הורשה

הורשה (Inheritance) היא מנגנון מרכזי ב-OOP המאפשר למחלקה חדשה (המחלקה הנגזרת או הבן) לקבל את התכונות והפעולות של מחלקה קיימת (המחלקה הבסיסית או האב).

מטרות עיקריות:

1. **שימוש חוזר בקוד (Code Reusability)**

המטרה הראשונה היא לחסוך בכתיבת קוד.

- במקום לכתוב את אותן תכונות ופעולות שוב ושוב עבור מחלקות שונות (לדוגמה: שם, גיל, ופעולת "הליכה" עבור מחלקת "תלמיד", "מורה" ו"רופא"), אנחנו כותבים אותן רק פעם אחת במחלקה המשותפת (האב).
- המחלקות החדשות (הבנים) יורשות את כל הקוד הזה אוטומטית. זה הופך את הקוד לקצר יותר וקל יותר לתחזוקה.

2. **מידול יחסי "IS-A" (היגיון וסדר)**

המטרה השנייה היא ליצור מבנה לוגי ברור בתוכנה.

- הורשה משמשת כדי למדל קשר טבעי שבו פריט אחד הוא סוג של פריט אחר.
  - "תלמיד הוא סוג של אדם" (Student IS-A Person).
  - "מכונית היא סוג של רכב" (Car IS-A Vehicle).
- הקשר הזה מאפשר לנו להתייחס לכל הבנים באמצעות האב המשותף שלהם, מה שמעניק גמישות רבה (פולימורפיזם) ומסדר את ההיררכיה של המחלקות בתוכנה בצורה הגיונית.

ב-C#, מחלקה יכולה לרשת רק ממחלקת אב יחידה.

הרשאות גישה:

נגישות מחוץ למחלקה	נגישות מהמחלקות היורשות	תיאור	מפתח
כן ✅	כן ✅	נגישות מלאה וללא הגבלה מכל מקום	public
לא ❌	לא ❌	נגישות מוגבלת: רק מתוך גוף המחלקה עצמה שהגדירה אותו	private
לא ❌	כן ✅	הגדרה ייחודית להורשה: מאפשר גישה מלאה לחבר מתוך המחלקה המגדירה ומכל המחלקות הנגזרות ממנה	protected

protected:

הרשאת הגישה protected היא הפתרון האמצעי בין public ל-private

protected מאפשר למחלקות הנגזרות (הבנים) גישה ישירה לתכונות ופעולות פנימיות של מחלקת האב, שהן חיוניות לבניית ההיררכיה.

פעולות בונות ואתחול עצמים בהורשה

כלל השרשרת הקפדני

כשיוצרים עצם ממחלקה יורשת, הפעולה הבונה של מחלקת הבסיס פועלת ראשונה.
היא נותנת ערכים לתכונות של מחלקת הבסיס. אחר כך פועלת הפעולה הבונה של המחלקה היורשת ונותנת ערכים לתכונות שלה.

קריאה לפעולה הבונה של מחלקת הבסיס בעזרת base

אם הפעולה הבונה של מחלקת הבסיס דורשת פרמטרים, משתמשים במילת המפתח base כדי להעביר לה את הערכים הדרושים.

בדוגמה, הערך של species מועבר לפעולה הבונה של Animal.

```
public class Animal
{
    protected string Species;
    1 reference
    public Animal(string species)
    {
        this.Species = species;
    }
}
1 reference
public class Dog : Animal
{
    private string Name;
    0 references
    public Dog(string name, string species) : base(species)
    {
        this.Name = name;
    }
}
```