About This Course

Course Goals

Upon completion of this course, you 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 a Java technology application from the command line
- 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 an event-driven graphical user interface (GUI) by using Java technology GUI components: panels, buttons, labels, text fields, and text areas
- Implement input/output (I/O) functionality to read from and write to data and text files
- Create multithreaded programs
- Create a simple Transmission Control Protocol/Internet Protocol (TCP/IP) client that communicates through sockets

The main goal of the *Java™ Programming Language* course is to provide you with the knowledge and skills necessary for object-oriented programming of advanced Java applications. In this course, you learn Java programming language syntax and object-oriented concepts, as well as more sophisticated features of the Java runtime environment, such as support for GUIs, multithreading, and networking. This course covers prerequisite knowledge to help prepare you for the Sun Certified Programmer for the Java™ Platform (SCJP) examination. For information about the exam, review the Web site:

`http://www.sun.com/training/certification/java/`

However, SL-275 is not sufficient to immediately pass the exam. You should spend several months practicing these techniques by building real programs. You should also review the exam objectives and study areas that were not discussed in this course. The SCJP exam objectives can be found at the web site listed previously.

Course Overview

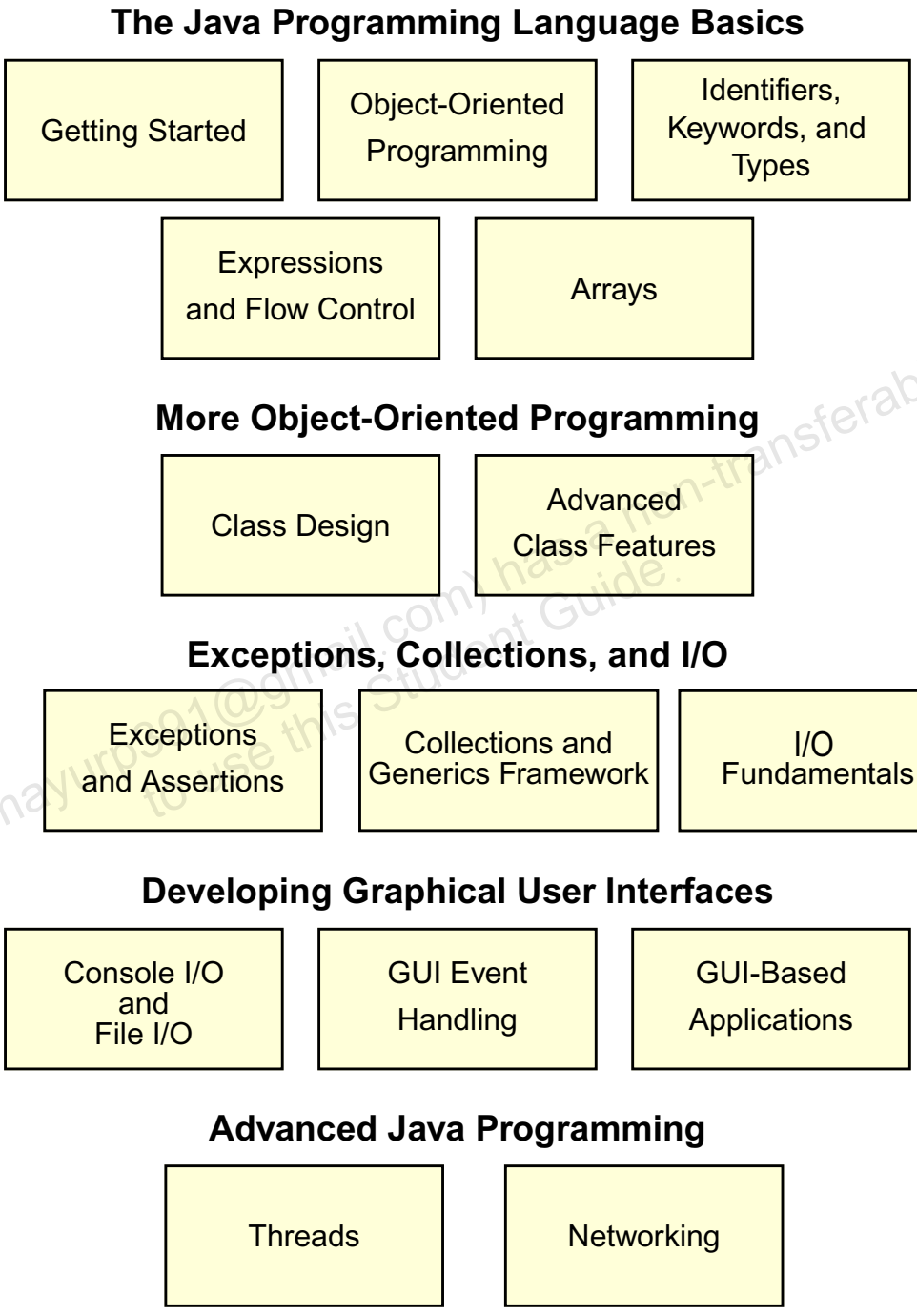
This course first describes the Java runtime environment and the syntax of the Java programming language. The course then covers object-oriented concepts as they apply to the Java programming language. As the course progresses, advanced features of the Java platform are discussed.

The audience for this course includes people who are familiar with implementing elementary programming concepts using the Java programming language or other languages. This is the follow-up course to SL-110: *Fundamentals of the Java™ Programming Language.*

While the Java programming language is operating-system-independent, the GUI that it produces can be dependent on the operating system on which the code is executed. In this course, code examples are run in the Solaris™ Operating System (Solaris OS) and in the Microsoft Windows operating environment; therefore, the graphics in this guide have both a Motif and a Windows GUI. The content of this course is applicable to all Java operating system ports.

Course Map

The following course map enables you to see what you have accomplished and where you are going in reference to the course goal.



Module-by-Module Overview

- Module 1 – “Getting Started”

  This module provides a general overview of the Java programming language and its main features and introduces a simple Java application.

- Module 2 – “Object-Oriented Programming”

  This module introduces basic object-oriented programming concepts and describes their implementation using the Java programming language.

- Module 3 – “Identifiers, Keywords, and Types”

  The Java programming language contains many programming constructs similar to the C language. This module provides a general overview of the constructs available and the general syntax required for each construct. It also introduces the basic object-oriented approach to data association using aggregate data types.

- Module 4 – “Expressions and Flow Control”

  This module looks at expressions, including operators, and the syntax of Java program control.

- Module 5 – “Arrays”

  This module describes how Java arrays are declared, created, initialized, and copied.

- Module 6 – “Class Design”

  This module builds on the information about object-oriented concepts provided in Module 2 and takes the information to the next level, including a discussion on subclassing, overloading, and overriding.

- Module 7 – “Advanced Class Features”

  This module completes the Java object-oriented programming model by discussing the concepts of static members, final variables, abstract classes, and interfaces.

- Module 8 – “Exceptions and Assertions”

  Exceptions provide you with a mechanism for trapping errors at runtime. This module explores both predefined and user-defined exceptions.

- Module 9 – "Collections and Generics Framework"

  This module examines the collections framework and the use of generics in the Java programming language.

- Module 10 – "I/O Fundamentals"

  This module describes the classes available for reading and writing both data and text. It also contains a discussion on object serialization.

- Module 11 – "Console I/O and File I/O"

  This module introduces topics that are useful in implementing large, text-based applications, such as console and file I/O.

- Module 12 – "Building Java GUIs"

  All graphical user interfaces in the Java programming language are built on the concept of frames and panels. This module introduces layout management and containers using Swing.

- Module 13 – "GUI Event Handling"

  Creating a layout of GUI components in a frame is not enough. You must write code to handle the events that occur, such as clicking a button or typing a character. This module demonstrates how to write GUI event handlers.

- Module 14 – "GUI-Based Applications"

  This module discusses a variety of GUI elements.

- Module 15 – "Threads"

  Threads are a complex topic; this module explains threading as it relates to the Java programming language and introduces a straightforward example of thread communication and synchronization.

- Module 16 – "Networking"

  This module introduces the Java network programming package and demonstrates a TCP/IP client-server model.

Course Objectives

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

- Describe key language features
- Compile and run a Java technology application
- Use the online hypertext Java technology documentation
- Describe language syntactic elements and constructs
- Describe the object-oriented paradigm
- Use the object-oriented features of the Java programming language
- Use exceptions
- Use the Collections application programming interface (API)
- Read and write to files
- Develop a GUI
- Describe the Java technology Abstract Window Toolkit (AWT)
- Develop a program to take input from a GUI
- Describe event handling
- Use the `java.io` package
- Describe the basics of multithreading
- Develop multithreaded Java technology applications
- Develop Java client and server programs by using TCP/IP

Topics Not Covered

This course does not cover the following topics. Many of these topics are covered in other courses offered by Sun Educational Services:

- Object-oriented analysis and design – Covered in OO-226: *Object-Oriented Application Analysis and Design Using UML*
- General programming concepts – Covered in SL-110: *Fundamentals of the Java™ Programming Language*

Refer to the Sun Educational Services catalog for specific information and registration.

How Prepared Are You?

Before attending this course, you should have completed SL-110: *Fundamentals of the Java Programming Language,* or have:

- Created and compiled programs with C or C++
- Created and edited text files using a text editor
- Used a World Wide Web (WWW) browser, such as Netscape Navigator™

Introductions

Now that you have been introduced to the course, introduce yourself to the other students and the instructor, addressing the following items:

- Name
- Company affiliation
- Title, function, and job responsibility
- Experience related to topics presented in this course
- Reasons for enrolling in this course
- Expectations for this course

How to Use Course Materials

To enable you to succeed in this course, these course materials contain a learning module that is composed of the following components:

- Goals – You should be able to accomplish the goals after finishing this course and meeting all of its objectives.
- Objectives – You should be able to accomplish the objectives after completing a portion of instructional content. Objectives support goals and can support other higher-level objectives.
- Lecture – The instructor presents information specific to the objective of the module. This information helps you learn the knowledge and skills necessary to succeed with the activities.
- Activities – The activities take various forms, such as an exercise, self-check, discussion, and demonstration. Activities help you facilitate the mastery of an objective.
- Visual aids – The instructor might use several visual aids to convey a concept, such as a process, in a visual form. Visual aids commonly contain graphics, animation, and video.

Typographical Conventions and Symbols

The following conventions are used in this course to represent various training elements and alternative learning resources.

Icons

Additional resources – Indicates other references that provide additional information on the topics described in the module.

Discussion – Indicates a small-group or class discussion on the current topic is recommended at this time.

Note – Indicates additional information that can help students but is not crucial to their understanding of the concept being described. Students should be able to understand the concept or complete the task without this information. Examples of notational information include keyword shortcuts and minor system adjustments.

Caution – Indicates that there is a risk of personal injury from a nonelectrical hazard, or risk of irreversible damage to data, software, or the operating system. A caution indicates that the possibility of a hazard (as opposed to certainty) might happen, depending on the action of the user.

This indicates that a slide is related to the material in the student guide at the location of the icon. The title of slide appears above the visual aid icon.

Typographical Conventions

`Courier` is used for the names of commands, files, directories, programming code, and on-screen computer output; for example:

> Use `ls -al` to list all files.
> `system% You have mail.`

`Courier` is also used to indicate programming constructs, such as class names, methods, and keywords; for example:

> The `getServletInfo` method is used to get author information.
> The `java.awt.Dialog` class contains `Dialog` constructor.

`Courier bold` is used for characters and numbers that you type; for example:

> To list the files in this directory, type:
> `# ls`

`Courier bold` is also used for each line of programming code that is referenced in a textual description; for example:

```
1 import java.io.*;
2 import javax.servlet.*;
3 import javax.servlet.http.*;
```

> Notice the `javax.servlet` interface is imported to allow access to its life cycle methods (Line 2).

`Courier italics` is used for variables and command-line placeholders that are replaced with a real name or value; for example:

> To delete a file, use the `rm` *`filename`* command.

`Courier italic bold` is used to represent variables whose values are to be entered by the student as part of an activity; for example:

> Type **`chmod a+rwx`** ***`filename`*** to grant read, write, and execute rights for *`filename`* to world, group, and users.

Palatino italics is used for book titles, new words or terms, or words that you want to emphasize; for example:

> Read Chapter 6 in the *User's Guide.*
> These are called *class* options.

Additional Conventions

Java programming language examples use the following additional conventions:

- Method names are not followed with parentheses unless a formal or actual parameter list is shown; for example:

  "The `doIt` method..." refers to any method called `doIt`.

  "The `doIt()` method..." refers to a method called `doIt` that takes no arguments.

- Line breaks occur only where there are separations (commas), conjunctions (operators), or white space in the code. Broken code is indented four spaces under the starting code.

- If a command used in the Solaris OS is different from a command used in the Microsoft Windows platform, both commands are shown; for example:

  If working in the Solaris OS

  ```
  $ cd $SERVER_ROOT/bin
  ```

  If working in Microsoft Windows

  ```
  C:\> cd %SERVER_ROOT%\bin
  ```