## המרות בין טיפוסי מחלקות

ב-C#, כאשר עובדים עם טיפוסים מבוססי מחלקות (Class Types), יש שני סוגים של המרות עיקריות:

1. **המרה מרומזת (Implicit Conversion) – "המרה כלפי מעלה" (Upcasting)**
2. **המרה מפורשת (Explicit Conversion) – "המרה כלפי מטה" (Downcasting)**

### המרה מרומזת (Upcasting)

Upcasting הוא תהליך שבו אובייקט של מחלקה נגזרת מומר אוטומטית (ללא צורך ב-Cast) לטיפוס של מחלקת הבסיס שלו. (הפניה מטיפוס אב-בסיס לעצם מטיפוס בן-מחלקה הנגזרת מהבסיס)

**מאפיינים**:

✔️ בטוח לשימוש – לא מאבד מידע.

✔️ מתבצע אוטומטית ע"י הקומפיילר.

✔️ ניתן לגשת רק לשדות ולמתודות הקיימות במחלקת הבסיס.

דוגמא:

```
class Animal
{
    public void MakeSound()
    {
        Console.WriteLine("Some generic animal sound");
    }
}

class Dog : Animal
{
    public void Bark()
    {
        Console.WriteLine("Woof woof!");
    }
}

class Program
{
    static void Main()
    {
        Dog dog = new Dog();
        Animal animal = dog; // Upcasting - המרה מרומזת
        animal.MakeSound();  // תקין
        // animal.Bark();    // לא ניתן לקרוא ל Bark כי השיטה לא מכירה את השיטה שנמצאת במחלקה הנגזרת!
    }
}
```

### מתי משתמשים ב-Upcasting?

- כאשר רוצים לאחסן אובייקטים מטיפוס נגזר בתוך משתנה מטיפוס בסיס.
- כאשר יש פונקציה שמקבלת את מחלקת הבסיס וניתן לשלוח אליה מופע של מחלקה נגזרת.

### המרה מפורשת (Downcasting)

Downcasting הוא תהליך שבו אובייקט של מחלקת בסיס מומר חזרה לטיפוס של מחלקה נגזרת.
**מאפיינים**:
⚠️ **לא בטוח** – אם האובייקט בפועל אינו מהטיפוס המומר אליו, תתרחש שגיאה בזמן ריצה.
⚠️ יש לבצע **Casting ידני** כדי לבצע את ההמרה.

```
class Program
{
    static void Main()
    {
        Animal animal = new Dog(); // Upcasting
        Dog dog = (Dog)animal; // Downcasting מפורש
        dog.Bark(); // תקין כי animal במקור היה Dog
    }
}
```

### בעיה אפשרית בהמרה מפורשת

אם ננסה להמיר מופע **שאינו** מהמחלקה המתאימה, תתרחש שגיאה:

```
Animal animal = new Animal();
Dog dog = (Dog)animal; // שגיאת זמן ריצה: InvalidCastException
dog.Bark();
```

### פתרון באמצעות is ו- as

כדי להימנע משגיאות, ניתן להשתמש ב-is או as:

```
// לבדוק אם ניתן לבצע המרה - שימוש ב-is
if (animal is Dog d)
{
    d.Bark(); // בטוח כי בוצעה בדיקה לפני
}

// שימוש ב-as (יחזיר null אם לא ניתן להמיר)
Dog dog2 = animal as Dog;
if (dog2 != null)
{
    dog2.Bark();
}
```