Object-Oriented Programming

Session: Week 7 Session 1
Instructor: Eric Pogue

LEWIS UNIVERSITY
A Catholic and Lasallian University

Agenda:

1. Review this week's Assignments
2. Introduce the week's Learning Objectives
3. Topics
4. Leaving time for you to start working on "Create a Picture Viewer" tutorial

Review Questions Assignment

Week 7 Questions Assignment [link]

Week 7 Questions Assignment:
http://www.epogue.info/CPSC-24500/Week07/2017SpringW07QuestionsAssignment.docx

Review Programming Assignment

Week 7 Programming Assignment [link]

Week 7 Programming Assignment:
http://www.epogue.info/CPSC-24500/Week07/2017SpringW07ProgrammingAssignment.pdf

Learning Objectives – Week 7

Using Visual Studio 2017, C#, and .NET we will:

1. Review and implement information hiding
2. Download documents from remote Web (HTTP) servers
3. Parse data expressed in XML format
4. Perform basic drawing operations
5. Separate an application's functionality among classes
6. Separate code among files and libraries that you can reuse in other applications
7. Review a Model-View-Controller application

Encapsulation & Information Hiding

Encapsulation is used to hide data from outside classes. C# has three primary (five total) types of access modifiers to encapsulate data. In order to better encapsulate our code and implement data hiding prioritize our access modifiers:

1. Private: only elements of the same class has access
2. Protected: only elements off the same class and descendent classes have access
3. Public: any code has access

Additional C# modifiers include:
Internal: only code in the same assembly has access
Protected Internal: Either code from the derived type or code in the same assembly has access

Review Java Setters & Getters

Setters and Getters are a practice where public Methods are put in place to control how private Attributes are updated.

They can be beneficial in:

- Validation
- Optimization
- Converting types (English to metric)
- Debugging breakpoints
- Some libraries expect setters and getters

Shapes with Setters and Getters:

```
//*******************************************************
// Shapes Step: Setters and Getters
abstract class Shape {
    private int positionX;
    private int positionY;

    public int getPositionX() {
        return positionX;
    }

    public void setPositionX(int positionXIn) {
        positionX = positionXIn;
    }

    public int getPositionY() {
        return positionY;
    }

    public void setPositionY(int positionYIn) {
        positionY = positionYIn;
    }

    public Shape() {
        positionX = 0;
        positionY = 0;
    }
}
```

Why use Setters & Getters? Because 2 weeks (months, years) from now when you realize that your setter needs to do more than just set the value, you'll also realize that the property has been used directly in 238 other classes. (Internet quote)

C# Setters & Getters

Setters and Getters are a practice where public Methods are put in place to control how private Attributes are updated.

They can be beneficial in:

- Validation
- Optimization
- Converting types (English to metric)
- Debugging breakpoints
- Some libraries expect setters and getters

C# Auto-Implement Properties

A property is a member that provides a flexible mechanism to read, write, or compute the value of a private field. Auto-Implement Properties provide a very concise syntax for implanting setters and getters.

```
// Partial class from RandomNumberThreadedCS project
class GetRandomNumbers {
    private long timesToLookFor1024;
    public long getTimesToLookFor1024() { return timesToLookFor1024; }
    public void setTimesToLookFor1024(long timesToLookFor1024In) {
        timesToLookFor1024 = timesToLookFor1024In;
    }

    ...
}

// Same partial call using C# Auto-Implment properties
// Auto-Implemented Properties:
//     https://msdn.microsoft.com/en-us/library/bb384054.aspx
class GetRandomNumbers {
    public long timesToLookFor1024 { get; set; }

    ...
}
```

Very concise, but really doesn't protect us from the most challenging parts of hiding data.

```
public string FirstName { get; set; } = "Jane";
```

Why use Setters & Getters? Because 2 weeks (months, years) from now when you realize that your setter needs to do more than just set the value, you'll also realize that the property has been used directly in 238 other classes. (Internet quote)

Encapsulation & Information Hiding Recommedation

1. Make everything local to a Method
2. Make everything a Method Parameter
3. Make everything Private…
4. If you must make it Protected or Public, provide “real” setters and getters

Encapsulation & Information Hiding Suggestions

1. Make everything local to a Method
2. Make everything a Method Parameter
3. Make everything Private…
4. If you must make it Protected or Public, provide “real” setters and getters

Download documents from remote Web (HTTP) servers

Multiple .NET (C#) classes and methods are provided that wrap various network protocols. For Web (HTTP) the .NET environment provides the WebClient class which:

- Is most often used to retrieve files
- Can access multiple Internet file types including HTML, XML, JSON, etc.
- Utilized HTTP or HTTPs for communication

There are a multitude of network and Internet protocols. It is beyond the scope of this class to cover them in detail.

XML

In computing, XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is an open standard that:

- Supports nearly all development languages and platforms
- Allows us to cross between many applications
- Can result in large files
- Supports schema to validate data

```xml
<?xml version="1.0" encoding="UTF-8"?>

<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is an open standard that:

- Supports nearly all development languages and platforms
- Allows us to cross between many applications
- Can result in large files

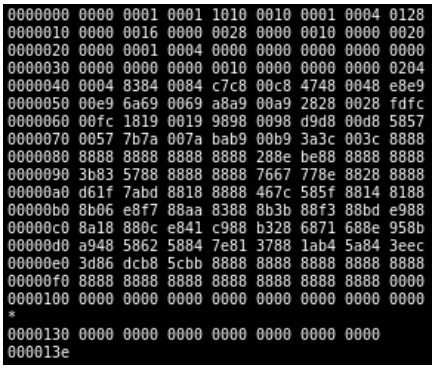
```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

JSON is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is a very common data format used for asynchronous browser/server communication, including as a replacement for XML in some web service style systems.

Binary Files

A binary file is a computer file that is not a text file. The term "binary file" is often used as a term meaning "non-text file". The can be open or closed formats that are generally:

- Fast, small, and efficient*
- Often not very portable across applications and platforms
- Difficult to maintain backward compatibility



Some people would say that binary files include all files, and that text files are just binary files that are being interpreted in a specific way.

Parse data expressed in XML format

Simple XML files can be parsed “by hand” without much difficulty. In addition, the .NET (C#) environment offers multiple classes that can assist in parsing XML including:

- XmlReader
- LINQ to XML [link]

Preview HideDataDownloadXML Example

Features:

1. Develop application entirely in Visual Studio 2017 and C#
2. Take in one command line argument that is the URL to download
3. Implement multiple C# classes that appropriately hide data
4. Download HTML and XML files from various URLs
5. Get ready for parsing XML

Object-Oriented Programming

Session: Week 7 Session 1
Instructor: Eric Pogue

Agenda:

1. Review this week's Assignments
2. Introduce the week's Learning Objectives
3. Topics
4. Leaving time for you to start working on "Create a Picture Viewer" tutorial

End of Session

Course Number: CPSC-24500

Week: 7

Session: 1

Instructor: Eric Pogue

End of Session

Course Number: CPSC-24500

Week: 6

Session: 2

Instructor: Eric Pogue

End of Session

Course Number: CPSC-24500

Week: 5

Session: 3

Instructor: Eric Pogue

End of Session

Course Number: CPSC-24500

Week: 5

Session: 4

Instructor: Eric Pogue