Object Oriented Programming

Meet the expert: Dr. Jack Purdum recently retired from Purdue University's College of Technology. Dr. Purdum has won several teaching awards and has over 25 years of university teaching experience. He has published numerous journal and magazine articles, was a frequent contributor to various software development conferences, owned and managed his own software house specializing in language compilers and programming tools for 17 years, and holds a U.S. patent for imaging software he developed. Dr. Purdum is also the author of 16 programming texts including the New York Times Best Seller "The C Programming Guide."

Prerequisites: This is course is intended for programmers who are interested in understanding object-oriented principles. The student should be familiar with using an Integrated Development Environment such as Visual Studio or Eclipse.

Runtime: 06:18:22

Course description: Object Oriented Programming will help you improve your programming skill set by giving you a thorough understanding of OOP regardless of your programming background. The training uses a non-jargon approach to teaching the backbone concepts of OOP which have been field tested in years of university-level teaching. While Visual Studio and C# form the instruction environment, the course is language agnostic as possible so the concepts are easily applied to other popular OOP languages and platforms.

Course outline:

History of Programming
- Introduction
- Binary Languages
- Assembly Language
- High Level Languages
- Structured Programming
- Object Oriented Programming
- Why Use OOP
- Summary

More History of Programming
- Introduction
- Objects Overview
- Example of an Object
- Object Methods
- Object Properties
- Class
- Class Versus Object
- Summary

Define Versus Declare
- Introduction
- Defining a Variable
- Bucket Analogy
- Summary

Designing Your Own Classes
- Introduction
- Overview
- Class Design Goals
- Designing Your Program
- Summary

UML Light
- Introduction
- UML Light Overview
- Access Specifiers
- Defining a Class Object
- Demo: UML Light
- Summary

More UML Light
- Introduction
- Demo: UML Light Continued
- Demo: Visual Interface
- Demo: Add Another Component
- Demo: Class Code
- Demo: Add a Property List
- Demo: Add a Constructor
- Demo: Multiple Constructors
- Summary

UML Light Finished
- Introduction
- Demo: Constructors Continued
- Demo: Overridden Constructor
- Demo: Write a Property Method
- Summary

Adding A General Method
- Introduction
- Demo: Write a General Method
- Demo: Change Properties
- Demo: Adding Another Method
- Summary

Dot Operator
- Introduction
- Dot Operator Overview
- Demo: Dot Operator
- Dot Operators Position
- Things to Keep in Mind
- Multiple Dot Operators
- Summary

Inheritance Fundamentals
- Introduction
- Inheritance Overview
- Data for Each Type of Property
- Inheritance Classes
- Summary

Base and Subclass Discussion
- Introduction
- Demo: The Base Class
- Demo: Run the Program
- Summary

Inheritance & 5 Program Steps
- Introduction
- Demo: Windows Form
- Demo: Button Click Event
- Summary

Delegate
- Introduction
- Delegate
- Delgate Syntax
- Delegate
- Demo: Delegate
- Demo: Derived Classes
- Summary

Interfaces
- Introduction
- Interfaces
- Demo: Interfaces
- Summary

Grand Project
- Introduction
- Grand Project
- Game Interface
- Initialization Step
- Input Step
- Process Step
- Display Step
- Termination Step
- Demo: Grand Project
- Summary

UML Game Design
- Introduction
- The Game Classes
- Grand Project UML
- Grand Project UML 2
- Summary

Code Walk Through UI
- Introduction
- Demo: Game Code
- Demo: Deal Button
- Summary

In Between Rules Class
- Introduction
- Demo: The Rules Class
- Demo: Helper Method
- Summary

In Between Deck Class
- Introduction
- Demo: The Deck Class
- Demo: Helper Methods

(Continued on page 2)

• Summary

Game Single Step
• Introduction
• Demo: Single Game Step
• Demo: Get Cards Left
• Summary