button [,] board = new Button[4,5];
int [,] cards = new int [4,5];
```

שני המערכים ביחד מספקים לנו פתרון מלא: מצד אחד כל התמיכה הגרפית במיקום, צבע, פונטים, לחיצה על עכבר ועוד. ומצד שני מערך השלמים מספק לנו את ערך הקלף במקום מסוים במערך.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

2. נגדיר מחלקה חדשה קלף, Card, היורשת מ-Button:

```
public class Card : Button
{
  public int cardValue;
  public bool exposed;
}
```

כעת נוכל להגדיר מערך דו-מימדי של קלפים:

```
Card [,] board = new Card[4,5];
```

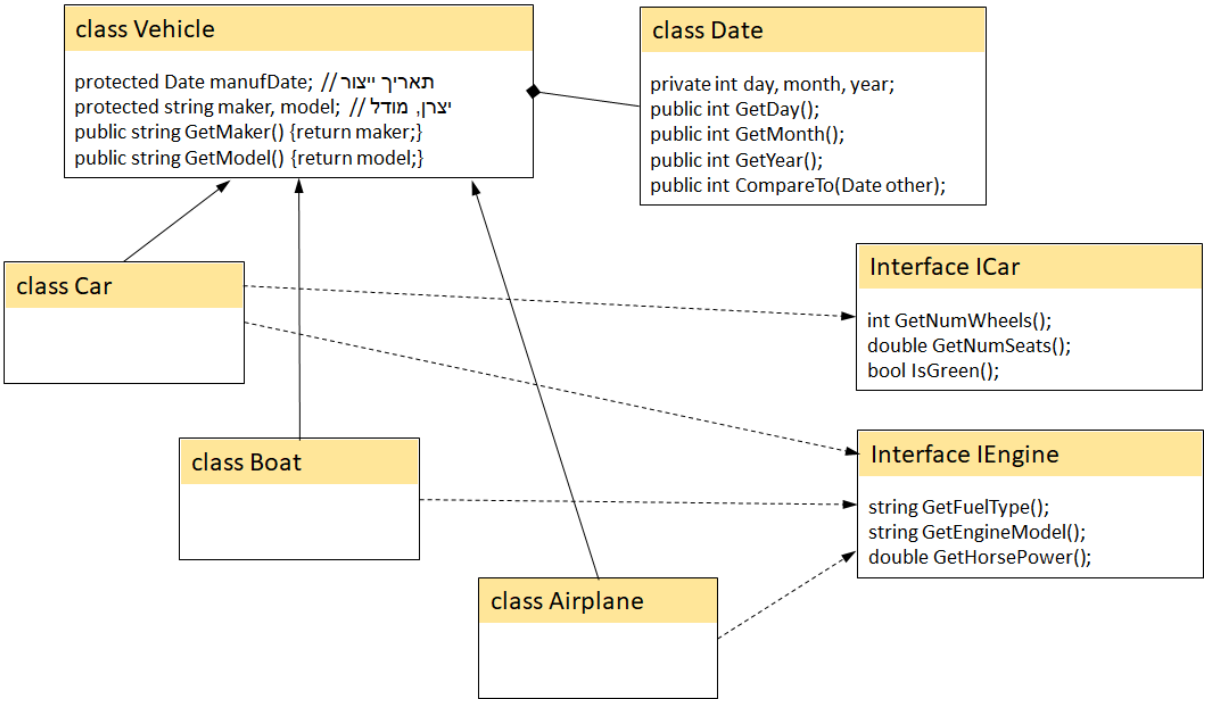
עכשיו בתכנית שלנו אנו מטפלים במערך אחד המכיל את כל היכולות של Button, עם כל התוספות שלנו: ערך הקלף, האם נחשף? ועוד…

נסו בעצמכם.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

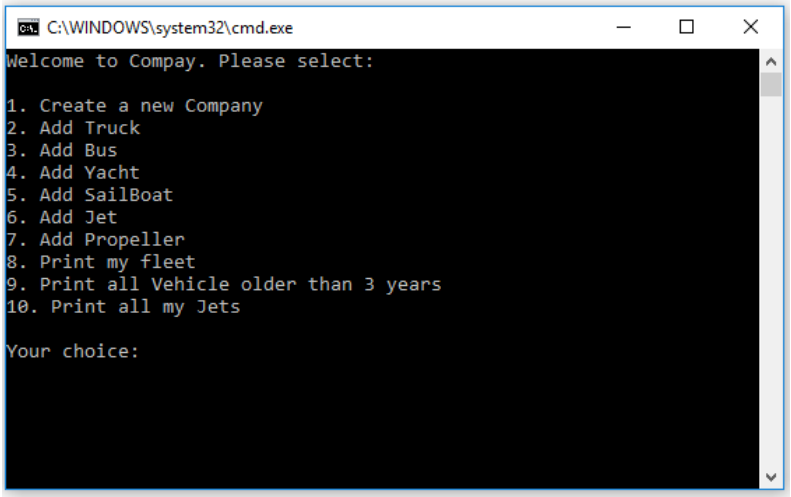
13. נספח - תרגיל מסכם [משה]

דוגמה למבנה אפשרי שיכול לשמש להרכבת שאלה פתוחה על כל הרמות בתמ"ע



התבוננו בתרשים ה-UML למעלה, ובתרשים במצורף לפרק 8, וענו על השאלות הבאות:

1. ממשו את המחלקה תאריך, Date. יש להוסיף:
   a. פעולה בונה המקבלת את כל התכונות: יום, חודש ושנה.
   b. פעולה בונה מעתיקה. המקבלת תאריך אחר, ובונה תאריך זהה.
   c. הפעולה CompareTo מקבלת תאריך אחר ומחזירה 0 אם שני התאריכים זהים, מספר חיובי אם התאריך שהתקבל מאוחר יותר (צעיר יותר) מהתאריך הנוכחי. אחרת תחזיר מספר שלילי.
2. הגדירו את הממשקים ICar, IEngine
3. בנו את המחלקות Vehicle, Car, Boat, Airplane. דאגו להוסיף את כל הפעולות הנחוצות, התכונות, והזימונים לפעולות הבונות.
4. הוסיפו את תת המחלקות המתוארות בתרשים בפרק 8.
5. כיתבו תכנית ראשית המציגה למשתמש את תפריט הפעולות הבא:



זו דוגמה בלבד! ניתן ורצוי להרחיב את מיגוון האפשרויות בתפריט.

הרעיון המרכזי:

בנו מחלקה חברה, Company, אשר לה מספר תכונות: שם החברה, שנת הקמה, שם המנהל, ורשימה של כלי תחבורה מסוג Vehichle.

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח

הפעולות השונות יוסיפו, ימחקו, ישנו אובייקטים הנמצאים ברשימה. האובייקטים יהיו כולם כלי תחבורה, אבל מסוגים שונים. יש פעולות שתעבודנה על כל כלי תחבורה, למשל גיל כלי התחבורה. ויש פעולות המתאימות רק לתת מחלקות.

פתרון [Moshe]

```
public class Vehicle
{
	protected int engineVol; //מנוע נפח
	protected string model; // דגם
	protected int year; // שנת יצור
	protected static string MANUF="Ishimoto";// יצרן.
	public Vehicle(int vol, string model, int year)
	{
			this.engineVol = vol;
		this.model = model;
		this.year = year;
	}
}

public class Truck : Vehicle
{
	private int maxLoad; // משקל מקסימלי
	public Truck(int vol, string model, int year, int load) :  base(vol, model, year)
	{
			this.maxLoad = load;
	}
}

public class Bus : Vehicle
{
	private int maxPassengers; // מספר נוסעים
	public Bus(int vol, string model, int year, int pass) : base(vol, model, year)
	{
		this.maxPassengers = pass;
	}
}
```

ניתן להשתמש בחומרים לצורך הוראה אבל אסור לעשות בהם כל שימוש מסחרי ללא קבלת אישור מצוות הפיתוח