موضوع درس اليوم هو الوراثة Inheritance، وهو من المواضيع المهمة في البرمجة الكائنية Object Oriented Programming – OOP. تساعدنا الوراثة على تنظيم الكود، إعادة استخدامه، وتقليل التكرار.

لنفترض أن لدينا فئة عامة اسمها Book تمثل كتابًا، ولدينا نوعان من الكتب:

كتاب ورقي Printed Book وكتاب إلكتروني E-Book.

يوجد بين النوعين خصائص مشتركة، مثل:

عنوان الكتاب.

اسم المؤلف.

بدلًا من كتابة هذه الخصائص مرة في فئة الكتاب الورقي ومرة أخرى في فئة الكتاب الإلكتروني، يمكننا تعريفها مرة واحدة في فئة أم، ثم نجعل الفئات الأخرى ترث منها.

الوراثة هي علاقة بين فئتين:

الفئة الأم: Base Class

الفئة الابنة أو المشتقة: Derived Class

الفئة الابنة ترث الخصائص والدوال الموجودة في الفئة الأم، ويمكنها أيضًا أن تضيف خصائص أو دوال خاصة بها.

الشكل العام للوراثة في لغة C# هو:

```
1  class BaseClass {}
2
3  class DerivedClass : BaseClass {}
```

مثال آخر:

إذا أردنا إنشاء برنامج يمثل أشخاصًا، مثل طلاب ومعلمين، يمكننا إنشاء فئة أم باسم Person تحتوي على الخصائص المشتركة مثل الاسم ورقم الهوية، ثم نجعل فئة Student وفئة Teacher ترثان منها.

بهذه الطريقة نستطيع كتابة الكود المشترك مرة واحدة فقط، مما يجعل البرنامج أكثر تنظيمًا وأسهل في الصيانة.

السؤال التطبيقي

صمم برنامجًا يحاكي نظام إدارة مكتبة.

المطلوب:

1. إنشاء فئة أم باسم Book تحتوي على الخصائص:
   - title
   - author
2. إنشاء فئة فرعية باسم Pbook تمثل كتابًا ورقيًا، وترث من الفئة Book، وتضيف الخاصية:
   - Pages
3. إنشاء فئة فرعية باسم Ebook تمثل كتابًا إلكترونيًا، وترث من الفئة Book، وتضيف الخاصية:
   - fileSize
4. إضافة دالة لطباعة معلومات الكتاب. من المفضل تعريف دالة طباعة مشتركة في الفئة الأم، ثم استخدامها أو تطويرها في الفئات الفرعية.
5. في البرنامج الرئيسي، قم بتعريف كائن من نوع Pbook وكائن من نوع Ebook، ثم اطبع جميع معلوماتهما.

```
using System;

class Book {
  public string title;
  public string author;

  public Book(string title, string author) {
    this.title = title;
    this.author = author;
  }

  public virtual void PrintInfo() {
    Console.WriteLine("Title: " + title);
    Console.WriteLine("Author: " + author);
  }
}

class Ebook : Book {
  public double fileSize;

  public Ebook(string title, string author, double fileSize)
      : base(title, author) {
    this.fileSize = fileSize;
  }

  public override void PrintInfo() {
    base.PrintInfo();
    Console.WriteLine("File Size: " + fileSize + " MB");
    Console.WriteLine("----------------------");
  }
}
```

```
class Pbook : Book {
  public int pages;

  public Pbook(string title, string author, int pages) : base(title,
      author) {
    this.pages = pages;
  }

  public override void PrintInfo() {
    base.PrintInfo();
    Console.WriteLine("Pages: " + pages);
    Console.WriteLine("----------------------");
  }
}
```

```
class Program {
  static void Main(string[] args) {
    Pbook printedBook = new Pbook("Learning C#", "Ahmad Ali", 250);
    Ebook electronicBook = new Ebook("OOP Basics", "Sara Khaled", 5.7);

    Console.WriteLine("Printed Book Information:");
    printedBook.PrintInfo();

    Console.WriteLine("Electronic Book Information:");
    electronicBook.PrintInfo();
  }
}
```

تلخيص الدرس

من خلال هذا الدرس تعلمنا أن
الوراثة تساعدنا على:

- تقليل تكرار الكود.
- إعادة استخدام الخصائص والدوال المشتركة.
- تنظيم العلاقة بين الفئات.
- جعل البرنامج أوضح وأسهل للتطوير.

ملاحظة مهمة:

في لغة C# لا يمكن للفئة أن ترث من أكثر من فئة أم واحدة مباشرة، أي أن C# لا تدعم الوراثة المتعددة بين الفئات.