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

Review Questions Assignment

Week 7 Questions Assignment [link]

Review Programming Assignment

Week 7 Programming Assignment [link]

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

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:

```java
//*******************************************************
// 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;
    }
}
```

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.

```csharp
// 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; }

    ...
}
```

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

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
}
```

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

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

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

Object-Oriented Programming

Session: Week 7 Session 2
Instructor: Eric Pogue

HideDataDownloadXML Example:

1. Develop application entirely in Visual Studio 2017 and C#

2. Take in one command line argument that is the URL to download

3. Utilize Web (HTTP) protocols to download HTML and XML files

4. Get ready for parsing XML

5. Implement multiple C# classes that appropriately hide data

Web (HTTP) Protocol

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web [link]:

- Network protocols like HTTP and HTTPs <u>ARE</u> used to protect data!

- HTTP defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource including:
  - GET: requests a resource
  - POST: requests that the server accept the entity enclosed in the request
  - Many, many more

- Most often uses a Web browser as a client

- A variety of Web servers are available

- TCP, IP, HTTP, HTTPs, HTML, XML, JSON

- Web Server: A server that utilizes TCP/IP and responds on Port 80 from a given IP address using HTTP or HTTPs and generally returns HTML (or XML or JSON)

End of Session

Course Number: CPSC-24500

Week: 7

Session: 2

Instructor: Eric Pogue

Object-Oriented Programming

Session: Week 7 Session 3
Instructor: Eric Pogue

HideDataDownloadXML Example:

1. Develop application entirely in Visual Studio 2017 and C#

2. Take in one command line argument that is the URL to download

3. Utilize Web (HTTP) protocols to download HTML and XML files

4. Parsing XML

5. Implement multiple C# classes that appropriately hide data

End of Session

Course Number: CPSC-24500

Week: 7

Session: 3

Instructor: Eric Pogue

Object-Oriented Programming

Session: Week 7 Session 1
Instructor: Eric Pogue

Agenda:

1. Comment on Requesting Graded Homework Assignments

2. Discuss this week's Assignments
   - Week 7 Questions Assignment [link]
   - Week 7 Programming Assignment [link]

3. Review the week's Learning Objectives

4. More Learning Objective Topics

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. Separate an application's functionality among classes

5. Separate code among files and libraries so that you can reuse in other applications

6. Review a Model-View-Controller application

7. Perform basic drawing operations… as time allows

Separate C# Files

C# code can be easily separated into files and be shared between application as source code. Pros and cons include:

- Utilizing source code management (GIT) to manage it within or between applications

- Sharing actual C# source code required (pro or con)

- Compiling required in order to utilize (con)

- Utilizing C# required (con)

- Very similar to how we did it with Java and other languages

Libraries and Components

The terms Libraries, <u>Components</u>, and Frameworks are often used interchangeably. For our purposes we will utilize the term Component. C# code can be compiled into Components that can then be utilized in other applications. Pros and cons include:

- Distributing source code is optional and can be done independently from the main application (pro)
- Hiding of information and implementation enforced (pro)
- Multiple (mostly incompatible) methods including DLLS, COM, .NET, etc. on Windows (con)
- Language agnostic (pro)

Service Oriented Architecture

Service Oriented Architectures (SOA) utilize standard network protocols to implement Encapsulation, Interface Inheritance (vs Implementation Inheritance), "limited" polymorphic abilities, operating system independence, and language independence. Pros and cons include:

- Security can be enforced at the network level (pro)

- "Components" do not have to run on the same operating system (pro)

- "Components" can be run remotely at different companies (pro)

- Cloud centric (pro)

- Performance can be an issue (con)

- Control and security is distributed (con… or pro)

Separate ShapeModel in HideDataDownloadXML

<u>Features:</u>

1. Develop application entirely in Visual Studio 2017 and C#

2. Take in one command line argument that is the URL to download

3. Utilize Web (HTTP) protocols to download HTML and XML files

4. Parsing XML

5. Implement multiple C# classes that appropriately hide data

6. Clone class source code

7. Separate ShapeModel into separate file called ShapeModel.cs

8. Update source code in Git repository

9. Implement, demonstrate and test ShapeModel new application

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. Separate an application's functionality among classes

5. Separate code among files and libraries so that you can reuse in other applications

6. Review a Model-View-Controller application

7. Perform basic drawing operations

Separate DownloadXML_MVC

Features:

1. Develop new application entirely in Visual Studio 2017 and C#

2. Implement ShapeModelViewController

3. Import ShapeModel into ShapeModelViewController

4. Implement ShapeConsoleView

End of Session

Course Number: CPSC-24500

Week: 7

Session: 4

Instructor: Eric Pogue