

Object Oriented Programming



K. Scott Allen

@OdeToCode

Pillars of OOP



Encapsulation



Inheritance



Polymorphism

Encapsulation

```
public class GradeBook
{
    public GradeBook()...

    public GradeStatistics ComputeStatistics()...

    public void WriteGrades(TextWriter destination)...
```

```
    public void AddGrade(float grade)...
```

```
    public string Name...
```

```
    public event NameChangedDelegate NameChanged;
```

```
    private string _name;
```

```
    private List<float> grades;
```

```
}
```

```
public void WriteGrades(TextWriter destination)
{
    for (int i = grades.Count; i > 0; i--)
    {
        destination.WriteLine(grades[i-1]);
    }
}
```

Inheritance

```
public class NameChangedEventArgs : EventArgs
{
    public string ExistingName { get; set; }
    public string NewName { get; set; }
}
```

```
public class A
{
    public void DoWork()
    {
        // ... work!
    }
}

public class B : A
{
}

public class C : B
{
}
```

Polymorphism

- Polymorphism == “many shapes”
 - One variable can point to different types of objects
 - Objects can behave differently depending on their type

```
public class A : Object
{
    public virtual void DoWork()
    {
        // ...
    }
}

public class B : A
{
    public override void DoWork()
    {
        // optionally call into base...
        base.DoWork();
    }
}
```

Abstract Classes

- Abstract classes cannot be instantiated
 - Can contain abstract members

```
public abstract class Window
{
    public virtual string Title { get; set; }

    public virtual void Draw()
    {
        // ... drawing code
    }

    public abstract void Open();
}
```

Interfaces

- **Interfaces contain no implementation details**
 - Defines only the signatures of methods, events, and properties
- **A type can implement multiple interfaces**

```
public interface Window
{
    string Title { get; set; }
    void Draw();
    void Open();
}
```

Important Interfaces

Name	Description
IDisposable	Release resources (files, connections)
IEnumerable	Supports iteration (foreach)
INotifyPropertyChanged	Raises events when properties change
IComparable	Compares for sorting

Where To Go From Here

C# Generics



LINQ Fundamentals



C# Programming Paradigms

The screenshot shows the Pluralsight website interface. The browser address bar displays 'www.pluralsight.com/courses/csharp-fundamentals-2'. The page title is 'C# Programming Paradigms' by Scott Allen. Below the title, it mentions 'Formerly titled "C# Fundamentals - Part 2," including dynamic, functional, and language-oriented paradigms.' A 'Table of contents' section lists five topics with their durations: 'C# and LINQ' (37:12), 'C# and the DLR' (35:54), 'Object Oriented Programming with C#' (27:07), 'Functional Programming with C#' (31:53), and 'Crafting C# Code' (32:36). A green button 'Start free trial now' is visible. On the right, a sidebar shows 'Course content' with links to 'Table of contents', 'Description', 'Exercise files', 'Assessment', and 'Discussion'. Below that, 'More info' includes 'Level: Intermediate', 'Rating: 4.5 stars', 'Duration: 2h 44m', and 'Released: 29 Sep 2011'. A vertical 'Feedback & Support' button is on the far right.

C# Programming Paradigms

Formerly titled "C# Fundamentals - Part 2," including dynamic, functional, and language-oriented paradigms.

by Scott Allen

Table of contents [Expand all](#) [Start free trial now](#)

▶ C# and LINQ	37:12
▶ C# and the DLR	35:54
▶ Object Oriented Programming with C#	27:07
▶ Functional Programming with C#	31:53
▶ Crafting C# Code	32:36

The trademarks and trade names of third parties mentioned in this course are the property of their respective owners, and Pluralsight is not affiliated with or endorsed by these parties.

Course content

- Table of contents
- Description
- Exercise files
- Assessment
- Discussion

More info

Level	Intermediate
Rating	★★★★★
Duration	2h 44m
Released	29 Sep 2011

Feedback & Support

Summary

```
internal interface IGradeTracker : IEnumerable
{
    void AddGrade(float grade);
    GradeStatistics ComputeStatistics();
    void WriteGrades(TextWriter destination);
    string Name { get; set; }
}
```