This course supports the assessment for Software I. The course covers 7 competencies and represents 6 competency units.

Introduction

Overview
The Java Standard Edition 6 Programmer Certified Professional Exam certification focuses on technology, and you will demonstrate proficiency in the fundamentals of the Java programming language.

Achieving this certification provides clear evidence that you understand the basic syntax and structure of the Java programming language. The competencies and skills you earn will make you valuable in the highly competitive world of information technology and are portable from one organization to another.

As a student seeking a degree in information technology, acquiring competencies with Java programming, especially hands-on practical skills, is one of the primary goals of the Software I Course at Western Governors University (WGU).

As you study, consider the following questions:

What are the competencies and their associated skill sets that you should have acquired by the time you finish this course?

How can you make use of the learning resources to help you successfully pass your assessment?

And, finally, where can you get help or technical assistance if you encounter issues in your studies of Software I?

It is the purpose of this course to provide the highlights that will facilitate your learning and mastering Java programming.

Learning and mastering Java programming requires basic working knowledge of the technology. Any basic concepts about hardware, software, drivers, applications, and knowledge in computer programming will definitely contribute to your learning process.

To facilitate your learning, this course has incorporated within it learning resources such as textbooks, hands-on practices, and preassessments in a way that will help you learn and master Java programming within 12 weeks. You can study the learning materials based on the recommendations specified in the Software I Course while having the opportunity to ask for help from your peers, your fellow students, and your course instructor.

Your course instructor is ready to assist you in your endeavors to learn and master Java programming.

Being competent in Java programming provides a number of benefits: you can use the skills acquired to develop software in Java both at home and in a work environment.

For many, passing the Java Standard Edition 6 Programmer Certified Professional Exam(ANV1) assessment provides them with a reputable certification that is highly valued in the IT industry today. This certification may help you get a start in the IT industry or help you obtain promotion in your current working environment.

Most important of all, mastering Java programming will help pave the way for your great success in the IT field, including the opportunity to become a senior programmer or a leader in an IT department. In a word, your hard work on Java programming is worth your time and effort in the years to come.

Different chapters of the textbook designated for studying the Software I Course are allocated to each of the 7 competencies accordingly. You can see detailed information on those competencies and their associated chapters while working on them as arranged.

Watch the following video for an introduction to this course:

Competencies
This course provides guidance to help you demonstrate the following 7 competencies:

- **Competency 430.2.1: Declaration of Classes, Interfaces, and Variables**
  The graduate develops and uses classes, interfaces, and variables in code development.
- **Competency 430.2.2: Object-Oriented Development**
  The graduate uses object-oriented concepts and programming techniques to develop applications that are flexible and maintainable.
- **Competency 430.2.3: Flow of Contr**ol
  The graduate applies appropriate control structures to develop robust applications.
- **Competency 430.2.4: Strings, Streams, and Parsing**
  The graduate uses appropriate Application Programming Interface (API) classes and interfaces to perform efficient string, pattern, and stream processing.
- **Competency 430.2.5: Threads**
  The graduate writes code in applicable languages to demonstrate concurrency.
- **Competency 430.2.6: Generics and Collections**
  The graduate writes code in specified languages to implement generics and collections.
- **Competency 430.2.7: Application Development Environment**
  The graduate writes code in specified languages using the application development environment.

Course Instructor Assistance
As you prepare to successfully demonstrate competency in this subject, remember that course instructors stand ready to help you reach your educational goals. As subject matter experts,

mentors enjoy and take pride in helping students become reflective learners, problem solvers, and critical thinkers. Course instructors are excited to hear from you and eager to work with you.

Successful students report that working with a course instructor is the key to their success. Course instructors are able to share tips on approaches, tools, and skills that can help you apply the content you're studying. They also provide guidance in assessment preparation strategies and troubleshoot areas of deficiency. Even if things don't work out on your first try, course instructors act as a support system to guide you through the revision process. You should expect to work with course instructors for the duration of your coursework, so you are welcome to contact them as soon as you begin. Course instructors are fully committed to your success!

Preparing for Success

The information in this section is provided to detail the resources available for you to use as you complete this course.

Learning Resources

The learning resources listed in this section are required to complete the activities in this course. For many resources, WGU has provided automatic access through the course. However, you may need to manually enroll in or independently acquire other resources. Read the full instructions provided to ensure that you have access to all of your resources in a timely manner.

Automatically Enrolled Resources

You will be automatically enrolled at the activity level for the following learning resources. Simply click on the links provided in the activities to access the learning materials.

SkillSoft and Books24x7

You will access the following SkillSoft items within this course. For more information on accessing SkillSoft items, please see the "**Accessing SkillSoft Learning Resources**" page.

The following Books24x7 e-texts will be used in this course:

- ***SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*** by Katherine Sierra and Bert Bates, June 2008, ISBN: 978-0-07-159106-5.

Note: These e-texts are available to you as part of your program tuition and fees, but you may purchase hard copies at your own expense through a retailer of your choice. If you choose to do so, please use the ISBN listed to ensure that you receive the correct edition.

uCertify

You will access the following resource from uCertify at the activity level within this course:

- 1Z0-851 Java Standard Edition 6 Programmer Certified Professional Exam

Note: The online version of the uCertify learning resource is provided to you by WGU as part of your enrollment. If you wish to use the offline version, you may purchase it yourself from

uCertify.

Additional Preparations

As you work through every chapter in the **SCJP Sun Certified Programmer for Java 6 Study Guide**, make note of the following pointers:

- Read and reread all of the two-minute drills. The two-minute drills material makes excellent flash cards.
- Read and reread the exam watch notes. Also, note any references to material that is specified in the textbook that is covered on the exam.
- Complete any exercises listed in the chapters.
- Take and retake the self-tests. It is recommended to take the chapter's self-test shortly after you have completed the chapter. Also take all of the self-tests in one sitting after you complete all of the chapters (pretend that you are taking the actual exam).
- Write lots of Java code. Experiment with the sample code in each chapter.

Taking the Diagnostic Test

Access the following uCertify courseware:

- **1Z0-851 Java Standard Edition 6 Programmer Certified Professional Exam**

Take the following preassessment in the uCertify PrepKit now:

- the Diagnostic Test in Test Mode

Once you have taken it, review your summary report with your mentor. This report will identify those exam objectives in which you need preparation. You should not continue with the uCertify PrepKit until instructed to do so later in this course.

Install NetBeans onto your PC

You will need to spend time writing Java code to practice the concepts and strengthen your knowledge of the Java syntax. NetBeans is the recommended Integrated Development Environment. This IDE will provide you an editor, debugger, and the Java Developers Kit. You will need NetBeans and the Java SE Developer's Kit. You can download the SDK bundled with NetBeans (recommended).

Download and follow installation instructions for **Java SE NetBeans IDE Download Bundle**.

Java SkillSoft eLearning

The following additional SkillSoft materials are available for your studies:

Mentoring: 1Z0-851 Sun Certified Programmer for the Java 2 Platform, Standard Edition 6.0

SkillSoft has mentors available to help 24 hours a day, 7 days a week, to address your Java-related questions. Access the **SkillSoft Home Page** and search for "IZO-851" in the

"Mentoring" category to utilize this resource.

Java Knowledge Center
Practice Labs, comprising sets of simulations and hands-on exercises, will help you to practice and assess your coding skills in engaging, real-world scenarios.

Access this resource at the following link:

- **Java Resource Center**

TestPrep 1Z0-851 Certified Programmer for the Java 2 Platform, SE 6.0
The practice exam is available in certification and study mode.

- **TestPrep CX-310-065 Certified Programmer for the Java 2 Platform, SE 6.0**

Note: Taking this in certification mode can be used as a secondary pre-assessment to the uCertify Diagnostic Test.

Use these results combined with the uCertify Diagnostic Test results to develop a study strategy that helps you strengthen weak areas.

Activities at a Glance

This course includes an outline to help you briefly visualize the elements of this course. This resource will also provide general guidelines for your work through the course and can be printed and used as a quick checklist for your progress.

- "**Software Activities at a Glance**"

Declarations, Initialization, and Scoping (Part I)

This section focuses on understanding main principles and elements of java, concepts related to packages and classes, declarations, and access control.

Declarations and Access Control (Chapter 1)

This topic will introduce you to basic Java concepts, declaring and developing classes and interfaces, using primitives and arrays, and using static methods. The objectives of chapter 1 instruct you on developing code that declares classes, interfaces, and enums and on using packages and import statements. You will develop code that declares, initializes, and uses primitives, arrays, instance, and local variables. You will develop code that declares static methods.

This topic addresses the following competency:

- **Competency 430.2.1: Declaration of Classes, Interfaces, and Variables**

The graduate develops and uses classes, interfaces, and variables in code development.

Chapter 1: Declarations and Access Control

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851):*

- chapter 1 ("Declarations and Access Control")

Complete exercise 1-1. Then complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self-test:

- What are the rules associated with creating legal identifiers and source code declarations?
- How does access control relate to classes, methods, and variables?
- What is the difference between abstract and nonabstract classes?
- How do you implement an interface?
- What are the different access modifiers, and how do they work?
- What are the relative sizes of the numeric primitives?
- What is an enum and how is it used?

Declarations, Initialization, and Scoping (Part II)

This section focuses on understanding assignments and operators.

Assignments and Operators (Chapters 3 and 4)

This topic will help you to understand using class members, developing wrapper and autoboxing code, determining the effects of passing variables into methods, recognizing when objects become eligible for garbage collection, and using operators. The objectives in chapter 3 will instruct you on how to write code that applies to assignment operators, pass variables into methods, develop code that uses wrapper classes, and recognize when an object is eligible for garbage collection. The objectives in chapter 4 will instruct you on how to write code that applies to assignment operators, arithmetic operators, relational operators, the instance of operator, and logical and conditional operators.

This topic addresses the following competencies:

- **Competency 430.2.1: Declaration of Classes, Interfaces, and Variables**
  The graduate develops and uses classes, interfaces, and variables in code development.
- **Competency 430.2.7: Application Development Environment**
  The graduate writes code in specified languages using the application development environment.

Chapter 3: Assignments

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 3 ("Assignments")

Complete exercises 3-1 and 3-2. Then complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self-test:

- What is the difference between stack and heap memory, and which objects are stored in each?
- How are values assigned to primitives and reference variables, and what are the rules for casting primitives?
- What are the implications of uninitialized variables?
- How do you declare, construct, and initialize a multi-dimensional array?
- How do you use wrappers?
- How do you use boxing?
- What is the purpose of garbage collecting?

Chapter 4: Operators

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 4 ("Operators")

Complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self- test.

- What are compound assignment operators?
- What are relational operators, and how are they used?
- What is the instance of operator, and how is it used?
- What are the arithmetic operators?
- How is an expression evaluated, and how do you change sequence on how it is evaluated?
- What is a string concatenation operator, and how is it used?
- How do the increment and decrement operators function, and what is the difference between prefix and postfix?
- What is a conditional operator, and how does it compare to an if statement?
- What are the logical operators, and how does each operate?

Flow Control (Part I)

This section focuses on understanding flow control, exceptions, and assertions.

Flow Control, Exceptions, and Assertions

This topic will help you to understand using if and switch statements; developing loop constructs; developing code with assertions; using try, catch, and finally statements; and recognizing common exceptions and their effects. The objectives in chapter 5 will instruct you on how to develop code that implements decision statements, develop code that implements all forms of loops and iterators, and develop code that makes use of exceptions and exception handling clauses.

This topic addresses the following competency:

- **Competency 430.2.3: Flow Control**
  The graduate applies appropriate control structures to develop robust applications.

Chapter 5: Flow Control, Exceptions, and Assertions

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 5 ("Flow Control, Exceptions, and Assertions")

Complete exercises 5-1, 5-2, 5-3, and 5-4. Then complete the associated two-minute drill and self-test at the end of the chapter.

Note: This chapter is scheduled for both sections on Flow Control

Consider the following questions as you work through the chapter and self-test:

- What is the basic format of an if statement?
- How do you write the code for if-else branching?
- How are switch statements used?
- How do you write the code for a switch statement?
- How do you control flow within a switch statement?
- How is the default case used?
- When is a while loop used?
- When is a do loop used?
- When is a for loop used?
- How do you write the code for a basic for loop and its iteration expression?
- How do you write the code using the for loop to iterate through arrays?
- What is the difference in the BREAK and CONTINUE?
- How are unlabeled and label statements handled?

Flow Control (Part II)

This section focuses on understanding flow control, exceptions, and assertions.

Flow Control, Exceptions, and Assertions (Continued)

This topic will help you to understand using if and switch statements; developing looping statements; developing code with assertions; using try, catch and finally statements; and recognizing common exceptions and their effects. The objectives in chapter 5 will instruct you on how to develop code that implements decision statements, develop code that implements all forms of loops and iterators, and develop code that makes use of exceptions and exception handling clauses.

This topic addresses the following competency:

- **Competency 430.2.3: Flow Control**
  The graduate applies appropriate control structures to develop robust applications.

Chapter 5: Flow Control, Exceptions, and Assertions

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 5 ("Flow Control, Exceptions, and Assertions")

Complete exercises 5-1, 5-2, 5-3, and 5-4. Then complete the associated two-minute drill and self-test at the end of the chapter.

Note: This chapter is scheduled for both sections on Flow Control.

Consider the following questions as you work through the chapter and self-test:

- How do you write the code to catch an exception using try and catch?
- How is a finally block used in an exception?
- How do you handle and propagate an uncaught exception?
- Where do exceptions come from?
- What are assertions?
- How do you enable assertions?
- What are the dos and don'ts in using assertions?

Practice - Assertions

Complete the following practice programming advanced Java programming exercises:

- Lab 1: Assertions

  Write a program that generates a random number between 1 and 50. If the number is larger than 30, the program throws an assertion. Print the number generated before using
  the assertion.

  Similar output will look like this:

  C:\Java Sandbox\UsingAssertions>javac -source 1.5 Using Assertions
  C:\Java Sandbox\UsingAssertions>java -ea UsingAssertions
  Practice using Assertions
  Random number = 3
  C:\Java Sandbox\UsingAssertions>java -ea UsingAssertions
  Practice using Assertions
  Random number = 18
  C:\Java Sandbox\UsingAssertions>java -ea UsingAssertions
  Practice using Assertions
  Random number = 36
  Exception in thread "main" java.lang.AssertionError: Rando
  at UsingAssertions.main(UsingAssertions.java:11)

Hints:

- Use this formula to truncate the generated double and return an integer between 1 and 50: int randomInt = (int)(Math.random()*50).
- Important compiler note: Don't forget to compile using javac-source <version> option and run the program using the -enableassertions or –ea argument. If you forgot what version of the java compiler you are running, first type javac -version. If you were running 1.5, you would type:
  javac -source 1.5 UsingAssertions.javac
  java -ea UsingAssertions

OO Concepts

This section focuses on understanding object orientation concepts.

Object Orientation

This topic will help you to understand object orientation concepts. The objectives covered in chapter 2 will instruct you on developing code that implements tight encapsulation, loose coupling, and high cohesion; develop code that implements inheritance; develop code that depicts polymorphism; develop constructors; and develop code that declares overloaded methods and constructors.

This topic addresses the following competencies:

- **Competency 430.2.1: Declaration of Classes, Interfaces, and Variables**
  The graduate develops and uses classes, interfaces, and variables in code development.
- **Competency 430.2.2: Object-Oriented Development**
  The graduate uses object-oriented concepts and programming techniques to develop applications that are flexible and maintainable.

Chapter 2: Object Orientation

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (CX-310-065)*:

- chapter 2 ("Object Orientation")

Complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self-test:

- What is the importance of encapsulation in good OO design?
- What is the importance of inheritance?
- How do you write code to implement Is-A and Has-A relationship?
- What are the key things to remember about references?
- What is the difference between overridden and overloaded methods?
- What is the difference between downcasting and upcasting?
- How do you write the code to implement an interface?
- What are the legal return types?
- What is a constructor?
- How do you instantiate a constructor?

- How do you write the code to overload a constructor?
- What are static variables and methods and how are they used?
- How is loose coupling and high cohesion used in good OO design?

Practice - Auto-boxing and Auto-unboxing

Complete the following practice programming advanced Java programming exercises:

- Lab 2: Auto-boxing and auto-unboxing

  Auto-boxing and auto-unboxing was introduced in JDK 1.5. It makes it easy to convert back and forth between a wrapper and its corresponding primitive:

Primitive	Wrapper
byte	Java.lang.Byte
char	Java.lang.Character
double	Java.lang.Double
float	Java.lang.Float
int	Java.lang.Integer
short	Java.lang.Short
boolean	Java.lang.Boolean
void	Java.lang.Void

  Without using auto-boxing and auto-unboxing, write a program that stores an int "a" into Vector "vt" by using a wrapper class. Print the element stored at position "0" of vector "vt".
  Notice that you have to use casting.

  Using auto-boxing and auto-unboxing, write a program that stores an int "a" into Vector "vt" to demonstrate that wrapper types are automatically converted into their primitive equivalents if needed for assignments or method or constructor invocations.

API Contents

This section focuses on understanding the use of strings, I/O, formatting, and parsing.

Strings, I/O, Formatting, Parsing

This topic will explain the use of String, StringBuilder, and StringBuffer; file I/O and serialization using java.io package; working with dates, numbers, and currencies; and the use of regular expressions. The objectives covered in chapter 6 will instruct you on how to develop code for string objects; develop code for navigating files systems and reading from and writing to files; develop code that serializes objects; and develop code that formats or parses dates, numbers, and currency values.

This topic addresses the following competency:

- **Competency 430.2.4: Strings, Streams, and Parsing**
  The graduate uses appropriate Application Programming Interface (API) classes and interfaces to perform efficient string, pattern, and stream processing.

Chapter 6: Strings, I/O, Formatting, and Parsing

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851):*

- chapter 6 ("Strings, I/O, Formatting, and Parsing")

Complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self- test:

- What are the differences between the String, StringBuilder, and StringBuffer classes?
- Why are strings immutable objects?
- How do you write code to create a new string?
- What are the important methods most commonly used in the String class?
- How are StringBuffer and StringBuilder used?
- What are the classes you need to know in java.io?
- What is the concept of serialization?
- How does inheritance affect serialization?
- What are the common use cases when working with dates and numbers?
- How do you find specific pieces of data in large data sources?
- How is a piece of source data tokenized?
- How would you format the source data?

Practice - DataInputStream and DataOutputStream

Complete the following practice programming advanced Java programming exercises:

- Lab 3: DataInputStream and DataOutputStream
- Write a program that reads a Java source file and writes it back without any changes under a new name. Use FileInputStream and FileOutputStream in your solution.

Practice - Locale Class

Complete the following practice programming advanced Java programming exercises:

- Lab 4: Locale class (format and parsing dates and currency)

  1. Write a program that will list all locales using getAvailableLocales().
  2. Write a program that will display the current date in US, FR, and DE.
  3. Write a program that formats the currency amount 1234.56 for ENGLISH (1234.56), CANADA (1234.56), FRANCE (1234,56), and ITALIAN (1234,56).

  For ideas on
  how to use the
  locale class,
  refer to the

examples found
on this website:
http://www.java2s.com/Code/Java/Development-Class/ListallLocale.htm.

Practice - Formatting Parameters

Complete the following practice programming advanced Java programming exercises:

- Lab 5: Formatting Parameters (%b, %c, %d, %f, %s)

  1. Create an application that prints the value of PI using the following formatting:
  double pi = Math.PI;
  %5.3f
  %8.3f
  %10.5f
  %f
  %b
  %s

- What happens if you try to print pi using %d?
- What happens if you use %d (int)pi?

  2. Create an application that uses %c to print the Unicode equivalent for:
  '\u0041'
  '\u0042'
  '\u0043'
  '\u0044'
  '\u0045'

  Your output should look like this:
  A
  B
  C
  D
  E

Collections and Generics

This section focuses on understanding the simplicity and complexity of generics, the use of generics in collections, and how to sort and search through collections.

Generics and Collections

The topics explain design using collections, override equals() and hashCode(); distinguishing == and equals(); using generic versions of collections including Set, List, and Map; using type parameters and writing generic methods; sorting and searching using java.util; and using comparable and comparator. The objectives covered in chapter 7 will instruct you on how to develop code to implement an equals() method, implement hashCode(), sort collections and arrays, and develop code that uses generic versions of the Collections API.

This topic addresses the following competency:

- **Competency 430.2.6: Collections and Generics**
  The graduate writes code in specified languages to implement generics and collections.

Chapter 7: Generics and Collections

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 7 ("Generics and Collections")

Complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self-test:

- What are the correct and incorrect overrides of corresponding hashCode and equals methods?
- What is the difference between == and the equals method?
- What do you do with a collection?
- How do you sort and search arrays and lists?
- What are the key methods for collections and arrays?
- How do you write code that uses the generic versions of the Collections API (the Set, List, and Map interfaces)?
- How do you mix generic and non-generic collections?
- How do you write code for generic methods and declarations?

Practice - Using the Set, List, and Map Interfaces

Complete the following practice advanced Java programming exercises:

- Lab 7: Using the Set, List, and Map Interfaces

  1. Create an application that takes the words in its argument list and prints out any duplicate words and a list of the words with duplicates eliminated. Make use of the HashSet.
  Refer to this
  link for
  inspiration: http://download.oracle.com/javase/tutorial/collections/interfaces/set.html2. 2. Create an application that uses an ArrayList that stores names entered by a user until the user is done entering names. When testing, try to use a list with at least 5 or more names.
- Using an iterator, print each element in the ArrayList.
- Create a subset of the list by removing the first and last element of the list and print the result.
- Sort the subset list created in b (use the sort() method) and print the result.
- Reverse the sorted subset list (use the reverse() method) and print the result.
- Cover the original list into an Array and print each element.

  1. Complete the application in part 2 but use a LinkedList.

2. Create an application that uses a HashMap to store last names and phone numbers. Use the name as the key.

- Write a test case that looks up a person's phone number.
- Print out the keys by looping over keySet().
- Print out the values by looping over values().
- Print out the key/value pairs by looping through entrySet().
- Sort the keys first and then print the sorted keys.

3. Complete exercise 4 using a TreeMap to store last names and phone numbers. Use the name as the key.
4. What did you notice about the key order for HashMap and TreeMap. Do they both keep keys in order?

Practice - Sorting an Array

Complete the following practice programming advanced Java programming exercises:

- Lab 8: Sorting an Array

1. Write an application that creates a person class that contains 3 attributes. Create the necessary setters and getters.
String firstName;
String lastName;
int age;

2. Create a test program that populates an array of type 'Person' class. Populate the array with at least 4 person objects.

- Print the contents of the array by the natural ordering of its elements.
- Sort the array by implementing the comparable interface and sort according to age. Print the sorted array.

The solution can be found at this link:
Making Java Objects Comparable

Practice - Generic Methods and Wildcards

Complete the following practice programming advanced Java programming exercises:

- Lab 9: Generic methods and Wildcards

1. Write a method that takes an array of objects and a collection and puts all objects in the array into the collection.

- First solve this by creating a generic method called fromArrayToCollection. Make use of the add() method. You should be able to call this method with any kind of collection whose element type is a supertype of the element type of the array.
- Solve this by using the Collections.copy() method which takes two parameters: dest (destination) and src (source). Create a signature of copy which the parameter dest expresses dependency between its type and its return type and use a wildcard for

the src.

Refer to this example for inspiration:

- Generic Methods

Declarations, Initialization, and Scoping

This section focuses on understanding the rules and syntax of inner classes.

Inner Classes

This topic will help you to understand the use of inner classes, method-local inner classes, anonymous inner classes, and static nested classes. The objectives covered in chapter 8 will instruct you on how to develop code which implements inner, anonymous, and static nested classes.

This topic addresses the following competency:

- **Competency 430.2.1: Declaration of Classes, Interfaces, and Variables**
  The graduate develops and uses classes, interfaces, and variables in code development.

Chapter 8: Inner Classes

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 8 ("Inner Classes")

Complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self-test:

- How do you declare and instantiate an inner class?
- How do you reference the inner or outer instance from within the inner class?
- What can a method-local inner object do and not do?
- What is an anonymous inner class and how is it declared?
- How do you instantiate and use static nested classes?

Concurrency

This section focuses on understanding the defining, instantiating, and starting threads.

Threads

This topic will help you to understand starting new threads, recognizing thread states and transitions, using object locking to avoid concurrent access, and writing code that uses wait(), notify(), or notifyAll(). The objectives covered in chapter 9 will instruct you on how to develop code to define, instantiate, and start new threads; develop code using object locking to protect static and instance variables from concurrent access issues; and develop code that implements thread interaction.

This topic addresses the following competency:

- **Competency 430.2.5: Threads**
  The graduate writes code in applicable languages to demonstrate concurrency.

Chapter 9: Threads

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 9 ("Threads")

Complete exercises 9-1 and 9-2. Then complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self-test:

- How do you define, instantiate, and start threads?
- How do you start and run multiple threads?
- How do you recognize thread states and transitions?
- How do you use object locking to avoid concurrent access?
- How do you write code that uses wait(), notify(), or notifyAll()?

Fundamentals

This section focuses on understanding the advanced capabilities of the java and javac commands, the use of packages, and how to search for classes that live in packages.

Development

The following topics will help you to understand the use of packages and imports, determine runtime behavior for classes and command-lines, understand the use of classes in JAR files, and understand the use of classpaths to compile code. The objectives in chapter 10 will instruct you on how to compile and launch Java applications, searching for other classes, using a JAR file to bundle up and distribute an application, and develop code that implements static imports.

This topic addresses the following competency:

- **Competency 430.2.7: Application Development Environment**
  The graduate writes code in specified languages using the application development environment.

Chapter 10: Development

Read the following in *SCJP Sun Certified Programmer for Java 6 Study Guide (1Z0-851)*:

- chapter 10 ("Development")

Complete the associated two-minute drill and self-test at the end of the chapter. Consider the following questions as you work through the chapter and self-test:

- What is the javac command and how is it used?
- How do you launch applications in Java?
- How do you determine runtime behavior for classes and command-lines?
- What are JAR files and how are they used in searching?
- What is the purpose of static imports?
- How do you use classpaths to compile code?

Practice - Using AWT and Swing

Complete the following practice programming advanced Java programming exercise:

- Lab 10: Using AWT and Swing
- Create a graphical user interface that computes student loan payments given the loan amount, interest rate, and number of months. The interface should include a combo box in a scroll pane, a button, and at least 1 text Area.

Solidify Your Preparation Using uCertify

The uCertify PrepKit is a hefty tool for exam preparation, but is not very useful without the grounding provided by the *SCJP Sun Certified Programmer for Java 6 Study Guide,* or solid experience with Java programming.

Using uCertify

The uCertify PrepKit combines targeted learning with a practice exam approach. The uCertify learning resource is meant to complement the other learning resources for this course.

uCertify

Complete the following uCertify review course:

- 1Z0-851 Java Standard Edition 6 Programmer Certified Professional Exam

Please use the steps listed on following webpage to guide your study of this course:

- "Using uCertify"

Final Steps

Congratulations on completing the activities in this course! This course has prepared you to complete the assessment associated with this course. If you have not already been directed to

complete the assessment, schedule and complete your assessment now.

Review of Competencies and Associated Chapters

This course of study covers 7 competencies of Soffware I Course of Study. Those 7 competencies and their associated chapters are:

Competency 430.2.1: Declaration of Classes, Interfaces, and Variables
The graduate develops and uses classes, interfaces, and variables in code development.

- Chapter 1: Declarations and Access Control
- Chapter 2: Object Orientation
- Chapter 3: Assignments
- Chapter 8: Inner Classes

Competency 430.2.2: Object-Oriented Development
The graduate uses object-oriented concepts and programming techniques to develop applications
that are flexible and maintainable.

- Chapter 2: Object Orientation

Competency 430.2.3: Flow of Control
The graduate applies appropriate control structures to develop robust applications.

- Chapter 5: Flow Control, Exceptions, and Assertions

Competency 430.2.4: Strings, Streams, and Parsing
The graduate uses appropriate Application Programming Interface (API) classes and interfaces to perform efficient string, pattern, and stream processing.

- Chapter 6: Strings, I/O, Formatting, and Parsing

Competency 430.2.5: Threads
The graduate writes code in applicable languages to demonstrate concurrency.

- Chapter 9: Threads

Competency 430.2.6: Generics and Collections
The graduate writes code in specified languages to implement generics and Collections.

- Chapter 7: Generics and Collections

Competency 430.2.7: Application Development Environment
The graduate writes code in specified languages using the application development

environment.

- Chapter 4: Operators
- Chapter 10: Development

Transfer/Application to Work

Successful completion of the ANV1 assessment validates the hands-on skills and knowledge that a Java programmer is expected to understand and use. Skills include knowledge of the Java programming language design principles and how to apply these principles to create software applications. The knowledge and skill base that are necessary to pass this assessment are valuable to individuals working in fields such as application development and software engineering.

Acquiring knowledge from textbooks is only part of the learning process. The ultimate goal of learning is to turn knowledge into skills that can be readily applied in the practical field. The following are some recommendations for transferring knowledge acquired through textbooks into practical skills:

- **Emphasis on hands-on practices:** Practice makes perfect. Hands-on practices help turn short-term memory into long-term memory. Finally, personal experiences also help reinforce learning outcomes.
- **Research on solutions through various channels:** There are many ways to research solutions on issues associated with Java programming, including using the Internet to look for solutions that are not addressed in textbooks. Since new issues may appear in the real work environment, textbooks are not good enough to cover all of them. Learning through research helps explore new solutions to new problems.
- **Collaborate and cooperate with peers or other students:** There are different ways of learning, including the use of cooperation and collaboration to facilitate learning processes. Working with your coworkers, fellow classmates, and even with other students on the message boards will definitely have a positive impact on your learning outcomes.

Next Steps

Once you have completed all the tasks associated with the competencies, chapters, activities, lesson reviews, hands-on practices, and pre-assessment(s), you can start scheduling for the actual assessment at a PearsonVue testing center.

Our best wishes for you as you take the Oracle Certified Professional Java Standard Edition 6 Programmer Certified Professional Exam (1Z0-851), which represents the objective assessment for this course.

Taking the Assessment

Schedule and take the following exam. If you have any questions about this process, talk to

your mentor first.

- Oracle Certified Professional Java Standard Edition 6 Programmer Certified Professional Exam (1Z0-851)

Note: This exam represents the objective assessment for this course of study.

Follow these directions for accessing your objective assessments.

Note: When scheduling, you may wish to choose a date that is about a month out to give you time to solidify your understanding of any material about which you may have doubt.

Following the Outside Vendor Assessment

You will need to submit your scores to WGU after completing your exam.

Follow these directions after completing an outside vendor assessment.