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

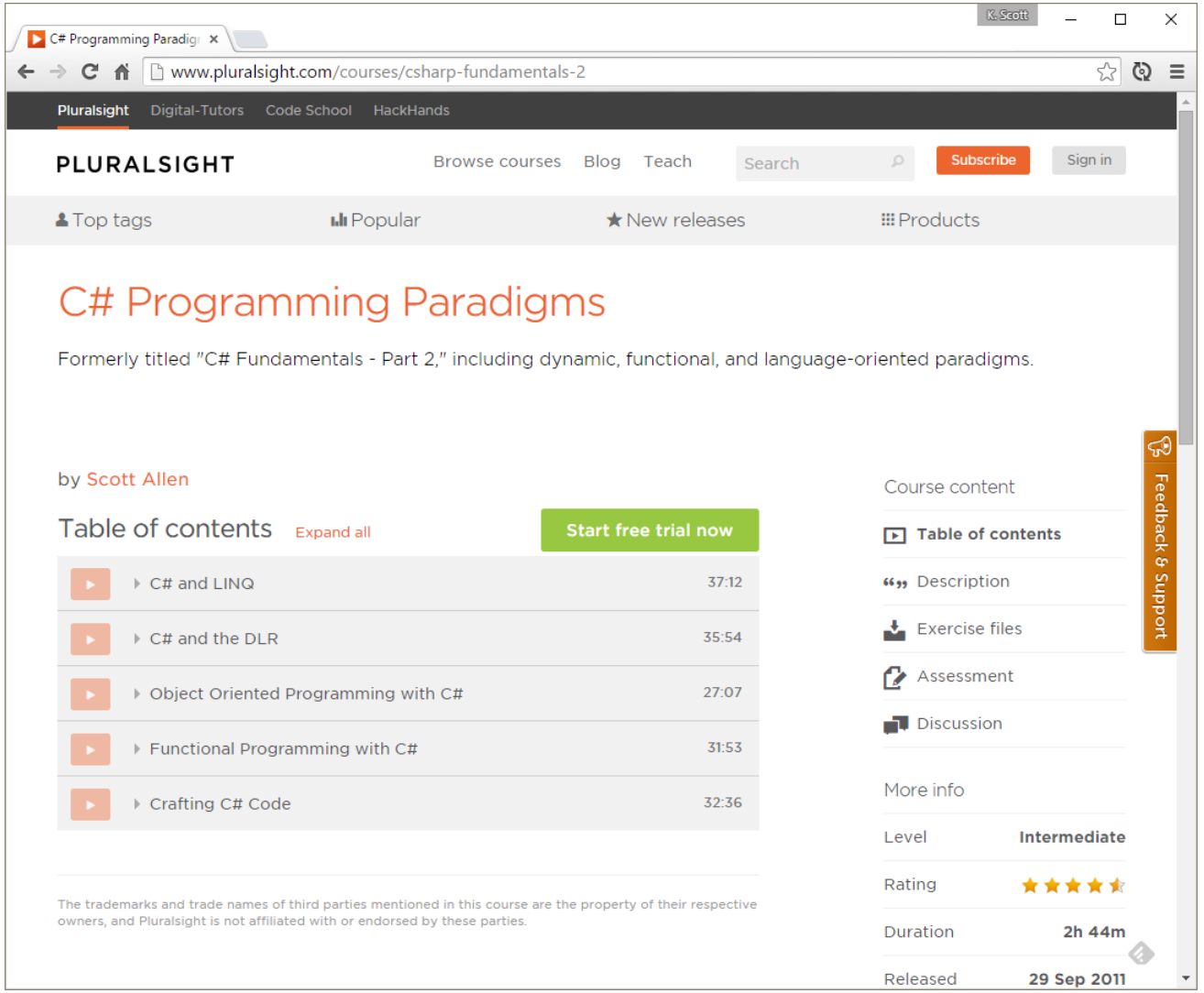
Name	Description
IDisposable	Release resources (files, connections)
IEnumerable	Supports iteration (foreach)
INotifyPropertyChange	Raises events when properties change
IComparable	Compares for sorting

Where To Go From Here

C# Generics

LINQ Fundamentals

C# Programming Paradigms



Summary

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