



Java™ Programming Language SL-275

The *Java™ Programming Language* course provides students with information about the syntax of the Java programming language; object-oriented programming with the Java programming language; and creating graphical user interfaces (GUIs), exceptions, file input/output (I/O), and threads and networking. Programmers familiar with object-oriented concepts can learn how to develop Java technology applications. The course uses the Java™ 2 Software Development Kit (Java™ 2 SDK).

This course includes the Sun™ Open Net Environment (Sun™ ONE) Starter Kit, which is a CD-ROM package containing the Java development and runtime environments for the Java™ 2 Platform, Standard Edition (J2SE™) and Java™ 2 Platform, Enterprise Edition (J2EE™) platforms, as well as the associated application programming interfaces (APIs) and documentation. It also contains reference books, sample code, and technical articles.

Who Can Benefit

Students who can benefit from this course are programmers who are interested in adding the Java programming language to their list of skills and students who are preparing for the Sun Certified Programmer for Java™ examination.

Prerequisites

To succeed fully in this course, students should be able to:

- Understand object-oriented principles
- Create or compile simple programs in a language, such as C or C++, or have completed the SL-110: *Java™ Programming for*

Non-Programmers course and have created and compiled simple Java programs

- Create and edit text files using a text editor
- Use an Internet browser, such as Netscape Navigator™

Skills Gained

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

- Create Java technology applications that leverage the object-oriented features of the Java language, such as encapsulation, inheritance, and polymorphism
- Execute and run a Java technology application
- Use Java technology data types and expressions
- Use Java technology flow control constructs
- Use arrays and other data collections
- Implement error-handling techniques using exception handling
- Create event-driven GUI using Java technology GUI components: panels, buttons, labels, text fields, and text areas
- Implement I/O functionality to read from and write to data and text files.
- Create multithreaded programs

50%
Lab
Lab

5
Days
Duration

- Create a simple Transmission Control Protocol/Internet Protocol (TCP/IP) client that communicate through sockets

Related Courses

Before:

- SL-110: *Fundamentals of the Java™ Programming Language*
- SL-210: *Migrating to Object-Oriented Programming With Java™ Technology*

After:

- OO-226: *Object-Oriented Application Analysis and Design for Java™ Technology (UML)*
- SL-285: *Java™ Programming Language Workshop*
- SL-291: *JavaBeans™ Component Development*
- SL-301: *Distributed Programming With Java™ Technology*
- SL-320: *GUI Construction With Java™ Foundation Classes*

Course Outline

Module 1 – *Getting Started*

- Describe the key features of Java technology
- Write, compile, and run a simple Java technology application
- Describe the Java™ virtual machine's (JVM™ machine's) function
- Define garbage collection
- List the three tasks performed by the Java platform that handle code security

Module 2 – *Object-Oriented Programming*

- Define modeling concepts: abstraction, encapsulation, and packages
- Discuss why you can reuse Java technology application code

- 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
- In a Java technology program, identify the following: the package statement, the import statements, classes, methods, and attributes, and constructors
- Use the Java technology API online documentation

Module 3 – *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

Module 4 – *Expressions and Flow Control*

- Distinguish between instance and local variables
- Describe how to initialize instance variables
- Identify and correct a possible reference before assignment compiler error
- 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

Module 5 – 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 type to another

Module 6 – Class Design

- Define *inheritance*, *polymorphism*, *overloading*, *overriding*, and *virtual method invocation*
- Use the access modifiers protected and “package-friendly”
- Describe the concepts of constructor and method overloading
- Describe the complete object construction and initialization operation
- In a Java program, identify the following: overloaded methods and constructors, the use of `this` to call overloaded constructors, overridden methods, invocation of super class methods, parent class constructors, and invocation of parent class constructors

Module 7 – Advanced Class Features

- Describe static variables, methods, and initializers
- Describe final classes, methods, and variables
- Explain how and when to use abstract classes and methods
- Explain how and when to use nested classes
- Distinguish between static and non-static nested classes
- Explain how and when to use an interface
- In a Java software program, identify: static methods and attributes, final methods and attributes, nested classes, interface and abstract classes, and abstract methods

Module 8 – 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
- Disable assertions at runtime

Module 9 – Text-Based Applications

- Write a program that uses command-line arguments and system properties
- Write a program that reads from standard input
- Write a program that can create, read, and write files
- Describe the basic hierarchy of collections in Java 2 SDK
- Write a program that uses sets and lists
- Write a program to iterate over a collection

- Describe the collection classes that existed before Java 2 SDK
- Identify deprecated classes and explain how to migrate from Java™ Development Kit (JDK™) 1.0 to JDK 1.1 to Java™ 2 JDK

Module 10 – *Building Java GUIs*

- Describe the Abstract Windowing Toolkit (AWT) package and its components
- Define the terms *containers*, *components*, and *layout managers*, and describe how they work together to build a GUI
- Use layout managers
- Use the `FlowLayout`, `BorderLayout`, and `GridLayout` managers to achieve a desired dynamic layout
- Add components to a container
- Use the `Frame` and `Panel` containers appropriately
- Describe how complex layouts with nested containers work
- In a Java technology program, identify the following: containers, the associated layout managers, and the layout hierarchy of all components

Module 11 – *GUI Event Handling*

- Define events and event handling
- Write code to handle events that occur in a GUI
- Describe the concept of adapter classes, including how and when to use them
- Determine the user action that originated the event from the event object details
- Identify the appropriate interface for a variety of event types
- Create the appropriate event handler methods for a variety of event types
- Understand the use of inner classes and anonymous classes in event handling

Module 12 – *GUI-Based Applications*

- Identify the key AWT components and the events that they trigger
- 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

Module 13 – *Threads*

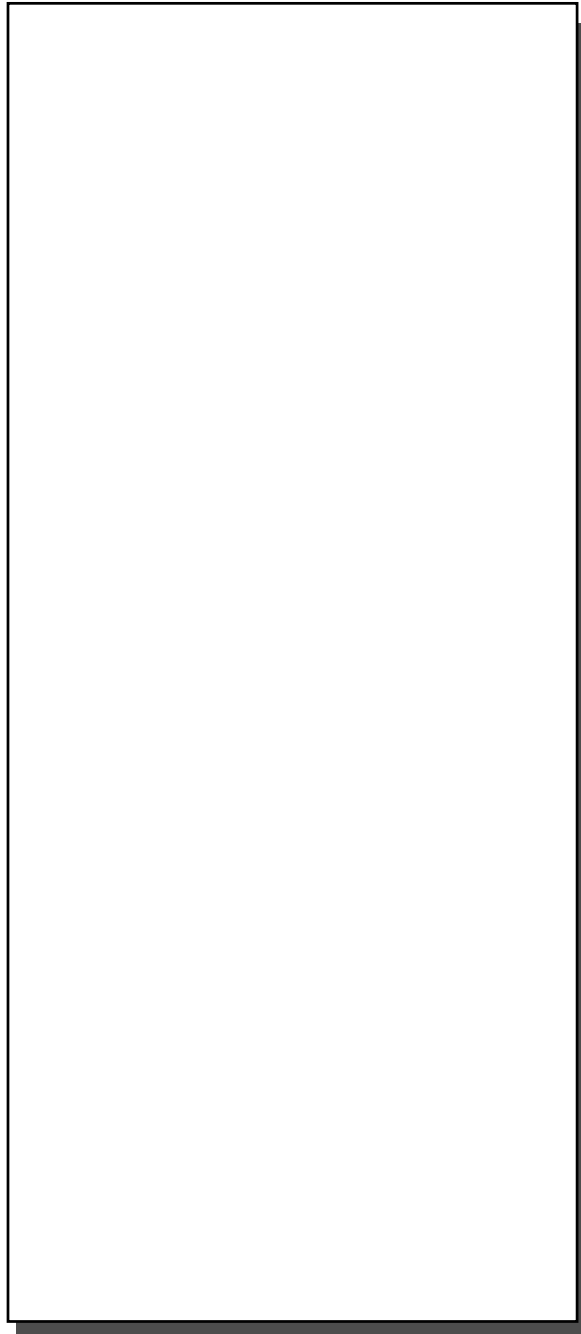
- Define a thread
- Create separate threads in a Java technology program, controlling the code and data that are used by that thread
- Control the execution of a thread and write platform-independent code with threads
- Describe the difficulties that might arise when multiple threads share data
- Use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

Module 14 – *Advanced I/O Streams*

- Describe the main features of the `java.io` package
- Construct node and processing streams, and use them appropriately
- Distinguish readers and writers from streams, and select appropriately between them

Module 15 – *Networking*

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



Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

Telephone: 800 422-8020

Specifications are subject to change without notice.

Copyright 2002 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Java, JavaBeans, JDK, JVM, J2EE, J2SE, and Sun Certified Programmer for Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Netscape Navigator is a trademark or registered trademark of Netscape Communications Corporation in the United States and other countries.