Java Programming Language (SL-275-SE6)

Carga horaria: 40Hs.

Descripción:

El curso del lenguaje de programación de Java provee a los estudiantes la información sobre la sintaxis del lenguaje de programación de Java; programación orientada a objeto con el lenguaje de programación de Java; crear los interfaces gráficos (GUIs), las excepciones, la entrada-salida del archivo (I/O), y threads; y establecimiento de una red. Los programadores con conceptos orientados a objetos pueden aprender cómo desarrollar usos de la tecnología de Java. El curso ofrece la plataforma de Java, edición estándar 6 (plataforma del SE de Java 6), y utiliza el kit 6 (producto del desarrollo del SE de Java del JDK 6). Los estudiantes realizan los ejercicios del laboratorio del curso usando el ambiente integrado NetBeans del desarrollo (IDE) 5.5.

Publico dirigido:

A estudiantes de programación que están interesados en el lenguaje de programación de Java y los estudiantes que se están preparando para el rendir la certificación de Sun.

Contenido

Getting Started

- Examine Java Technology
- Analyze a simple Java Technology Application
- Execute a Java technology application

Object-Oriented Programming

- Define modeling concepts: abstraction, encapsulation, and packages
- Discuss Java technology application code reuse
- Define class, member, attribute, method, constructor, and package
- Use the access modifiers private and public as appropriate for the guidelines of encapsulation
- Invoke a method on a particular object
- Use the Java technology API online documentation

Identifiers, Keywords, and Types

- Use comments in a source program
- Distinguish between valid and invalid identifiers
- Recognize Java technology keywords
- List the eight primitive types
- Define literal values for numeric and textual types
- Define the terms primitive variable and reference variable
- Declare variables of class type
- Construct an object using new
- Describe default initialization
- Describe the significance of a reference variable
- State the consequence of assigning variables of class type

Expressions and Flow Control

- Distinguish between instance and local variables
- Describe how to initialize instance variables
- Recognize, describe, and use Java software operators
- Distinguish between legal and illegal assignments of primitive types
- Identify boolean expressions and their requirements in control constructs
- Recognize assignment compatibility and required casts in fundamental types
Use if, switch, for, while, and do constructions and the labeled forms of break and continue as flow control structures in a program

**Arrays**
- Declare and create arrays of primitive, class, or array types
- Explain why elements of an array are initialized
- Explain how to initialize the elements of an array
- Determine the number of elements in an array
- Create a multidimensional array
- Write code to copy array values from one array to another

**Class Design**
- Define inheritance, polymorphism, overloading, overriding, and virtual method invocation
- Use the access modifiers protected and the default (package-friendly)
- Describe the concepts of constructor and method overloading
- Describe the complete object construction and initialization operation

**Advanced Class Feature**
- Create static variables, methods, and initializers
- Create final classes, methods, and variables
- Create and use enumerated types
- Use the static import statement
- Create abstract classes and methods
- Create and use an interface

**Exceptions and Assertions**
- Define exceptions
- Use try, catch, and finally statements
- Describe exception categories
- Identify common exceptions
- Develop programs to handle your own exceptions
- Use assertions
- Distinguish appropriate and inappropriate uses of assertions
- Enable assertions at runtime

**Collections and Generics Framework**
- Describe the general purpose implementations of the core interfaces in the Collections framework
- Examine the Map interface
- Examine the legacy collection classes
- Create natural and custom ordering by implementing the Comparable and Comparator interfaces
- Use generic collections
- Use type parameters in generic classes
- Refactor existing non-generic code
- Write a program to iterate over a collection
- Examine the enhanced for loop

**I/O Fundamentals**
- Write a program that uses command-line arguments and system properties
- Examine the Properties class
- Construct node and processing streams, and use them appropriately
- Serialize and deserialize objects
- Distinguish readers and writers from streams, and select appropriately between them

**Console I/O and File I/O**
- Read data from the console
- Write data to the console
- Describe files and file I/O

Building Java GUIs Using the Swing API

- Describe the JFC Swing technology
- Define Swing
- Identify the Swing packages
- Describe the GUI building blocks: containers, components, and layout managers
- Examine top-level, general-purpose, and special-purpose properties of container
- Examine components
- Examine layout managers
- Describe the Swing single-threaded model
- Build a GUI using Swing components

Handling GUI-Generated Events

- Define events and event handling
- Examine the Java SE event model
- Describe GUI behavior
- Determine the user action that originated an event
- Develop event listeners
- Describe concurrency in Swing-based GUIs and describe the features of the SwingWorker class

GUI-Based Applications

- Describe how to construct a menu bar, menu, and menu items in a Java GUI
- Understand how to change the color and font of a component

Threads

- Define a thread
- Create separate threads in a Java technology program, controlling the code and data that are used by that thread
- Control the execution of a thread and write platform-independent code with threads
- Describe the difficulties that might arise when multiple threads share data
- Use wait and notify to communicate between threads
- Use synchronized to protect data from corruption

Networking

- Develop code to set up the network connection
- Understand TCP/IP
- Use ServerSocket and Socket classes to implement TCP/IP clients and servers