# Table of Contents

## Software Development for Non-Programmers
- Software Development for Non-Programmers ................................................................. 1
- Software Development for Non-Programmers Using Python ............................................. 2
- Software Development for Non-Programmers Using Java ................................................ 3

## C
- C Programming .................................................................................................................. 4
- Advanced C Programming .................................................................................................. 5

## C++
- C++ Programming .............................................................................................................. 6
- Advanced C++ Programming .............................................................................................. 7

## Java Technologies
- Java Programming .............................................................................................................. 8
- Advanced Java Programming ............................................................................................. 9
- SWING .................................................................................................................................. 10
- Introduction to Hibernate .................................................................................................. 11
- Introduction to Spring Framework ..................................................................................... 12
- Hibernate/Spring Combo ................................................................................................. 13
- JavaServer Faces (JSF) ....................................................................................................... 14
- EJB 3.x Update for EJB 2.x Developers ............................................................................. 15
- Developing Java Enterprise Applications Using EJB3 ..................................................... 16

## Python
- Introduction to Python 3 .................................................................................................... 17
- Advanced Python 3 ............................................................................................................ 18
- Python 3 and Advanced Python 3 ..................................................................................... 19

## Perl
- Perl Programming ............................................................................................................. 20
- Advanced Perl Programming ............................................................................................ 21

## Ruby
- Ruby Programming ............................................................................................................ 22

## Groovy
- Groovy Programming ....................................................................................................... 23
- Groovy and Grails .............................................................................................................. 24
Software Development for Non-Programmers

Course Description: This class has a wealth of knowledge that aspiring programmers need to master before tackling their first programming language. This course consists of topics such as: hardware and software components of a computer system, the CPU, registers, operating systems, processes, number systems, data type formats, conversions among number bases, logic and reasoning, flow charting, pseudo code, control structures, differentiation among programming languages, from machine language to assemblers and compilers, data structures, libraries, networking, web programming, and a host of other topics. After learning the fundamentals, students will be given a chance to show their knowledge by writing some simple to intermediate programs in the C Programming Language.

Who Should Attend: This class has a wealth of knowledge that aspiring programmers need to master before tackling their first programming language. This course consists of topics such as: hardware and software components of a computer system, the CPU, registers, operating systems, processes, number systems, data type formats, conversions among number bases, logic and reasoning, flow charting, pseudo code, control structures, differentiation among programming languages, from machine language to assemblers and compilers, data structures, libraries, networking, web programming, and a host of other topics. After learning the fundamentals, students will be given a chance to show their knowledge by writing some simple to intermediate programs in the C Programming Language.

Prerequisites: There are no prerequisites for this course.

Benefits of Attendance: Upon completion of this course, students will be able to:

- List the major components of a computer system.
- Differentiate between various kinds of computer storage.
- Distinguish between system software and application software.
- List the steps in the instruction/execution cycle.
- Differentiate between various kinds of programming language translators and programming paradigms.
- Convert from and to the following bases: binary, octal, decimal, and hexadecimal.
- Demonstrate how negative numbers are stored in memory.
- Demonstrate how various data types are stored in memory.
- State the difference between a program and a process.
- Use various graphical techniques, such as flow charting and pseudo code, to specify the logic of a program.
- Read BNF notation that describes correct grammar for a programming language.
- Write C programs that use conditionals and loops.
- Demonstrate the benefits of using functions.
- Understand the relationship between arguments and parameters.
- Effectively use arrays and strings in C.
- Build and use static libraries in C.
- Discern the various interfaces in launching programs.

Course Outline:

**Introduction**
- Introduction
- What is a Computer System?
- Input Units
- Output Units
- Memory
- Central Processing Unit
- Components of the CPU
- Instruction Execution Cycle
- Cache
- Software
- Functions of an Operating System
- Loading the Operating System
- Loading an Application Program
- Timesharing

**Programming Languages**
- Introduction
- Machine Language
- Assembly Language
- Compilers
- BNF
- Language Classification
- Procedural vs. Object Oriented
- Static vs. Dynamic Typing
- Scripting vs. Non-Scripting
- Choosing a Programming Language

**The Programming Cycle**
- The Software Development Cycle
- The Programming Cycle - Overview
- Edit
- Compile
- Execute

**Number Systems and Data Types**
- Number Bases
- Base 10
- Base 2
- Base 8 and Base 16
- Conversion Between Number Bases
- Relationships Between Number Bases
- Data Types
- Negative Numbers
- Sign Magnitude
- Complement Arithmetic
- Floating-Point Values
- String Data
- Data Types in C

**Fundamentals of C - Part 1**
- C Data Types
- Variables
- printf
- Arithmetic Operators
- Control Flow - Decisions
- Relational Operators
- Control Flow - Loops
- The while Loop
- The for Loop
- for Loop vs. while Loop

**Fundamentals of C - Part 2**
- Simple Conditions
- Compound Conditions
- Operators
- Truth Conditions
- Logical and Operator
- Logical or Operator
- Logical not Operator
- Loop Considerations
- exit
- break
- continue

**Using Functions in a C Program**
- Introduction
- Abstraction
- Functions
- Arguments and Parameters
- return Statement
- Using a Function
- Some Coding Techniques

**Arrays**
- Arrays vs. Non-Array Variables
- Arrays
- Array Subscripts
- Iterating Through an Array
- Finding a Value in an Array
- Finding the Largest Value in an Array

**Appendix A: User Interfaces**
- Interfaces
- Character Based - Command Line
- Character Based - Interactive
- Graphical User Interfaces
- Client/Server Computing
- The Client
- The Server
- Running the Client/Server Application
- Web Based Applications

**Appendix B: Creating Libraries**
- Creating Libraries
- Using Libraries

**Appendix C: Strings in C**
- Character Arrays
- String Input Functions
- How String Functions Work
- Other String Functions
- String Output Functions
- Handling the End of File
- Converting a String to a Number
Software Development for Non-Programmers
Using Python

Course Description: This class has a wealth of knowledge that aspiring programmers need to master before tackling their first programming language. This course consists of topics such as hardware and software components of a computer system, the CPU, registers, operating systems, processes, number systems, data type formats, conversions among number bases, logic and reasoning, flow charting, pseudo code, control structures, differentiation among programming languages (from machine language to assemblers and compilers), data structures, libraries, networking, web programming, and a host of other topics. After learning the fundamentals, students will be given a chance to show their knowledge by writing some simple to intermediate programs in a programming language. The course is given using C, Java, or Python depending on student demand.

Who Should Attend: This course is intended as a prerequisite for anybody who wishes to become a programmer or who needs to know key programmer issues and who does not have the necessary background for such an undertaking. It is also ideal for those who wish to begin a programming career or for those web designers who wish to add a programming component to their skills. This course is also intended for those tasked with managing programmers despite having no programming experience. Likewise, testers will gain invaluable experience from the subject matter in this course.

Prerequisites: There are no prerequisites for this course.

Benefits of Attendance: Upon completion of this course, students will be able to:

- List the major components of a computer system.
- Differentiate between various kinds of computer storage.
- Distinguish between system software and application software.
- List the steps in the instruction/execution cycle.
- Differentiate between various kinds of programing language translators and programming paradigms.
- Convert from and to the following bases: binary, octal, decimal, and hexadecimal.
- Demonstrate how negative numbers are stored in memory.
- Demonstrate how various data types are stored in memory.

Course Outline:

Introduction
Introduction
What is a Computer System?
Input Units
Output Units
Memory
Central Processing Unit
Components of the CPU
Instruction Execution Cycle
Cache
Software
Functions of an Operating System
Loading the Operating System
Loading an Application Program
Timesharing

Programming Languages
Introduction
Machine Language
Assembly Language
Compilers
BNF
Language Classification
Procedural vs. Object Oriented
Static vs. Dynamic Typing
Scripting vs. Non-Scripting
Choosing a Programming Language

The Programming Cycle
The Software Development Cycle
The Programming Cycle - Overview
Compile

Number Systems and Data Types
Number Bases
Base 10
Base 2
Base 8 and Base 16
Conversion Between Number Bases

Fundamentals of Python - Part 1
Python Data Types
Variables
Print
Arithmetic Operators
Control Flow - Decisions
Control Flow - Loops
The while Loop
The range Function
The for Loop
Indenting Control Decisions

Logical Operators
Logical and Operator
Loop Considerations
continue vs break

Using Functions in a Python Program
Introduction
Abstraction
Defining a Function
More on Functions

Lists
Lists
Accessing Elements of a List
List Functions
Finding a Value in a List
Finding the Largest Value in a List
Passing a List to a Function

Collections
Boolean Values
Bitwise Operators
Bitwise Examples
The int and chr Functions

Appendix A: User Interfaces - Other Languages
Interfaces
Character Based - Command Line
Character Based - Interactive
Graphical User Interfaces
Client/Server Computing
The Client
The Server
Running the Client/Server Application
Web Based Applications

Appendix B: Creating Libraries
Creating Libraries
Standard Python Libraries
Built-in Functions

Appendix C: Strings in Python
Strings
String Operations
String Functions
Converting a String to a Number

Relationships Between Number Bases
Data Types
Negative Numbers
Sign Magnitude
Complement Arithmetic
Floating-Point Values
String Data
Data Types in Python

Logical not Operator

Fundamentals of Python - Part 2
Simple Conditions
Compound Conditions
Operators
Truth Conditions
Logical Operators
Logical Operators and Logical or Operator

http://www.trainingetc.com
Software Development for Non-Programmers Using Java

Course Description:

Who Should Attend:

Prerequisites:
C Programming

Course Description: This course provides students with a comprehensive study of the C programming language. Classroom lectures stress the strengths of C, which provide programmers with the means of writing efficient, maintainable, and portable code. The lectures are supplemented with non-trivial lab exercises.

Who Should Attend: This course is for programmers who have had experience in any programming language or have been tasked with a C programming project, and other technical types including managers and customer support engineers who need to know C.

Prerequisites: Students should have taken the Software Development for Non-Programmers course or have experience with a programming or an assembly language.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Write C programs that are non-trivial.
- Use the variety of data types appropriate to specific programming problems.
- Utilize the modular features of the language.
- Demonstrate efficiency and readability.
- Demonstrate the use of the various control flow constructs.
- Use arrays as part of the software solution.
- Utilize pointers to efficiently solve problems.
- Include the structure data type as part of the solution.
- Create their own data types.
- Use functions from the portable C library.

Course Outline:

Getting Started
What is C?
Background
Sample Program
Components of a C Program
Examples
Data Types
Variables
Naming Conventions for C Variables
Printing and Initializing Variables
Array Examples
Compiling and Executing a C Program

Functions and Operators
Functions
Invoking Functions
Elementary Operators
The Assignment Operators
Number of Operands
Increment and Decrement Operators

Control Flow Constructs
Examples of Expressions
if
if-else
while
for
Endless Loops
do while
break and continue
switch
else if

The C Preprocessor
#define
Preprocessor Macros
#include
Conditional Compilation
#undef

Simple I/O
Character I/O
End of File
Simple I/O Examples
Simple I/O Redirection
I/O with Character Arrays

More on Functions
Introduction
Function Declarations
Returning a Value or Not
Function Prototypes
Arguments and Parameters
Organization of C Source Files

Extended Example
The strcpy Function
The strlen Function
The check Function
The atof Function
The average Function
Summary

Bit Manipulation
Defining the Problem Space
A Programming Example
Bit Wise Operators
Bit Manipulation Functions
Circular Shifts

Strings
Fundamental Concepts
Aggregate Operations
String Functions
String Functions Example

Higher Dimensional Arrays
Array Dimensions
An Array as an Argument to a Function
Two-Dimensional Array Example
String Arrays

Separate Compilation
Compiling Over Several Files
Function Scope
File Scope
Program Scope
Local static register and extern
Object Files
Libraries
The C Loader
Header Files

Pointers (Part 1)
Fundamental Concepts
Pointer Operators and Operations
Changing an Argument with a Function Call
Pointer Arithmetic
Array Traversal
String Functions with Pointers
Pointer Difference
Pointers for String Functions
Relationship Between an Array and a Pointer
The Pointer Notation *p++

Pointers (Part 2)
Dynamic Storage Allocation - malloc
Functions Returning a Pointer
Initialization of Pointers
gets - a Function Returning a Pointer
An Array of Character Pointers
Two Dimensional Arrays vs. Array of Pointers
System Line Arguments
Pointers to Pointers
Practice with Pointers
Function Pointers

Structures
Fundamental Concepts
Describing a Structure
Creating Structures
Operations on Structures
Functions Returning Structures
Passing Structures to Functions
Pointers to Structures
Array of Structures
Functions Returning a Pointer to a Structure

Structure Related Items
typedef - New Name for an Existing Type
Bit Fields
unions
Non-Homogeneous Arrays
Enumerations

File I/O
System Calls vs. Library Calls
Opening Disk Files
fopen
I/O Library Functions
Copying a File
Character Input vs. Line Input
scanf
fprintf
fclose
feof

Signal Functions
exit and abort
signal
memsys
memset
gsget
Binary Search - bsearch

Appendix A: C Language
Precedence Chart

Appendix B: Useful Library Functions
strcat
strchr, strcspn
system
strtok
strcpyn
strtok

Information About Files
The stat Function
File Existence
Telling Time - time and ctime
Telling Time - localtime

I/O With Structures
A Database Application
The menu Function
fwrite
The create_db Function
fread
The print_db Function
fseek

http://www.trainingetc.com
Phone: 410-290-8383 / Fax: 410-290-9427
Course Description: This course broadens the skills of a C language programmer by introducing sophisticated problem solving techniques including the advanced use of pointers, abstract data types, data structures, portability, and optimization techniques. Skills are reinforced by hands-on laboratory exercises.

Who Should Attend: This course is for anybody who has been programming in C for at least six months or who is a skilled programmer and has taken a C introductory course.

Prerequisites: Students should have at least six months of C programming experience.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Master the use of pointers in a wide variety of problems.
- Use sophisticated pointer techniques to solve problems involving advanced data structures such as lists, stacks, queues and trees.
- Choose from a wide variety of data structures to implement the most efficient solution to a problem.
- Apply the proper optimization technique to your C code.
- Apply many portability techniques to your C code.
- Use bit manipulation techniques for efficient solutions to problems.
- Write programs which emphasize modern program design techniques which emphasize code reuse.
- Write powerful C programs which make “calls” directly into the UNIX operating system through the use of system calls.
- Decrease development time in writing applications through a more thorough understanding of sophisticated concepts in C.

Course Outline:

A Review of C
Data Types
Operators
Control Flow Constructs - if
Loops
switch
Derived Data Types
Arrays
Array vs. Pointer
Arrays and Pointers
Structures
Header File for a Structure
Use of Structures
Structure References
Structure Assignments
Unions
Bitfields
Enumerations

Functions
Function Fundamentals
Function Prototypes
Function Invocation and Definition
Function Prototypes
Subprogram Examples
Functions Returning a Value
Return Value Considerations
Recursive Functions
Evaluation of Function Arguments
Variable Number of Arguments
Scope of Variables
Storage Class Attributes
Initialization

Bit Manipulation
Characteristics of Bitwise Problems
Defining the Problem Space
Bitwise Operators
Readability Aids
Assigning Bit Values
Writing Bitwise Functions
Circular Shifts
Character Information Array
Direct Lookup
Mapping With Bits
Radix Sort

Pointers
Common Pointer Constructions
Pointer Arithmetic
Binary Search
Command Line Arguments
The Environment Pointer

Changing a Pointer through a Function Call
Processing Arrays With Pointers
Simulation Example
Simulating Higher Dimensional Arrays
Two Dimensional Arrays
Complex Declarations
Pointers to Functions
Sorting with Large Records

Designing Data Types:
Structures
Steps in Creating Data Types
Rationale for a New Data Type
The File fraction.h
Operations on the Fraction Data Types
Implementation of the Functions
Example Program Using Fractions
Applications with Fractions
Set Notation Examples
Creating the Set Type
Set Representation Example
Set Representation
Set Function Implementations
A Program That Uses the Set Data Type

Data Structures - Part 1
Potential List Elements
Lists - What are They?
Lists as Arrays
Benefits of Linking Elements
A List of Linked Elements
Defining the List Data Type
Implementations of List Functions
A Simple Program With a List
Other Types of Lists
Ordered Lists
The rand Function
Circular Lists
Circular List Code
Circular Lists Principles
Two Way Lists
Structures for Networks
Networks
Stacks
Stack Representation
Solving Problems With Stacks
Picture of the Stack
Push and Pop Functions
The Calculator Driver Program
Queues
Queue Driver Program

Data Structures - Part 2
Hashing is Close to Direct Lookup
Hashing / Searching
The Hashing Algorithm
Hashing main Program
Linear Collision Handling
Chaining Collision Handling
Chaining - lookup Function
Chaining Code - main
Binary Trees
Traversing Trees
Left-Root-Right Traversal
Tree Algorithms

Optimization Techniques
Knowing When to Optimize
Where to Optimize
Macros
Knowing When to Initialize
Initialization
Modifying the Data Structure
Caching - Ready Access
Invariant Expressions
Logical Inefficiencies
Odds and Ends

Portability
Different Kinds of Portability
Source Code Portability
Prototype Problems
Portability with Functions
Problems with int
Arithmetic Data Types
Problems with Bits
Bit Manipulation
Portable Masks
Pointer Problems
Odds and Ends

Appendix A: Software Tools
The cc Command
Different C Compilers
Compiler Options
Conditional Compilation
The assert Macro
Libraries
Header File Support
The main Command
An Example Makefile
The make Dependency Tree
Source Code Control System

Appendix B: Library Functions
Building Command Strings

Appendix C: File Access
I/O From Applications Programs
System Calls vs. Library Calls
The fopen Function
Access Modes
Errors in Opening Files
Example: Copying a File
I/O Library Calls
Character Input vs. Line Input
Interpreting Input
The scanf Function
scanf Variants
fprintf Variants
The fclose Function
Servicing Errors
Application for Binary I/O
create_db Function - fwrite
print_db Function - fwrite
create_db Function - fread
retrieve_db Function
fseek
fflush and ftell

system
exit and _exit
signal
strtok
mempy and memmove
gsort
buscar
strstr
strchr and strcstr
Data Validation Example
strspn and strcspn

http://www.trainingetc.com Phone: 410-290-8383 / Fax: 410-290-9427
Course Description: C++ is the object oriented superset of ANSI C. This course provides students with a comprehensive study of the C++ Programming Language. The course stresses the object paradigm including classes, inheritance, virtual functions, and templates in the development of C++ programs. Lab exercises reinforce the lectures.

Who Should Attend: Anybody who has the need to write programs in the C++ language including programmers, engineers, scientists, or other technical support personnel will benefit from this course.

Prerequisites: Students should have taken the Software Development for Non-Programmers and Introduction to C courses or have equivalent knowledge.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Explain how object-oriented software engineering enhances the software development process.
- Identify the major elements in an object-oriented programming language.
- Implement the concepts of data abstraction and encapsulation in the creation of abstract data types.
- Implement operator overloading.
- Use inheritance in C++.
- Select the proper class protection mechanism.
- Demonstrate the use of virtual functions to implement polymorphism.
- Write programs utilizing the I/O classes in C++.
- Understand some advanced features of C++ including templates, exceptions, and multiple inheritance.
- Compare the object vs the procedural approach to writing software.
- Use correct object oriented terminology.
- Define and use classes in a C++ program.
- Create and use abstract data types.
- Derive classes using inheritance in C++.
- Implement polymorphism by using virtual functions in a program.

Course Outline:

Perspective
- The Software Crisis
- Design Techniques
- Large Software Systems
- Roots of Object Technology
- What is Object-Oriented Programming?
- C++ and Object-Oriented Programming
- Why C++?
- Features of C++
- Pros and Cons of C++

The Language of Object-Oriented Programming
- What is an Object?
- What is a Class?
- Encapsulation
- Data Hiding
- The Public Interface
- Relationships Among Classes
- Inheritance
- Polymorphism
- Object-Oriented Design

C vs. C++
- Comments
- Namespaces
- Simple Output
- Simple Input
- Definitions Near to First Use
- Function Prototypes
- The inline Specifier
- const
- Structure Members
- The Reference Type
- Overloading Function Names
- Default Parameters
- The Scope Resolution Operator
- Aggregates
- Operators new and delete
- The bool Data Type
- The string Data Type

Fundamentals of Classes
- Data Types
- User Defined Data Types
- Using the Class Concept
- Defining a Class
- public and private Access Levels
- The Scope Resolution Operator ::
- Using Class Objects Like Built-In Types

Scope
- Constructors
- Member Initialization Lists
- Destructors
- Array of Objects
- Pointers
- The this Pointer
- Passing Objects to Functions
- Returning Objects From Functions
- static Class Members

Operator Overloading
- Introduction
- Rules for Operator Overloading
- Rationale for Operator Overloading
- Overloading Member Functions
- Overloading Non-Member Functions
- friend Functions
- The Copy Constructor
- The Assignment Operator
- Overloading [ ]
- Overloading Increment and Decrement Operators
- const: Objects and References

Composition of Classes
- Relationships
- Composition of Classes
- The Point Class
- The Line Class
- Member Initialization Lists
- An Application With Composition
- The Copy Constructor Under Composition
- operator= Under Composition

Inheritance
- Introduction
- Public Base Classes
- The protected Access Level
- Member Initialization Lists
- What Isn’t Inherited
- Assignments Between Base and Derived Objects
- Compile-Time vs. Run-Time Binding
- virtual Functions
- Polymorphism
- virtual Destructors
- Pure virtual Functions
- Abstract Base Classes
- An Extended Inheritance Example

I/O in C++
- The iostream Library
- Predefined Streams
- Overloading operator<<
- Overloading operator>>
- Manipulators
- Stream States
- Formatted I/O
- Disk Files
- Reading and Writing Objects

Advanced Topics
- Template Functions
- Template Classes
- Multiple Inheritance
- User-Defined Conversions
- Data Structures
- An Iterator Class
- Exceptions

Introduction to the Standard Template Library
- Introduction
- The Standard Template Library
- Design Goals
- STL Components
- Iterators
- Example: vector
- Example: list
- Example: set
- Example: map
- Example: find
- Example: merge
- Example: accumulate
- Function Objects
- Adaptors

Function Prototypes
- Simple I/O

Appendix B: More I/O in C
- The printf Function
- The scanf Function
- The preprocessor
- Conditional Compilation
- Avoiding Multiple Inclusion for the Same File

Appendix C: Aggregates in C
- Data Types Revisited
- Aggregate Types
- Arrays
- Structures
- Structures and Functions
- Bit Fields
- Enumeration Types

Appendix D: Pointers in C
- Fundamental Concepts
- Pointer Operations
- Using Pointers to Alter a Function Argument
- Using Pointers for Array Traversal
- Pointer Arithmetic
- Sending an Array to a Function
- Command Line Arguments
- Pointers vs. Arrays
- Sending an Aggregate to a Function
- Summary of the Uses of Pointers

Appendix E: Bibliography
- Bibliography
Advanced C++ Programming

Course Description: This course broadens the skills of a C++ language programmer by examining sophisticated C++ concepts such as templates, exceptions, memory management, advanced inheritance issues, disambiguation of overloaded functions, private and protected inheritance, binary I/O and class libraries.

Who Should Attend: This course is for anybody who has programmed in C++ and wishes to enhance their knowledge of the language.

Prerequisites: Students should have completed an introductory C++ programming course or have equivalent knowledge.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Differentiate between global functions, friend functions and member functions.
- Use the C++ Standard Template Library.
- Write classes and functions with parameterized types.
- Understand and handle exceptions in C++ programs.
- Disambiguate data and functions using multiple inheritance.
- Understand the difference between various kinds of inheritance.
- Use pointers to class member functions.
- Understand the C++ mechanism to resolve overloaded functions.

Course Outline:

What You Should Already Know - A Review
Rationale for a New Programming Language
The Language of Object-Orientation
A Typical C++ Class
Issues Regarding Member Functions vs. Non-Member Functions
friend or non-friend
Functions Returning References
Relationships
Initialization Lists
Inheritance In C++
Access Levels
Simple C++ I/O
The Many Uses of const

Parameterized Types - Templates
Templates
Overloading Functions
Template Functions
Specializing a Template Function
Disambiguation Under Specialization
Template Classes
Instantiating a Template Class Object
Rules for Template Classes
A Non-Member Function with a Template Argument
Friends of Template Classes
Templates with Multiple Type Parameters
Comments Regarding Templates

Relationships of all Kinds
Uses of Member Initialization Lists
Initialization Lists Under Composition
Initialization Lists Under Inheritance
Efficiency
Dynamic Memory Allocation
operator= and Composition
Constructors and Composition
What is Not Inherited?
Summary - Constructors and operator=
Public Inheritance
virtual Functions
A Shape Class Hierarchy
Polymorphism
Pure Virtual Functions
Abstract Base Classes
Private Inheritance
"Using" Relationships
Associations

Multiple Inheritance
Multiple Inheritance
Ambiguities
Ambiguities
virtual Base Classes
The Dominance Rule
Member Initialization Lists

Multiple Inheritance and Composition
operator=

Data Structures
Introduction
A Simple List
Layering Type-safe Classes Upon List
A Template List Class
Iterators
A Template Iterator
Stack and Queue Classes
Templates and Inheritance

Function Pointers
Why Have Function Pointers?
Passing Functions as Arguments
Registering Functions
Function Pointers in C++
Callback Functions
A Class with a Callback Object
Registration of Exception Handlers

Exceptions
What Are Exceptions?
Traditional Approaches to Error Handling
try, catch, and throw
A Simple Exception Handler
Multiple catch Blocks
The Exception Specification List
Rethrowing an Exception
Cleanup
Exception Matching
Inheritance and Exceptions
Resource Allocation
Constructors and Exceptions
Destructors and Exceptions
Catch by Reference

The C++ Standard Template Library
Introduction
The Standard Template Library
Design Goals
STL Components
Iterators
vector
list
set
map
find
merge
accumulate
Adaptors

Disambiguation
Conversion
int: Conversion
float: and double: Conversions
Arithmetic and Pointer Conversion

Inheritance Based Conversion
Overloaded Functions
Exact Match
Match with Promotion
Match with Standard Conversion
User Defined Conversion
Constructors as Conversion Operators
Ambiguities

I/O
Introduction
Manipulators
Writing Your Own Manipulators
Overloading the I/O Operators
Disk Files
Reading and Writing Objects
Internal Transmission of Data
A Spell Checker
Handling Streams in Constructors and Destructors
Treating a File as an Array

Miscellaneous Topics
Namespaces
Reference Counts
RTTI
Casts
Having a Limited Number of Objects
Smart Pointers
Java Programming

Course Description: This course teaches students how to develop Java applications. Topics covered include the Java programming language syntax, OO programming using Java, exception handling, file input/output, threads, collection classes, and networking. Students will develop and test Java applications (typically) using Eclipse. This course is a pre-requisite to all Application Server courses, and speciality Java Technology courses such as Struts, Spring, and Hibernate.

Who Should Attend: This course is designed for applications programmers and designers planning to develop applications using the Java Virtual Machine.

Prerequisites: Students should have taken the Software Development for Non-Programmers course or have programmed in at least one programming language - preferably C or C++. Some familiarity with Object Oriented Programming is desired but not required.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Compile and run a Java application.
- Understand the role of the Java Virtual Machine in achieving platform independence.
- Navigate through the API docs.
- Use the Object Oriented paradigm in Java programs.
- Understand the division of classes into Java packages.
- Use Exceptions to handle run time errors.
- Select the proper I/O class among those provided by the JDK.
- Use threads in order to create more efficient Java programs.

Course Outline:

Introduction
What is Java?
History
Versioning
The Java Virtual Machine
Writing a Java Program
Packages
Simple Java Programs

Language Components
Primitive Data Types
Comments
The for Statement
The if Statement
The while and do while Statements
The break Statement
The continue Statement
Operators
Casts and Conversions
Keywords

Object-Oriented Programming
Defining New Data Types
Constructors
The String Class
String Literals
Documentation
Packages
The StringBuffer Class
Naming Conventions
The Date Class
The import Statement
Deprecation
The StringTokenizer Class
The DecimalFormat Class

Methods
Introduction
Method Signatures
Arguments and Parameters
Passing Objects to Methods
Method Overloading
Static Methods
The Math Class
The System Class
Wrapper Classes

Arrays
Introduction
Processing Arrays
Copying Arrays
Passing Arrays to Methods
Arrays of Objects

Encapsulation
Introduction
Constructors
The this Reference
Data Hiding
public and private Members
Access Levels
Composition
Static Data Members

Inheritance & Polymorphism
Introduction
A Simple Example
The Object Class
Method Overriding
Polymorphism
Additional Inheritance Examples
Other Inheritance Issues

Abstract Classes and Interfaces
Introduction
Abstract Classes
Abstract Class Example
Extending an Abstract Class
Interfaces

Exceptions
Introduction
Exception Handling
The Exception Hierarchy
Checked Exceptions
Advertising Exceptions with throws
Developing Your Own Exception Classes
The finally Block

Input and Output in Java
Introduction
The File Class
Standard Streams
Keyboard Input
File I/O Using Byte Streams
Character Streams
File I/O Using Character Streams
Buffered Streams
File I/O Using a Buffered Stream
Keyboard Input Using a Buffered Stream
Writing Text Files

Threads
Threads vs. Processes

Creating Threads by Extending Thread
Creating Threads by Implementing Runnable
Advantages of Using Threads
Daemon Threads
Thread Stacks
Thread Problems
Synchronization

Collections
Introduction
Vectors
HashTables
Enumerations
Properties
Collection Framework Hierarchy
Lists
Sets
Maps
The Collections Class

Networking
Networking Fundamentals
The Client/Server Model
InetAddress
URLs
Sockets
A Time-of-Day Client
Writing Servers
Client/Server Example
Advanced Java Programming

Course Description: Advanced Java is a comprehensive study of many advanced Java topics. These include assertions, collection classes, searching and sorting, regular expressions, logging, bit manipulation, serialization, with sockets, Remote Method Invocation, and Java Database Connectivity.

Who Should Attend: This course is intended for Java programmers who wish to write programs using many of the advanced Java features.

Prerequisites: Students should have completed a beginning Java course or have programmed in Java for at least three to six months.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Document and package a Java application.
- Use many of the new enhancements added to the Java API.
- Use assertions to write robust Java code.
- Use regular expressions for efficient pattern matching.
- Choose appropriate data structures from the Java Collection API.
- Sort and search arrays and lists using a variety of techniques.
- Capture configuration and debugging information using the Java Logging APIs.
- Use Generics to create type safe collections.
- Serialize Java objects.
- Use features of the new I/O API.
- Write TCP/IP Client Server applications using sockets.
- Write multi-threaded Java applications.
- Execute methods on a remote object using RMI.
- Perform database queries and updates using JDBC.

Course Outline:

**Review of Java Fundamentals**
- The Java Environment
- Data Types
- The String Class
- The StringBuffer Class
- Arrays
- Passing Data Types to a Method
- Constructors and Initialization
- Inheritance
- Abstract Classes
- Interfaces
- Static Data, Methods, and Blocks
- Wrapper Classes
- I/O

**Packaging and Distributing a Java Application**
- Packages
- Managing Source and Class Files
- The javadoc Utility
- Documenting Classes and Interfaces
- Documenting Fields
- Documenting Constructors and Methods
- Running the javadoc Utility
- jar Files
- The Manifest File
- Bundling and Using Jar-Packaged Resources

**Miscellaneous Enhancements**
- Enhanced for Loop
- Autoboxing and Auto-Unboxing
- Static Imports
- varArgs
- Typeface Enums
- Formatted Strings
- Format Specifier Syntax
- Format Specifier Conversions
- Format Specifier Flags
- Formatted Integers Example
- Formatted Floating Points Example
- Formatted Strings Example
- Formatted Dates Example
- Complex Formatted Example

**Assertions**
- Introduction
- Assertion Syntax
- Compiling with Assertions
- Enabling and Disabling Assertions
- Assertion Usage

**Regular Expressions**
- Regular Expressions
- String Literals
- Character Classes
- Quantifiers
- Capturing Groups and Backreferences
- Boundary Matchers
- Pattern and Matcher

**The Java Collection Classes**
- Introduction
- The Arrays Class
- Searching and Sorting Arrays of Primitives
- Sorting Arrays of Objects
- The Comparable and Comparator Interfaces
- Sorting - Using Comparable
- Sorting - Using Comparator
- Collections
- Lists and Sets
- Iterators
- Lists and Iterators Example
- Maps
- Maps and Iterators Example
- The Collections Class
- Rules of Thumb

**Generics**
- Introduction
- Defining Simple Generics
- Generics and Subtyping
- Wildcards
- Bounded Wildcards
- Generic Methods

**Advanced I/O**
- Introduction
- Basic File I/O Example
- Buffered I/O
- The Console Class
- Object Serialization
- Serialization Issues
- Compressed Files
- Zip File Example
- Writing Your Own I/O Classes
- Property Files
- The Preferences Class

**Enhanced I/O**
- Introduction
- Channels
- Buffers
- Typed Buffers
- Direct Buffers

**Logging API**
- Introduction
- Loggers
- Logger Levels
- Logger Handlers
- Specifying Handlers and Formatters
- Configuring Handlers
- LogManager

**Networking**
- Networking Fundamentals
- The Client/Server Model
- InetAddress
- URLS
- Sockets
- A Time-of-Day Client
- Writing Servers
- Client/Server Example

**Threads and Concurrency**
- Review of Fundamentals
- Creating Threads by Extending Thread
- Creating Threads by Implementing Runnable
- Advantages of Using Threads
- Daemon Threads
- Thread Stacks
- Thread Problems
- Synchronization
- Performance Issues

**Remote Method Invocation (RMI)**
- Introduction
- RMI Architecture
- The Remote Interface
- The Remote Object
- Writing the Server
- The RMI Compiler
- Writing the Client
- Remote Method Arguments and Return Values
- Dynamic Loading of Stub Classes
- Remote RMI Client Example
- Running the Remote RMI Client Example

**Java Database Connectivity (JDBC)**
- Introduction
- Relational Databases
- Structured Query Language
- A Sample Program
- Transactions
- Meta Data

http://www.trainingetc.com
**Course Description:**
This 5-day course will introduce Java programmers to the Swing package. The Swing package, extending the AWT, provides efficient and easy-to-use tools for manipulating graphics. Students completing this course will have the ability to create fully-functional, customizable, graphical user interfaces.

**Who Should Attend:**
This course is designed for Java programmers of any level.

**Prerequisites:**
Java programming experience is required. Experience with the AWT is helpful, but not required.

**Benefits of Attendance:**
Upon completion of this course, students will be able to:

- Describe the key features of Swing
- Contrast the use of Swing and the AWT
- Add components to a JFrame using the ContentPane
- Write Swing Applications using Swing components such as JLabel, JButton, JTextField, JTree, etc.
- Implement keystroke handling in response to keyboard events from a user
- Understand InputMap and ActionMap

- Use the printing capability to print GUIs
- Describe the Model View Controller Architecture
- Encode Model View Architecture components
- Write code that uses Border objects to alter the appearance of components
- Create programs using various Pane components such as JTabbedPane, JSplitPane, JScrollPane, and JOptionPane
- Write programs that support cut/paste and drag/drop

**Course Outline:**

**An Introduction to Swing**
- Rational for the Swing Components
- AWT Foundation Classes
- Lightweight Components
- Features of the Swing Components
- The AWT Components
- An Example Using the AWT Components
- Building a Minimum Swing Application

**Window Events**
- Introduction
- Window Positioning
- Handling Window Events
- Window Closing Styles
- Inner Classes
- Anonymous Inner Classes

**Simple Swing Components**
- Introduction
- JButton with Selected Fonts
- JButton with an Icon
- The Icon Interface
- JButton with a Mnemonic Labels
- JLabel, JTextField, JTree, etc.

**Keystrokes**
- The Focus
- Focus Traversal
- Keyboard Input
- Keyboard Shortcuts
- Keystrokes and Actions

**Dialogs**
- What is a Dialog?
- A Simple Dialog
- Adding Components to a Dialog

**Menus and Toolbars**
- AWT vs. Swing Menus
- Adding Mnemonics
- Adding Accelerators
- Adding Colors Fonts, and Icons Toolbars

**Painting and Graphics**
- Painting Basics
- The paintComponent Method
- Some Methods of the Graphics Class
- Creating a Drawing Area
- Drawing Area Source Code

**Printing**
- Printing Basics
- A Printing Example

**Using the Book Class**

**Layout Managers**
- Introduction
- FlowLayout
- GridLayout
- BorderLayout
- CardLayout
- GridBagLayout
- GridBagConstraints
- BoxLayout
- OverlayLayout

**Models, Views, and Controllers**
- Introduction
- The MVC Process of JButton
- Advantages to using the MVC Architecture
- Pluggable Look and Feel
- An MVC Example
- Linking the Components
- MVC Diagram
- ListModel
- AbstractListModel
- JTable
- TableModel
- MVCList Source Code

**Using the JTree**
- Tree Basics
- The DefaultMutableTreeNode
- DefaultTreeModel
- Constructing a JTree
- Customizing the JTree
- Tree Listeners

**Text Components**
- Various Text Components of Swing
  - JTextField
  - JPasswordField
  - JTextArea
  - Using a JScrollPane
  - JTextPane
  - StyleConstants
  - StyleTest Source Code

**Utility Panes, Borders, and Focus**
- Various Utility Panes
  - JTabbedPane
  - JSplitPane
  - JFileChooser
  - JTextPane
  - JComponent
  - Using Borders

**Creating Custom Components**
- Extending JButton
- Creating JButton
- RButton's Model to View
- The Constructor
- More with Listeners
- RButton Source Code

**Clipboard and Drag & Drop**
- Clipboard
- DataFlavors
- System Clipboard vs. Custom Clipboard
- More with DataFlavors
- Drag & Drop

**Appendix A: Events and Their Listeners**
- AWT Events
- Swing Events

**Appendix B: Accessibility**
- Accessible Applications
- The ADA and Section 508 Regulations
- Accessibility Needs
- Java Support for Accessibility
- The Java Accessibility API
-AccessibleContext
- Section 508 Accessibility Standards
- Section 1194.21 (a) - Keyboard Access
- Section 1194.21 (b) - Accessibility Features
- Section 1194.21 (c) - Input Focus
- Section 1194.21 (d) - Object Information
- Section 1194.21 (e) - Bitmap Images
- Section 1194.21 (f) - Textual Information
- Section 1194.21 (g) - User Selected Attributes
- Section 1194.21 (h) - Animation
- Section 1194.21 (i) - Color Coding
- Section 1194.21 (j) - Color and Contrast
- Section 1194.21 (k) - Flicker Rate
- Section 1194.21 (l) - Electronic Forms

**Resources**
- The Java Accessibility Bridge
**Introduction to Hibernate**

**Course Description:** This course introduces Hibernate, a popular open-source object/relational mapping (ORM) tool that helps Java developers store and access persistent objects. Topics covered include Hibernate configuration, the Hibernate mapping file, inheritance, collections, associations, and the Hibernate Query Language (HQL).

**Who Should Attend:** This course is for Java developers creating or maintaining applications that use a relational database and Java SE or Java EE, Java developers and architects investigating ORM alternatives.

**Prerequisites:** Students should have Java programming experience and knowledge of Structured Query Language (SQL).

**Benefits of Attendance:** Upon completion of this course, students will be able to:

- Describe the purpose and benefits of an object/relational mapping tool
- Configure database connection properties in the Hibernate configuration file
- Use JPA and Hibernate Annotations to map Java classes to database tables
- Create, save, update, and delete entities
- Distinguish between entity and value types
- Configure primary key generators for persistent classes
- Describe and use the Hibernate strategies for mapping inheritance hierarchies
- Map collections and associations
- Write queries using Hibernate Query Language (HQL)

**Course Outline:**

**Getting Started with Hibernate**

- What is Hibernate?
- Preparing to Use Hibernate
- Configuring Hibernate
- JDBC and Datasource Properties
- Other Configuration Properties
- Hibernate Sessions
- Writing Classes for Hibernate Applications
- Sample Class and Database Table
- Sample Class and Mapping Diagram
- Creating and Saving a New Entity
- Locating an Existing Entity
- Updating an Existing Entity
- Deleting an Entity
- Executing an HQL Query
- Schema Generation
- Programmatic Configuration

**Mapping Persistent Classes**

- Class Annotations
- Access Type
- Property Annotations
- Hibernate Types
- Entities and Values
- Mapping Embeddable Types
- Compound Keys
- Generated Keys

**Inheritance**

- Mapping Class Inheritance
- Table Per Class Hierarchy
- Table Per Subclass
- Table Per Concrete Class

**Collections and Associations**

- Mapping Collections
- Initializing Collections
- Sample Application - UML Diagram
- Sample Application - Database Schema
- Implementing Associations
- Mapping Associations

**Hibernate Query Language**

- HQL Basics
- HQL Expressions
- HQL Functions
- Polymorphic Queries
- Executing Queries
- Scrollable Results
- Named Queries
- Associations and Joins
- Inner Joins
- Outer Joins
- Sample Queries

**Appendix A: Working with the Lab Files**

- Importing the Eclipse Project
- Working with MySQL
- Creating Tables for the Training Application
- Hibernate Console Configuration
Introduction to Spring Framework

Course Description: This course introduces the Spring Framework, the leading full-stack framework for Java EE applications. Topics covered include the Spring container, dependency injection, data validation, aspect-oriented programming, the JDBC Template, and the Hibernate Template. A Web application is also presented to illustrate the use of the Spring Web MVC framework.

Who Should Attend: This course is for Java developers and architects who wish to explore a popular, open-source alternative to traditional Java EE programming.

Prerequisites: Students must have strong Java programming skills and exposure to Java EE technology.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Describe the seven component modules of the Spring Framework.
- Understand the basic philosophies of Spring.
- Explain the purpose and benefits of dependency injection.
- Configure beans in a Spring configuration file.
- Use setter and constructor injection with Spring beans.
- Create property files for error messages and to support internationalization.
- Write validators and property editors for user-defined data.
- Understand basic concepts of aspect-oriented programming.
- Use the JDBC template to simplify database access.
- Use the Hibernate template to integrate Hibernate and Spring.
- Create Web applications using the Model-View-Controller architecture.
- Write forms and controllers for Spring Web applications.

Course Outline:

Introduction to Spring
What is Spring?
Overview of the Spring Framework
Spring Philosophies
Spring Documentation
Java 5 Language Features

A First Look at Spring
A Simple Example
Wiring Beans
Configuring a Properties File
Schema-Based Configuration

Beans and Containers
Spring Containers
Spring Configuration File
Spring Beans
Using the Container
The BeanFactory Interface
Singleton vs. Prototype
Bean Naming
Dependency Injection
Setter Injection
Constructor Injection

The Application Context
The ApplicationContext Interface
Accessing Application Components
Accessing Resources
Internationalization with MessageSource
Application Events

Data Validation and Conversion
The Validator Interface
The Errors Interface
The ValidationUtils Class
Validator Example
Testing the Validator
Property Editors
Custom Property Editors

Aspect-Oriented Programming
Aspect-Oriented Programming
AOP Concepts
AOP Proxy
The AOP Alliance
Types of Advice
AOP Example
Introductions

Using JDBC with Spring
A Simpler Approach

Using Hibernate with Spring
What is Hibernate?
Hibernate Sessions
The HibernateTemplate
Sample Class and Mapping File
Creating and Saving a New Entity
Locating an Existing Entity
Updating an Existing Entity
Hibernate Query Language
Executing Queries

Spring Web MVC - Part 1
Spring Web MVC
The DispatcherServlet
Writing a Controller
A Simple View Page
Configuring the Controller
Adding a View Resolver
Adding a Message Bundle
Adding Business Classes
Adding Test Data
Accessing a Database
Adding a Form
Updating the Database
Integrating Hibernate

Spring Web MVC - Part 2
Handler Mappings
View Resolution
Chaining View Resolvers
Controllers
AbstractWizardFormController

Appendix A: Resources
Resources
Course Description: This course combines the following courses into a single course:

Introduction to Spring Framework
Introduction to Hibernate

The description of each course is listed below:

This course introduces the Spring Framework, the leading full-stack framework for Java EE applications. Topics covered include the Spring container, dependency injection, data validation, aspect-oriented programming, the JDBC Template, and the Hibernate Template. A Web application is also presented to illustrate the use of the Spring Web MVC design pattern.

This course introduces Hibernate, a popular open-source object-relational mapping (ORM) tool that helps Java developers store and access persistent objects. Topics covered include Hibernate configuration, the Hibernate mapping file, inheritance, collections, associations, and the Hibernate Query Language (HQL).

Who Should Attend: This course combines several courses, the descriptions for who should attend are listed below: This course is for Java developers and architects who wish to explore a popular, open-source alternative to traditional Java EE programming. This course is for Java developers creating or maintaining applications that use a relational database and Java SE or Java EE, Java developers and architects investigating ORM alternatives.

Prerequisites: Students must have strong Java programming skills and exposure to Java EE technology. Students should have Java programming experience and knowledge of Structured Query Language (SQL).

Benefits of Attendance: Upon completion of this course, students will be able to:

- Describe the seven component modules of the Spring Framework.
- Understand the basic philosophies of Spring.
- Explain the purpose and benefits of dependency injection.
- Configure beans in a Spring configuration file.
- Use setter and constructor injection with Spring beans.
- Create property files for error messages and to support internationalization.
- Write validators and property editors for user-defined data.
- Understand basic concepts of aspect-oriented programming.
- Use the JDBC template to simplify database access.
- Use the Hibernate template to integrate Hibernate and Spring.
- Create Web applications using the Model-View-Controller architecture.
- Write forms and controllers for Spring Web applications.
- Describe the purpose and benefits of an object/relational mapping tool
- Configure database connection properties in the Hibernate configuration file
- Use a Hibernate mapping file to map a Java class to a database table
- Create, save, update, and delete entities

Course Outline:

Introduction to Spring

What is Spring?
Overview of the Spring Framework
Spring Philosophies
Spring Documentation
Java 5 Language Features

A First Look at Spring
A Simple Example
Wiring Beans
Configuring a Properties File
Scope-Based Configuration

Beans and Containers
Spring Containers
Spring Configuration File
Spring Beans
Using the Container
The ApplicationContext
Singleton, Prototype
Bean Naming
Dependency Injection
Setter Injection
Construction Injection

The Application Context
The ApplicationContext Interface
Accessing Application Components
Accessing Resources
Internationalization with MessageSource
Application Events

Data Validation and Conversion
The Validator Interface
The ErrorHandler Interface
The FieldError Class
Validator Example
Testing the Validator
Property Editors
Custom Property Editors

Aspect-Oriented Programming
Aspect-Oriented Programming

Spring Web MVC

AOP Concepts
AOP Process
AOP Process
The AOP Alliance
Types of Advice
AOP Example
Introductions

Using JdbcTemplate
Using JdbcTemplate
A Simple Approach
Working with the HSQLDB Database
The JdbcTemplate Case
Exception Handling
Updating with the JdbcTemplate
Mappings fromHibernate to Java Classes

Using Hibernate with Spring
What is Hibernate?
Hibernate Sessions
The HibernateTemplate
Sample Class and Mapping File
Creating and Saving a New Entity
Locating an Existing Entity
Updating an Existing Entity
Hibernate Query Language
Executing Queries

Spring Web MVC - Part 1
Spring Web MVC
The DispatcherServlet
Writing a Controller
A Simple View Page
Configuring the Controller
Adding a View Resolver
Adding a Message Bundle
Adding Business Classes
Adding Test Data
Accessing a Database
Adding a Form
Updating the Database
Integrating Hibernate

Spring Web MVC - Part 2
Handler Mappings
View Resolution
Chaining View Resolvers
Controllers
AbstractWizardFormController

Appendix A: Resources
Resources

Appendix B: Spring IDE
Spring IDE
Adding the Spring Project
Managing Spring Configuration Files
Visualizing a Configuration File

Appendix C: Creating a Spring Project in Eclipse
Creating a Spring Project
Configuring the Build Path

Appendix D: Running the Examples in Eclipse
Running the Standalone Applications
Starting the Server
Running the Web Application
Modifying the Web Application
Stopping the Server

Getting Started with Hibernate
What is Hibernate?
Using Hibernate
Configuring Hibernate
Hibernate Sessions
Writing Classes for Hibernate Applications
Sample Class and Mapping File
Creating and Saving a New Entity
Locating an Existing Entity
Updating an Existing Entity
Deleting an Entity
Executing an SQL Query
Programmatically Configuring

Mapping Persistent Classes
The Hibernate Mapping File
Entities and Values
Class Mappings
Properties
Defined and Generated Properties
Mapping Value Types
Key Generators
Composite Keys
Hibernate Types

Inheritance
Mapping Class Inheritance
Table Per Class Inheritance
Table Per Subclass
Table Per Concrete Class
Using Inverse Polymorphism

Collections and Associations
Mapping Collections
Collections of Components
Sample Application - UML Diagram
Sample Application - Database Schema
Implementing Associations
Mapping Associations
The ImportFromConcept

Hibernate Query Language
HQL Basics
HQL Expressions
HQL Functions
Polymorphic Queries
Executing Queries
Scrollable Results
Named Queries
Associations and Join Keys
Inner Join
Outer Join
Sample Queries

Appendix A: XDoclet and Java Annotations

Using XDoclet Markup
Using Annotations
JavaServer Faces (JSF)

Course Description: The JavaServer Faces framework is establishing itself as the new standard for the development of web applications. Designed under Sun’s Java Community Process by many of the same people that developed the Jakarta Struts framework, Faces is proving itself to be the next step in the web application evolutionary process. JavaServer Faces, or simply JSF, combines Java Servlets and JavaServer Pages into a server-side implementation of the Model-View-Controller Design Pattern. The JSF framework provides developers with a unified infrastructure upon which Internet applications can be constructed.

The course also introduces the developer to the JSF architecture and provides the basis for planning, developing, and deploying Web based applications using the JSF framework. After taking this class, the developer will be able to quickly construct dynamic server-side web pages using JSF and integrate the Web application with many of the other Java2 Enterprise Edition application server methodologies such as Enterprise Java Beans, JavaMail, and SOAP. This class combines lecture with a unifying, hands-on experience, and open discussion that will help the developer quickly understand the benefits of JSF and how to use the framework.

Who Should Attend: This course is for Java Developers.

Prerequisites: Participants should already have a solid understanding of Java programming and understand the basics of XML. The course also assumes a basic understanding of HTML syntax and JavaServer Pages syntax. Understanding of Enterprise Java Beans (EJB) is also a plus.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Write web applications that take advantage of the FacesServlet, FacesContext and Action Java classes to control the user experience of the web application.
- Write JSF applications that gather and update information from external application servers such as EJBs, CORBA servers, and database servers.
- Create and use custom Tag Libraries in JavaServer Pages.
- Understand the basics of web security and learn to take advantage of the security features provided by the Web Server.
- Understand the use of the standard JSF Validators and how to write custom Validators.
- Understand the use of the standard JSF Data Conversion classes and how to write custom Data Converters.
- Take advantage of the JSF architecture that supports rendering output in several formats from the same application. Such as: HTML, WML, XML, etc.

Course Outline:
Introduction and Overview
The JSF Architecture
JSF Request Objects
Simple JSF User Interface components
The EL Expression Language and Advanced User Interface components
Event Handling
Data Validation
Data Conversion
Rendering Custom User Interface Objects

http://www.trainingetc.com  Phone: 410-290-8383 / Fax: 410-290-9427
This course is a comprehensive introduction to writing Enterprise JavaBeans (EJB) using the EJB 3.x specification. An overview of EJB3 is provided, followed by hands-on experience with session beans, entity beans, and message-driven beans. Topics also include container-managed relationships (CMR) and the EJB Timer Service. This course can be taught using IBM WebSphere, Oracle WebLogic, or the JBoss application server.

This course is for Java programmers and software engineers with experience writing enterprise JavaBeans and who wish to update from EJB2 to EJB3.

Students should be comfortable with Java programming and object-oriented concepts. In addition, students should have prior experience developing EJBs according the EJB 2.x specification.

Upon completion of this course, students will be able to:

- Start, stop, and configure an application server
- Use JNDI to access database and EJB resources
- Create JDBC data sources
- Write stateless and stateful session beans
- Use container-managed persistence
- Use standard annotations for EJBs

- Configure and deploy EJB applications
- Assemble Java EE-compliant enterprise applications
- Implement container-managed relationships
- Write message-driven beans
- Use the EJB Timer Service

### Course Outline:

**Java EE Development using Eclipse**
- Starting Eclipse
- Creating a Java Project
- Importing Existing Java Code
- Creating a New Java Program
- Java EE-Compliant Application Servers
- Creating a Server Instance
- Starting and Stopping the Server

**Using JDBC Data Sources**
- A Simple JDBC Program
- JDBC Driver Types
- Using the Derby Database
- JDBC Data Sources
- Data Source Example
- Configuring a JDBC Data Source
- Running the JDBC Examples
- Executing a Query
- Using the Database Explorer

**Introduction to EJB 3**
- Limitations of EJB 2
- EJB 3 Feature Overview
- Comparing EJB 2 and 3
- The EJB 3 Business Interface
- Review of Java Annotations
- The Annotated EJB Class
- Dependency Injection
- Container Callback Methods

**Session Beans**
- Session Bean Lifetime
- Session Bean Interface
- Session Bean Lifecycle
- Stateless Session Bean Example
- Accessing Environment Entries
- Stateful Session Bean Example
- EJB Exceptions - Examples
- Testing the Session Beans
- Creating a New Session Bean

**CMP Entity Beans**
- Entity Beans
- Entity Bean Interface
- Lifecycle of an Entity Bean
- Container-Managed Persistence
- Primary Key Class
- Implementing CMP Entity Bean Methods
- Container-Managed Persistence Example
- Custom Finders
- EJB Query Language
- Mapping Container-Managed Fields

**Container-Managed Relationships**

### Benefits of Attendance:

- Start, stop, and configure an application server
- Use JNDI to access database and EJB resources
- Create JDBC data sources
- Write stateless and stateful session beans
- Use container-managed persistence
- Use standard annotations for EJBs

- Configure and deploy EJB applications
- Assemble Java EE-compliant enterprise applications
- Implement container-managed relationships
- Write message-driven beans
- Use the EJB Timer Service

### Course Length
- 3 Days
Developing Java Enterprise Applications Using EJB3

Course Description: This course is a comprehensive introduction to writing Enterprise JavaBeans (EJB) using the EJB 3.x specification. An overview of Java EE technology is provided, followed by hands-on experience with JNDI, JDBC, JMS, session beans, entity beans, and message-driven beans. Topics also include container-managed persistence (CMP), container-managed relationships (CMR), and the EJB Timer Service. This course can be taught using IBM WebSphere, Oracle WebLogic, or the JBoss application server.

Who Should Attend: This course is for experienced Java programmers and software engineers preparing to write Enterprise JavaBeans for Java EE applications hosted on a Java EE-compliant application server.

Prerequisites: Students should be comfortable with Java programming and object-oriented concepts. A minimum of six months coding experience is suggested. In addition, students should have prior experience using JDBC and SQL.

Benefits of Attendance: Upon completion of this course, students will be able to:
- Start, stop, and configure an application server
- Use JNDI to access database and EJB resources
- Create JDBC data sources
- Write stateless and stateful session beans
- Use container-managed persistence
- Use standard annotations for EJBs
- Configure and deploy EJB applications
- Assemble Java EE-compliant enterprise applications
- Use the Session Facade pattern
- Implement container-managed relationships
- Create JMS destinations
- Write message-driven beans
- Use the EJB Timer Service

Course Outline:

Overview of Java EE
- Java Platforms
- Characteristics of “Enterprise” Computing
  - Java EE Technologies
- Multi-Tier Architectures
- Advantages of Multi-Tier Architectures
- Container-Based Approach
- Parties Involved in Java EE Deployment
- Java EE Application Models
  - HTTP Services Application Model
  - N-Tiered Application Model
- Java EE Development using Eclipse
  - What is Eclipse?
  - Starting Eclipse
  - Creating a Java Project
  - Importing Existing Java Code
  - Creating a New Java Program
  - Java EE-Compliant Application Servers
  - Creating a Server Instance
  - Starting and Stopping the Server
- Java Naming and Directory Interface
  - What is JNDI?
  - Benefits of JNDI
  - Naming Services
    - Directory Services
  - Using JNDI
  - Context Operations
    - JNDI Utility Class
    - JNDI Example
  - Naming Exceptions
  - Running the JNDI Example
- Using JDBC Data Sources
  - A Simple JDBC Program
    - JDBC Driver Types
  - Using the Derby Database
    - JDBC Data Source
    - Data Source Example
  - Configuring a JDBC Data Source
    - Running the JDBC Examples
  - Executing a Query
    - Using the Database Explorer
- RMI and IIOP
  - Object Serialization
  - Remote Method Invocation
  - RMI Architecture
- The Remote Interface
  - CORBA
  - Enterprise JavaBeans
    - Enterprise JavaBeans Component Model
    - Types of Enterprise Beans
  - EJB Wrapper Interfaces
  - Deployment Descriptors
  - Context and Environment Objects
  - EJB Runtime Environment
    - The Remote Interface
    - The Home Interface
    - The Enterprise Bean Class
    - The Client Test Program
  - The ejb-jar.xml File
  - The Vendor-Specific Deployment Descriptor
  - Creating an Enterprise Application Project
  - Deploying the Enterprise Application
  - Introduction to EJB 3
    - Limitations of EJB 2
      - EJB 3 Feature Overview
      - Comparing EJB 2 and 3
      - The EJB 3 Business Interface
      - Review of Java Annotations
      - The Annotated EJB Class
      - Dependency Injection
      - Container Callback Methods
- Session Beans
  - Session Bean Lifecycle
  - Session Bean Interface
  - Session Bean Lifecycles
  - Stateless Session Bean Example
  - Accessing Environment Entries
  - Stateful Session Bean Example
  - EJB Exceptions - Examples
  - Testing the Session Beans
  - Creating a New Session Bean
- CMP Entity Beans
  - Entity Beans
  - Entity Bean Interface
  - Lifecycle of an Entity Bean
  - Container-Managed Persistence
  - Primary Key Class
  - Implementing CMP Entity Bean Methods
  - Container-Managed Persistence Example
  - Custom Finders
  - EJB Query Language
  - Mapping Container-Managed Fields
- Session Facade Pattern
  - J2EE Design Patterns
  - Session Facade Pattern
    - Local Interfaces
    - Example - ItemOrderer Bean
  - Deployment Settings for ItemOrderer Bean
  - Testing the Session Bean
- Container-Managed Relationships
  - Container-Managed Relationships
    - Local Interfaces
    - Local Home Interfaces
    - CMR Example - Entity Bean Classes
    - Transfer Object Pattern
  - CMR Example - Session Bean
  - Creating New CMP Entity Beans
  - Creating a Relationship
- Message-Driven Beans
  - Message-Driven Beans
  - Message-Driven Bean Lifecycle
  - Message-Driven Bean Example
  - Deploying Message-Driven Beans
  - Creating a New Message-Driven Bean
- EJB Timer Service
  - Overview of the Timer Service
  - Timer Service API
  - Creating Timers
  - Canceling and Saving Timers
- Appendix A: Web Resources
  - Java Technology
  - Application Servers
  - Derby Database
- Appendix B: Using EJBs in a Web Application
  - Using Web Components as EJB Clients
  - Servlet Code for the Survey Application
  - Session Bean for the Survey Application
  - Using Web Components as EJB Clients
- Appendix C: EJB Transactions
  - Transactions
  - Container-Managed Transactions
  - Transaction Attributes
  - System vs. Application Exceptions
  - Rolling Back a Container-Managed Transaction
- Appendix D: EJB Security
  - Java EE Security
  - Specifying Permissions for EJBs
- Appendix E: Java Message Service
  - Introduction
  - JMS and the Java EE Platform
  - Basic JMS Concepts
  - The JMS Programming Model
  - Point-to-Point Example
  - Configuring JMS
  - Publish/Subscribe Example
  - Reliable Message Delivery

http://www.trainingetc.com
Introduction to Python 3

**Course Description:** This course introduces the student to the Python language. Upon completion of this class, the student will be able to write non trivial Python programs dealing with a wide variety of subject matter domains. Topics include language components, the IDLE environment, control flow constructs, strings, I/O, collections, classes, modules, and regular expressions. The course is supplemented with many hands on labs using either Linux or Windows.

**Who Should Attend:** This course is designed for anyone who needs to learn how to write programs in Python.

**Prerequisites:** Students should have taken the Software Development for Non-Programmers course or have some experience with at least one programming language. Typically, students in this course will have already programmed in either C, C++, Java, Perl, Ruby, VB, or anything equivalent to these languages.

**Benefits of Attendance:** Upon completion of this course, students will be able to:

- Execute Python code in a variety of environments
- Use correct Python syntax in Python programs
- Use the correct Python control flow construct
- Write Python programs using various collection data types
- Write home grown Python functions
- Use many of the standard Python modules such as os, sys, math, and time
- Trap various errors via the Python Exception Handling model
- Use the IO model in Python to read and write disk files
- Create their own classes and use existing Python classes
- Understand and use the Object Oriented paradigm in Python programs
- Use the Python Regular Expression capabilities for data verification

**Course Outline:**

**An Introduction to Python**
- Introductory Remarks about Python
- Strengths and Weaknesses
- A Brief History of Python
- Python Versions
- Installing Python
- Environment Variables
- Executing Python from the Command Line
- IDLE
- Editing Python Files
- Getting Help
- Dynamic Types
- Python Reserved Words
- Naming Conventions

**Basic Python Syntax**
- Introduction
- Basic Syntax
- Comments
- String Values
- String Operations
- The `format` Method
- String Slices
- String Operators
- Numeric Data Types
- Conversions
- Simple Input and Output
- The `print` Function

**Language Components**
- Introduction
- Control Flow and Syntax
- Indenting
- The `if` Statement
- Relational Operators
- Logical Operators
- True or False
- Bit Wise Operators
- The `while` Loop
- `break` and `continue`
- The `for` Loop

**Collections**
- Introduction
- Lists
- Tuples
- Sets
- Dictionaries
- Sorting Dictionaries
- Copying Collections
- Summary

**Functions**
- Introduction
- Defining Your Own Functions
- Parameters
- Function Documentation
- Keyword and Optional Parameters
- Passing Collections to a Function
- Variable Number of Arguments
- Scope
- Functions - "First Class Citizens"
- Passing Functions to a Function
- Mapping Functions in a Dictionary
- Closures

**Modules**
- Modules
- Standard Modules - `sys`
- Standard Modules - `math`
- Standard Modules - `time`
- The `dir` Function

**Exceptions**
- Errors
- Run Time Errors
- The Exception Model
- Exception Hierarchy
- Handling Multiple Exceptions
- `raise`
- `assert`
- Writing Your Own Exception Classes

**Input and Output**
- Introduction
- Data Streams
- Creating Your Own Data Streams
- Access Modes
- Writing Data to a File
- Reading Data From a File
- Additional File Methods
- Using Pipes as Data Streams
- Handling IO Exceptions
- Working with Directories
- Metadata
- The `pickle` Module

**Classes in Python**
- Classes in Python
- Principles of Object Orientation
- Creating Classes
- Instance Methods
- File Organization

**Special Methods**
- Class Variables
- Inheritance
- Polymorphism
- Type Identification
- Custom Exception Classes
- Class Documentation - `pydoc`

**Regular Expressions**
- Introduction
- Simple Character Matches
- Special Characters
- Character Classes
- Quantifiers
- The Dot Character
- Greedy Matches
- Grouping
- Matching at Beginning or End
- Match Objects
- Substituting
- Compiling a String
- Compiling Regular Expressions
- Flags

---

[http://www.trainingetc.com](http://www.trainingetc.com)  17  Phone: 410-290-8383 / Fax: 410-290-9427
Advanced Python 3

Course Description: The course covers a handful of various Python advanced topics including high level data structures, network programming, writing GUI's in Python, and CGI programming.

Who Should Attend: This course is for students wanting to further their knowledge of Python.

Prerequisites: Students should have taken an introductory Python course or have six months of Python programming experience.

Benefits of Attendance: Upon completion of this course, students will be able to:
- Write Network Programs in Python
- Write CGI programs in Python
- Write GUI programs in Python
- Differentiate between the proper use of Python collection classes
- Use advanced data structures
- Become proficient in the use of bit variables
- Use Python list comprehensions
- Use Python generators
- Use the most popular Python modules
- Create and execute processes

Course Outline:

What You Should Already Know About Python
Introduction
Language Evolution
Python Reserved Words and Other Rules
Documentation
The string Module
Variables
Data Types
Boolean and Numeric Types
Strings
Lists
Sets
Sequences
Looping Through Sequences
Dictionaries
Bit Manipulation
Functions
Function Arguments
Modules
Standard Input and Output
File Input and Output
Some File Tests

Data Structures
List Comprehensions
Nested List Comprehensions
Dictionary Comprehensions
Dictionaries with Compound Values
Processing Lists in Parallel
Specialized Sorts
Time Functionality
Generators

Writing GUIs in Python
Introduction
Components and Events
An Example GUI
The Tk Widget
Button Widgets
Entry Widgets
Text Widgets
Checkbox Widgets
Radiobutton Widgets
Listbox Widgets
Frame Widgets
Menu Widgets
Toplevel Widgets
Dialogs

Python and CGI Scripts
What is CGI
HTML
HTML Forms
A Guestbook Application
What Can Go Wrong!
HTML Tables
The CGI Script

The os Module
The Environment
Creating a Process
Listing Files
Other Process Methods
File Information (Metadata)
Working With Directories

Network Programming
Networking Fundamentals
The Client/Server Model
The socket Module
A Client Program
A Server Program
An Evaluation Client and Server
A Threaded Server

Appendix A: What You Might Not Already Know
What is an iterable?
Creating Your Own Iterators
Generators
The Functions any and all
Thread Fundamentals
Synchronization
Signals
The Python Debugger
The with Statement
Data Compression

Appendix B: Python and Databases
Introduction
DBM Operations
Pickling
Pickling with Complex Objects
Shelves
Using sqlite3
Executing Queries
Table Descriptions
Writing Database Scripts
Python 3 and Advanced Python 3

Course Description:
This course combines the Introduction to Python 3 and Advanced Python 3 courses. The introductory course introduces the student to the Python language. Upon completion of this class, the student will be able to write non-trivial Python programs dealing with a wide variety of subject matter domains. Topics include language components, the IDLE environment, control flow constructs, strings, I/O, collections, classes, modules, and regular expressions. The course is supplemented with many hands on labs using either Linux or Windows. The advanced course covers a handful of various Python advanced topics including high level data structures, network programming, writing GUI's in Python, and CGI programming.

Who Should Attend:
This course is designed for anyone who needs to learn how to write programs in Python.

Prerequisites:
Students should have taken the Software Development for Non-Programmers course or have some experience with at least one programming language. Typically, students in this course will have already programmed in either C, C++, Java, Perl, Ruby, VB, or anything equivalent to these languages.

Benefits of Attendance:
Upon completion of this course, students will be able to:

- Execute Python code in a variety of environments
- Use correct Python syntax in Python programs
- Use the correct Python control flow construct
- Write Python programs using various collection data types
- Write home grown Python functions
- Use many of the standard Python modules such as os, sys, math, and time
- Trap various errors via the Python Exception Handling model
- Use the IO model in Python to read and write disk files
- Create their own classes and use existing Python classes
- Understand and use the Object Oriented paradigm in Python programs
- Use the Python Regular Expression capabilities for data verification
- Write Network Programs in Python
- Write CGI programs in Python
- Write GUI programs in Python
- Differentiate between the proper use of Python collection classes
- Use advanced Data Structures
- Become proficient in the use of bit variables

Course Outline:

An Introduction to Python
Introductory Remarks about Python
Strengths and Weaknesses
A Brief History of Python
Python Versions
Installing Python
Environment Variables
Executing Python from the Command Line
IDLE
Editing Python Files
Getting Help
Dynamic Types
Python Reserved Words
Naming Conventions
Language Evolution
Python Reserved Words and Other Rules
Documentation
The string Class
DataTypes
Boolean and Numeric Types
Strings
Lists and Tuples
Sets
Sequences
Looping Through Sequences
Dictionaries
Bit Variables
Modules
Reading Files
Some File Tests

Basic Python Syntax
Introduction
Basic Syntax
Comments
String Values
String Operations
The format Method
String Slices
String Operations
Numeric Data Types
Conversions
Simple Input and Output
The print Function

Language Components
Introduction
Control Flow and Syntax
Indenting
The if Statement
Relational Operators
Logical Operators
True or False

Bit Wise Operators
The while Loop
break and continue
The for Loop

Collections
Introduction
Lists
Tuples
Sets
Dictionaries
Sorting Dictionaries
Copying Collections
Summary

Functions
Introduction
Defining Your Own Functions
Parameters
Function Documentation
Keyword and Optional Parameters
Pasting Collections to a Function
Variable Number of Arguments
Scope
Functions
Passing Functions to a Function
Mapping Functions in a Dictionary
Lambdas
Closures

Modules
Modules
Standard Modules - sys
Standard Modules - math
Standard Modules - time
The dir Function

Exceptions
Errors
Run Time Errors
The Exception Model
Exception Hierarchy
Handling Multiple Exceptions
raise
assert
Writing Your Own Exception Classes

Input and Output
Introduction
Data Streams
Creating Your Own Data Streams
Access Modes
Writing Data to a File
Reading Data From a File

Additional File Methods
Using Pipes as Data Streams
Handling IO Exceptions
Working with Directories
Metadada
The print Function

Classes in Python
Classes in Python
Principles of Object Orientation
Creating Classes
Instance Methods
File Organization
Special Methods
Class Variables
Inheritance
Polymorphism
Type Identification
Custom Exception Classes
Class Documentation - pydoc

Regular Expressions
Introduction
Simple Character Matches
Special Characters
Character Classes
Quantifiers
The Dot Character
Greedy Matches
Grouping
Matching at Beginning or End
Match Objects
Substituting
Splitting a String
Compiling Regular Expressions
Flags

Data Structures
range
List Comprehensions
Nested List Comprehensions
Dictionary Comprehensions
Dictionaries with Compound Values
Processing Lists in Parallel
Functions
Default Parameters
Variable Arguments
A Dictionary of Dictionaries
Specialized Sorts
The in Statement
Time Functionality
Using Generators

Writing GUIs in Python
Introduction
Component and Events
An Example GUI
The root Component
Adding a Button
Entry Widgets
Text Widgets
dialogs
Radio buttons
Listboxes
Frames
Menus
Binding Events to Widgets

Python and CGI Scripts
Introduction
HTML
HTML Forms
A Graphical Application
What Can Go Wrong!
HTML Tables
The CGI Script
Rendoring of the Script

The os Module
The Environment
Launching Commands
Creating a Process
Directory Commands
Other Process Methods
File Information (Metadata)
Miscellaneous os Calls
Walking Through Directories

Network Programming
Introduction
A Daytime Server
Clients and Servers
The Client Program
The Server Program
A Server Program
An Evaluation Client and Server
A Threaded Server

Appendix A: What You Might Not Already Know
What is an iterator?
Creating Your Own Iterators
Generators
The Functions and all
Thread Fundamentals
Synchronization
Formatting Records
Perl Programming

Course Description: Perl is a scripting language which allows for rapid prototyping of projects formerly done with a programming language or a shell. It incorporates all the functionality of C (including a UNIX system interface), the Shells, grep, sed, and awk. The topics in the course will aid all computer users - from end user to programmer to administrator alike. Many in-class labs support the course material.

Who Should Attend: This course is for programmers, end users, system administrators, network administrators, CGI script writers, or anybody who is interested in automating tasks but doesn't want to learn all the details of a full blown programming language.

Prerequisites: Students should have some experience with either a programming language (preferably C), or any of the UNIX shells.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Manipulate files and directories
- Use arrays and array functions to solve a variety of problems
- Use the powerful regular expression capabilities of Perl
- Generate reports
- Use hashes to solve commonly encountered problems
- Take advantage of Perl's powerful system interface
- Write programs that solve many system administrator problems
- Use Perl to write CGI applications
- Use Modules from the standard Perl distribution
- Use Perl references

Course Outline:

Getting Started with Perl
What is Perl?
Where Can I Get Perl?
A Simple Perl Program
Simple I/O
Perl Variables
Control Flow - Decisions
Control Flow - Loops
Looping Loop Control Flow
Statement Modifiers
What Is True And What Is False?
The Special Variable $_

Perl Operators
Introduction
Table Of Perl Operators
Arithmetic Operators
String Operators
Relational Operators
Logical Operators
Sillywise Operators
Assignment Operators
The Conditional Operator
Range Operator
String Functions
The eval Function

I/O
Introduction
String Literals
The print Function
Here Documents
The printf Function
Filehandles
Opening Disk Files
File Open Errors
The die and warn Functions
File Operators

Arrays
Basic Concepts
Assigning Values To An Array
Accessing Array Elements
Array Functions
push and pop
sort, reverse, and chop
split and join
grep
splice
Command Line Arguments

Associative Arrays
Basic Concepts
Associative Array Functions
Updating Associative Arrays
Accessing Environment Variables

Subroutines
Calling Subroutines
Passing Arguments to Subroutines
Returning Values from Subroutines
The require Function
Packages and Modules
The @INC Array
Predefined Subroutines
Comparison Subroutines for Sorting

Pattern Matching and Regular Expressions
Introduction
Regular Expression Syntax
The Match Operator
Regular Expression Meta-Characters
Anchors
Single Character Matches
Some Special Issues
Character Classes
Multiple Character Matches
Alternation
The Substitution Operator
The Translation Operator

Accessing System Resources
Introduction
File and Directory System Calls
The stat Function
The utime Function
The fork Function
The exec and wait Functions
Handling Signals
The system Function
Command Substitution
Opening Pipe Files

Generating Reports with Perl
Formats
Formatting Examples
Multi-Line Values
Multi-Line Text Blocks
Sending a Report to a File
The select Function
The Special Variable $-
Top-of-Page Formats
Bottom-of-Page Formats
A Sample Report

Perl and CGI
What is CGI?
Advanced Perl Programming

Course Description: The course begins with a thorough treatment of packages, modules, and libraries. Next, Perl references are studied. This gives students the necessary background to write object-oriented Perl. Various applications and areas that use object orientation are studied next. These modules include the Tk.pm module for building Graphical User Interfaces, the DBI.pm module, which provides a portable way of querying databases, the CGI.pm module for writing CGI programs, and the Socket.pm module used in client server networking applications. Finally a treatment of XML and Perl is undertaken.

Who Should Attend: Programmers, end users, system administrators, network administrators, and CGI script writers should attend this course.

Prerequisites: Participants should be well-versed in the fundamentals of Perl.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Download, install, and use modules from the Comprehensive Perl Archive Network (CPAN).
- Use the modules in the Standard Perl Distribution.
- Write POD (Plain Old Documentation) sections of Perl modules.
- Use Perl references to solve many programming problems including those problems involving arbitrarily complex data structures.
- Distinguish among packages, modules, libraries, and classes and use each one effectively.
- Write client/server applications using the Socket.pm module.
- Write Graphical User Interfaces (GUIs) using the Tk.pm module.
- Write Perl CGI (Common Gateway Interface) scripts.
- Write Perl applications that make queries to real databases through the use of the DBI.pm module.
- Write Perl applications that produce and process XML documents.

Course Outline:

What You Should Already Know
- Introduction
- A Quick Review of Perl
- Perl Libraries
- The Standard Perl Library
- Packages
- Modules
- Using .pm Modules
- Exporter.pm
- Standard Perl Modules
- Comprehensive Perl Archive Network (CPAN)
- Roman.pm
- Miscellaneous Perl Topics - wantarray

Associative Arrays
- Introduction
- Associative Arrays as Dual Arrays
- A Hashing Algorithm
- Collisions
- Associative Arrays
- Sorting by Keys or Values
- Finding Unique Tokens in a File
- Reverse Lookups
- Selecting the Top n Elements

References
- Introduction
- Summary of References
- Array References
- Anonymous Arrays
- Anonymous Hashes
- Prototypes
- Higher Dimensional Arrays
- Complex Hashes
- References and Subroutines
- Anonymous Subroutines
- Lists of References

Object-Oriented Programming
- Introduction
- Object-Oriented Vocabulary
  - The class
  - Definition
  - Defining and Using Objects
  - Information Hiding
  - Instance Methods
  - Destructors
- Class Methods
- Inheritance
- Polymorphism
- Documenting Perl Code
- IO.pm

The TK.PM Module
- Introduction
- Event Driven Programming
- Geometry Management
- pack()
- grid()
- grid() Options
- place(): Absolute Coordinates
- place(): Relative Coordinates
- The Label Widget
- The Button Widget
- The Checkbutton Widget
- The Radiobutton Widget
- The Dialog Widget
- Text Input Widgets
- The Listbox Widget
- Menus
- Frames
- Toplevel Widgets
- Bind

Client-Server Applications and CGI
- Introduction
- Internet Terminology
- Data Delivery
- Writing a Simple Client
- Writing a Simple Server
- Writing an Iterative Server
- The Common Gateway Interface
- HTML Forms
- The CGI Environment
- Administering the Server
- The HTTP Protocol
- Header Information
- The CGI Script
- Extracting Form Data
- The CGI Response
- CGI Output
- Database Access
- What Can Go Wrong?
- Images
- Extra Path Information
- CGI.pm
- Using CGI.pm?
- Simple Form Elements
- Parameters
- HTML Tags
- Form Processing
- checkbox_group and radio_group
- Text Areas
- Popup Menus and Scrolling Lists
- Debugging

Accessing Real Databases in Perl
- Introduction
- Architecture
- Review of SQL
- Accessing Databases from Perl
- Executing a Query in Perl
- Accessing Database Metadata
- Interactive Requests
- Adding a Graphical Front-End
- Accessing a Real Database via a Web Form

XML Fundamentals
- Introduction
- What is a Markup Language?
- SGML vs. HTML
- Sample HTML Document
- XML
- Creating Semantic Tags
- XML Syntax
- Elements
- Attributes
- Comments
- Unicode and Character Sets
- Character Referencess
- Entity References
- Character Data Sections (CDATA)
- Processing Instructions
- Parsing XML

Processing XML With Perl
- Creating an XML Document With Perl
- Creating an XML Document
- Using an XML Parser

Appendix A: An HTML (P)review
- An HTML (P)review of  Basics
- Appendix B: An HTML (P)review
- XML::Parser
- XML::Simple
- XML::Parser
- XML::Parser

Course Length: 5 Days
Course Description: This course covers the fundamental components of the Ruby Programming Language. Emphasis is placed on the object oriented aspects of Ruby. Topics include arrays, hashes, regular expressions, io, exceptions, modules, and applications areas.

Who Should Attend: This course is intended primarily for those who have programmed in other programming languages such as, but not limited to, C, C++, Java, or Perl.

Prerequisites: Students should have taken the Software Development for Non-Programmers course or have at least six months of programming experience in at least one programming language.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Distinguish and use various Ruby datatypes
- Master the use of arrays and hashes
- Build home grown classes
- Use the extensive pre bundled classes
- Use the I/O facilities of Ruby to read and write binary and text files
- Master the use of Iterators to loop through various data structures
- Use Exceptions in handling various run time errors
- Create Ruby modules
- Use the wide variety of Ruby Modules that come with the Ruby distribution

Course Outline:

An Introduction to Ruby
- What is Ruby?
- Installing Ruby
- Executing Ruby Code
- Getting Help
- Dynamic Types
- Ruby Reserved Words
- Naming Conventions
- Comments

Standard Ruby Data Types
- Numbers
- Strings
- Simple Input and Output
- Converting String Input
- Regular Expressions
- Time Methods

Language Components
- The if Statement
- Logical Operators
- The case Construct
- Loops
- Iterators
- Numeric Iterators
- String Iterators
- Methods
- Odds and Ends

Collections
- Arrays
- Array Operator Methods
- Array Equality Operator
- Arrays as Stacks and Queues
- Higher Dimensional Arrays
- Other Useful Arrays Methods
- Command Line Arguments
- Hashes
- Common Hash Methods
- Sorting Hashes
- Iterators with Arrays and Hashes
- Arrays and Methods
- Hashes and Methods
- Named Parameters
- Symbols
- Procs
- Closures

Classes
- Objects
- Brief History of OOP
- OOP Vocabulary
- Creating a New Class
- Using Objects
- Defining Operator Methods
- Inheritance
- Ancestors

Odds and Ends
- Ruby Conventions

http://www.trainingetc.com

Phone: 410-290-8383 / Fax: 410-290-9427
Course Description: Groovy is a dynamic scripting and programming language for the Java platform. It combines the dynamic features of modern scripting languages such as Ruby and Python with familiar Java syntax. To quote one of the Groovy developers: “Groovy is what Java would have been if it had been created in the 21st century.”

This course introduces the Java developer to the Groovy language. The course focuses on understanding the internals of how Groovy works in addition to understanding the Groovy language syntax. After taking this course developers will understand the Groovy syntax and be able to leverage existing Java classes within Groovy.

This class combines lecture with a unifying, hands-on experience, and open discussion that will help the developer quickly understand the benefits of Groovy and how to use the language.

Who Should Attend: This course is intended for experienced Java developers.

Prerequisites: Participants should have taken the Software Development for Non-Programmers course or have a solid understanding of Java programming.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Write applications using Groovy.
- Understand how Groovy operates within the Java Virtual Machine.
- Incorporate existing Java classes and libraries within Groovy applications.
- Add new methods and member variables to existing Java or Groovy classes dynamically.
- Understand the role of Closures within Groovy.
- Take advantage of Groovy’s simplified object configuration syntax.
- Override operators for Groovy or Java classes.
- Understand the concept of Metaprogramming and how to leverage it to simplify application development.
- Explore Groovy’s Regular Expression syntax for easily managing String processing.
- Calling methods that do not exist - ExpandoMetaClass, Categories

Regular Expressions
- Regular Expression syntax
- The =~ operator
- The ==~ operator
- Common methods that use Regular Expressions

Builders and Slurpers
- What are Builders and Slurpers?
- NodeBuilder
- MarkupBuilder
- AntBuilder
- Using the ConfigSlurper

Course Outline:

Language Overview
- What is Groovy?
- What Groovy can do
- Installing
- Running Groovy scripts

Basic Syntax and Scalar Variables
- Syntax rules
- Numbers - Integers, Floats, and BigDecimal
- Strings - Double quoted, Single quoted, Here documents, Slash quoted, GStrings
- Operators

Collections
- Lists
  - Coding a closure - The it parameter, Passing multiple parameters, Naming parameters (the - > operator)
  - Maps
  - Ranges

Flow Control
- If statements - The truth in Groovy
- Switch statements
- While loops
- For loops
- Exceptions

Classes
- Defining classes - File-to-class relationships
  - Member variables - Automatic getter/setter generation, Default visibility, Safe dereferencing with ? operator
- Methods - Optional parameters and default parameters
- Operator overloading
- Automatic constructor generation - Initializing property values in the constructor
- The Closure Groovy class
- Coding a method that expects a closure - Calling into the closure, Passing parameters

Advanced Classes and Closures
- Closures - Using methods as closures, Polymorphic closures
- Operator overloading
- Metaprogramming - Discovering a class, Discovering fields, Discovering methods, Method resolution
- Pointers - Method, Field
Course Description:
Groovy is a dynamic scripting and programming language for the Java platform. It combines the dynamic features of modern scripting languages such as Ruby and Python with familiar Java syntax. To quote one of the Groovy developers: “Groovy is what Java would have been if it had been created in the 21st century.” This course introduces the Java developer to the Groovy language and to Grails (an Open Source, full stack, web application framework that takes advantage of the Groovy programming language and convention over configuration to provide a productive and stream-lined development experience). The course focuses on understanding the internals of how Groovy works in addition to understanding the Groovy language syntax.

After taking this course developers will understand the Groovy syntax and be able to leverage existing Java classes within Groovy. Students will also be able to Install Grails and understand the main components of a Grails application. This class combines lecture with a unifying, hands-on experience, and open discussion that will help the developer quickly understand the benefits of Groovy and Grails.

Who Should Attend:
This course is intended for experienced Java developers.

Prerequisites:
Participants should have taken the Software Development for Non-Programmers course or have a solid understanding of Java programming.

Benefits of Attendance:
Upon completion of this course, students will be able to:

- Write applications using Groovy.
- Understand how Groovy operates within the Java Virtual Machine.
- Incorporate existing Java classes and libraries within Groovy applications.
- Add new methods and member variables to existing Java or Groovy classes dynamically.
- Understand the role of Closures within Groovy.
- Take advantage of Groovy’s simplified object configuration syntax.
- Override operators for Groovy or Java classes.
- Understand the concept of Metaprogramming and how to leverage it to simplify application development.
- Explore Groovy’s Regular Expression syntax for easily managing String processing.
- Install Grails and understand the main components of a Grails application.
- Create a simple Grails application.
- Perform domain queries.
- Return data to the view.
- Call a controller method from the view

Course Outline:

Language Overview
What is Groovy?
What Groovy can do
Installing Groovy scripts

Basic Syntax and Scalar Variables
Syntax rules
Numbers - Integers, Floats, and BigDecimal
Strings - Double quoted, Single quoted, Here documents, Slash quoted, GStrings
Operators

Collections
Lists
Coding a closure - The it parameter, Passing multiple parameters, Naming parameters (the -> operator)
Maps
Ranges

Flow Control
If statements - The truth in Groovy
Switch statements
While loops
For loops
Exceptions

Classes
Defining classes - File-to-class relationships
Member variables - Automatic getter/setter
generation, Default visibility, Safe
dereferencing with ? operator
Methods - Optional parameters and default parameters
Operator overloading
Automatic constructor generation -

Initializing property values in the constructor
The Closure Groovy class
Coding a method that expects a closure - Calling into the closure, Passing parameters

Advanced Classes and Closures
Closures - Using methods as closures, Polymorphic closures
Operator overloading
Metaprogramming - Discovering a class, Discovering fields, Discovering methods, Method resolution
Pointers - Method, Field Calling methods that do not exist - ExpandoMetaClass, Categories

Regular Expressions
Regular Expression syntax
The == operator
Common methods that use Regular Expressions

Builders and Slurpers
What are Builders and Slurpers?
NodeBuilder
MarkupBuilder
AntBuilder
Using the ConfigSlurper

Grails Installation and Overview
Installing Grails - GRAILS_HOME
variable, Updating PATH
Technologies included with Grails - Spring, Hibernate, Site Mesh, AJAX libraries, Jetty/Tomcat
Main components of a Grails application - Domain, Controller, View, Configurations

Creating a simple Grails application
The create-app command
Configuring the application
Create a Domain
Getting started with the Grails scaffolding
Generate views and a controller with create-all
Create a basic Controller
Running the application
Unit Tests

The Domain
Modeling relationships – belongsTo, mapping, and hasMany - One-to-one, One-to-many, Many-to-Many
Validators - constraints
GORM - Using the dynamic finder methods, Criteria objects, HQL

Domain Queries
Using the dynamic finder methods
Criteria objects
HQL

Controllers
Navigation
Returning data to the view
Calling a controller method from the view
Scopes - Request, Flash, Session, servletContext
Working with redirects
Creating and using services

Filters
Creating simple filters
Assigning a filter to actions on a controller
Assigning a filter to a uri
Defining when a filter executes - Before, After, afterView

Plug-ins and simple Web 2.0 application
What is a plug-in
Installing a plug-in - The Zk plug-in, What is provided by a plug-in
Integrating the Zk framework - Composers, Facades, Zul pages

Binding properties to domain objects
Command objects

GSP basics
How GSPs work
Common GSP tag libraries
Custom error messages
Rendering and processing forms
Writing custom tag libraries
Working with layouts