

Object-Oriented Programming

Session: Week 8 Session 1

Instructor: Eric Pogue



Agenda:

1. Review this week's Assignments
2. Introduce the week's Learning Objectives
3. Review Learning Objectives
 - Threading
 - Database
 - Network Programming
4. Review Session 1 Topics

Our learning objectives from the syllabus assumed that we would not have focused quite so much on threading earlier in the session, so our threading discussion will be brief this week. Previous session of the course also focused somewhat more on database and SQL programming and a little less on network programming. With the continued expansion of Web Service and Cloud Computing in the industry, I am going to reverse that priority. We will still cover both topics; however, are focus will be Web Services and network programming while still recognizing the importance of databases.

Review Questions Assignment

Week 8 Questions Assignment [\[link\]](#)

Week 7 Questions Assignment:

<http://www.epogue.info/CPSC-24500/Week07/2017SpringW07QuestionsAssignment.docx>

Review Programming Assignment


Week 8 Programming Assignment [\[link\]](#)

Week 7 Programming Assignment:

<http://www.epogue.info/CPSC-24500/Week07/2017SpringW07ProgrammingAssignment.pdf>

Learning Objectives – Week 8

1. Describe what a thread is and why it can be useful to distribute tasks among multiple threads
2. Review our multi-threaded application development activities
3. Explain why it is important to synchronize threads that need to share data source access
4. Review Object Oriented Programming benefits including the associating Data & Functionality, Encapsulation & Information Hiding, Inheritance, and Polymorphism
5. Review databases, database servers, and the SQL language
6. Understand how databases support (or don't support) work within a Object Oriented Programming environment
7. Understand client-server (two-tier), three-tier, and n-tier architectures
8. Introduce network programming concepts
9. Understand Web Services network programming
10. Develop a middle-tier data server using network programming

 - We covered these topics in week 5/6 and week 1 respectively.

Performance Optimization and Threading

Performance is critical in application development... the focus of performance optimization continues to evolve, but the criticality remains very high! Multithreading is one very important way that we can optimize CPU performance; however, there are many other performance bottlenecks and optimization techniques:

- CPU... threading
- Memory... optimize disk usage, buy more memory
- Disk IO... buffering, file size, or faster (more expensive) disks
- Network bandwidth... "file" or package size
- Network latency... pray for a miracle!
- User Interaction and Capabilities

I believe that performance optimization is one of the most important and challenging aspects of developing high quality software.

Consider mobile phone networks and satellite networks... and latency.

User experience.

Know the difference between latency and bandwidth and how it impacts network and application performance:

https://en.wikipedia.org/wiki/Network_performance

The following measures are often considered important:

Bandwidth commonly measured in bits/second is the maximum rate that information can be transferred

Latency the delay between the sender and the receiver decoding it, this is mainly a function of the signals travel time, and processing time at any nodes the information traverses

Throughput is the actual rate that information is transferred

Jitter variation in packet delay at the receiver of the information

Error rate the number of corrupted bits expressed as a percentage or fraction of the total sent

You can usually buy more bandwidth. Fixing a latency issue might require you to change the speed of light.

Threads & Multithreaded Applications

Multithreading: A technique by which a single set of code can be used by several processors or cores at different stages of execution.

- Threads and application performance are becoming nearly synonymous
- Moore's law only remains achievable if we can effectively utilize multi-processor, multi-core, and multi-threaded applications
- Our performance principles that we discuss will be applicable across platforms and environments
- Our practical focus will be on Java multi-threading
- Parallel processing has become the focus of the computing and software development industry
- The rise of big data, artificial intelligence, virtual/augmented reality, and dedicated graphical processing units (GPUs) have made that a nearly guaranteed trend for years to come

The difference between user perceived (real or perceived) is as important as actual performance. Optimizing application performance to reflect user capabilities is challenging and necessary. It doesn't matter if you optimize an application to be 10 times faster if the bottleneck is how fast the user.

Multithreading is the primary method of optimizing CPU performance. It is unlikely to be of benefit if the bottleneck is elsewhere. For example, if you are disk bound, splitting your process into multiple threads is unlikely to be of benefit.

Processors, Cores, and Threads

- Computers have one or more Processors (CPUs)
- Processors each have one or more Cores
- Cores can create one or more Threads
- An application running only on one thread of a dual cpu, quad-core, single thread can utilize only a portion of 1/8th of the processing power of that machine

For this discussion we will use multithreading and multiprocessing terms synonymously for our purposes.

Modern central processing units (CPUs) are made up of cores. A core is like a mini-processor that works with its fellow cores to perform the work that applications request of the CPU. In the old days, a CPU had just one core, a single channel through which all requests would pass. This was how we optimized applications... CPU, memory, fast disk, slow disk. Today, though, with multiple CPUs and multiple cores, a CPU can pay attention to and do many things at once.

This architecture, in turn, allows today's applications to perform multiple tasks at once. This ability is called multitasking. Multitasking enables an

Multi-Threaded Development

Now for the bad news. Multi-Threaded Development is really hard!



Image sources: Google

- Rigidity - It is hard to change because every change affects too many other parts of the system
- Fragility - When you make a change, unexpected parts of the system break
- Immobility - It is hard to reuse in another application because it cannot be disentangled from the current application

Stadia add-in net change example.

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other. Deadlock occurs when multiple threads need the same locks but obtain them in different order.

Multi-Threaded Development

Now for the bad news. Multi-Threaded Development is really hard!

- Developing commercial quality multi-threaded applications makes Rigidity, Fragility, and Immobility much harder to avoid
- Many of the 3rd party professional libraries that the industry had come to rely on came into question as multi-threading application became required
- Testing becomes harder when a sequence of events becomes variable
- What if your automated unit test results might be different depending on which thread finishes first?
- What about deadlock?
- **Performance** is so important that we will need to understand be able to effectively utilize, test, and deploy effective multi-threaded applications

- Rigidity - It is hard to change because every change affects too many other parts of the system
- Fragility - When you make a change, unexpected parts of the system break
- Immobility - It is hard to reuse in another application because it cannot be disentangled from the current application

Increased complexity is the primary disadvantage for developing multithreaded applications.

Some languages have come into existence in order to try to reduce the complexity of writing, enhancing, and supporting multithreaded applications. For example, Scala has implemented specific parallelization features in the core language that make it a first class threading language. Note that Scala also targets the Java runtime environment.

C++ would be an example of a language that has implemented a plethora of threading mechanisms for various platforms and implementations. Recent versions have introduced more common approaches.

Stadia add-in example.

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other. Deadlock occurs when multiple threads need the same locks but obtain them in different order.

Interesting threading article:

<http://blog.smartbear.com/programming/why-johnny-cant-write-multithreaded-programs/>

When reviewing libraries look for something like “This class is immutable and thread-safe.” before you use it in multithreaded development.

FastPrime in C#

Write a performance optimized command line C# application that will programmatically find prime numbers and store those numbers sorted in an output file.

In FastPrime we will create a command line Java application that will:

1. Use multiple threads to find the prime numbers between two numbers
2. Sort those results and store them to a file
3. Perform some timings
4. ... And do this all very fast

See the details in this week's assignment

Learning Objectives – Week 8

1. Describe what a thread is and why it can be useful to distribute tasks among multiple threads
2. Review our multi-threaded application development activities
3. Explain why it is important to synchronize threads that need to share data source access
4. Review Object Oriented Programming benefits including the associating Data & Functionality, Encapsulation & Information Hiding, Inheritance, and Polymorphism
5. Review databases, database servers, and the SQL language
6. Understand how databases support (or don't support) work within a Object Oriented Programming environment
7. Understand client-server (two-tier), three-tier, and n-tier architectures
8. Introduce network programming concepts
9. Understand Web Services network programming
10. Develop a middle-tier data server using network programming

We are reviewing this so that we can relate it back to databases and client-server development.

Object-Oriented Programming [\[link\]](#)

Object-oriented programming (OOP) is a programming model based on the concept of "objects", which contain both Attributes (data) and Methods (procedures) that operate on those attributes.

Most popular OOP languages are class-based, meaning that objects are instances of classes.

It includes concepts, patterns, and principles for designing and implementing modern software products.

- Concepts – powerful features that prove indispensable to modern software development, brought to us automatically by object-oriented programming.
- Patterns – tried-and-true templates for forging relationships between classes
- Principles – guidelines that help you determine what classes are needed and how they should divide up the work

I will often start with definitions from Wikipedia and other sources. If you are struggling with a topic and/or would like more information, it can often be valuable to review the references. You should be able to click on the [link] tag (possibly while holding the shift key down) in order to open the reference in a browser. Please let me know if this is not working for you.

Walking through the slide, you will see that words and terminology will be important as we discuss and learn new concepts. During the course I am sure you will notice that at times I will struggle with the attribute/property vs. data and method vs. procedure distinction when I am talking. I would like for us to try to make that distinction in our work as we go through the term.

For the purposes of this class, “attributes” and “properties” will be used interchangeably to describe variables belonging to a class or object. Also “procedures” and “functions” will be used interchangeably.

The “six” (Three plus) Object-Oriented Concepts

Object-oriented concepts:

1. **Encapsulation...** and Information Hiding
2. **Inheritance...** and Abstraction
3. **Polymorphism**

Plus... Composition & Aggregation

Helpful Interview Hint: Whenever you are asked a conceptual question about object-programming in an software development interview (and you will be), answer confidently “Encapsulation”, “Inheritance”, and “Polymorphism”.

When asked what is Encapsulation (or how would you implement it), say, “I would limit or minimize variable scope and keep data attributes private as often as possible.”

Now as we are going through our object-oriented examples, be thinking about how you would answer the “What is Inheritance?” and “What is Polymorphism?” interview questions. Note that answering them both with very brief examples can be very effective... and it is always best to use animals in you OOP interview examples.

Now we just need to make sure that we are able to effectively utilize these concepts after we get the job. Let’s start by walking through an example.

Encapsulation & Information Hiding

Encapsulation: Wrapping properties and methods into a class and minimizing the scope of those properties and methods.

Information Hiding: Minimize visibility/scope of data, attributes, functions, and methods.

Inheritance & Abstraction

Inheritance: When one class acquires the properties and methods of another class it is called Inheritance.

Abstraction: Something is abstract when it is a concept but is not concrete or defined enough to actually be built. Generally, in OO design, we start with abstract things, and then we build on them through Inheritance.

```
// Inheritance
class Shape {
}

class Circle extends Shape {
}

class Rectangle extends Shape {
}
```

The first of many “Shape” examples. We will get to a Abstraction example in a few minutes.

Polymorphism

Polymorphism: Polymorphism enables you to process collections of related things generically. This is particularly useful when you want to use a loop to march through a collection of items.

```
// Polymorphism
Shape[] shapes = new Shape[3];
shapes[0] = new Circle();
shapes[1] = new Rectangle();
shapes[2] = new Triangle();
for (Shape s : shapes) {
    System.out.println(s.area());
}
```

Example of polymorphism: a for loop that moves through the entries of a list. The list might be of a collection of related kinds of object. We can refer to each of the objects in the list through a generic variable (whose data type matches the one that all are ultimately related to). But, when we invoke a particular function that all members of the family share, each will respond by performing that function in their own specific way. For example, we could have a collection of Shape objects. We could refer to each entry in the Shape list through a generic Shape variable, even though the actual entries in the list are specific kinds of shapes – Circle, Rectangle, etc. All Shape objects might have the ability to calculate their own area. When we refer to an object in the list through a generic Shape variable and tell it to calculate its area, thanks to polymorphism, the circle version of the area() function will be called when we're dealing with a circle, and the Rectangle version of area() will be called when we're dealing with a rectangle, etc.

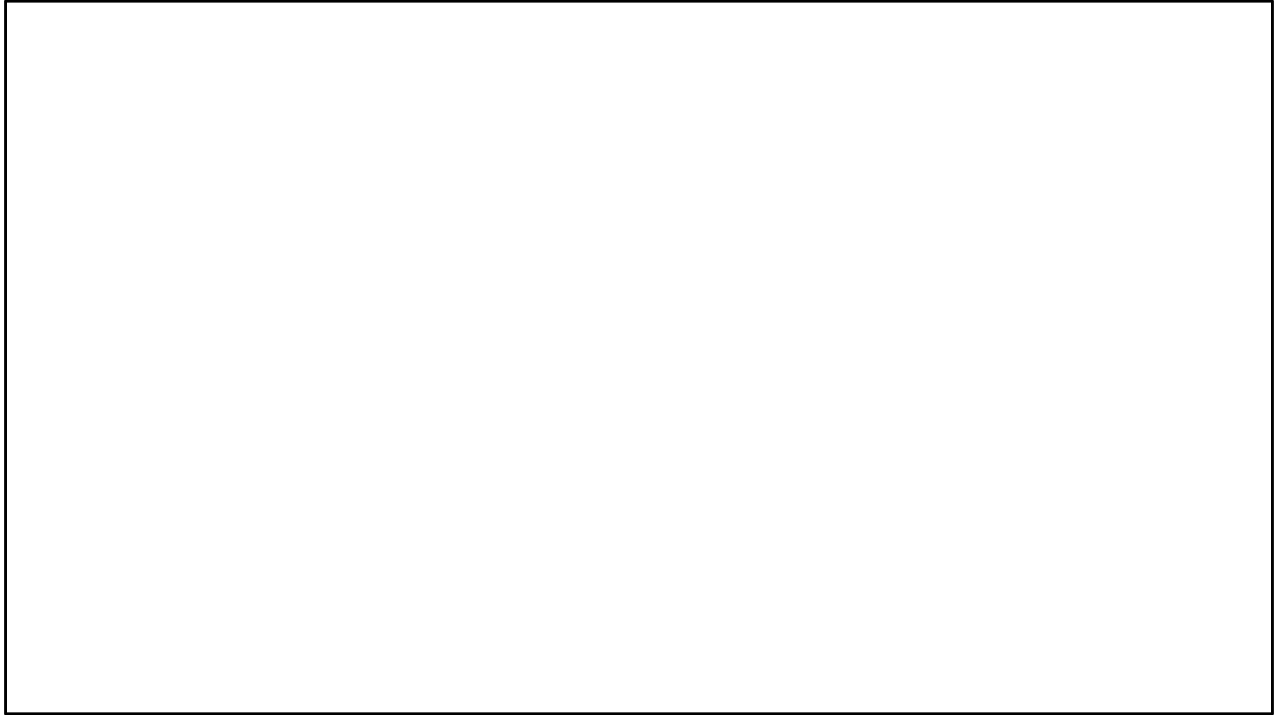
The first time through the for loop, we'll call the Circle.area() function – actually, we won't; it will happen automatically. The next time through, we'll call the Rectangle version, and then we'll call the Triangle version.

Polymorphism is implemented behind the scenes using a Virtual Method Table (VMT). The VMT keeps track of where various related classes' same-named functions are located in memory. Using the VMT, the operating system is able to figure out which code to implement when we tell each shape to fire its area() function.

Learning Objectives – Week 8

1. Describe what a thread is and why it can be useful to distribute tasks among multiple threads
2. Review our multi-threaded application development activities
3. Explain why it is important to synchronize threads that need to share data source access
4. Review Object Oriented Programming benefits including the associating Data & Functionality, Encapsulation & Information Hiding, Inheritance, and Polymorphism
5. Review databases, database servers, and the SQL language
6. Understand how databases support (or don't support) work within a Object Oriented Programming environment
7. Understand client-server (two-tier), three-tier, and n-tier architectures
8. Introduce network programming concepts
9. Understand Web Services network programming
10. Develop a middle-tier data server using network programming

We are reviewing this so that we can relate it back to databases and client-server development.



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 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.

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 [\[link\]](#)

End of Session

Course Number: CPSC-24500

Week: 8

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 ARE 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)

TCP/IP: Transmission Control Protocol / Internet Protocol

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

SSL: Secure Sockets Layer

HTML: Hypertext Markup Language

XML: Extensible Markup Language

JSON:

Web Server: A server that utilizes TCP/IP and responds on Port 80 from a given IP address using HTTP and generally returns HTML.

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 Discussion & Lecture

Instructor: Eric Pogue



Agenda:

1. Reminder on Requesting Graded Homework Assignments
2. Review Week 7 To-do List
 - Recognize that it may be valuable to review items “8a” and “8b” before this “Week 7 session 1” video
 - Don’t forget your Bb postings
3. Discuss this week’s Assignments
 - Week 7 Questions Assignment
 - Week 7 Programming Assignment
4. Review the week’s Learning Objectives
5. Continue with 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

"git clone <https://github.com/EricJPogue/CPSC-24500.git>

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 shared code (con)
- Utilizing C# required (con)
- Very similar to how we did it with Java

Libraries and Components

The terms Libraries, Components, 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 (pro)
- Hiding of information and implementation enforced (pro)
- Multiple (often incompatible) methods including DLLS, COM, .NET, etc. on Windows (con)
- Language agnostic (pro)

Microsoft has a long history of providing (mostly incompatible) mechanisms to develop, deploy, and utilize component architectures. Over time component architectures have evolved into Service Oriented Architectures (SOA).

Components: A binary (compiled) package that contains local application functions/methods or APIs (Application Programming Interfaces) that can be utilized during development or run-time. JAR files are an example of a component. The functions/methods run locally on the same computer as the application. Note: the fact that they run on the same computer is why information hiding and components do not provide data security.

Service: A remote API that (generally) runs on a separate machine accessed by a network protocol (HTTP, REST, SOAP).

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)
- Legal & Privacy (pro... or con)

Components: A binary (compiled) package that contains local application functions/methods or APIs (Application Programming Interfaces) that can be utilized during development or run-time. JAR files are an example of a component. The functions/methods run locally on the same computer as the application. Note: the fact that they run on the same computer is why information hiding and components do not provide data security.

Service: A remote API that (generally) runs on a separate machine accessed by a network protocol (HTTP, REST, SOAP). XML or JSON are generally used within SOAP and REST.

Note: The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) designed to facilitate the communication of systems that are deployed on diverse platforms. It was a mostly failed attempt to implement “full” OOP across the network in a SOA implementation. Over time this gave way to simpler implementation (with less OOP functionality) like SOAP and REST.

Implement ShapeModel in DownloadAndParseXML

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. 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. Review ToString method that was added to Shape class
8. Separate ShapeModel into separate file called ShapeModel.cs
9. Update source code in Git repository

Steps:

1. git clone <https://github.com/EricJPogue/CPSC-24500.git>
2. Open DownloadAndParseXML solution
3. Add new ShapeModel class .cs file
4. Cut and past ShapeModel source into new file
5. Change NameSpace to ShapeModelXML
6. Update Program.cs to with "Using ShapeModelXML;"
7. Compile & debug
8. git add ShapeModel.cs
9. git commit
10. git push

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

`"git clone https://github.com/EricJPogue/CPSC-24500.git`

Implement DownloadAndParseXML_MVC

Features:

1. Develop new application entirely in Visual Studio 2017 and C#
2. Implement ShapeController using "Project|Add New Item"
3. "Import" ShapeModel into ShapeController using "Project|Add Existing Item"
4. Implement ShapeConsoleView

Recognize that we may have multiple Models, Views, and Controllers in a complex application. We could end up with names like "SimpleShapeModel_ShapeConsoleView_Controller". We will keep it very simple for our example.

Steps:

1. Create a new Visual Studio 2017 project named "DownloadAndParseXML_MVC"
2. Add a new Class and .cs file called ShapeController
3. Create a new "ShapeController" in Main
4. "Import" ShapeModel from DownloadAndParseXML... copy ShapeModel.cs file
5. Add "Using ShapeModelXML;" to ShapeController
6. Enhance ShapeController with ShapeModel
7. Implement ShapeConsoleView

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

Implement DrawShapes

Features:

1. Develop new application entirely in Visual Studio 2017 and C#
2. Create a new Windows Forms (.NET Framework) application called DrawShapes
3. Add a button called DrawNow with button text of "Draw"
4. Edit the button pressed code to draw Ovals and Rectangles
5. Create separate methods to draw and an Oval and a Rectangle
6. Draw a few Ovals and Rectangles

End of Session

Course Number: CPSC-24500

Week: 7

Session: 4

Instructor: Eric Pogue

Object-Oriented Programming

Session: Week 7 Session 5

Instructor: Eric Pogue



InternetShapeDrawLite:

1. Develop new application entirely in Visual Studio 2017 and C#
2. Create a new Windows Forms (.NET Framework) application called EJPIInternetShapeDrawLite
3. Override OnPaint()
4. Implement graphical "Hello World!!!"
5. Draw Rectangles
6. Draw Ovals
7. Implement Loading and Parsing of Shapes... by copy/past importing from previous example
8. Draw Shape in ShapeModel
9. Review application requirements... add comments
10. Compile & Test release build

End of Session

Course Number: CPSC-24500

Week: 7

Session: 5

Instructor: Eric Pogue