הורשה- משימת סיכום

למדנו בתהליך ההורשה שניתן להשתמש במחלקה הקרויה מחלקת בסיס בתוך מחלקה נגזרת.
כלומר למחלקה יש יכולת לקבל ממחלקה אחרת(ונקרא לה מחלקה נגזרת) את כל התכונות של מחלקה אחרת(ולזו נקרא מחלקת בסיס).

נסתכל על הקוד הבא של המחלקה Animal

```
class Animal
  {
    protected string name;
    protected char gender;
    public Animal(string name, char gender)
    {
      this.name = name;
      this.gender = gender;
    }
  public virtual void print()
    {
      Console.WriteLine("i am an animal");
    }
  }
```

מחלקה פשוטה שיש בה רק פעולה בונה ופעולת הדפסה למסך. זו תשמש אותנו כמחלקת בסיס.

נביט במחלקה Dog הבאה – המחלקה הנגזרת- מחלקה זו יורשת מהמחלקה Animal את

התכונות שם ומגדר ואת כל הפעולות. המחלקה Dog מרחיבה את המחלקה Dog בתכונה speed

```
class Dog: Animal
  {
   private int speed;
    public Dog(int speed, string name, char gender) : base(name, gender)
    {
      this.speed = speed;
    }
    public int GetSpeed()
        {
              return this.speed;
        }
```

```
        public override void print()
    {
      base.print();
      Console.WriteLine("i am a dog");
    }
}
```

שימו לב למספר דברים שלמדנו בשיעור.

1. הגדרת הירושה באופן הבא Dog: Animal
2. למחלקה יש רק תכונה אחת אבל הפעולה הבונה מאתחלת את כל התכונות- גם של המחלקה הנגזרת וגם של מחלקת הבסיס שאותה היא יורשת. ייתכן כמובן שהתכונות יאותחלו בקבוע ולא רק בערך שמתקבל כפרמטר
3. הפעולה print דורסת את הפעולה שאותה היא יורשת ולכן יש לה את אותה החתימה. כדי לקבל את הערכים שיש במחלקת הבסיס נוכל- לא חובה לזמן בתוך הפעולה printשבמחלקה הנגזרת , את הפעולה print שבמחלקת הבסיס אבל... היא מזמנת שוב את הפעולה print הנמצאת במחלקת הבסיס. ושיש לה את אותו שם של פעולה.. הפעולה print במחלקת הבסיס הוגדרה כפעולה וירטואלית והפעולה printבמחלקה הנגזרת מבצעת דריסה override לפעולה הוירטואלית שבמחלקת הבסיס. מעצם העובדה שזו פעולה עם אותה החתימה שבמחלקת הבסיס והיא דורסת פעולה אחרת- היא גם כן הופכת לפעולה וירטואלית שניתן לדרוס אותה .

משימה 1

הוסיפו לתכנית מחלקה דג ומחלקה ציפור. לדג צריך להגדיר האם הוא דג של מים מתוקים או לא. לציפור צריך להגדיר משקל. הוסיפו את כל תכונות Get Set החליפו את הפעולה print בפעולה ToString. התאימו את המחרוזת כך שהיא תחזיר את המחרוזת הקיימת במחלקת הבסיס תוך כדי שירשור המחרוזת למחרוזת במחלקה הקונקרטית.

משימה 2

צרו מופע של שני ציפורים, שני כלבים שני דגים ושתי חיות. הריצו בסביבת העבודה ובחנו את הפלט. השתמשו בפעולה ToString עבור הפלט

פולימורפיזם

ניצור מערך של חיות בתכנית הראשית באופן הבא. העתיקו את הקוד כפי שהוא. שימו לב שבמשימה 2 לא יצרתם מופע של המחלקות באמצעות מערך. אתם יכולים להוסיף אותם כאן- כמובן שיש להתאים את גודל המערך לנתונים שאתם רושמים.

```
Animal[] arr = new Animal[4];

    arr[0] = new Animal(" stam", 'm');
    arr[1] = new Bird(3.5, "pashosh", 'f');
    arr[2] = new Dog(30, "suzy", 'f');
    arr[3] = new Bird(7, "tuky", 'm');
```

משימה 3

עצרו שניה ובחנו היטב את הקוד. שימו לב- סביבת העבודה- יכולה לסבול את השעונות שלנו. בפועל- אין דבר כזה שנקרא חיה. על המשמעות של זה נלמד בהמשך כשנגיע לנושא מחלקות מופשטות- מחלקות אבסטרקטיות. כרגע הניחו לנושא.

משימה 4

צרו מערך בגודל 8 שיכיל את כל החיות שיצרתם במשימה מספר 2.

מדוע ניתן לייצר מערך אחד עבור חיות שונות?

התשובה היא משום שאנחנו אומרים שציפור – זה בעצם סוג של חיה, גם כלב הוא סוג של חיה, כנל גם דג וציפור. למעשה תוך כדי השמה של כלב לאיבר במערך של חיה אנחנו מבצעים המרה כלפי מעלה(upcasting) , המרה של מחלקה נגזרת למחלקת הבסיס. המרה זו מוגדרת כהמרה מרומזת. תהליך זה קרוי פולימורפיזם. רב צורתיות. איבר במערך Animal יכול להיות פעם אחת ציפור ופעם אחרת דג ופעם אחרת כלב. בתוך המערך יהיו אובייקטים של חיה או של נגזרות של חיה

משימה 5

הריצו בתכנית הראשית את הקוד הבא על מערך החיות שיצרתם במשימה 4

```
for (int i = 0; i < arr.Length; i++)
  arr[i]. ToString ();
```

מהן הפעולות שניתן לזמן עבור המערך? רמז- מה הפעולות שאתם רואים אחר הכתיבה

arr[i].

מה שקורה כאן זה שהויזואל סטודיו מסתכל על המערך ובודק מהו כל פריט שיש לנו- והתשובה היא שיש לנו פריטים מסוג חיה. כל כלב/דג/ציפור זה בעצם סוג של חיה והויזואל סטודיו יציג לנו את הנתונים שבוודאות הם נכונים כלומר הנתונים של חיה. מבחינת הויזואל סטודיו מה שיש בכלב- יש גם בחיה

<u>משימה 6</u>

שנו את החתימה של המחלקה ToString בכל המחלקות באופן הבא: מחקו ממחלקת הבסיס את המילה virtual ומהמחלקות הנגזרות מחקו את המילה override ובחנו את הפלט המתקבל. חקרו את השגיאה המתקבלת. הסבר לשגיאה תמצאו בספריית MSDN

פעולה המוגדרת כפעולה וירטואלית נדרסת במחלקות הנגזרות_מחלקת העצם קובעת איזה פעולה תתבצע במצב של פולימורפיזם בו דרסנו על הפעולות של מחלקת האב.

ההסבר למה שהיה לפני שביצענו את השינוי נעוץ בעובדה שאנחנו מזמנים את הפעולה השייכת לעצם המוגדר בזיכרון – Animal. יש לנו אפשרות להציג את הנתונים של המחלקה של העצם המוגדר בזיכרון ולא של הייחוס שלו. אם למחלקות נגזרות יש פעולות הייחודיות רק להן ושלא מוגדרות במחלקה Animal נצטרך לבצע תהליך שנקרא down casting עם שימוש באופרטור is.

<u>משימה 7</u>

לכל אחת מהחיות הוסיפו תכונה ופעולות GetSet המגדורות את הקול שמשמיע החיה.

עבור הכלב - bark

עבור הציפור-tweet

עבור הדג- silence

התאימו גם את הפעולה ToString בהתאם- בכל אחת מהמחלקות. כיצד נוכל לדעת עבור כל חיה בנפרד- מהו הקול שמשמיע אותה החיה?

שימו לב לקוד הבא

```
for (int i = 0; i < arr.Length; i++)
  {
  if (arr[i] is Bird)
    ((Bird)arr[i]).print();
  }
```

הלולאה עובר על מערך מטיפוס Animal ובודקת האם האיבר שעליו האינדקס i מצביע עליו הוא איבר שמוגדר בזיכרון מסוג Bird. אם כן- מתבצעת המרה כלפי מטה down casting. המרה של איבר Animal לאיבר Bird

בשלב זה אם תבחנו את arr[i]. תוכלו לראות את כל הפעולות של המחלקה האמיתית

בזיכרון.

הריצו את הקוד .

<u>מספר נקודות חשובות.</u>

1. רצוי מאוד להימנע מהמרות כלפי מטה. זה תהליך מסוכן שיש להימנע ממנו. סכנה זו נובעת מהעובדה שיש לנו מספר מגוון של מחלקות ולא כל אחת – נרצה להמיר כלפי מטה. למשל- שימו לב לקוד התקין הבא:

```
class Animal
{
  public void run()
  {
    Console.WriteLine("Animal: run()");
  }
}
class Cat : Animal
{
  public void sound()
  {
    Console.WriteLine("Cat: meow()");
  }
}
class Program
{
  static void Main(string[] args)
  {
    Animal a = new Cat();
    execute(a);
  }
  public static void execute(Animal a)
  {
    Cat c = (Cat)a;
    c.sound();
  }
}
```

הפעולה execute מקבלת ייחוס של מחלקת בסיס ומבצעת המרה כלפי מטה לייחוס של מחלקה נגזרת

אבל- מה יקרה אם בפעולה הראשית נגדיר את הקוד הבא

```
static void Main(string[] args)
```

```
{
    Animal a = new Dog();
    execute(a);
     }
```

אפשר להמיר כלפי מטה חיה מסוג חתול לחתול אבל להפוך כלב לחתול?

הקומפיילר יצעק לנו שלא ניתן לבצע סוג כזה של המרה. ולכן ככלל רצוי מאוד לתכנן את העבודה באופן כזה שההמרות כלפי מטה יהיו עד כמה שפחות . ריבוי בצורך בהמרות כלפי מטה מעיד על תכנון לא נכון של המחלקות.

2. המרות כלפי מטה שימושיות באוספים – דוגמת מערך.
3. המרות כלפי מטה הנן המרות מפורשות- חייבים לציין לאיזה מחלקה נגזרת אנו רוצים להמיר את מחלקת הבסיס.
4. המרה כלפי מטה תעשה כאשר אנו רוצים להפעיל אלמנטים ייחודיים של מחלקה נגזרת.

<u>משימה 8</u>

הוסיפו למערך עוד שני כלבים ועוד שני ציפורים. חשבו והציגו את המהירות הממוצעת של כל הכלבים הנמצאים במערך ואת המשקל הממוצע של כל הציפורים שבמערך. בצעו זאת בלולאה אחת בלבד.

<u>משימה 9- משימת חקר כולל בונוס למבחן</u>

קיים אופרטור נוסף as . כיצד אופרטור זה יוכל לסייע לנו בתהליך ההמרה כלפי מטה?

בונוס ינתן עבור מי שיציג עבור הדוגמה שלנו קוד תקין ועובד - כולל צילומי מסך של פלטים.

גודל הבונוס יקבע בהמשך